Python 2 Birinici buraxılış.....

Python proqramlama dili

Rəşad Qarayev

Python

programlama dili

Müəllif : Rəşad Qarayev

əlaqə üçün https://www.facebook.com/rashad.garayev

email pythonaz@yahoo.com

blogu izləyərək daima python-a daha yaxın olun

http://rpirates.wixsite.com/alternativ-az

pythonla bağlı müzəkirəli mövzular üçün facebook səhifəmiz

https://www.facebook.com/python-azcom

Kitab orta məktəb şagirdləri və ali təhsil tələbələri üçün nəzərdə tutulub.
Kitabda olan bəhsləri qavramaq üçün ilkin orta məktəb riyazi biliy,eləcədə linux əməliyyat sistemindən müəyyən qədər məlumat olmağı tələb edir.
Kitab,yüksək səviyyədə ingilis dili tələb etmir.
Çalışıb sizlərə python 2 və proqramlamanın ilkin addımlarını izah etmişəm.
Kitab haqqında təklif və rəylərinizi yuxarıda qeyd etdiyim ünvanlara bildirə bilərsiniz.

Mündəricat

1. Python haqqında
2. Yükləmə qaydaları və versiya
3. Pythonda sabit ifadələr
4. print() funksiyası
5. Riyazi operatorlar
6. input() və raw_input() funksiyaları
7. Cinslərin dəyişdirilməsi
8. if,else elif operatorları
9. Xüsusi işarələr.
10. len() funksiyası
11. while operatoru
12. for operatoru
13. range() funksiyası
14. break operatoru
15. continue operatoru
16. in ifadəsi
17. Listlər
18. List metodları
19. Tupllar
20. Tupl metodları
21. Dictlər
22. Funksiyalar
23. def ifadəsi
24. global operatoru
25. Xətalar.try except blok operatoru
26. Xətaları qabaqlama metodları
27. return operatoru
28. pass operatoru
29. Modullar(importing modules)
30. Sabit modullar və operatorları
31. os modulu
32. os modul funksiyaları
33. Fayllar.Fayl yaratma
34. String modulu
35. string modul metodları
36. re modulu
37. re modulu metodları
38. Xüsusi simbol və xarakterlər
39.Siniflər (class)
40.math modulu
39.Qrafik proqramlama.Tkinter bəhsi

Python

Python,**Guido Van Rossum** (Hollandiyalı) tərəfindən 1990-cı ildə labaratoriyada yazılmışdır.
1. Az müddətə öyrənilə bilinir.
2.Az kodlarla böyük layihələrə imza atılır
3.Sistemdə kökdən işləyə bilir.
Eləcədə pythonu bir çox şirkətlər hal-hazırda istifadə etməkdə qərarlıdırlar.Google,Yahoo,NASA,Dropbox kimi bir çox şirkətlər python programçılarına iş vakansiyaları ayırırlar.Guido V.Rossum özüdə 2005-ci ildən 2011-ci ilə qədər Google şirkətində çalışmışdır.
Eləcədə YouTube şəbəkəsində videroliklərin hazırlanmasında geniş istifadə olunur.

BitTorrent pythondan faylların dəyişdirilməsi üçün istifadə edir.
App Engine,EVE Online,Massively Multiplayer Online Game(MMOG),Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm və IBM
Animasiyalı filmlərin hazırlanmasında Industrial Light & Magic, Pixar və s geniş istifadə edir.
Finans sektorlarında JPMorgan Chase, UBS, Getco, Citadel eləcədə NASA, Los Alamos, Fermilab, JPL şifrələmə əməliyyatlarında pythondan istifadə edir.

Rəsmi ünvanı
http://www.python.org.

Terminalı açın və

```
 $python -V
Python 2.7.12rc1
```

Deməli python2 bizdə yüklü haldadır.
Əgər siz

```
$ python -V
bash: Python: command not found
```

xəta ilə qarşılaşarsanız deməli python sisteminizə yüklənməmişdir.
Terminalı açın

apt-get install python
və ya

sudo apt-get install python
Əgər python3- ü yükləmək istəyirsinizsə eyni qayda ilə
sudo apt-get install python3
Python bir çox əməliyyat sistemlərində yüklü haldadır.GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS2,Amiga,AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks,
PlayStation, Sharp Zaurus, Windows CE və PocketPC.
.Eləcədə http://www.python.org/download ünvanından əməliyyat sisteminizə uyğun olaraq(for Linux, Unix or Mac OS X) tar.gz yükləyib
və ya
/tmp wget 'url'
./configure
make
sudo make install
Tək yuxarıdakı link deyil,araşdırsanız başqa adreslərdən də pythonu yükləyə bilərsiniz.Amma təklif edirəmki rəsmi saytından yükləsəniz daha məqsədəuyğundur

Pythonu öyrəndiyimiz ərəfədə bizə idle və yagrafik programlar lazım olacaq.idle,yazacağınız kod bloklarını test etmək üçün ideal grafik programıdır..İnternet üzərindən **idle** haqqında bir çox mənbələrə rast gələ və ətraflı məlumat ala bilərsiniz.Bütün sistemlərdə yüklü olan idle bir mətn yaza biləcəyimiz idle-dir.Və ya terminaldan çağıra biləcəyimiz python.Digər idle-dən Geany,Eric python idle,python3 idle və s misal çəkmək olar.Əgər geany sisteminizə yükləmək istəyirsinizsə (Backbox-da yüklü haldadır) sudo apt-get install geany.
sudo apt-get install idle #bu sisteminizə python2xxx versiyası yüklənəcək
sudo apt-get install idle3 #sisteminizə python3xx versiyası yüklənəcək
sudo apt-get install eric
sudo apt-get install pycrust
İdle -lərin bir çoxu C və C++ dillərində yazılmışdır.
Pythonla biz nələr edə bilərik?!Pythonla sistem programlaşdırmaq,GUİ programlama,internet səhifələri,pygame -oyunlar,eləcədə UNİX-i pythondan çalışdırma raspberry pi və arduino mikrokontrol -ı python ilə programlaşdırmaq,eləcədə pythonun təklif etdiyi micropythonu python ilə programlaşdrma,mobil programlar və sairə.
Pythonun bazası böyükdür.Sisteminizə kivy yükləyib ios,ipad və android sistemləri üçün pythonla oyunlar yaza bilərsiniz.
Saytların hazırlanmasında isə Django daha çox istifadə olunur.Django ilə hazırlanan saytlardan biri də 'The Washington post' u misal çəkmək olar.

Terminalı açın və python yazın,sisteminizdə pythonun hansı versiyası yüklü olduğu göstəriləcək

```
backbox@backbox-pc:~$ python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Əgər python 3 ün yüklü olduğunu test etmək istəyirsinizsə eyni qayda ilə python3 yazın.

```
backbox@backbox-pc:~$ python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
backbox@backbox-pc:~$ python3
Python 3.4.3 (default, Oct 14 2015, 20:28:29)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Deməli sistemimizdə python 2 və 3 yüklüdür.Bunun başqa yolu **sudo apt-get install python** yazmağla da olur.

```
$ sudo apt-get install python
[sudo] password for backbox:
Reading package lists... Done
Building dependency tree
Reading state information... Done
python is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
backbox@backbox-pc:~$
```

Pythonun sabit ifadələri.Operatorlar və funksiyalar

Aşağıdakı ifadələri əzbərləməyə gərək yoxdur.İrəlidə bunlara dair misallar çəkiləcək,eləcədə ayrıca bölümlərdə bunlar haqqında geniş danışılacaq.

and	break	in	global
exec	continue	return	pass
for	class	not	elif
else	finally	yield	

```
import          lambda          def
raise           while           from
is              assert          or
try             except          del
```

bunu python vasitəsilə görmək istəyirsinizsə

```
   $python
Python 2.7.12rc1 (default, Jun 13 2016, 09:20:59)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print kwlist
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'kwlist' is not defined
>>> from keyword import*
>>> print kwlist
['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except',
'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or',
'pass', 'print', 'raise', 'return', 'try', 'while', 'with', 'yield']
>>>
 >>> len(kwlist)
31
>>>
```

deməli 31 standart xüsusi ifadə sayına malikdir.

Pythonu çalışdırdıqdan sonra >>> terminalda görülən bu işarə,bizdən komanda verməyimizi gözləyir.
Python versiyasını təyin etmək üçün

```
>>> import sys
>>> sys.version
'2.7.12rc1 (default, Jun 13 2016, 09:20:59) \n[GCC 5.4.0 20160609]'
>>>
```

import ifadəsində çox dayanmayın.Bu ifadə modulları çağırma operatorudur.Python versiasını təyin etmək üçün istifadə etdik.Modullar bəhsində bundan ətraflı danışacağıq.Ekran görüntüsündən məlum olduki biz hal-hazırda python 2.7.12 versiası üzərindəyik.
Qeyd edimki cmd ,terminalda normal qaydada istifadə etdiyiniz Linux və Windows komandalarını python programına yazmayın,hər halda bu başarılı olmayacaq,pythonun özünə məxsus sabit ifadələri,modulları,operatorları və

funksiyaları var.
Pythonun hansı kitabxanalarının yüklü olduğunu öyrənmək istəyirsinizsə terminalı açın və ardından

$pydoc -g

və qarşınıza

çıxacaq.

Open browser butonuna basın,qarşınıza pythonla bağlı bir çox modullar və paketlər listələnəcək.

ilk başlayacağımız print operatorudur.print operatoru artıq python3-də funksiya olduğundan gəlin bizdə funksiya olaraq adlandıraq.

print funksiyası

```
>>> print "sanaye"
```

```
sanaye
>>>
```
Gördüyünüz kimi sanaye sözünü ekrana yazdıra bildik.Dırnaq işarələri isə tək-dırnaq,cüt dırnaq və üçəm-dırnaq işarələrindən istifadə olunur.Gəlin bunları ayrı-ayrı test edək.
Məsələn(tək dırnaq)

```
>>> print 'məlahət'
məlahət
>>>
```
Gördüyümüz kimi tək dırnaq, ekrana yazdığımız sözü yazacaq.

Məsələn(cüt dırnaq)

```
>>> print "Bill Gates"
Bill Gates
>>>
```

Bunda da bir fərq görmədik.

Məsələn(üçəm-dırnaq)

```
>>> print """
Pythonu öyrəndiyiniz müddətdə copy paste qısayollardan istifadə etməyə çalışmayın.Bu sizin zərərinizədir."""

Pythonu öyrəndiyiniz müddətdə copy paste qısayollardan istifadə etməyə çalışmayın.Bu sizin zərərinizədir.
>>>
```

Yenə də bir fərq görmədik
Bu dırnaq işarələrin arasındakı fərqlərə nəzər salaq

```
>>> print 'Linux'un möhtəşəmliyi'
SyntaxError: invalid syntax
>>>
```
Yuxarıda in şəkilşi tipini apastrofla linux sözünə birləşdirməyə çalışsaqda bu baş vermədi.Çünki string cinsini tək dırnaqla başladıq.
Gəlin cüt dırnaq içində apastrofdan istifadə edək.

```
>>> print "linux'un möhtəşəmliyi"
linux'un möhtəşəmliyi
>>>
```

Gördüyünüz kimi bu baş tutdu.
Üçəm-dırnaq işarələrindən banner kimi də istifadə edə bilərsiniz.

```
>>> print"""
################
#              #
#              #
#    Xoş gəlmisiniz!   #
#              #
#              #
#              #
################
"""



################
#              #
#              #
#    Xoş gəlmisiniz!   #
#              #
#              #
#              #
################


>>>

>>> a="texnologiya"
>>> print a
texnologiya
>>>
```

və ya

```
>>> a="texnologiya" ;print a
texnologiya
>>>
```

Əgər yuxarıda print funksiyasından əvvəl nöqtəli-vergül yazmassanız xəta ilə qarşılaşacaqsınız.

```
>>> a="texnologiya" print a
SyntaxError: invalid syntax
>>>
```

xətamız print funksiyasındadır.

Yuxarıdakı a-hərfinə texnologiya ifadəsi atdıq.Daha sonra bu sözü(texnologiya)monitorda göstərmək məqsədilə print funksiyası vasitəsilə çap

etdirdik və nəticədə uğurla əmri başa vurduq.

Hesablamalar
(matematic operators)

Kompyuterinizdə çox gümanki hesablayıcılar var.Bunlar adi riyazi qaydalarla verdiyiniz bütün əmrləri səlis yerinə yetirir.Fərqli nəticə ala bilmərsiniz,çünki riyazi qanunlar bütün dünyada hal-hazırda sabitdir,pi ədədi toplamalar,çixma,vurma,bölmə logarifma,bucaq dərəcələri və s. kimi verdiyiniz əmrləri yerinə yetirir.
da da bunlar dəyişməzdir və verdiyiniz əmrləri yerinə yetirir.Əgər kompyuterinizdə hesablayıcı program yoxsa python yüklüdürsə pythondan istifadə edə bilərsiniz.
Terminaldan python(fərqi yoxdur python2 yada python3 olsun.)yazaraq >>> bu işarədən sonra sadəcə istədiyimiz rəqəmləri, əməli qeyd etməklə pythondan cavabı ala bilərik.
Misal

```
>>> 12+3
15
>>> 4+6
10
>>>
```

Gördüyünüz kimi hesablama tamamilə doğrudur.

Riyazi hesablamalar zamanı riyazi ifadələrimiz,orta məktəbdə tədris olunan qaydada dəyişməz olaraq qalır.

Toplama	+	(müsbət)
Çıxma	-	(mənfi)
Vurma	*	(vurma)
Bölmə	/	(bölmə)

```
>>> 6**2
36
>>> 4**2
16
>>>
```

Yuxarıda gördüyünüz iki-ulduz işarəsi isə qüvvəti təmsil edir.Bütün riyazi işarələri yazmadımki,irəlidə misallarda rastlaşacağıq.
Qüvvətdən başqa faiz işarəsi(%) hesablama maşınlarından fərqli olaraq pythonda başqa funksiyanı icra edir.

```
>>> 14%2
0
>>> 15%2
1
>>> 17%3
2
>>>
```

Yuxarıda ilk yazdığımız 14%2-i bu o deməkdirki 14 ikiyə bölündükdə qalan qalıq 0-dır.Eləcədə 15%2 nəticəsindən əldə etdiyimiz 1-ədədi də həmçinin.

Eləcədə pythonda istədiyiniz bir ifadəni başqa bir ifadədə qeydləyə bilərsiniz.Bunu daha açıq şəkildə məsələn sizin adınız olsun Fuad.Fuad daxilində bir alt ifadələr,tanımlamalar qeyd edə bilərik.
Fuad=soyad,təvəllüd
Pythonda da bu belədir sadəcə olaraq qeyd etdiyimiz bu alt ifadələr cinslərinə görə fərqli şəkildə qeyd olunacaqlar.
Məsələn

```
>>> a=fuad

Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    a=fuad
NameError: name 'fuad' is not defined
```

Bəli bir səhv var.Dediyim kimi a-ya fuad adlı ifadə atdıq amma bunu dırnaq içində göstərmədik və bu səhv anlaşıldı.

```
>>> a="fuad"
>>>
```

Və alt sətrə səhvsiz keçid etdik.

```
>>> print a
fuad
>>>
```

```
>>>10/6
1
>>>
```

Yuxarıda qeyd olunan hesablamadan aldığımız nəticəyə tərəddüdlə yanaşacaqsız.

Python 3-də bunu hesablasanız cavabı doğru alacaqsınız.Amma python 2-ni istifadə etdiyimiz üçün bizə bu cavabı verəcək.
Bu problemin həlli,bölmə əməliyyatı apararkən ədədlərin yanına .0 artırmağınız yetərlidir.

```
>>>10.0/6
1.6666666666666667
>>>
```

Əgər hər dəfə 0-ədədini artırmağı istəmirsinizsə hesablama apararkən ilk başda

```
 >>> from __future __ import division
```

yazmağınız yetərlidir.Qeyd edimki future ifadəsinin önündə və sonra gələn alt-tire işarəsini iki dəfə yazın.
Hesablamalara dair bir neçə misallar yazın.
Rəqəmi bir ifadəyə qeyd edib print lə manitora çap edin.

```
>>>n=5
>>>print n
5
>>>n=5
>>>a=4
>>>a*n
20
>>>print a
4
>>>n=2
>>>a=2
>>>cavab=a*n+n
>>>print cavab
6
```

input() **və** raw_input() funksiyası

raw_input() funksiyası istifadəçilərlə əlaqə yaradır,yəni istifadəçinin şərti olaraq qeyd etdiyi rəqəm və ifadə deyil,yazdığı programda istifadə edə biləcəyi sonsuz dəyərləri qeyd etmə imkanı yaradır.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
raw_input("Zəhmət olmasa adınızı yazın-->");
print "Minnətdaram"
```

```
>>> raw_input('Zəhmət olmasa adınızı yazın-->')
Zəhmət olmasa adınızı yazın-->rashad
'rashad'
>>> print 'Minnətdaram'
Minnətdaram
>>>
```

raw_input() funksiyası söz birləşmələri ilə yanaşı ədədləri də qəbul edir.

```
>>> raw_input('write you age:');
write you age:45
'45'
>>>
```

Amma hər dəfə ədədlərdən bu funksiyada istifadə edə bilmirik.Məsələn bir hesablama programı zamanı raw_input() vasitəsilə istifadəçidən ədədi alıb toplamaya məcbur etsəniz bu xəta verəcək.Bunu aşağıdakı misalda daha aydın başa düşəcəksiniz.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
a=raw_input('bir ədəd girin:');
b=raw_input('bir ədəd  girin:');cavab=a+b;
print cavab

bir ədəd girin:23
bir ədəd  girin:17
2317
>>>
```

Yuxarıda gördüyümüz kimi nəticəmiz 40 olmaq əvəzinə daxil etdiyimiz 17 və 23 ədədini birləşdirdi.

Bu problemlərdən qaçmaq üçün python bizə cins tiplərinin dəyişdirilməsini təklif edir.

Cinslərin dəyişdirilməsi

Pythonda əsasən

int()
str()
float()
complex()
bool()
cinsləri mövcuddur.

int() cinsi **integer** sözünün qısaltmasıdır,ədədləri ifadə edir.

```
>>> a=23
>>> type(a)
<type 'int'>
>>> b=2
>>> type(b)
<type 'int'>
>>>
```

a-ya 23 ədədini verdik,daha sonra tipini soruşduq.Və bizə integer cinsindən olduğunu ifadə etdi.

str() cinsi **string** sözünün qısaltmasıdır,söz və ya söz birləşmələrin ifadə edir.

```
>>> a='Sameddin'
>>> type(a)
<type 'str'>
>>> type('Elena')
<type 'str'>
>>>
```

float() cinsi kəsirli ədədlərdə eləcədə tam ədədin bölmə zamanı ala biləcəyimiz dəqiq nəticəni ifadə etməkdə yardımçımız olacaq.

```
>>> type(5.0)
<type 'float'>
>>> a=23
```

```
>>> b=7
>>> a/b
3
>>> float(a/b)
3.0
>>> a=1
>>> a=34
>>> b=17
>>> float(a,b)
>>> float(a)
34.0
>>> float(b)
17.0
>>> a/b
2
>>> a=33
>>> b=17
>>> float(a)
33.0
>>> float(b)
17.0
>>> a/b
1
>>> a=33.0
>>> b=17.0
>>> a/b
1.9411764705882353
```

complex() cinsi qarışıq ədədlər sisteminə daxildir.

```
   >>> a=33.0
>>> complex(a)
(33+0j)
>>> complex(23)
(23+0j)
>>>
```

Biraz öncə raw_input() funksiyasından istifadə edərək program yazdıq amma hesablama zamanı problemlə qarşılaşdıq,əslində problem deyildə bizim istədiyimiz nəticəni vermədi.cins dəyişmələri öyrəndiyimiz üçün artıq bu hesablamanı dəqiqliklə aparaq
Yazdığımız program

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
a=raw_input('bir ədəd girin:');
b=raw_input('bir ədəd  girin:');cavab=a+b;
print cavab

bir ədəd girin:23
bir ədəd  girin:17
2317
>>>
```

raw_input() əvvəlinə int əlavə edərək integer cinsinə çevirək

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
a=int(raw_input('bir ədəd girin:'));
b=int(raw_input('bir ədəd  girin:'));cavab=a+b;
print cavab
```

Alacağımız nəticə

```
bir ədəd girin:23
bir ədəd  girin:17
40
>>>
```

tamamilə doğrudur.cavab=40

input() funksiyası

input() funksiyası da raw_input() funksiyasına bənzərdir.Aralarındakı fərq isə hesablama apararkən daha rahat istifadə olunur.Yuxarıda cins dəyişmələri input() funksiyasında ehtiyac olmur.Buna dair bir neçə misal çəkək.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
a=input('bir ədəd girin:');
b=input('bir ədəd  girin:');cavab=a+b;
print cavab
bir ədəd girin:23
bir ədəd  girin:17
40
>>>
```

Yazdığımız kodlara biraz aydınlıq gətirək.İlk öncə raw_input() funksiyasında ədədi integer cinsinə çevirib,daha sonra əməliyyatı yerinə yetirirdik.input() funksiyasında isə gördüyümüz kimi buna ehtiyyac yoxdur.input() funksiyası misaldan aydın olduğu kimi ədədlərlə işləməyində möhtəşəm funksiyadır.

if else elif **operatorları**

Bu üçlük demək olarki bir biriylə hər zaman istifadə olunur.üçü istifadə olunmasada if və else funksiyaları daha çox istifadə olunmaqdadır.Hər gün internetdə rastlaşdığımız bizə verilən suallar-dəyişikliyi yadda saxlamağa əminsinizmi və ya programdan çıxmaq istəyirsiniz kimi suallar əsasən bu üçlük üzərində yazılır.Və ya GUİ programlamada hazır xəta mesajları bu funksiyaları əvəz edir.
if() ingiliscədən ‘əgər’ kimi tərcümə olunur
Belə bir misal yazaq.
İstifadəçi parolu daxil etsin,əgər parol daha öncəki parolla uyğun gəlirsə ekrana xoş gəldiniz ifadəsini versin.Buna dair bir misal kompyuterinizi açanda(əgər parol tətbiq etmisinizsə) sizdən giriş üçün parol istəməsi.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
passwd='aliyevabdurrahman';
a=raw_input('please write your passwd:');
if a==passwd:
    print 'xoş gəldiniz!';
```

```
please write your passwd:aliyevabdurrahman
xoş gəldiniz!
>>>
```

Gördüyümüz kimi parolu öncədən qeyd etdik daha sonra istifadəçidən girməyini tələb etdik,sonra əgər a passwd bərabərdirsə(eynidirsə) ekrana xoş gəldin yaz Düşünürəm yazdığımız kodlarda heç bir yenilik yoxdur və hər biri sizə agahdır. Qeyd edimki if operatorunu istifadə etdikdə iki-nöqtədən sonra bir tab-düyməsi aşağı düşəcək kursor.əgər siz python üçün grafik idledən istifadə etsəniz idle özü avtomatik aşağı sətrə keçəcək enter düyməsindən sonra təbiki.
Yox əgər bir txt mətni açıb sonunu py qeyd etməsəniz bu prosesi manual yəni əllə

edəcəksiniz.Bir txt mətni açın və bu kodları əllə yazın ən son print funksiyası ilk sətirdən başlayacaq,dərhal print funksiyasının əvvəlinə kursoru atın və dörd dəfə space düyməsinə basın və ya birbaşa tab düyməsini.
if operatoru üstündə çox qalmayaraq növbəti else funksiyasına keçmək istəyirəm.çünki pythonu öyrəndiyimiz ərəfədə bu üçlü operatorlardan daima istifadə edəcəm.

else() operatoru ingiliscədən **‘və ya’** kimi tərcümə olunur
Yuxarıda yazdığımız kod nəzəriyyəsinə belə bir ifadə əlavə edərək

İstifadəçi parolu daxil etsin,əgər parol daha öncəki parolla uyğun gəlirsə ekrana xoş gəldiniz ifadəsini versin.Əgər parol yanlışsa ekrana ‘səhv parol’ yazsın.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
passwd='aliyevabdurrahman';
a=raw_input('please write your passwd:');
if a==passwd:
    print 'xoş gəldiniz!';
else:
    print 'səhv parol';
```

```
please write your passwd:fghtyydgdfg
səhv parol
>>>
```

Göründüyü kimi biz parolu aliyevabdurrahman olaraq tanıdmışdıq amma istifadəçi tamamən başqa bir parol daxil etdiyindən **else()** operatoru özünü göstərdi.

Üçlü operatorlara (if,else,elif) dair misallar

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
eded=raw_input('bir ədəd daxil edin:');
if eded<=0:
```

```
    print '0 və ya 0-dan kiçik ədəd daxil etməyin!'

else:
    print int(eded)**2


bir ədəd daxil edin:2
4
>>>
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
parol=str(raw_input('write you passwd:'))
if len(parol)<8:
    print 'wrong passwd lenth';
else:
    print  'welcome'

write you passwd:star
wrong passwd lenth
>>>
#!/usr/bin/env python
# -*- coding: utf-8 -*-
parol=str(raw_input('write you passwd:'))
if len(parol)<8:
    print 'wrong passwd lenth';
else:
    print  'welcome'

write you passwd:aquamarine
welcome
>>>
```

if else elif operatorlarını bildiyimiz üçün bir hesablama programı yaza bilərik.

Program yazarkən ilk öncə onun nəzəriyyəsini bir qaralamaya yazın.Daha açıq desək ideyanızı bir vərəqə köçürün.

Hesablayıcı program üçün bizə riyazi əməllər lazım olacaq bu əməllər pythonda da dəyişməz olaraq

Riyazi ifadə	Python	Misal
+ Addition(toplama)	+	20+10=30
- Subtraction(çıxma)	-	20-10=10
* Multiplication(vurma)	*	20*10=200
/ Division(bölmə)	/	20/10=2
% Modulus(qalıq)	%	20%10=0
** Exponent(qüvvət)	**	20**2=400
// (qalıqsız göstəriş)	//	20//11=1

Və programımızın kodunu yazmağa başlayaq.İlk program açılışında istifadəçini salamlayaq.Daha sonra əməlləri ekranda göstərərək istifadəçidən əməllərdən birini seçməsiniz istəyək.Və python-a girişdə qeyd etdiyimiz kimi kodları yazmadan öncə əvvələ from __future__ import division yazmağımız daha məqsədə uyğundur.Çünki hesablama ərəfəsində python nə etdiyini daha dəqiq anlayacaqdır.
Kodlar aşağıdakı kimidir

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import division
print ''Salam,hesablayıcı programa xoş gəlmisiniz!''

musbet='(1) toplama\n'
menfi='(2) cixma\n'
vurma='(3) vurma\n'
```

```
bolme='(4) bolme\n'
faiz='(5) faiz'

print musbet,menfi,vurma,bolme,faiz
sec=raw_input("Yuxarıdakı əməllərdən birini seçin və enter'ə basın:")
```

buraya qədər düşünürəm hər biri aydındır.Sadəcə \n -i başa düşməyə bilərsiniz,irəlidə bunun haqqında geniş yazacam.Və davam edək,

```
if sec=='1':
    print 'siz toplama əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a+a1
```

İstifadəçidən əməli seçməyini istədik.Və ilk əməliyyatımızı toplama üzərində başladıq
Programın ekran görüntüsü

```
Salam,hesablayıcı programa xoş gəlmisiniz!
(1) toplama
(2) cixma
(3) vurma
(4) bolme
(5) faiz
Yuxarıdakı əməllərdən birini seçin və enter'ə basın:1
siz toplama əməlini seçdiniz
ilk ədədi daxil edin:12
ikinci ədədi daxil edin:15
cavab= 27
```

Və toplama əməli üçün hesablayıcı hazırdır.
İkinci əməlimiz çıxma əməlidir.Eyni qayda ilə elif() operatorundan istifadə edərək kodları yazmağa davam edirik.
Bütünlüklə kodlarımız aşağıdakı şəkildə olacaq

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import division
print'''Salam,hesablayıcı programa xoş gəlmisiniz!'''#bu sadəcə fikir bildirmək üçündür

musbet='(1) toplama\n'
menfi='(2) cixma\n'
```

```
vurma='(3) vurma\n'
bolme='(4) bolme\n'
faiz='(5) faiz'

print musbet,menfi,vurma,bolme,faiz
sec=raw_input("Yuxarıdakı əməllərdən birini seçin və enter'ə basın:")

if sec=='1':
    print 'siz toplama əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a+a1

elif sec=='2':
    print 'siz çıxma əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a-a1
```

Və ekran görüntüsü

```
Salam,hesablayıcı programa xoş gəlmisiniz!
(1) toplama
(2) cixma
(3) vurma
(4) bolme
(5) faiz
Yuxarıdakı əməllərdən birini seçin və enter'ə basın:2
siz çıxma əməlini seçdiniz
ilk ədədi daxil edin:12
ikinci ədədi daxil edin:4
cavab= 8
>>>
```

Və keçək üçüncü əməlimizə,vurma əməliyyatı

Kodlar aşağıdakı kimidir

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import division
print'''Salam,hesablayıcı programa xoş gəlmisiniz!'''#bu sadəcə fikir bildirmək
üçündür
```

```
musbet='(1) toplama\n'
menfi='(2) cixma\n'
vurma='(3) vurma\n'
bolme='(4) bolme\n'
faiz='(5) faiz'

print musbet,menfi,vurma,bolme,faiz
sec=raw_input("Yuxarıdakı əməllərdən birini seçin və enter'ə basın:")

if sec=='1':
    print 'siz toplama əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a+a1

elif sec=='2':
    print 'siz çıxma əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a-a1

elif sec=='3':
    print 'siz vurma əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a*a1
```

Biz başqa fərqli bir şey əlavə etmədik.eyni qayda ilə elif() operatorundan istifadə edərək istifadəçidən daha sonra rəqəmləri istədik və nəticəni python rahatlıqla hesablayaraq ekrana çap etdi.

Növbəti kod blokumuz

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import division
print'''Salam,hesablayıcı programa xoş gəlmisiniz!'''#bu sadəcə fikir bildirmək üçündür

musbet='(1) toplama\n'
```

```
menfi='(2) cixma\n'
vurma='(3) vurma\n'
bolme='(4) bolme\n'
faiz='(5) faiz'

print musbet,menfi,vurma,bolme,faiz
sec=raw_input("Yuxarıdakı əməllərdən birini seçin və enter'ə basın:")

if sec=='1':
    print 'siz toplama əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a+a1

elif sec=='2':
    print 'siz çıxma əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a-a1

elif sec=='3':
    print 'siz vurma əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a*a1
```

və sonuncu faiz ifadəsinə biraz diqqət yetirin.Bildiyimiz kimi bir ədədin faizini tapmaq üçün ədədi faizinin ədədinə vurub,100-ə bölürdük.Bunu düsturla ifadə etsək,

20-nin 4%-i
a=20
b=4
a*b/100

Və bu qayda ilə faizə dair kodlarımızı yazaq

```
elif sec=='5':
    print 'siz faiz əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a*a1/100
```

Yuxarıda yazdığımız kodu bütünlüklə programımızda qeyd edərək

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import division
print'''Salam,hesablayıcı programa xoş gəlmisiniz!'''#bu sadəcə fikir bildirmək
üçündür

musbet='(1) toplama\n'
menfi='(2) cixma\n'
vurma='(3) vurma\n'
bolme='(4) bolme\n'
faiz='(5) faiz'

print musbet,menfi,vurma,bolme,faiz
sec=raw_input("Yuxarıdakı əməllərdən birini seçin və enter'ə basın:")

if sec=='1':
    print 'siz toplama əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a+a1

elif sec=='2':
    print 'siz çıxma əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a-a1

elif sec=='3':
    print 'siz vurma əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a*a1

elif sec=='4':
    print 'siz bölmə əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a/a1

elif sec=='5':
    print 'siz faiz əməlini seçdiniz'
```

```
a=input('ilk ədədi daxil edin:')
a1=input('ikinci ədədi daxil edin:')
print 'cavab=',a*a1/100
```

Ekran görüntüsü

```
Salam,hesablayıcı programa xoş gəlmisiniz!
(1) toplama
(2) cixma
(3) vurma
(4) bolme
(5) faiz
Yuxarıdakı əməllərdən birini seçin və enter'ə basın:5
siz faiz əməlini seçdiniz
ilk ədədi daxil edin:12
ikinci ədədi daxil edin:3
cavab= 0.36
>>>
```

Və aldığımız nəticə doğrudur.
Bu sadə hesablama programını birazda uzadaraq qüvvət və s daxil edə bilərsiniz.Eləcədə üçdırnaq işarəsindən istifadə edərək programın əvvəlinə bəzək(banner) əlavə edə bilərsiniz.

Aşağıda göstərilən riyazi ifadələrin pythonda istifadə edə biləcəyiniz qarşılığı

Riyazi	Python
=	==
≠	!=
<	<
>	>
≤	<=
≥	>=

Operatorlar	Misallar

+=	**a +=c və ya a=a+c**
-=	**c -= a və ya c=c-a**
***=**	**c *= a və ya c = c * a**
/=	**c /= a və ya c = c / a**
%=	**c %= a və ya c = c % a**
****=**	**c **= a və ya c = c ** a**
//=	**c //= a və ya c = c // a**

Cədvəldə qeyd olunan operatorlara dair misallar

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
a=12
b=10
c=0
c=a+b
print 'c is ',c;
c=a-b;
print 'c is ',c;
c=a*b;
print 'c is ',c;
c=a/b;
print 'c is ',c;
c=a%b;
print 'c is ',c;
a=2;
b=3;
c=a**b;
print 'c is ',c;
a=10;
b=5;
c=a//b;
print 'c is ',c;
```

ekran görüntüsü

```
c is  22
c is  2
c is  120
c is  1
c is  2
c is  8
c is  2
>>>

#!/usr/bin/env python
# -*- coding: utf-8 -*-
a=21
b=10
c=0
if(a==b):
    print'a is equal to b'
else:
    print 'a is not equal to b'

if(a!=b):
    print'a is not equal to b'
else:
    print'a is equal to b'
if(a<>b):
    print'a is not equal to b'
else:
    print'a is equal to b',True
if(a<b):
    print'a is less than b'
else:
    print 'a is not less than b'
if ( a > b ):
    print ' a is greater than b'
else:
    print ' a is not greater than b'
a = 5;
b = 20;
if ( a <= b ):
    print 'a is either less than or equal to b'
else:
    print 'a is neither less than nor equal to b'
```

```
if ( b >= a ):
    print 'b is either greater than or equal to b'
else:
    print 'b is neither greater than nor equal to b'
```

ekran görüntüsü

```
a is not equal to b
a is not equal to b
a is not equal to b
a is not less than b
 a is greater than b
a is either less than or equal to b
b is either greater than or equal to b
>>>
```

Pythonda sətrdən-sətrə keçid işarələri.Xüsusi işarələr

\, \n, \t, \a, \\

Bundan öncəki bəhslərin birində kodlarımız arasında \n işarəsindən istifadə etdik.Bu işarədən başqa bir neçə sabit işarələr də var.Bunlar xüsusi işarələrdir və kodlar arasında string və ya integer,eləcədə ifadələr kimi rol oynamır,istifadə olunacaq xüsusi yerləri və qaydaları var.Python-a həmən yeni başladığımızda apastroflu sözlərin necə dırnaq içində yazılmasından qısaca danışmışdıq.Bu xüsusi işarələrdə o kimi xarakter daşıyır.
Bir misala baxaq.leafpad (siz istərsəniz kwrite və gedit açın) açırıq və əvvəlinə python path,daha sonra ascii kodlamamısı əlavə edirik.Aşağıdakı kimi

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
```
Daha sonra bir ad verərək(mən syut.py olaraq qeyd etdim) py sonluqlu sistemimizdə yadda saxlayırıq.Sonra bir sadə program yazaq.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
a=raw_input('write you name:')
b=raw_input('write you surname:')
print 'you name->',a,'you surname→',b
```

```
write you name:Mahmud
write you surname:Alimamedov
you name-> Mahmud you surname-> Alimamedov
>>>
```

Yuxarıda istifadəçidən adını,soyadını soruşduq və ekrana çap etmək üçün print funksiyasından istifadə etdik.programın görünüşü çox əcaib olduğu ortadadır.Məsələn biz bunu you name : Mahmud
you surname : Alimamedov
kimi göstərə bilərik.Bunun üçün python bizə xüsusi işarələrdən istifadə etməyi təklif edir.İlk öyrənəcəyimiz işarəmiz \n işarəsidir.Bu işarə sətrin əvvəlində və sonunda fərqli nəticələr verir.Bu nəticələri əyani görmək üçün gəlin yuxarıdakı kodlara daxil edək.

İlk sətrin əvvəlində istifadə edərək

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
a=raw_input('write you name:')
b=raw_input('write you surname:')
print '\nyou name ',a,'\nyou surname ',b
```

```
write you name:Eldar
write you surname:Faracov


you name Eldar
you surname Faracov
>>>
```

Göründüyü kimi hər iki sətrin əvvəlinə atdıq və programın görünüşü əla təəssürat bağışladı.Sətrin əvvəlində bu işarə-yəni işarədən sonra gələn ifadə bir alt sətrə keçəcək.Əgər sətrin sonunda qeyd etsəniz,yəni işarədən sonra gələn istənilən bir ifadə bir alt sətrə keçəcək.əslində biz bu işarəni ilk ifadənin sonunda qeyd etsəydik yetərli olacaqdır.Yəni bir dəfə istifadə edə bilərik.Qeyd edimki \-slash işarəsinin hansı yönə yıxıldığına diqqət edin.səhv yazsanız dırnaq içindəki ifadəylə bərabər string kimi çap olunacaq.Bu xüsusi işarələrin çoxu dırnaq içində istifadə olunur.Dırnaqdan kənarda heç bir xüsusi ,funksional bacarığa malik deyil.

Tək slash işarəsi. \
Slash işarəsini n-hərfi ilə(\n) istifadə edirdiksə tək slash işarəsi də python da xüsusi bir işarədir.Bu işarə ifadənin istənilən yerində istifadə oluna bilinir.print funksiyasından sonra dırnaq içində yazdığımız ifadələər uzun uzadı ola bilər.Bunların görünüşlü,rahat olmasını təmin etmək məqsədilə \ işarəsindən istifadə edilir.Gəlin misallara baxaraq nə kimi rol oynadığını təyin edək.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
print "Azərbaycan Dəniz Akademiyası nəznində bir neçə fakültələr olmağla fəaliyyət göstərməyə davam edir"



Azərbaycan Dəniz Akademiyası nəznində bir neçə fakültələr olmağla fəaliyyət göstərməyə davam edir
>>>
```

Yuxarıda gördüyünüz kimi uzun bir ifadə yazdıq,bu ifadə hətda davam edə də bilərdi.Gəlin \ işarəsindən istifadə edərək bunu görünüşlü hala gətirək.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
print  "Azərbaycan Dəniz Akademiyası nəznində\
bir neçə fakültələr olmağla fəaliyyət göstərməyə davam edir"
```

"Azərbaycan Dəniz Akademiyası nəznində\
bir neçə fakültələr olmağla fəaliyyət göstərməyə davam edir" --- cümləsinin içində istifadə etdiyimiz \ növbəti sətrə fikirlərimizi yazmağa yardımçı oldu və heç bir xəta vermədən ekrana çap etdi.Bu slash işarəsi şeir yazarkən istifadə edə biləcəyiniz ən ideal xüsusi işarələrdən biridir.Düşünürəm \-işarəsinin nə rol oynadığını anlamışıq.

\t xüsusi işarəsi.
\t xüsusi işarəsi tab düyməsinin ifadə formasıdır.Yəni bir cümləni və ya ifadəni bir tab irəli atma imkanına malikdir.bir tab irəli atma prosesi istənilən bir complier,idle grafik programın 0-sütunundan hesablayaraq yerinə yetirilir.Misala baxaq

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
print '\tFridrix Nitshe'
```

ekran görüntüsü

```
        Fridrix Nitshe
>>>
```

Göründüyü kimi ilk sütundan bir tab irəli çap etdi.

\a xüsusi işarəsi.
Bu işarə öz növbəsində siqnal səsi verir.Misala baxaq.

```
>>> print('\a'*5)

>>>
```

Linuksda bu siqnal səsi çıxmaya bilər,amma Windows əməliyyat sistemində bu işarə çalışır.Bu işarəyə misal bir programı bağladığınız zaman səhv proseslərin gedişində bip siqnalını eşitməniz.

Pythonda xüsusi işarələrə burada son qoyaq.Çünki biz python2 -ni öyrənirik.Pythonda xüsusi işarələr bunlarla bitmir,Python3-ə daxil olan bir sıra xüsusi işarələr vardır \r. \v ,\N, \u, \x ,\U , r və sairə.Python3 də olan xüsusi işarələr python2-də çalışmır.amma python2-də olan bütün xüsusi işarələr python3-də çalışır.Məntiqi yöndən bu belə olmalıdır.

Gnu\linux da terminaldan programı açmaq.yazdığımız kodları bir gedit leafpad və ya kwrite -yə yazıb yadda saxlayırıq.(mən syut.py olaraq qeyd etdim)
Daha sonra terminalı açırıq və ardından

```
$pwd
/home/panda
```

```
$ls
Desktop    PycharmProjects  book    eric    photo       py_docs  syut.pyc
Downloads  Templates        django  music   programlar  syut.py  videos
```

və ardından

```
$sudo chmod +x syut.py
$
```

Alt sətrə səhvsiz keçdisə deməli syut.py programımıza write-read komandalarını aktive etdik.
Ardınca

```
$python syut.py
Viktor    Armani
$
```

gördüyümüz kimi programımız terminaldan rahatlıqla açıldı.

Qeyd edimki python program_adı.py verdikdə bu yalnız python2-də açılacaq.

len() **funksiyası.**uzunluq anlayışını ifadə edir.

Əsasən string cinslərlə istifadə olunur.Bunun integer qarşılığı yoxdur.Integer cins sayısını öyrənmək üçün cins dəyişmələrindən istifadə olunur.
Məsələn

```
>>> len('alimamedov')
10
>>> len(23)

Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    len(23)
TypeError: object of type 'int' has no len()
```

Yuxarıda len() vasitəsilə 23 ədədin uzunluğunu soruşduq amma xəta verdi.

```
>>> a=1039539259
>>> i=str(a)
>>> len(i)
10
>>>
```

gördüyümüz kimi cins dəyişdirmələrindən istifadə edərək a-nı string -ə çevirdik və len() ilə uzunluğunu soruşduq.

```
>>> len('dict')
4
>>> len('a')
1
>>> a='bakielektriksebekesi'
>>> len(a)
20
>>>
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

a=raw_input('write you name:')
if len(a)<8:
    print 'adiniz cox qisadir'
else:
    print 'davam et'


write you name:ali
```

```
adiniz cox qisadir
>>>
```

və ya

```
write you name:felakettin
davam et
>>>
```

Bu funksiyaya bənzər hal sosial şəbəkələrdə az sayda parol daxil etdikdə bizə verilən xəta mesajları.parol ən az 8-dən az olmamalıdır kimi

while **operatoru**.
Ingiliscədən ‘nə üçün,niyə’ kimi tərcümə olunur.
Bunu sirkulyasiya kimi də başa düşmək olar.kodları təkrar başa döndərməyə imkan yaradır.Bu funksiyanı misallarla daha aydın izah edək.

Məsələn biz 100-yə qədər olan rəqəmləri ekrana çap etmək istəyirik.Buraya qədər öyrəndiyimiz metodlardan bilirikki print vasitəsilə hər bir rəqəmi bir hərflərə qeyd edərək çap edə bilərik.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
a=1
b=2
c=3
d=4
print a,b,c,d

1 2 3 4
>>>
```

Yuxarıda yazdığımız kodların nə qədər əhəmiyyətsiz eləcədə çətin yol olduğuna əminik.Bu vəziyyətdə while() köməyimizə çatacaq

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
a=0
while a<100:
    a+=1
```

```
    print a


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
.....
98
99
100
>>>
```

Yuxarıda yazdığımız kodlara açıqlama verək.
A-ya 0 verdik,daha sonra ühile funksiyası vasitəsilə a-ın 100-dən kiçik olduğu müddətdə 100 daxil olmaqla qədər bütün rəqəmlərin üzərinə 1 əlavə və ekrana çap et.ühile funksiyası kodlarda yerinə yetirdiyi son əməliyyatdan sonra başa dönərək prosesi təkrar edir.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

a=raw_input('write you name:')
if len(a)<8:
    print 'adiniz cox qisadir'
else:
    print 'davam et'
```

Yuxarıda yazdığımız kodları çalışdırdıqda yalnız bir dəfə sual verim cavab aldıqdan sonra program bağlanacaq,bunun üçün əvvəlinə while True: əlavə edərək

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
while True:
    a=raw_input('write you name:')
    if len(a)<8:
        print 'adiniz cox qisadir'
    else:
        print 'davam et'
```

```
write you name:lengo
adiniz cox qisadir
write you name:rahmanasadov
davam et
write you name:
```

Gördüyümüz kimi program hələ də davam etməkdədir o vaxta qədərki biz məcburi olaraq ctrl+c komandası verməyək

hesablayıcı program yazdığımızı yenidən xatırlayaq.Biz əməli secib prosesi yerinə yetirdikdən sonra program bağlanırdı və əgər əvvəlinə while True: əlavə etsək programımız tamamlanmış halda olacaq

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import division
print'''Salam,hesablayıcı programa xoş gəlmisiniz!'''#bu sadəcə fikir bildirmək üçündür

musbet='(1) toplama\n'
menfi='(2) cixma\n'
vurma='(3) vurma\n'
bolme='(4) bolme\n'
faiz='(5) faiz'
while True:
    print musbet,menfi,vurma,bolme,faiz
    sec=raw_input("Yuxarıdakı əməllərdən birini seçin və enter'ə basın:")
```

```
if sec=='1':
    print 'siz toplama əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a+a1

elif sec=='2':
    print 'siz çıxma əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a-a1

elif sec=='3':
    print 'siz vurma əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a*a1
elif sec=='4':
    print 'siz bölmə əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a/a1




elif sec=='5':
    print 'siz faiz əməlini seçdiniz'
    a=input('ilk ədədi daxil edin:')
    a1=input('ikinci ədədi daxil edin:')
    print 'cavab=',a*a1/100
else:
     print 'səhv'
```

```
Salam,hesablayıcı programa xoş gəlmisiniz!
(1) toplama
(2) cixma
(3) vurma
(4) bolme
(5) faiz
Yuxarıdakı əməllərdən birini seçin və enter'ə basın:6
```

```
səhv
(1) toplama
(2) cixma
(3) vurma
(4) bolme
(5) faiz
Yuxarıdakı əməllərdən birini seçin və enter'ə basın:4
siz bölmə əməlini seçdiniz
ilk ədədi daxil edin:23
ikinci ədədi daxil edin:45
cavab= 0.511111111111
(1) toplama
(2) cixma
(3) vurma
(4) bolme
(5) faiz
Yuxarıdakı əməllərdən birini seçin və enter'ə basın:
```

Sonda else əlavə edərək,əgər istifadəçi 5-sayıdan başqa bir sayı girərsə ekrana səhv yazısını yazdırdıq.

for **operatoru**

İngiliscədən üçün kimi tərcümə olunur.while() ilə demək olarki eyni prosesləri yerinə yetirir.Amma fərqləridə varki aşağıdakı misallardan daha aydın olacaq

```
>>> a='aviabiletlerinsatisi'
>>> for i in a:
        print i


a
v
i
a
b
i
l
e
t
l
e
r
```

```
i
n
s
a
t
i
s
i
>>>
```

Göründüyü kimi y-oxu üzrə ekrana çap etdi.

```
>>> for i in range(1,100):
        print i


1
2
3
4
5
6
7
8
9
10
.....
96
97
98
99
>>>
```

Bu bəhslərin üstündə çox qalmağın vacib əhəmiyyəti yoxdur,cünki irəlidə modullarda GUİ programlamalarda istifadə edəcəyimiz alt-programların kod içəriyində bu tip funksiyalara böyük yer ayıracağıq və zamanla əliniz bunlara alışacaq.

range() **funksiyası**.
Aralıq kimi tərcümə olunur.Ala biləcəyi maksimum 3 argumentdir.

```
>>> for i in range(20):
        print i


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
>>>


>>> print range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>


>>> print range(1,10,2)
[1, 3, 5, 7, 9]
>>>
```

1 və 10 rəqəmi arasındakı ədədlərə,1-dən başlayaraq üzərinə 2 əlavə et

```
>>> print range(3,20,4)
[3, 7, 11, 15, 19]
>>>
```

3 və 20 ədədləri arasındakı rəqəmlər,3-daxil olmaqla üzərlərinə 4 -əlavə et

və ya iki sayda rəqəm də ala bilir.

```
>>> print range(20,50)
[20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49]
>>>
```

birbaşa print range() metodundan istifadə etdiyimiz üçün ekrana list halında çap etdi.Əgər y-oxu istiqametində çap etdirməyi planlasanız əminəmki bunu yerinə yetirə biləcəksiniz.Burada for() köməyinizə çatacaq.

```
>>> for ali in range(1,10,2):
        print ali


1
3
5
7
9
>>>
```

```
>>> for a in range(1,30):
        print a


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

```
19
20
21
22
23
24
25
26
27
28
29
>>>
```

break **operatoru.**
İngiliscədən ‘kəsmək,dayandırmaq,sona endirmək’ kimi tərcümə olunur.

Biz bir vkontakt hesabımıza daxil olmaq istəyirik,əgər parolu və emaili doğru girərsək,hesaba daxil ola biləcəyik,əgər ikisindən biri yalnışsa program bağlansın.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
while True:
    parol='almaniya1990'
    email='almaniya@gmail.com'
    a=raw_input('please write you passwd:')
    b=raw_input('please write you email_adress:')
    if a == parol and b==email:
        print 'welcome vkontakte account!'
    else:
        print 'wrong passwd or email!'
        break
```

```
please write you passwd:df
please write you email_adress:dfg@gmail.com
wrong passwd or email!
>>>

please write you passwd:almaniya1990
please write you email_adress:dfg@gmail.com
wrong passwd or email!
>>>
```

```
please write you passwd:almaniya1990
please write you email_adress:almaniya@gmail.com
welcome vkontakte account!
please write you passwd:
```

continue ‘davam etmək ’ kimi tərcümə olunur.
and ‘və’ kimi tərcümə olunur
or ‘və ya’ kimi tərcümə olunur
not ‘yox’ kimi tərcümə olunur
True ‘doğru’ mənasını ifadə edir
False ‘yalnış’ mənasını ifadə edir

continue operatoru
İngiliscədən ‘davam,davam etmək’ mənasını ifadə edir.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
while True:

    a=raw_input('please write you passwd:')
    if len(a)<8:
        print 'you passwd very small at least 8'
    else:
        print 'welcome sosial network!'
        continue
```

```
please write you passwd:akade
you passwd very small at least 8
please write you passwd:akademiya123
welcome sosial network!
please write you passwd:
```

```
a='Yes'
b='No'
while True:
    sistem=raw_input('sistemdən çixmaq [Yes]..\ndavam etmək üçün [No]
yazın  :')

    if b==sistem:

        print '[....Siz sistemi tərk etdiniz....]'
        break
    if a==sistem:
        print '[.......prosses davam edir,lütfən gözləyin...]'
        continue


sistemdən çixmaq [Yes]..
davam etmək üçün [No] yazın  :No
[....Siz sistemi tərk etdiniz....]
>>>

sistemdən çixmaq [Yes]..
davam etmək üçün [No] yazın  :Yes
[.......prosses davam edir,lütfən gözləyin...]
sistemdən çixmaq [Yes]..
davam etmək üçün [No] yazın  :

#!/usr/bin/env python
# -*- coding: utf-8 -*-
while True:
    s = raw_input('write something : ')
    if s == 'exit':
        break
    if len(s) < 3:
        print  'very small'
        continue
    print 'Okay!'


write something : exit
>>>
write something : 3
very small
write something : 6
very small
```

```
write something : 78
very small
write something :
```

```
in
```

İngiliscədən içində,da,də kimi tərcümə olunur.
Davamlı istifadə edə biləcəyiniz bu deyim tərcüməsi qədər də pythonda iş görür.

```
>>> a='Azərbaycan'
>>> 'a' in a
True
>>>
```

Yuxarıda a-ya Azərbaycan verdik,sonra kiçik a-nın(string) Azərbaycan sözünün içində olub-olmadığını soruşduq və nəticədə bizə True verdi,yəni doğru.

```
>>> 'p' in a
False
>>>
```

Burda isə p-nin Azərbaycan sözünün içində olub-olmadığını soruşduq və bizə False verdi,yəni səhv.Təbiki bu doğrudur.Azərbaycan sözündə p hərfi yoxdur.
Əgər hər hansısa bir sözü y-oxu üzrə alt-alta düzmək istəyirsinizsə in deyimindən istifadə edə bilərsiniz.

```
>>> a='Azadliq'
>>> for i in a:
        print i


A
z
a
d
l
i
q
>>>
```

```
#!/usr/bin/env python
```

```
# -*- coding: utf-8 -*-
sual=raw_input('sistemden cixmaq üçün? [Y/N]')
if 'Y' in sual or 'y' in sual:
    print 'you system is updating now'
else:
    print 'exiting....'

sistemden cixmaq üçün? [Y/N]y
you system is updating now
>>>

sistemden cixmaq üçün? [Y/N]n
exiting....
>>>
```

List

Bəzi sabit sözləri Azərbaycan dilinə tərcümə etməyin və vacib olduğunu düşünmürəm.Çünki list sözünü tərcümə etsək qat,lay mənasını ifadə edəcək.Bizə isə əksinə tərcüməsi deyil tamamilə ingiliscəsi lazımdır.Çünki python da başqa dillərdə olduğu kimi (c++,perl,ruby,php,html,java və s) ingilis sözlərinin üzərində labaratoriya şəraitində yazılmışdır.List nə mənanı ifadə etdiyinə gəlin baxaq

```
list=[]
>>> type(list)
<type 'list'>
>>>
```

list-in tipini soruşduq və list olduğunu python bizə söylədi

```
>>> list
[]
>>>
```

Burda isə list-in boş olduğu aşkar olur.Listlərin ala biləcəyi bir necə deyilişlər var. Yəni alt funksiyaları.
Bunun üçün

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__delslice__',
'__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
'__getslice__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__iter__', '__le__',
'__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',
'__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__setslice__',
'__sizeof__', '__str__', '__subclasshook__', 'append', 'count', 'extend', 'index',
'insert', 'pop', 'remove', 'reverse', 'sort']
>>>
```

dir(list) yazaraq listlərin ala biləcəyi metodları listələdik.Yuxarıdakı metodlardan sacədə bizə
'append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort'
metodları lazım olacaq.

Listlərə ifadələr,rəqəmlər(string və integer) əlavə edək.

Listlərə string və ya integer əlavə etmək üçün istifadə edəcəyimiz metodlardan biri olan **append** metodudur.

append() metodu

İngiliscədən ‘əlavə etmək’ kimi tərcümə olunur.Ala biləcəyi yalnız bir ifadədir.Bir neçə ifadə əlavə etmək üçün append() metodunu bir neçə dəfə yazmalı olacaqsınız.

```
>>> list.append(25,4444,80,587)

Traceback (most recent call last):
  File "<pyshell#62>", line 1, in <module>
    list.append(25,4444,80,587)
TypeError: append() takes exactly one argument (4 given)
>>>
```

Qırmızıya boyadığımız ifadə bizə deyirki append yalnız bir argument ala bilir(biz 4 argument yazmışıq)

```
>>> list=[]
>>>
```

enter-ə basdıqdan sonra aşağı xətasız keçdisə demək bir boş listimiz yaradıldı.Daha sonra append metodundan istifadə edərək ifadə və ya ədəd əlavə edək.

```
>>> list.append('mayk')
>>> list
['mayk']
>>>
```

mayk adlı söz deməli əlavə olundu.Qeyd edimki boş bir list yaratdıqda qapalı mötərizədən istifadə etdik amma metodları istifadə edəndə list-dən sonra nöqtə yazaraq metodumuzu daha sonra sadə mötərizə açıq dırnaq içində yalnız string cinsini əlavə edirik.Diqqətli olun metodlarda stringlər dırnaq içində yazılır integer cinsi isə birbaşa.Əgər hər hansısa bir integer-i dırnaq içinə alsanız metod bunu string cinsi kimi başa düşəcək.Misallarla buna baxaq

```
>>> list.append(23)
>>> list.append('elitar')
>>> list.append('23')
>>> list
['mayk', 23, 'elitar', '23']
>>>
```

Qeyd edimki append() metodu yalnız bir string və ya integer ala bilmə bacarığı var.Yuxarıda gördüyünüz kimi biz əlavələrimizi append metodunu bir neçə dəfə istifadə etmək məcburiyyətində olduq.Qayıdaq cinsləri listlər necə qəbul

etməsinə
İlk başda bir 23 rəqəmini daxil etdik daha sonra elitar sonra 23 rəqəmini bir string cinsi olaraq əlavə etdik və görüntüdə listin içində sadəcə bir rəqəm və üç sayda söz(string) var.Buna əmin olmaq istəyirsinizsə tiplərini sorğuya çəkə bilərsiniz.

```
>>> list
['mayk', 23, 'elitar', '23']
>>> type('mayk')
<type 'str'>
>>> type(23)
<type 'int'>
>>> type('elitar')
<type 'str'>
>>> type('23')
<type 'str'>
>>>
```

əlavə etdiyimiz son ifadələr ən sona əlavə olunur,Yəni alfabe sıralaması etmir.listlər daxlindəki ifadələri sayı sistemi üzrə 0-dan başlayaraq sayır.

0	1	2	3
‘mayk’	23	‘elitar’	‘23’

Listlərin içindəkiləri görmək üçün başqa bir üsul

```
>>> list[0]
'mayk'
>>> list[1]
23
>>> list[2]
'elitar'
>>> list[3]
'23'
>>> list[4]

Traceback (most recent call last):
  File "<pyshell#55>", line 1, in <module>
    list[4]
IndexError: list index out of range
>>>
```

Gördüyümüz kimi listəmizin içində 4-ədəd ifadə varkən amma list 4-cü ifadəni soruşduqda bizə 4-cü ifadənin olmadığını(normadan kənarda) bildirdi.Amma len() funksiyasını keçdiyimiz üçün gəlin listin neçə ifadədən olduğunu soruşaq.

```
>>> len(list)
4
>>>
```

Listlərin başdan yəni 0-dan sıralama üsulu soldan-sağa sadəcə aiddir,amma gəlin -1 və ya -2 verərək soruşaq

```
>>> list[-1]
'23'
>>> list[-3]
23
>>> list[-4]
'mayk'
>>> list[-5]

Traceback (most recent call last):
  File "<pyshell#60>", line 1, in <module>
    list[-5]
IndexError: list index out of range
>>> list[0]
'mayk'
>>>
```

Bizdə 23string cinsi sonda yer aldığından -1 ilə sorduqda bizə bunu göstərdi.Belə aydın olurki listlər geri saymada 0-dan deyil ilk ifadəni -1-dən başlayır.Geri saymada isə 4 ifadənin olduğu bizə məlum oldu.Düşünürəm append() metodunu başa düşdünüz.İndi isə ikinci bir metodumuz keçək

insert() metodu

Bu metodda **append()** metoduna bənzəyir.Mənası tərcümədə yerləşdirmək anlamına gəlir.Bu metodun **append()** metodundan fərqi isə istədiyimiz bir argumenti istədiyimiz bir yerə yerləşdirə bilirik.Misallara baxaq.

```
>>> list.append('suheng')
>>> list.insert(0,'suheng')
>>> list
['suheng', 'mayk', 23, 'elitar', '23', 'suheng']
>>>

>>> portList=[21,22,80,110]
>>> type(portList)
<type 'list'>
>>> portList = []
>>> portList.insert(0,21)
>>> portList.insert(1,80)
>>> portList.insert(2,443)
>>> portList.insert(3,25)
>>> print portList
[21, 80, 443, 25]
>>> portList
[21, 80, 443, 25]
>>>
>>> portList.insert(1,23)
>>> portList.insert(2,45)
>>> portList
[21, 23, 45, 80, 443, 25]
>>>
```

extend() metodu
Ingiliscədən genişləndirmək,genişlətmək mənasını ifadə edir.

Yəni əgər siz ayrı ayrı iki list hazırlayıb birləşdirmək istəsəniz bu zaman **extend()** metodu yardımınıza gələcək.Aşağıdakı misallara baxaq.

```
>>> yeniportList=[]
>>> yeniportList.append(26)
>>> yeniportList.insert(1,4444)
>>> yeniportList.insert(1,15)
```

```
>>> portList.extend(yeniportList)
>>> portList
[21, 23, 45, 80, 443, 25, 26, 15, 4444]
>>>
```

Gördüyümüz kimi yeniportList yaratdıq və sonra öncəki dərslərdə keçdiyimiz **insert()** və **append()** metodundan istifadə edərək bir neçə port daxil etdik daha sonra **extend()** metodu vasitəsilə **yeniportList**-i **portList**-yə əlavə etdik,tərcüməsindən anlaşıldığı kimi listi genişlətdik.Bunun başqa bir yolu isə hər iki list-i toplamaqdan keçir.

```
>>> a=[]
>>> a.append('jhon')
>>> a.append('Kevin Mitnhik')
>>> a.append('Pentest')
>>> a.insert(3,'Python')
>>> a
['jhon', 'Kevin Mitnhik', 'Pentest', 'Python']
>>> b=[]
>>> b.append('rubby')
>>> b.append('c++,c')
>>> b.append('java')
>>> b
['rubby', 'c++,c', 'java']
>>> a+b
['jhon', 'Kevin Mitnhik', 'Pentest', 'Python', 'rubby', 'c++,c', 'java']
>>>
```

a verdiyimiz yeni bir list yaratdıq daha sonra **b** adlı bir yeni list.Sonra bu hər iki list-i normal riyazi qaydada topladıq.Yalnız bu üsul müvəqqəti olaraq keçərlidir çünki daha sonra **a və ya b**-ni ayrı-ayrılıqda ekrana çap edə bilərsiniz.

```
>>> a
['jhon', 'Kevin Mitnhik', 'Pentest', 'Python']
>>> b
['rubby', 'c++,c', 'java']
>>>
```

remove() metodu

Daha bir metodumuz **remove()** metodudur.İngiliscədən tərk etmək,çıxartmaq

mənasını ifadə edir.Yaratdığımız listlərə argumentlər verirdiksə bu metod vasitəsilə onları listdən çıxarda biləcəyik.

```
>>> portList
[21, 23, 45, 80, 443, 25, 26, 15, 4444]
>>> portList.remove(21)
>>> portList.remove(4444)
>>> portList
[23, 45, 80, 443, 25, 26, 15]
>>> a
['jhon', 'Kevin Mitnhik', 'Pentest', 'Python']
>>> a.remove('jhon')
>>> a.remove('Pentest')
>>> a
['Kevin Mitnhik', 'Python']
>>>
```

Gördüyümüz kimi listlərimizdəki argumentlərdən integer cinslərini dırnaqsız,string cinslərini isə dırnaq içində verərək listdən remove etdik,yəni çıxartdıq.remove metodu listdən çıxartdığınız hər argumenti proses zamanı ekrana çap etmir.Bunu monitorda görmək üçün növbəti **pop()** metodu vasitəsilə yerinə yetirəcik.

pop() metodu

pop metodu listdən argumentin sıra nömrəsini verərək çıxarda və monitorda çap edə bilərik.

```
>>> a.pop(0)
'Kevin Mitnhik'
>>> a
['Python']
>>>

>>> portList
[23, 45, 80, 443, 25, 26, 15]
>>> portList.pop(3)
443
>>> portList
[23, 45, 80, 25, 26, 15]
>>>
```

index() metodu
index() metodu bir argumentin neçənci sırada olduğunu təyin edir.

```
>>> portList
[23, 45, 80, 25, 26, 15]
>>> portList.index(80)
2
>>> portList.index(15)
5
>>> portList[-1]
15
>>>
```

sort() metodu

Biz listlərə argumentlər əlavə etdikdə bu argumentlərin alfabe sıranı gözləmədən sonuncu sırada özünə yer tuturdu.Əgər listmizi alfabe sırasıyla düzmək istərsən bu zaman sort() metodu işimizə yarayacaq.

```
>>> list=[]
>>> list.append('Almaniya')
>>> list.insert(1,'manat')
>>> list.insert
>>> list.insert(2,'Vahid')
>>> list
['Almaniya', 'manat', 'Vahid']
>>> list.append('Hex')
>>> list.append('Engel')
>>> list.append('Albert')
>>> list
['Almaniya', 'manat', 'Vahid', 'Hex', 'Engel', 'Albert']
>>> list.sort()
>>> list
['Albert', 'Almaniya', 'Engel', 'Hex', 'Vahid', 'manat']
>>>
```

reverse() metodu
İngiliscədən tərs çevirmək mənasını ifadə edir.

```
>>> list.reverse()
>>> list
['manat', 'Vahid', 'Hex', 'Engel', 'Almaniya', 'Albert']
>>>
```

count() metodu
İngiliscədən saymaq kimi tərcümə olunur.
Pythonda da mənanı ifadə edir.Yəni bir argumentin list içində neçə dəfə olduğunu təyin edərək sayıyla monitora çap edir.

```
>>> list
['manat', 'Vahid', 'Hex', 'Engel', 'Almaniya', 'Albert']
>>> list.count('Hex')
1
>>> list.count('Engel')
1
>>> list.append('Hex')
>>> list
['manat', 'Vahid', 'Hex', 'Engel', 'Almaniya', 'Albert', 'Hex']
>>> list.count('Hex')
2
>>>
```

Burada list metodlarını bitirdik,Gəlin bir tabel formasında list metodlarını yazaq

append()	Yalnız bir argument alır	list.append(‘for example’)
insert()	İki argument alır	list.insert(0,’for example’)
extend()	Yalnız bir argument alır	list.extend(new_list)
remove()	Yalnız bir argument alır	list.remove(‘for example’)
pop()	Yalnız bir argument alır	list.pop(2)

index()	Yalnız bir argument alır	list.index(‘for example’)
sort()	Ala biləcəyi argument yoxdur	list.sort()
reverse()	Ala biləcəyi argument yoxdur	list.reverse()
count()	Yalnız bir argument alır	list.count(‘for example’)

tuple(tupl)

```
>>> dir(tuple)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__',
'__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__gt__',
'__hash__', '__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', 'count', 'index']
>>>
```

Tupllar listlərə bənzəyir,amma aralarında fərqlər var.Biz listlərə istədiyimiz an argument əlavə edə və silə bilirdiksə tupllarda bu mövcud deyil.tuplları ən

başdan hazır şəkildə argumentlərinizi yazıb yaradırsınız.
Yuxarıda istifadə edə biləcəyimiz tupl metodları count və index -dir.
Boş bir list yaratmaq üçün xatırlayırsınızsa list=[] kimi olurdu
boş bir tupl isə tupl=()
Bir argumentli tupl üçün isə argumentdən sonra vergül işarəsi qoyulur

```
>>>tupl=(‘Axundov’,)
>>> tupl='elementary','aquamarine','heinken'
>>> tupl
('elementary', 'aquamarine', 'heinken')
>>> type(tupl)
<type 'tuple'>
>>>
```

Gördüyünüz kimi açıq şəkildə mötərizəsiz argumentlərimizi daxil etdik və tipini soruşduqda bizə tuple olduğunu bildirdi.tuplları mötərizə içində də göstərə bilərik.

```
>>> tupl=('elementary','aquamarine','heinken',10,1,12)
>>> tupl
('elementary', 'aquamarine', 'heinken', 10, 1, 12)
>>> type(tupl)
<type 'tuple'>
>>>
```

Bir argumentli tupl yaratmaq üçün isə argumentdən sonra vergül işarəsi qoyulur.

```
>>> tupl='ex'
>>> type(tupl)
<type 'str'>
>>> tupl=('ex')
>>> type(tupl)
<type 'str'>
>>> tupl=('ex',)
>>> type(tupl)
<type 'tuple'>
>>>
```

Tuplların içərisindəki argumentləri başqa bir arqumentlə qeyd edib monitora çap edə bilərik

```
>>> tupl='tank','silah','top'
>>> type(tupl)
```

```
<type 'tuple'>
>>> t,s,t=tupl
>>> t
'top'
>>> t
'top'
>>> s
'silah'
>>> t[2]
'p'
>>> t[0]
't'
>>> t[1]
'o'
>>> s[0]
's'
>>> s[1]
'i'
>>>
```

count() və index() metodu .Tupllar üçün

```
>>> tupl.count('silah')
1
>>> tupl.count('top')
1
>>> tupl.index('top')
2
>>>
```

set() metodu

```
>>> a=set('asdfghjklzxcvbnm')
>>> a
set(['a', 'c', 'b', 'd', 'g', 'f', 'h', 'k', 'j', 'm', 'l', 'n', 's', 'v', 'x', 'z'])
>>>
```

dict

dictionary sözünün qısaltmasıdır.Dilimizə **lüğət** kimi tərcümə olunur.

```
>>> dir(dict)
```

```
['__class__', '__cmp__', '__contains__', '__delattr__', '__delitem__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__',
'__init__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
'__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'has_key', 'items', 'iteritems',
'iterkeys', 'itervalues', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values',
'viewitems', 'viewkeys', 'viewvalues']
>>>
```

Yuxarıda istifadə edə biləcəyimiz dict() metodları göstərilmişdir.dir() funksiyasının nə olduğunu dair irəlidə geniş bəhs edəcəyik.

Və ya

```
>>> help(dict)
Help on class dict in module __builtin__:

 values(...)
      D.values() -> list of D's values


  D.viewitems() -> a set-like object providing a view on D's items

  viewkeys(...)
    D.viewkeys() -> a set-like object providing a view on D's keys

  viewvalues(...)
     D.viewvalues() -> an object providing a view on D's values
.................
```

və s

```
>>>
```

yazaraq metodların istifadə qaydalarını görə bilərsiniz.
Gəlin lüğətlərin(dict) nə olduğuna baxaq.İlk öncə boş bir lüğət yaratmaq üçün

```
>>> dict={}
>>> dict
{}
>>> type(dict)
<type 'dict'>
>>>
```

Və boş bir lüğət yaratdıq ardınca lüğət olduğuna əmin olmaq üçün tipini

soruşduq.

```
>>> phone={'Newman':'+9942123456','Ballack':'+99421245678'}
>>> phone
{'Newman': '+9942123456', 'Ballack': '+99421245678'}
>>> type(phone)
<type 'dict'>
>>>
```

bir telefon kitabçası yaratdıq və Newman,Ballack adları üçün ayrı-ayrı telefon nömrələrini qeyd etdik.

Lüğətin istifadə edə biləcəyimiz metodlarından biri **keys()** metodudur.

```
>>> phone.keys()
['Newman', 'Ballack']
>>>
```

'Newman', 'Ballack' adları lüğətin içində **açar** sözlərdir,onların alt ifadələri isə həcmləri(**values()**) telefon nömrələridir.İndidə gəlin alt ifadələrini monitora yazdıraq.

```
>>> phone.values()
['+9942123456', '+99421245678']
>>>
>>> len(phone.values())
2
>>>
```

alt ifadələri **values()** metoduyla çap etdik ardından neçə ifadə olduğunu **len()** funksiyasıyla təyin etdik.

Dict içindəki açar sözləri silək üçün del ifadəsindən istifadə edəcəyik.del tək dictlərə aid olmadığı üçün bunu dict metodu kimi qeyd etmədim.Sadəcə olaraq del funksiyasından lüğətlərdə də istifadə edə biləcəyik.

```
>>> del phone['Ballack']
>>> phone
{'Newman': '+9942123456'}
>>>
```

gördüyümüz kimi del vasitəsilə Ballack adını sildik eləcədə Ballack adına dair telefon nömrəsidə silindi.İndidə Ballack -in lüğətimizdə olub olmadığını təyin edək

```
>>> 'Ballack' in phone
False
>>> 'Newman' in phone
True
>>> '+9942123456' in phone
False
>>> +9942123456 in phone
False
>>>
```

Qeyd edimki in funksiyası lüğətlərdə yalnız içəriyin **keys**-**açar** sözləri təyin etmə qabiliyyətinə malikdir.Yuxarıda açar sözün alt ifadəsini yoxladıqda false xətası verdi.çünki in ifadəsi **values()** hissəsini heç oxumadı bilə.

Gəlin iki sayda açar söz qeyd edək və **items()** metodundan istifadə edərək monitorda çap edək

```
>>> example={'freedom':'azadliq','excelent':'mohtesem'}
>>> example
{'excelent': 'mohtesem', 'freedom': 'azadliq'}
>>> type(example)
<type 'dict'>
>>> for a,b in example.items():
        print a,b


excelent mohtesem
freedom azadliq
>>>

>>> sorted(example)
['excelent', 'freedom']
>>>
```

Yuxarıda sorted vasitəsilə açar sözlərimizi alfabe sırasıyla düzdük.

```
>>> example={'john','enumerate','exam','snapshot','woodalias'}
>>> example
set(['snapshot', 'john', 'woodalias', 'exam', 'enumerate'])
>>> sorted(example)
['enumerate', 'exam', 'john', 'snapshot', 'woodalias']
>>>
```

sorted-dən başqa python2 -də OrderedDict metodu vardırki lüğətlərdə istifadə etmə imkanı verir.

```
>>> from collections import OrderedDict
>>>
```

Yarada biləcəyimiz funksiyalar.
Funksiya yaratma

Buraya qədər pythonda mövcud olan bir sıra funksiyalar keçdik.Python istifadəçilərin də funksiya yaratma imkanını mümkün qılır.
Funksiya yaratmaq üçün def...() sabit sözlüyündən istifadə olunur.Funksiyaların bizə yararı çox böyükdür.Həm yazdığımız kod bloklarını bir yerə toplamağa,rahat anlamağa yardımçı olur.Eləcədə GUİ,tkinter bəhsində funksiyalardan geniş istifadə edəcəyik.Buttonlara command vermək üçün ayrıca yazdığımız funksiya vasitəsilə olacaq.
İlk funksiyamızı yaradaq.
Bir mətin faylı və ya idle-file-new ardıcıllığıyla fayl açırıq.

Ilk sətrə həmişə olduğu kimi path və ascii kodlamanı qeyd edirik.Ardından

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
def ilk_funksiya():
    print 'Hello world!'
```

yazırıq və bunu çalışdırdıqda bir şey olmadığının fərqinə varacıq.Çünki

funksiyamız natamamdır,yəni açıq qalmışdır.def sözündən sonra qeyd etdiyimiz ilk_funksiya() vasitəsilə funksiyamızı bağlayırıq.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
def ilk_funksiya():
    print 'Hello world!'

ilk_funksiya()


Hello world!
>>>
```

ilk_funksiya -nı hara yazdığımıza diqqət edin.def sözüylə eyni sütunda olmalıdır.istərsəniz print funksiyasının altında qeyd edib nə kimi xəta verəcəyinə baxın.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
def ilk_funksiya():
    print 'Hello world!'

    ilk_funksiya()

>>>
```

göründüyü kimi alt sətrə boş keçdi və heç bir şey monitora yazmadı.

İndidə gəlin iki print funksiyasından istifadə edərək monitora bir şeylər yazdıraq

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
def ilk_funksiya():
    print 'Hello world!'
    print 'Hello user!'
ilk_funksiya()

Hello world!
Hello user!
>>>
```

Arqumentli funksiya yaratmaq.

```
def ilk_funksiya(name):

    print 'you name:',name

ilk_funksiya('eldar')

you name: eldar
>>>
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
def multi(tuple):
    a=1
    for i in tuple:
        a=a*i
    print a
eded=(1,2,4)
multi(eded)

8
>>>
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
def multi(a, b):
    print a * b
multi(10, 20)

200
>>>
```

Yuxarıdakı funksiya isə a=10,b=20 kimi qeyd etdik.python bunları a*b bir birinə vuraraq nəticəni monitora verdi.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def multi(a, b):
```

```
    print a * b
a = 12
b = 16
multi(a, b)


192
>>>
```

Yuxarıdakı kodlardan əvvəlki funksiyamızda a və b-ni funksiyanı bağlarkən qiymətlərin qeyd etmişdiksə sonrakı kodlarımızda isə bunu funksiyanı bağlamamışdan öncə qeyd etdik

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def allegator(info,time=1):
    print info*time
allegator('hello')
allegator('world',4)

hello
worldworldworldworld
>>>
```

Öncəki dərslərdə keçdiyimiz list metodundan istifadə edərək bir funksiya yazaq.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def list_funk(list):
    list=[]
    name=raw_input('write you name:')
    for i in name:
        list.append(i) # insert() metodunu da istifadə edə bilərsiniz
        print list
list_funk(list)

write you name:rashad
['r']
['r', 'a']
['r', 'a', 's']
['r', 'a', 's', 'h']
['r', 'a', 's', 'h', 'a']
```

```
['r', 'a', 's', 'h', 'a', 'd']
>>>
```

def vasitəsilə list_funk(list) funksiyasını qeyd etdik.daha sonra boş bir list yaratdıq və ardınca istifadəçidən adını girməyi tələb etdik,istifadəçi adını yazdıqdan sonra **append()** metodyla listmizə əlavə etdik və ardınca **print()** -lə monitora çap etdik.Biz funksiyamızda **for** operatorundan istifadə etdiyimiz üçün monitora yuxarıdakı formada çap etdi,Bir list formatında çap etmək istəyirsinizsə

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def list_funk(list):
    list=[]
    name=raw_input('write you name:')
    list.append(name)
    print list
list_funk(list)

write you name:Vahid
['Vahid']
>>>
```

Yuxarıdakı funksiyamız hər zaman istifadəyə artıq hazırdır.Yuxarıdakı funksiyanı bir fayla yazın və adını bir şey yazaraq,sonunu py dosyasında qeyd edin.Yəni funk.py
Mən syut.py kimi yadda saxladım.Və terminalı açın ardınca python yazın.Önünüzə gələn python komanda sətrinə import syut verin əgər alt sətrə səhvsiz keçsə deməli yazdığınız funksiyanı python rahatlıqla gördü

```
$python
Python 2.7.12rc1 (default, Jun 13 2016, 09:20:59)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import syut
write you name:Eldar
['Eldar']
>>>
```

ekran görünüşü yuxarıdakı kimidir.Yəni yazdığımız funksiya həm də bir moduldur.Sizə qaranlıq tərfi import bölümüdürki heç tələsməyin modullar haqqında növbəti bəhslərimizdə ətraflı danışacağıq.

global operatoru
Global ifadəsi pythonda mühüm ifadələrdən biridir.Biz bir funksiya daxilində yazdığımız integer və stringlər,eləcədə sabit ifadələri monitora çap edə biliriksə global ifadəsi funksiya xaricində də cavabı olduğu kimi ekranda çap etməyə yardım edir.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def funk():
    i=12
    print i
funk()
print i
```

```
Traceback (most recent call last):
  File "/home/panda/syut.py", line 7, in <module>
    print i
NameError: name 'i' is not defined
>>>
```

Yuxarıda funksiyamızın xaricindədə i-nin qiymətini çap etdirmək istəsəkdə xəta aldıq.Xətada deyilirki i adlı dəyişkən qeyd olunmayıb.
Gəlin bunu global ifadəsiylə yenidən yazaq.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def funk():
    global i
    i=12
    print i
funk()
print i
```

```
12
12
>>>
```

Gördüyümüz kimi global vasitəsilə -i-nin qiymətini funksiya xaricində də monitora

çap edə bildik.

Xətalar. try except blok operatorları

Hər zaman uazdığımız programlar istifadəçilər tərəfindən bizim düşündüyümüz kimi çalışmır.Biz bir programda istifadəçidən yalnız rəqəm və ya hərf tələb etsəkdə istifadəçi bilmədən və ya bilərək hər hansısa bir düymə və ya uzun bir integer,string cinsi daxil edə bilər.Bunlar da program çalışdığında xətalar verəcək.Bu kimi xətaları önləmək üçün python bizə try və except operator bloklarını təklif edir.
Məsələn

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def integer():
    a=input('write you number:')
    if a>10:
        print 'You number:',a
    else:
        print 'You loss!'
integer()
```

```
write you number:45
You number: 45
>>>

write you number:df
```

```
Traceback (most recent call last):
  File "/home/panda/syut.py", line 9, in <module>
    integer()
  File "/home/panda/syut.py", line 4, in integer
    a=input('write you number:')
  File "<string>", line 1, in <module>
NameError: name 'df' is not defined
>>>
```

Gördüyümüz kimi input() funksiyası vasitəsilə istifadəçidən(user) bir rəqəm girməyini tələb etdik ilk başda rəqəm daxil etdi ardından ikinci dəfə programı çalışdırdıqda hərf birləşməsi daxil edərək programı çökdürdü.Və aldığımız xətanın tipi NameError .Və xətada deyilirki df qeyd olunmayıb və string cinsidir.Bəli string cinsidir,Çünki input funksiyası integer cinsləri ilə çalışır.Bu tip xətanı önləmək üçün try operatorundan istifadə edəcəyik.Program içində try və except blok operatorları necə istifadə olunduğuna aşağıdakı misala diqqət yetirək.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def integer():
    while True:
        try:

            a=input('write you number:')
            if a>10:
                print 'You number:',a
            else:
                print 'You loss!'
        except NameError:
            print'please write only number'

integer()
```

```
write you number:df
please write only number
write you number:34
```

```
You number: 34
write you number:56
You number: 56
write you number:rturthth
please write only number
write you number:
```

Yuxarıdakı kodların ekran görüntüsündən göründüyü kimi istifadəçi hərf və ya hərf birləşməsi girərsə program çökməmiş istifadəçiyə xəta mesajı verdi.

ValueError xətası
ValueError: invalid literal for int() bu kimi xətaya bəlkədə rast gəlmisinizdir.integer cinsi ilə bağlı bir xəta mesajıdır.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def integer():
    a=int(raw_input('write you number:'))
    if a>10:
        print 'You number:',a
    else:
        print 'You loss!'

integer()



write you number:a


Traceback (most recent call last):
  File "/home/panda/syut.py", line 10, in <module>
    integer()
  File "/home/panda/syut.py", line 4, in integer
    a=int(raw_input('write you number:'))
ValueError: invalid literal for int() with base 10: 'a'
>>>
```

Programımızı çalışdırdıqda yuxarıdakı ValueError xətası aldıq.yəni istifadəçinin

girdiyi hərf olduğundan ondan öncə gələn int() operatoru cins dəyişdirməni yerinə yetirə bilmədiyi üçün xəta mesajını aldıq

Eyni qayda ilə bu programın çökməməsi üçün try və except opertorlarından istifadə edəcik.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def integer():
    try:
        a=int(raw_input('write you number:'))
        if a>10:
            print 'You number:',a
        else:
            print 'You loss!'
    except ValueError:
        print 'please write only number'
integer()
```

```
write you number:s
please write only number
>>>
```

Program davamlı çalışması üçün while True kodların əvvəlinə yazaraq davamlı tuta bilərsiniz.Beləcə program çökmədən istifadəçi hərf girərsə xəta mesajı vermə imkanına malik olduq.

ZeroDivisionError xətası

Bu xəta növü ilk ifadəsi olan zero-yəni sıfır ilə bağlıdır.Bilirikki riyaziyyatda bölmə əməliyyatında bir ədəd sıfıra bölünməz.python-u hesablayıcı maşın kimi istifadə

edərək ədədi 0-a böldüyümüz zaman bu xətanı alacıq.

```
>>> 5/0

Traceback (most recent call last):
  File "<pyshell#34>", line 1, in <module>
    5/0
ZeroDivisionError: integer division or modulo by zero
>>>
```

Gəlin hesablama programı yazaq və bölmə əməliyyatını aparsın.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def division():
    while True:
        a=input('ilk rəqəmi yazın:')
        b=input('ikinci rəqəmi yazın:')
        print a/b
division()



ilk rəqəmi yazın:23
ikinci rəqəmi yazın:0

Traceback (most recent call last):
  File "/home/panda/syut.py", line 8, in <module>
    division()
  File "/home/panda/syut.py", line 7, in division
```

```
    print a/b
ZeroDivisionError: integer division or modulo by zero
>>>
```

ZeroDivisionError xətasından istifadə edərək

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
from __future__ import division
def division():
    while True:
        try:

            a=input('ilk rəqəmi yazın:')
            b=input('ikinci rəqəmi yazın:')
            print a/b
        except ZeroDivisionError:
            print "0-a bölmə əməlliyyatı doğru deyil\nyenidən rəqəmləri daxil edin"
division()
```

```
ilk rəqəmi yazın:34
ikinci rəqəmi yazın:45
0.755555555556
ilk rəqəmi yazın:23
ikinci rəqəmi yazın:0
0-a bölmə əməlliyyatı doğru deyil
yenidən rəqəmləri daxil edin
ilk rəqəmi yazın:56
```

ikinci rəqəmi yazın:13

4.30769230769

ilk rəqəmi yazın:

return operatoru

Dilimizə geri dönmək ,qayıtmaq kimi tərcümə olunur.Gələcəkdə yazacağımız kod bloklarında bu ifadədən çox istifadə edəcəyik.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def integer_a():

    ques=int(raw_input('rəqəm yazın:'))
    return ques
a=integer_a()
print 'yazdığınız rəqəm:',a
if a%2==0:
    print 'daxil etiyiniz rəqəm cütdür'
else:
    print 'daxil etdiyiniz rəqəm təkdir'
```

```
rəqəm yazın:2
yazdığınız rəqəm: 2
daxil etiyiniz rəqəm cütdür
>>>

rəqəm yazın:23
yazdığınız rəqəm: 23
daxil etdiyiniz rəqəm təkdir
```

```
>>>


#!/usr/bin/env python
#-*- coding:utf-8 -*-
def maximum(x,y):
    if x>y:
        return x
    elif x==y:
        return "all right"
    else:
        return y
print(max(2,3))

3
>>>


#!/usr/bin/env python
#-*- coding:utf-8 -*-
def maximum(x,y):
    if x>y:
        return x
    elif x==y:
        return "all right"
    else:
        return y
print(min(2,3))


2
>>>
```

Yuxarıda gördüyünüz min() , max() operatorları minimum və maximum sözlərinin qısaltmasıdır.Funksiya içində integerlərin ən böyüyünü max() ən kiçiyini min() operatoru təyin edir.

pass operatoru

Dilimizə ötürmək,keçmək kimi tərcümə olunur.Pythonda da bu mənanı ifadə edir.Misallarla bunu aydın başa düşəcəyik.
Biz bir hesablama programı yazmaq istəyirik,əgər istifadəçi 0-rəqəmini daxil edərsə program avtomatik 0-ı görmədən keçsin növbəti komandaya.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def division():
    while True:
        a=input('rəqəm daxil edin:')
        if a==0:
            pass
        else:
            print a,'\nqüvvət=',a**2
division()
```

```
rəqəm daxil edin:2
2
qüvvət= 4
rəqəm daxil edin:0
rəqəm daxil edin:3
3
qüvvət= 9
rəqəm daxil edin:
```

Pythonda modullar (importing modules)

Gəlin bir program yazaq və py sonluqlu yadda saxlayaq.

```
def ad(list):
    list=[]
    a=raw_input('write you name:')
    list.append(a)
    print list
ad(list)
```

Mən syut.py olaraq yadda saxladım.
Sonra python komanda sətrində import program_adi yazın.Yəni bu şəkildə

```
>>> import syut
```

və alt sətrə xətasız keçdisə deməli modulunuzu çağıra bildiniz.əgər xəta ilə qarşılaşarsınızsa programınızı yadda saxladığınız yerə gedin və ordan cut(kəsmək) verərək götürüb işçi stoluna yerləşdirin və ya home/user/program_adı.py

Daha sonra komanda sətrindən dir(syut) yazın

```
>>> dir(syut)
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'ad']
>>>
```

Nəticə yuxarıdakı kimi olacaq.Fikir verirsinizsə öncə syut.py programımızın içindəki ad() funksiyası sonda göründü.Deməli biz ad funksiyamızı hər yerdə istifadə etmə imkanımız var.

komanda sətrində

```
>>> import syut
>>>syut.ad(list)
write you name:John
['John']
```

Deməli biz istənilən an komanda sətrindən bu programımızı çalışdıra bilərik.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
def version():
    print 'Bu bizim ilk modulumuzdur'
__version__='0.1'
```

yuxarıdakı kodları bir adla py sonluqlu yadda saxlayırıq.(bunu syut.py kimi qeyd etdim)

Yeni bir gedit açaraq ad verib bunu da py sonluqla yadda saxlayırıq(mən mymodul.py kimi qeyd etdim).Sonuncu açdığımız gedit-ə yazırıq

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import syut
syut.version()
print 'version',syut.__version__
```

və yadda saxlayıb bağlayırıq.Ardından terminalı açırıq python mymodul.py programımızı çalışdırırıq.

```
$python '/home/panda/mymodul.py'
Bu bizim ilk modulumuzdur
version 0.1
$
```

Qeyd edimki bu hər iki fayl eyni bir yerdə yəni ya işçi stolunda(Desktop) yada home/user içində olmalıdır.Ayrı-ayrı yerdə olarsa aşağıdakı kimi xəta alacaqsınız.

```
 $python '/home/panda/Desktop/mymodul.py'
Traceback (most recent call last):
  File "/home/panda/Desktop/mymodul.py", line 3, in <module>
    import syut
ImportError: No module named syut
$
```

Sabit modullar və operatorlar

from ….import…..

Yuxarıda import program_adı yazaraq çağırırdıqsa from import operator və moduldan istifadə edərək modul içindəki funksiyanı istifadə edə biləcik.

Python komanda sətrindən

>>> from syut import ad verərək alt sətrə səhvsiz keçirik.
>>> ad(list)

write you name:Eldar
['Eldar']
>>>

from -dan dən şəkilçisi kimi tərcümə olunur.yəni syutdan ad funksiyasını import

et
Bu metoda siz tez-tez rast gələcəksiniz.

Hazır modullardan math moduluna dair bir misal yazaq.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
from math import*
n=input('write number:')
p=[2,3]
count=2
a=5
while (count<n):
    b=0
    for i in range(2,a):
        if(i<=sqrt(a)):
            if a%i==0:
                print a
                b=1
            else:
                pass
    if b !=1:
        print a
        p=p+[a]
    count=count+1
    a=a+2
print p
```

```
write number:12
5
7
9
11
13
15
17
19
21
23
[2, 3, 5, 7, 11, 13, 17, 19, 23]
>>>
```

os modulu

Os modulu kompyuterinizdə os.py adında bir python programı olub /usr/lib/python2.7/ içində yer almışdır.Windows-da isə C:/Python27/Lib Modulumuzu python komanda sətrindən çağıraq.

```
>>> import os
>>> dir(os)
```

 yazaraq önümüzə gələn bir çox operatorları ekranda görəciyik.Bu uzun sətrləri bura qeyd etmədim.
Bir öncəki bəhsdə yazdığımız modulun içindəki funksiya kimi,os modulunun da içində funksiya deyil funksiyalar toplusu var.Yuxarıda dir(os) komandası ilə verdiyimiz əmrlə os modulunun içindəki funksiyaları ekrana tökdük.
Os modulunun içərisində yer alan name funksiyasına baxaq.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
from os import name
if name=='posix':
    print 'Hello linux'
elif name=='nt':
    print 'hello windows'
elif name=='mac':
    print 'hello macintosh'
```

```
Hello linux
>>>
```

Yuxarıdakı kodlarımızda from os import name yazmaqla os modulundan sadəcə name funksiyasını istifadə etdik.
Və ekran görüntüsündə Hello linux ifadəsi çıxdı.Çünki mənim əməliyyat sistemim linux olduğundan irəli gəlir.Əgər windows altında çalışırsanız windows və sairə görünəcəkdir.Qeyd edimki

windows üçün	"nt ", "ce "
Macintosh üçün	"mac "
OS/2 üçün	"os2"

yazılır.

Əməliyyat sistemini sorğuya çəkməyin başqa bir yoluda,sadəcə os modulunu çağırmaqdan keçir.Bu zaman isə os.name ifadəsini istifadə edəciyik.

Sadəcə

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
import  os
if os.name=='posix':
    print 'Hello linux'
elif os.name=='nt':
    print 'hello windows'
elif os.name=='mac':
    print 'hello macintosh'
```

```
Hello linux
>>>
```

Os modulunun içində yer alan növbəti funksiya listdir funksiyasıdır.listdir funksiyasının rolu əməliyyat sisteminizdəki dosyaları göstərməyə imkan yaradır.Qovluqlar(Desktop,Home,user,tmp,C və s)

```
>>> import os
>>> a = os.listdir("/home/")
>>> print a
```

```
['panda']
>>>
```

İndi isə istifadəçidən bunu alaraq ekrana yazacaq bir program yazaq.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
import os
def listdir():
    qovluq=raw_input('görmək istədiyiniz dosyanın adını yazın:')
    a=os.listdir('/'+dosya+'/')

    print a
listdir()
```

```
görmək istədiyiniz dosyanın adını yazın:home
['panda']
```

```
>>>
```

Gördüyünüz kimi raw_input() funksiyası vasitəsilə istifadəçidən dosya adını girməyi istədik daha sonra os.listdir funksiyasını print vasitəsilə ekrana çap etdik. Sizə qaranlıq qalan bölümü ('/'+qovluq+'/') ola bilər.Listdir funksiyası dırnaq içində iki tərəfi də slash işarəsiylə dosyana ekrana yazdırır.Həm də listdir funksiyası yalnız bir argument aldığından mötərizə içindəki iki + toplama işarəsini birlikdə qəbul edərək toplam bir argument dosya-argumenti olaraq görür.İstərsəniz mötərizə içində slash işarələrindən sonra vergul qoyub yazaq

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
import os
def listdir():
    qovluq=raw_input('görmək istədiyiniz dosyanın adını yazın:')
    a=os.listdir('/',qovluq,'/')

    print a
listdir()
```

```
görmək istədiyiniz dosyanın adını yazın:home

Traceback (most recent call last):
  File "/home/panda/syut.py", line 9, in <module>
    listdir()
  File "/home/panda/syut.py", line 6, in listdir
    a=os.listdir('/',dosya,'/')
TypeError: listdir() takes exactly 1 argument (3 given)
>>>
```

Gördüyümüz kimi toplamanı silib vergüllə əvəz etdik və program çalışdığında xəta aldıq.Xətada deyilirki listdir funksiyası yalnız bir argument alır,siz 3 argument daxil etmisiniz.Burda toplama işarəsi bizim yardımımıza gəldi.Tək bu funksiyada deyil bir argument ala bilən başqa funksiyalarda da toplama işarəsindən istifadə edəcik.

getcwd() funksiyası

Os modulunun içində yer alan növbəti funksiyamız getcwd funksiyasıdır.Bu funksiyanın rolu,siz hal hazırda hansı qovluq altında çalışdığınızı ekrana verəcəkdir.

```
>>> import os
>>> os.getcwd()
'/home/panda'
```

chdir() funksiyası

Bu funksiya çalışma zamanı olduğunuz qovluqdan bir addım geridəki qovluqa keçid edir.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
import os
def chdir():
    a=os.chdir(os.pardir)

    print os.getcwd()
chdir()

/home/panda
>>>
```

yuxarıda göstərilən os.pardir funksiyası bir addım geridəki qovluqa keçməyə yardımçı olur.parent directory sözünün qısaltmasıdır.

Bu funksiyalardan əlavə curdir(os.curdir) funksiyası da var.
curdir funksiyasının rolu istədiyimiz qovluqu manuel,əllə yazaraq keçid ala bilərik.

```
>>> import os
>>> os.listdir(os.curdir)
['.gscriptor', 'untitled', '.ophcrackrc', '.xsession-errors.old', '.gnome2', '.kde',
'programlar', '.wine', '.recently-used', '.grc_gnuradio', '.gconf', 'Desktop', '.swt',
'.eclipse', '.gnupg', '.config', '.zuluCrypt-socket', '.ssh', '.bash_history', '.local',
'.links2', 'PycharmProjects', '.icedove', 'eclipse', 'Downloads', '.ipython',
'.subversion', '.pycrust',
.....
>>>
```

```
>>> import os
>>> os.chdir('/tmp')
>>>
```

Əgər alt sətrə xətasız keçdisə,deməli mötərizə daxilində yazdığımız qovluq düzgündür.

Qovluq yaratma.mkdir və makedirs() funksiyaları.
Os modulunun içində yer alan bu iki funksiya vasitəsilə qovluq yaradacağıq.
Qovluq yaratmanın iki yolu var.Birincisi birbaşa çalışmaqda olduğununz ana qovluğun içində python özü yaratsın.
İkincisi isə istifadəçinin göstərdiyi qovluq yolu ilə yaratma.

```
>>> import os
>>> print os.getcwd()
/tmp
>>> os.mkdir('qovluq')
>>>
```

Biraz öncə chdir funksiyası vasitəsilə /tmp qovluquna getdik.Və çalışma qovluğumuz /tmp qovluğu olduğundan yuxarıdakı kodlarda os.getcwd() vasitəsilə ekrana hansı qovluqda olduğumuzu görə bildik.Daha sonra os.mkdir('test') komandası verərək tmp qovluqunda yeni bir alt test qovluğu yaratdıq.

Və ya

```
>>> import os
>>> os.mkdir('/home/user/qovluq')
>>>
```

qeyd etdiyim kimi home/user/qovluq_adı yarada bilərsiniz.
Bir də yaratdığımız qovluqların içində yeni qovluq yaratma yolları var.Gəlin bunu mkdir() funksiyasından istifadə edərək yaradaq.

```
>>> import os
>>> os.mkdir('/home/user/python/elementary')

Traceback (most recent call last):
```

```
  File "<pyshell#14>", line 1, in <module>
    os.mkdir('/home/panda/python/elementary')
OSError: [Errno 2] No such file or directory: '/home/panda/python/elementary'
>>>
```

Yuxarıda mkdir vasitəsilə qovluq altında qovluq yaratmağa cəhd etsəkdə bu baş tutmadı,xəta verdi.Ən başda mkdir() funksiyası ilə yanaşı makedirs() funksiyası yazmışdıq.Bu funksiya son yaradacağımız qovluqlar üçün işimizə yarayacaq.

```
>>> import os
>>> os.makedirs('/home/user/python/elementary')
>>>
```

və alt sətrə xətasız keçdi.makedirs() funksiyasının da nə rol oynadığını gördük.

rmdir() və removedirs() funksiyaları.

Os modulunun içində yer alan növbəti funksiyalardandır.Dilimizə tərcümədə kənarlaşdırmaq,çıxartmaq kimi tərcümə olunur.Düşünürəm aşağıdakı yazdığımız kodlara diqqətlə yanaşacaqsıız.tərcüməsindən anladığımız kimi hansısa qovluğu silməyə,kənarlaşdırmağa səy göstəririk.Yəni ehtiyatlı olmağınız məqsədə uyğundur.Ək s halda ana qovluqlardan silə,əməliyyat programınızı çökdürə bilərsiniz.

Gəlin home/user/ altında bir Axundov qovluğu yaradaq.

```
>>> import os
>>> os.mkdir('/home/user/Axundov')
>>>
```

və bu yaratdığımız Axundov qovluğunu siləkk.

```
>>> import os
>>> os.rmdir('/home/user/Axundov')
```

```
>>>
```

və alt sətrə səhvsiz keçdik.Gördüyümüz kimi ana user qovluğu altında öncə yaratdığımız Axundov qovluğunu sildik.
İndi də Axundov qovluqunun içində başqa bir qovluq yaradaq.

```
>>> import os
>>> os.makedirs('/home/panda/Axundov/kitabxana')
>>>
```

Yuxarıda Axundov qovluğunun içində yeni,kitabxana qovluqu yaratdıq.mkdir() və makedirs() funksiyasının bir birindən fərqi bunda idiki,mkdir() funksiyası vasitəsilə sadəcə bir qovluq,makedirs() funksiyası vasitəsilə isə qovluq içində qovluqlar yarada bilirdik.Eləcə də rmdir() və removedirs() funksiyaları da bunlara bənzər iş prinsipləri var.rmdir() funksiyası vasitəsilə yalnız bir qovluğu,qovluq içində qovluqları removedirs() funksiyası vasitəsilə siləcəyik(kənarlaşdırmaq).
Hansı xəta verə biləcəyini gəlin test edək.

```
>>> os.rmdir('/home/panda/Axundov')

Traceback (most recent call last):
  File "<pyshell#24>", line 1, in <module>
    os.rmdir('/home/panda/Axundov')
OSError: [Errno 39] Directory not empty: '/home/panda/Axundov'
>>>
```

Yuxarıda xətadan göründüyü kimi -Axundov qovluğunun boş olmadığını- xəta kimi verdi.

```
>>> os.rmdir('/home/panda/Axundov/kitabxana')
>>>
 >>> import os
>>> os.removedirs('/home/panda/Axundov/kitabxana')
>>>
```

və alt sətrə hər iki prosesdə xətasız keçdik.

Fayllar.Fayl yaratmaq.
open() funksiyası
Faylların sonu müxtəlif olur.Məsələn txt,py,cpp,lst və sairə kimi.Bu faylların sonluqları da hansısa bir dilə(c və c++,ruby,python,perl,html və s) xidmət edir.
Fayl yaratmaq üçün sonluğu nə istərsəniz seçə bilərsiniz.
Ilk öncə hansı qovluq altında çalışmağımıza baxaq

```
>>> import os
>>> os.getcwd()
'/home/user'
>>>
```

Deməli home/user – qovluqunun içindəyik.Və yaradacağımız fayl,bu qovluqda olacaq.

```
>>> open('new.txt','w')
<open file 'new.txt', mode 'w' at 0x7f00afb938a0>
>>>
```

Faylın adını new.txt yazaraq açdıq.fayl adını siz nə istəsəniz yaza bilərsəniz,yetərki sonluğunu(txt,cpp,rb,lst və s) doğru qeyd edəsiniz.Daha sonra 'w' tipində olmasını əmr etdik.Faylların müxtəlif tipləri var gəlin bu tiplərə nəzər salaq.
'w' tipi write sözünün baş hərfidir,yəni yazma tipində açılıb.Bundan başqa
r tipi.read sözünün baş hərfidir.Oxumaq mənasını ifadə edir.
a tipi.append sözünün baş hərfidir.Əlavə etmək mənasını ifadə edir.

r tipində açdığımız bir fayla heç bir şey yazmaq və əlavə etmək mümkün deyil.Çünki bu yalnız read tipində,faylı oxumaq mənasını ifadə edir.a tipi isə açdığımız bir fayla daha sonra əlavələr etmək(kök fayl qalır və yenisi əlavə olunmur.) üçün istifadə olunur.

Yuxarıda new.txt faylı açmışdıq.tipi isə 'w' qeyd edik.İndidə bu eyni adla 'r' tipində açaq.

```
>>> open('new.txt','r')
<open file 'new.txt', mode 'r' at 0x7f00afa23e40>
>>>
```

Və faylımızı r tipində açdıq.

Əgər biz bir faylı r tipində açmağa cəhd etsək,

```
>>> open('new_1.txt','r')

Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    open('new_1.txt','r')
IOError: [Errno 2] No such file or directory: 'new_1.txt'
>>>
```

Problemlə qarşılaşacağıq.Çünki r tipi mövcud olan fayla əmr verir.Əksinə yeni fayl açmır.Bu yolla gəlin w tipinə baxaq

```
>>> open('elementary.txt','w')
<open file 'elementary.txt', mode 'w' at 0x7f00afa23e40>
>>>
```

Gördüyümüz kimi w tipində yeni fayl yaratma imkanımız var.Bunu zatən ən başda istifadə etmişdik.

Əgər bu tiplərin heçbirini istifadə etməsək python birbaşa faylı r tipində açacaqdır.

Fayl açarkən os modulunun funksiyalarındakı kimi faylı istədiyimiz qovluqda açma imkanımız olacaq

```
>>> a=open('/home/panda/ekstra.txt','a')
>>>
```

Və user qovluquna gedib ekstra.txt faylının olub-olmadığını yoxlayırıq.Və bəli ekstraştxt faylı a tipində açılmışdır.
Qeyd edimki pythonu windows əməliyyat sistemində çalışdırırsanız,yuxarıdakı yol sizə xəta verəcəkdir..

```
>>> a= open("C:\Documents and Settings\user\Desktop\ekstra.txt", "a")
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
```

IOError: [Errno 22] invalid mode ('a') or filename: 'C:\\\\Documents and Settings\\user\\\\Desktop\\ekstra.txt'

Bunun üçün xüsusi işarələrdən olan r-i istifadə etməyiniz məsləhət görülür.
Və ya \\ -işarəsindən istifadə edə bilərsiniz.

```
>>> a= open("C:\\Documents and Settings\\user\\Desktop\\ekstra.txt", "a")
```
və ya
```
>>> a = open(r"C:\Documents and Settings\user\Desktop\ekstra.txt", "a")
```

Fayllara ifadə əlavə etmək.

Bildiyimiz kimi yaratdığımız hər hansı bir sonluqlu fayllara üstündən iki dəfə basmaqla açıb,istədiyimiz ifadələri daxil edə bilərik.Bizim məqsədimiz isə daha asan variantla,python dilinin köməyi ilə açdığımız fayllara istədiyimiz ifadəni daxil edək.
Bu minvalla yeni bir txt sonluqlu mətn faylı açırıq.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
file=open('new.txt','w')
file.write('Azərbaycanın ölməz şairi Ramiz Rövşəndən sətrlər')
file.close()
```

new.txt adlı yeni fayl açdıq.Ardından write metoduyla fayla ifadə əlavə etdik,daha sonra faylımızı close() metoduyla bağladıq.Əgər sonda close() yazmasaydıq faylımıza heç bir ifadə daxil olmayacaqdı.close() metodunu sonda yazmağı unutmayın.
Və ardından ikinci sətrinə yeni bir ifadələr əlavə edək.Bunun üçün xüsusi işarəmiz olan \n istifadə edəcik.Əgər siz istərsəniz yeni ifadəni ilk ifadədən bir tab düyməsi qədər də yazdıra bilərsiniz.(Bunun üçün ifadənin əvvəlinə \t işarəsini əlavə eməliyik)
Bu dəfə isə artıq 'w'(write) tipində deyil,a(append) tipində var olan faylımıza əlavələrimizi edirik.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
file=open('new.txt','a')
file.write('\nAzərbaycan Dövlət Dəniz akademiyası')
file.close()
```

Bəli,faylı açırıq və gördüyümüz kimi son ifadəmiz ikinci sətrə əlavə olunub. Hətda bir tupl və ya list içindəki ifadələrimizi faylımıza əlavə edə bilərik.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
tuple='flight','elder','armagedon','monliya'
file=open('new.txt','a')
for i in tuple:
    file.write(i+'\n')
file.close()
```

write() metodundan əlavə writelines() metodu da var.Bu metod vasitəsilə ifaələrin arasına vergül ataraq istədiyiniz sayda cümlələr əlavə edə bilərsiniz.

Faylı və ya fayldan ifadələri oxumaq.
Bir öncəki bəhsdə öyrəndiyimiz kimi ya yeni fayl aça bilərsiniz və ya hazır bir fayldan ifadələri oxuya bilərsiniz.Biz yuxarıda açdığımız new faylın içindəki ifadələri oxuyaraq ekrana yazdıracıq.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
file=open('new.txt','r')
print file.read()
```

```
Azərbaycanın ölməz şairi Ramiz Rövşəndən sətrlər
Azərbaycan Dövlət Dəniz akademiyasıflight
elder
armagedon
monliya

>>>
```

read() metodundan başqa readlines() metodu var.

Yuxarıdakı kodlar içində read() metodu yerinə readlines() yazaraq nə kimi nəticə olduğuna baxaq.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
file=open('new.txt','r')
print file.readlines()
```

Ekran görüntüsündən bəlli olurki ifadələr bir list şəklində çap olunur.readlines() metodundan başqa readline() metodu da var.Bunu da kodlara daxil edərək nəticəyə baxaq.

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
file=open('new.txt','r')
print file.readline()


Azərbaycanın ölməz şairi Ramiz Rövşəndən sətrlər

>>>
```

Yuxarıda ekran görüntüsündən məlum olurki readline() metodu yalnızca fayl içindəki ilk sətri oxumaq hüququna malikdir.Bu metoddan ikinci dəfə istifadə edərsək ekran görüntüsündə fayl içindəki ikinci sətr görünəcəkdir.Bu metod sıra-sıra faylı oxuyur.İlk sətrə dönmənin yolu isə seek() metodundan keçir.seek() metoduna sətir sayını bildirməklə readline() metodunu ora yönləndirə bilərsiniz.

```
(seek(0),seek(2))
>>> file.seek(0)
>>>
```

remove() metodu.

Os modulunun içində yer alan metodlardandır.listlərdə buna dair misallar yazmışdıq və nə məna ifadə etdiyi bizə məlumdur.Burda da faylları silməyə yardımçımız olacaq.

```
>>> import os
>>> os.remove('new.txt')
>>>
```

və ya faylın ünvanın bildirərək də silə bilərsiniz.(məsələn.
os.remove(‘/home/user/fayl_adi.txt’)

del metodu
del delete sözünün qısaltmasıdır.Dilimizə silmək kimi tərcümə olunur.fayl içindəki ifadələri,sətirləri silkmək üçün bu metoddan yararlanacağıq.

String modulu,cinsi və metodları

```
>>> dir(str)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__',
'__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__gt__',
'__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__',
'_formatter_field_name_split', '_formatter_parser', 'capitalize', 'center', 'count',
'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index', 'isalnum',
'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
>>>
```

Yuxarıdakı list içindəki istifadə edə biləcəyimiz metodları aşağıda göstərək.

'capitalize', 'center', 'count', 'decode', 'encode', 'endswith','expandtabs', 'find',

'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

capitalize() metodu
Bu metod vasitəsilə cümlənin ilk hərfini böyüdə bilirik.Bu metodu for() operatoru vasitəsilə istifadə edərsənizsə y-oxu üzərində bütün hərflər böyüyəcəkdir.Aşağıdakı misallarla bunu daha aydın izah edək.

```
>>> 'almaniya'.capitalize()
'Almaniya'
>>> i='almaniya'
>>> i.capitalize()
'Almaniya'
>>> for i in 'almaniya':
        print i.capitalize()


A
L
M
A
N
I
Y
A
>>>
```

```
>>> tuple='random','randit','exception','elevator'
>>> tuple.capitalize()

Traceback (most recent call last):
  File "<pyshell#26>", line 1, in <module>
    tuple.capitalize()
AttributeError: 'tuple' object has no attribute 'capitalize'
```

Yuxarıda gördüyünüz kimi tuplarda xəta verdi.Əgər bir for() operatoru vasitəsilə bir string cinsinə atıb çap etsək,

```
>>> for i in tuple:
        print i.capitalize()


Random
Randit
Exception
Elevator
>>>
```

nəticəni alarıq.

upper() metodu

Bu metod öz növbəsində klaviaturadakı CapsLk(capslock) düyməsinin eynisidir.Yəni eyni rol oynayır.İfadənin və ya cümlənin bütün hərflərini böyüdür. Misallara baxaq

```
>>> tuple='Azerbaijan','Marine','Academy'
>>> for i in tuple:
        print i.upper()


AZERBAIJAN
MARINE
ACADEMY
>>>
```

```
>>> 'azerbaijan'.upper()
'AZERBAIJAN'
>>> a='allegator'
>>> a.upper()
'ALLEGATOR'
>>> list=['elementary','second','hacker','seaman']
>>> for i in list:
        print i.upper()
```

```
ELEMENTARY
SECOND
HACKER
SEAMAN
>>> dict={'fruit':'cherry','погода':'wather'}
>>>

>>> dict.values()
['cherry', 'wather']
>>> for i in dict.values():
        print i.upper()


CHERRY
WATHER
>>>
```

lower() metodu

Bu metod öz növbəsində upper() metodunun tam tərsinə olaraq,böyük hərflərdən ibarət cümlənin bütün böyük hərflərini kiçildir.

```
>>> a='ELEGANT'
>>> a.lower()
'elegant'
>>> 'ELEGANT'.lower()
'elegant'
>>> tuple='Vasif','Eldar','Nadir','elena','vladimr'
>>> for i in tuple:
        print i.upper()


VASIF
ELDAR
NADIR
ELENA
VLADIMR
>>>
>>> for i in tuple:
        print i.lower()
```

```
vasif
eldar
nadir
elena
vladimr
>>>
```

Qeyd edimki string metodları argument alma imkanına malik deyil.Buna misal olaraq aşağıdakı

```
>>> i.lower('Eldar')

Traceback (most recent call last):
  File "<pyshell#68>", line 1, in <module>
    i.lower('Eldar')
TypeError: lower() takes no arguments (1 given)
>>>
```

kimi xətanı alacaqsınız.Xətada deyilirki lower() argument almayır,siz 1 argument daxil etmisiniz.

Swapcase metodu

Bu metod ifadə və ya cümlə daxilində hərflərin böyük və ya kiçik olmasını təyin edərək,tərsinə hərəkət edir.Yəni cümlə içində bütün hərflər əgər böyükdürsə-onları kiçildəcək,əgər bütün hərflər kiçikdirsə onları böyüdəcək.Misallara baxaq

```
>>> a='float'
>>> a.swapcase()
'FLOAT'
>>> a='agNarom'
>>> a.swapcase()
'AGnAROM'
>>> a='fliGHT'
>>> a.swapcase()
'FLIght'
>>>
```

Bu metod üçün fərq etmirki əgər cümlədə hərflərin bəzisi böyük ya kiçik olmasına

rəğmən-böyükləri kiçildəcək,kiçikləri isə böyüdəcəkdir.

Bir öncəki bəhslərdəki metodları list tuple və dict -lərə tətbiq etdiyimiz kimi bu metoduda tətbiq edə bilərik.Siz buna dair misallar yazaraq əlinizi öyrəşdirin.

title() metodu

Bu metod capitalize() metoduna yaxın iş yerinə yetirir.Sadəcə olaraq bu metod cümlələrin hər birinin ilk hərfini böyüdür.Misallardan daha aydın olacaq.

```
>>> a='avrora','elementary','snapshot'
>>> for i in a:
        print i.title()


Avrora
Elementary
Snapshot
>>> 'vanadium'.title()
'Vanadium'
>>> title('albert')

Traceback (most recent call last):
  File "<pyshell#80>", line 1, in <module>
    title('albert')
NameError: name 'title' is not defined
>>>
```

center() metodu

Bu metod dilimizə mərkəz kimi tərcümə olunur.metodun rolu bir ifadənin əvvəl və sonuna argument və ya boşluq əlavə etməkdən ibarətdir.Bu metod argument alır.

```
>>> 'akkra'.center(34)
'             akkra               '
>>>
>>> 'akkra'.center(4,'*')
'akkra'
>>> 'a'.center(4,'*')
```

```
'*a**'
>>> 'a'.center(3,'*')
'*a*'
>>>
```

ljust() metodu

Bu metod ifadəni sol kənar başlığında tutaraq sağa boşluq və ya işarələr əlavə edə bilər.

```
>>> 'a'.center(4,'*')
'*a**'
>>> 'a'.center(3,'*')
'*a*'
>>> 'a'.ljust(24)
'a’                      '
>>> 'a'.ljust(4,'*')
'a***'
>>> 'a'.ljust(5,'*')
'a****'
>>> 'a'.ljust(3,'#')
'a##'
>>>
```

rjust() metodu

Bu metod isə ifadəni sağ son kənara sıxışdırar və sol tərəfə boşluq və ya ifadə əlavə edər.

```
>>> 'a'.rjust(12)
'           a'
>>> 'a'.rjust(3,'*')
'**a'
>>>
```

zfill() metodu

Bu metod öz növbəsində sağa sıxışdırılar və soluna 0-ədədi əlavə oluna bilər.

```
>>> 'a'.zfill(3)
'00a'
>>> '4'.zfill(4)
'0004'
>>>
```

replace() metodu

Dilimizə əvəz etmək kimi tərcümə olunur.Metodun rolu,ifadə içindəki hərfləri silib-yazmadan başqa hərflərlə ifadə etməkdir.

```
>>> string='Azerbaijan Maritime Administration'
>>> print string.replace('a','ə')
Azerbəijən Məritime Administrətion
>>>
>>> a='flanga voter'
>>> print a.replace('a','')
flng voter
>>>
```

Yuxarıdakı kodlarda a-nı boşluqla əvəz etdik,bu bir növ silmək kimi də başa düşülür.

startswith() metodu

Bu metod vasitəsilə bir ifadənin ilk hərfi göstərdiyimiz hərflə başlayıb-başlayamadığını təyin edir.

```
>>> a='Azerbaycan'
>>> print a.startswith('a')
False
>>> print a.startswith('A')
True
>>>
```

Gördüyünüz kimi bu metod ilk kodlarda kiçik hərf olmadığı üçün False(səhv) verdi,ikinci dəfə böyük A soruşaraq,bizə True verdi.Bu metod tam dəqiqliklə təyin edir.

endswith() metodu

Adından da bəlli olduğu kimi end-yəni son,startswith() metodunun tam əksinə hansı hərflə bitdiyini təyin edir.

```
>>> i='excelent'
>>> i.endswith('t')
True
>>> i.endswith('v')
False
>>>
```

count() metodu

Bu metod ifadənin içində bir hərfdən neçə dənə olduğunu təyin edir.

```
>>> a='qarğabazarı'
>>> a.count('a')
4
>>> a.count('ğ')
1
>>>
```

isalpha() metodu

Bu metod vasitəsilə bir ifadənin alfabe sinfinə aid olub-olmadığını təyin edə bilərik.

```
>>> 'academy'.isalpha()
True
>>> 'academy12a'.isalpha()
False
>>>
```

isdigit()   metodu

Bu metod vasitəsilə isə ifadənin rəqəmsal olub-olmadığını təyin olunur.

```
>>> i='121243'
>>> i.isdigit()
True
>>> i='Alasker'
>>> i.isdigit()
False
>>>
>>> a.isdigit()
False
>>>
```

isalnum()   metodu

Bu metod vasitəsilə,ifadənin həm alfabe,həm də rəqəmsal olub-olmadığını təyin edəcəyik.

```
>>> i='ty780u'
>>> i.isalnum()
True
>>> i='5674'
>>> i.isalnum()
True
>>> i='/\asd'
>>> i.isalnum()
False
>>>
```

islower() metodu

Bu metod vasitəsilə ifadənin bütünlüklə kiçik hərflərdən ibarət olduğunu təyin edəcik.

```
>>> eko='Philadelphiya'
>>> eko.islower()
False
>>> eko='isaak'
>>> eko.islower()
True
>>> eko='ELEMENTAR'
>>> eko.islower()
False
>>>
```

isupper() metodu

Bu metod islower() metodunun tərsi olaraq,ifadənin tamam böyük hərflərdən olduğunu təyin edir.

```
>>> a='elitar'
>>> a.isupper()
False
>>> a='ECONOMY'
>>> a.isupper()
True
>>> a='ECOnomy'
>>> a.isupper()
False
>>>
```

istitle() metodu

Bu metodla bir ifadənin ilk hərflərinin böyük olub-olmadığı təyin olunur.

```
>>> 'mcdonald'.istitle()
False
>>> 'Mcdonald'.istitle()
True
>>>
```

isspace() metodu

Bu metod vasitəsilə ifadənin boşluqlardan təşkil olunub-olunmadığı təyin olunur.

```
>>> '  '.isspace()
True
>>> '  rt'.isspace()
False
>>>
```

expandtabs() metodu

Bu metod vasitəsilə ifadə içindəki sözlərə ara boşluqları əlavə etmək olur.

```
>>> 'world\tship\twar'.expandtabs(10)
'world     ship      war'
```

```
>>>
```

find() metodu

Bu metod vasitəsilə ifadə daxilində argumentin neçənci sırada olduğu təyin olunur.

```
>>> 'Azerbaijan'.find('b')
4
>>> i='Albaniya'
>>> i.find('n')
4
>>> list=['avropa','america','asiya']
>>> for i in list:
        print i.find('america')


-1
0
-1
>>>
```

rfind () metodu

Bu metod find () metoduyla demək olarki eyni işi görür,fərq isə rfind() metodu ifadəni sağdan-sola oxuyur.

```
>>> i='diskusiya'
>>> i.rfind('s')
5
>>> i.rfind('i')
6
>>>
```

index() metodu

index() metodu da find() və rfind() metoduyla eyni işi yerinə yetirir.

```
>>> i.index('a')
8
>>> i.index('d')
0
>>>
```

rindex() metodu

Bu metod növü də index() metodu ilə eyni funksiyanı yerinə yetirir,fəqət fərq rindex() metodunu ifadəni sağdan sola oxumasındadır.

```
>>> i.rindex('i')
6
>>> i.rindex('k')
3
>>>
```

join() metodu

join() metodu vasitəsilə ifadə daxilinə istədiyiniz arqument əlavə edə bilərsiniz.

```
>>> '.'.join(a)
'd.e.v.e.l.o.p.e.r'
>>>
```

yuxarıda qeydimizə diqqət edin,öncə daxil edəcəyimiz argumenti yazdıq sonra bu arqumenti daxil edəcəyimiz ifadəni mötərizə içərisində dırnaqsız qeyd etdik.

```
>>> '&'.join(a)
'd&e&v&e&l&o&p&e&r'
>>>  list=['Baki','Sumqayit','Sabuncu']
>>> ','.join(list)
'Baki,Sumqayit,Sabuncu'
>>>
```

translate() metodu

Bu metod pythonda çox qarışıq metodlardan biridir.Eləcədə bu metoda şifrələmə metodu da deyə bilərik.Aşağıdakı misallardan aydın olacaq

```
>>> from string import maketrans
>>> alfabe='asdfg'
>>> codin='12345'
>>> netice=maketrans(alfabe,codin)
>>> str='Salam Azerbaycan,sabahiniz xeyir!'
>>> print str.translate(netice)
S1l1m Azerb1yc1n,21b1hiniz xeyir!
>>>
```

İlk öncə biz string modulundan maketrans funksiyasını çağırdıq.Daha sonra alfabe-yə 5 hərf daxil etdik və sonra ilk hərfdən
a – 1
s --2
d –3
f –4
g –5

qeyd edərək,Salam azerbaycan,sabahiniz xeyir ifadəsini yazdıq.Və çap etdikdə yuxarıda qeyd etdiyimiz kimi a hərfinin yerinə -1 ,s hərfinin yerinə -2 və s nəticəni aldıq.Bu metodda hərfləri başqa hərflərlə də əvəz edə bilərsiz,yəni azərbaycan alfabsindəki e-hərfi yerinə ə-hərfini və sairə.

partition() metodu

Bu metod ifadəni bir neçə hissəyə bölür.Misallara baxaq.

```
>>> i.partition('lo')
('deve', 'lo', 'per')
>>> 'fundamental'.partition('un')
('f', 'un', 'damental')
```

```
>>>
```

rpartition() metodu

Bu metod partition() metodu ilə eyni işi görür,fərq isə rpartition() metodu ifadəni sağdan-sola oxuyur.

```
>>> i.rpartition('an')
('Salam', 'an', 'dra')
>>>
```

strip () metodu

bu metod ifadə daxilindəki istənilən tərəfdə olan(baş və son,sol və sağ tərəf) boşluqları silir.

```
>>> '   dfhfhdvcb[]]  '.strip()
'dfhfhdvcb[]]'
>>> ' sdfdsf  ]][['.strip()
'sdfdsf  ]][['
>>>
```

rstrip () metodu

Bu metod isə ifadənin sadəcə sonunda(sağ tərəfdə) gələn boşluqları silər.strip metodunun əvvəlinə artırılan r- hərfi right,yəni sağ tərəf mənasını ifadə edir.

```
>>> '   aadaakddf   '.rstrip()
'   aadaakddf'
>>> '  ere'.rstrip()
'  ere'
```

```
>>> 'fgf   '.rstrip()
'fgf'
>>>
```

lstrip () metodu

Bu metod vasitəsilə ifadənin sadəcə baş(sol) tərəfində olan boşluqlar silinir. Əvvəlində olan l-hərfi left,yəni sol mənasını ifadə edir.

```
>>> '  django '.lstrip()
'django '
>>>
```

splitlines () metodu

Bu metod vasitəsilə ifadə daxilindəki söz birləşmələrini ayırmaq,eləcədə bunları bir list şəklində göstərmək olur.

```
>>> a='Azerbaijan State marine academy\nUSA EMbassy'
>>> a.splitlines()
['Azerbaijan State marine academy', 'USA EMbassy']
>>>
```

split () metodu

Bu metot vasitəsilə ifadələr list halına çevirmək olar

```
>>> a='blue,red,green'
>>> a.split(',')
['blue', 'red', 'green']
>>>
```

metoda daha bir argument əlavə edib,ayrı-ayrı çap etmək olar

```
>>> b=a.split(',',1)
>>> print b
['blue', 'red,green']
>>> b[0]
```

```
'blue'
>>> b[1]
'red,green'
>>>
```

bu metod həm də ifadə içində kəlimə axtarışına da çıxa bilir.

```
>>> i='fashion tv on katv'
>>> i.split()[0]=='fashion'
True
>>>
```

Və bizə True ifadəsini verdi,yəni fashion sözü ifadədə 0-ci yerdə başda yer alır.

rsplit () metodu

Bu metod split() metodu ilə eyni işi görür,sadəcə ifadəni sağdan-sola(r-hərfi right yəni sağ mənasını ifadə etdiyi üçün) oxuyur.

```
>>> 'Azerbaijan.marine.academy'.rsplit('.')
['Azerbaijan', 'marine', 'academy']
>>>
```

re modulu

Bu modul da əvvəlki modullar kimi import re metodu ilə çağırılır,eləcədə daxilində bir neçə funksiya və metod daşıyır.

Terminalı açın python-u çağırın və ya bir idle açın

```
>>> import re
>>>
```

daha sonra

```
>>> dir(re)
```

['DEBUG', 'DOTALL', 'I', 'IGNORECASE', 'L', 'LOCALE', 'M', 'MULTILINE', 'S', 'Scanner', 'T', 'TEMPLATE', 'U', 'UNICODE', 'VERBOSE', 'X', '_MAXCACHE', '__all__', '__builtins__', '__doc__', '__file__', '__name__', '__package__', '__version__', '_alphanum', '_cache', '_cache_repl', '_compile', '_compile_repl', '_expand', '_pattern_type', '_pickle', '_subx', 'compile', 'copy_reg', 'error', 'escape', 'findall', 'finditer', 'match', 'purge', 'search', 'split', 'sre_compile', 'sre_parse', 'sub', 'subn', 'sys', 'template']

və ya

```
>>> for i in dir(re):
        if '_' not in i:
                print i
```

```
DEBUG
DOTALL
I
IGNORECASE
L
LOCALE
M
MULTILINE
S
Scanner
T
TEMPLATE
U
UNICODE
VERBOSE
X
compile
error
escape
findall
finditer
match
purge
search
split
sub
subn
sys
```

```
template
>>>
```

match() metodu

Bu metod vasitəsilə bir ifadə daxilində bir sözün olub-olmadığını təyin edə bilərik.Aşağıdakı misaldan bu metodun nə olduğu daha aydın olacaq.

```
>>> import re
>>> i='New version is available'
>>> re.match('New',i)
<_sre.SRE_Match object at 0x7fbd9b508ac0>
>>>
```

yuxarıda gördüyümüz kimi <_sre.SRE_Match object at 0x7fbd9b508ac0> görüntüsü axtardığımız sözün varlığının olmağına işarədir.

Əgər siz

```
>>> i='Azərbaycan ədəbiyyatında Məhəmmədhüseyn Şəhriyarın böyük rolu var'
>>> re.match('böyük',i)
>>> i='New version is available'
>>> re.match('is',i)
```

İfadə daxilində ikinci və ya son sözləri axtarsaq bir başa alt sətrə keçəcək.Bu o deməkdirki bu söz yoxdur.Doğrusu axtarış verdiyimiz is kəliməsi ifadəmizdə var,sadəcə match() metodu yalnız ifadənin ilk sətrinə-ilk sözə nəzər yetirir. Əgər print funksiyasından istifadə edərək çap etdirsək ekrana None yazılacaq

```
>>> print re.match('is',i)
None
>>>
```

Yuxarıda yazdığımız kodların ekran görüntüsündə axtardığımız söz ekrana çap olunmur.bunun üçün group() metodu təklif olunur.

```
>>> import re
>>> i='discovery channel Azerbaijan'
>>> x=re.match('discovery',i)
>>> print x.group()
discovery
>>>
>>> type(x)
<type '_sre.SRE_Match'>
>>>
```

search () metodu

Bu metod match() metodundan fərqlənir.Adından da məlum olduğu kimi,ingiliscədən axtarmaq mənasını ifadə edir.match() metodu ifadənin sadəcə ilk sətrinə baxırdısa,search() metodu bütünlüklə ifadə daxilində axtarma edir.

```
>>> i='elementar sistem programlama'
>>> x=re.search('sistem',i)
>>> print x.group()
sistem
>>>
```

search() metodundan listlər içində də axtarışlar etmək mümkündür.

```
>>> import re
>>>list=['texnologiya','kitab','ixtiralar','rubby','feride','nigar']
>>> for i in list:
        x=re.search('feride',i)
        if x:
                print x.group()
```

```
feride
>>>
```

başqa bir misal

```
>>> import re
>>> import urllib
>>> i=urllib.urlopen('http://www.unibank.az')
>>> for x in i.readlines():
        v=re.search('money',x)
        if v:
                print v.group()


money
>>>
```

Gördüyünüz kimi money sözü unibank.az saytında yalnız bir dəfə istifadə olunub.

Və ya axtaracağımız kəliməni, istifadəçidən alaq

```
# -*- coding: utf-8 -*-
import re
import urllib


while True:
        word=str(raw_input('write word :'))
        i=urllib.urlopen('http://www.python.org')
        for url in i.readlines():
                x=re.search(word,url)
                if x:
                    print x.group()
```

findall () metodu

Bu metod search() metoduna bənzəyir,yəni eyni işi görür amma findall() metodu kəliməni tapdığında ekrana həmin kəliməni list şəklində çap edir.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import re
text='Python is a superb language for in teaching programming,\
 both at the introductory level and in more advanced courses.'
print re.findall('in',text)

>>>
['in', 'in', 'in', 'in', 'in']
>>>
```

group () və groups () metodları

group () metodu

Bu metod öz növbəsində çap etmə funksiyası kimi özünü biruzə verir.Aşağıda

yazacağımız kodlardan bunu daha aydın başa düşəcik

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
text='Being a very high level language,\
 Python reads like English, which takes a lot of syntax-learning stress off coding beginners'
i=re.search('level',text)
print i.group()
```

```
level
```

Yuxarıdakı mətndən level sözünü axtarışa verdik və ekrana print i.group() funksiyası ilə çap etdirdik.əgər biz print I ifadəsi istifadə etsəydik,python bizə yenə də o sözün(level) olduğunu ekrana çap edəcəkdir amma qarşılığında <_sre.SRE_Match object at 0x7f27c45722a0> ifadə ilə rastlaşacaqdıq.i.group() metodundan biz ona görə istifadə etdikki axtardığımız sözü ifadədə tapdıqdan sonra ekrana dolğun çap etsin.
<_sre.SRE_Match object at 0x7f27c45722a0> ifadəsi xəta ifadəsi deyil,yəni axtardığımız sözün varlığına dair məna verir.

group() metodu ədəd argumenti alaraq ifadə daxilində sıra ilə sözləri ekrana çap edəcək

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
text='high level language'
i=re.search('(high) (level) (language)',text)

print i.group()
print i.group(0)
print i.group(1)
print i.group(2)
```

```
>>>
high level language
high level language
```

```
high
level
>>>
```

Və gördüyümüz kimi 0-high 1-high 2-level yerlərində sıra ilə durur və ekrana çap edir.

groups() metodu

Bu metod ifadəni eləcədə ifadə daxilindəki sözləri tupl formasında çap etmə imkanına malikdir.Yuxarıdakı kodlarımızı təkrar yazaraq

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
text='high level language'
i=re.search('(high) (level) (language)',text)

print i.groups()

>>>
('high', 'level', 'language')
```

Və gördüyümüz kimi print i.groups() yazaraq,ekrana ifadələri bir tupl formatında çap etdi.

Xüsusi simbol və xarakterlər (Special Symbols and Characters)

re modulu ilə bərabər işlək hal alan bəzi simbol və xarakterlər var.Bunlardan ilk başlayacağımız qapalı mötərizə([]) işarəsidir.Bu simbolun vəzifəsi ifadə daxilindəki string-i heca formasına bölmə imkanı var.Sözlə ifadə etmək ilk başqan qəliz olsada,gəlin misallar yazaraq bunu aydınlaşdırmağa çalışaq.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import re
list=['john','mohn','leon','julion','jon']
for i in list:
```

```
x=re.search('jo[nh]n',i)
if x:
    print x.group()
```

john

TypeError: expected string or buffer
Yuxarıdakı kodlarda ilk öncə re modulunu çağırdıq(import re) daha sonra list tərtib edib,bir neçə isim daxil etdik.Qapalı mötərizənin işi isə jo[nh]n ifadə jo -ilə başlayıb,ortasında n və ya h hərfi keçən və sonu n -ilə bitən sözləri axtarışa verdik.Və cavabında sadəcə john ifadəsini ala bildik.
Bu simbolun üzərindən ötəri keçirəmki xarakterləri keçdikdən sonra bir daha geniş bəhslə buna qayıdacıq.

Nöqtə işarəsi (.)

bu xarakter qapalı mötərizə xarakterindən fərqli olaraq ifadə daxilində kəlimədən sadəcə tək hərfi (və ya rəqəmi) ayıra bilir.misallara baxaq

belə bir kod yazaq

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['john','mohn','leon','julion','jon']
for i in list:
        if re.match('.o',i):
                print i
```

john
mohn
jon

yuxarıdakı kodlrımızda ilk əvvəl list tərtib edərək bir neçə ifadə daxil etdik.

re.match('.o',i) ifadədə isə əvvəli hansı hərflə başlayır başlasın sonrakı hərfi o ilə başlayan sözləri qarşımıza töksün.Və nəticədə 3 ifadəni ekrana çap etdi.
Başqa bir misala baxaq

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['amanda','adams','amazon','alabama','amalia']
for i in list:
        if re.match('.m',i):
                print i
```

```
amanda
amazon
amalia
```

yuxarıdakı kodlarda isə biz ilk hərfi nə olur olsun ikinci hərfində m- keçən ifadələri bizə göstər əmrini verərək qarşımıza 5 ifadədən sadəcə 3-nü(amanda , amazon ,amalia) ekrana çap etdi.Bu misalları hər hansısa bir ünvana da tətbiq edə bilərik

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
import urllib
url=urllib.urlopen('https://www.facebook.com')
for i in url.readlines():
        if re.search('photo',i):
                print i
```

yuxarıdakı link yerinə bir şəxsin id-nömrəsiylə bərabər link yazıb,səhifəsindəki bütün şəkilləri ekrana tökə bilərsiniz.
search() metodu ilə yanaşı findall () metodundan da istifadə edə bilərsiniz

Ulduz işarəsi (*)

Bu metod ifadə daxilində bir hərf və ya çox sayda olmasına rəğmən tapıb ekrana çap etmə imkanına malikdir.Daha doğrusu tapmağda yardımçı olan işarədir,çap

etməkdə print funksiyası rol oynayır

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['eal','eaal','eaaal','ealata','eafrodita']
for i in list:
        if re.match('ea*l',i):
                print i
```

```
eal
eaal
eaaal
ealata
```

Yuxarıdakı kodlarımızda ulduz işarəsi özündən əvvəl gələn ea hərflərini qarşılaşdırır və sonra l-hərfi keçən sözü ekrana çap edir.Bütövlükdə daxilində ea və l hərfi olan ifadələr çap olundu.
Indidə gəlin keçdiyimiz işarələri bir yerdə istifadə edək

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['eal','eaal','eaaal','ealata','eafrodita']
for i in list:
        if re.match('e.*f',i):
                print i
```

```
eafrodita
```

yuxarıdakı kodlarımıza e hərfi ilə başlayıb f ilə bitən ifadələri təyin etdik.nöqtə və ulduz işarəsini bərabər istifadə etdik.Əvvəlin və son hərflərin dəyişərək hansı nəticələr alacğınıza dair kod yazaraq təkrar edin.

Yuxarıda nöqtə işarəsi qeyd etdiyimiz kimi ifadə daxilində nöqtədən əvvəl ona maraqlı olmadığı üçün sonrakı hərf olan ifadə üzərində axtarışa çıxır.
Bunun vasitəsilə sisteminizdə kökdə bəzi şəkillər və musiqilər axtarışına çıxa

bilərsiniz.yuxarıda os modulunu keçərdən içərisində listdir funksiyasından bəhs etmişdik və bilirikki listdir olduğumuz qovluğu göstərir və ya qovluğu manuel yazaraq keçid alma imkanın malikiy(os.listdir('/home/user/Pictures/').
Məsələn

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
import os
qovluq=os.listdir('/home/panda/Pictures/')

for i in qovluq:
        if re.match('.*jpg',i):
                print i
```

```
Hack_The_Planet_by_HATE_LOVE_FEAR_ANGER.jpg
285654_258256810945404_802988727_n.jpg
5033403-python-wallpapers.jpg
color-blue-04.jpg
blue-coloured-wallpapers-wallpaper.jpg
319741_329632687141149_1558238888_n.jpg
Atasun_Optik-selfie.jpg
```

Qeyd edimki yuxarıda user- qovluqu mənim əməliyyat sistemimə aiddir.Sizdə bunun adı fərqli ola bilər.Və gördüyümüz kimi Pictures qovluğunun içindəki şəkilləri ekrana çap etdirdik.jpg formatını dəyişib flv və ya iso,mp3 kimi də qeyd edə bilərsiniz.Bu növ databeyslərə giriş sosial şəbəkələrdə şəkil yüklə və birbaşa sizin jpg olan kök qovluğunuza giriş etməsini misal çəkə bilərəm.Son dövrlərdə bəzi sosial şəbəkələr sizin kök qovluqlarınızı,siz giriş edərkən oxuyub yuxarı hissədə şəkil paylaş kimi scriptlər yazırlar.Başqa dillərdə olduğu kimi pythonla da bir web səhifəsi yazarsanız bu kimi kök qovluqları oxuma yollarından istifadə edə bilərsiniz.

Üstəgəl işarəsi (+)

Bu işarə isə öz növbəsində, (+) işarəsindən sonra axtaracağımız hər hansısa bir sonluğu dəqiq olaraq axtarma imkanına malikdir,eləcədə əvvəlin qeyd etməsəkdə,başlanğıcı nə olur,fərqinə varmadan axtarışını davam etdirəcəkdir.

Misallara baxaq

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['manqan','qamma','alan','tiran','elite']
for i in list:
        if re.search('.+an',i):
                print i
```

ekran görüntüsü

```
manqan
alan
tiran
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['manqan','qamma','alan','tiran','elite']
for i in list:
        if re.search('a.+an',i):
                print i
```

```
manqan
alan
```

Sonuncu kodlarımızda isə a hərfi olan və aradakı hərflərin nə olduğu bizə maraqlı olmayan və sonu an ilə bitən ifadələri axtar əmri verdik.

Və ya search () metodundan istifadə edərək

```
#!/usr/bin/env python
```

```
# -*- coding: utf-8 -*-
import re
list=['manqan','qamma','alan','tiran','elite']
for i in list:
            x=re.search('.+an',i)
            if x:
                print x.group()
```

```
manqan
alan
tiran
```

Sual işarəsi (?)

İlk əvvəl keçdiyimiz xüsusi simbollar içində * və + işarələri ifadə daxilində bir və ya bir neçə sayda hərfləri axtarış edirdisə, sual (?) işarəsi isə yoxluğu və ya bir sayda olduğu hərfləri belə nəzərə alaraq axtarışa keçir.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['fl','ful','fuul','fuuul','flang','fuuuuto']
for i in list:
    if re.match('fu?l',i):
        print i
```

```
fl
ful
flang
>>>
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['fl','ful','fuul','fuuul','flang','fuuuuto']
for i in list:
    if re.match('fu ?l',i):
        print i
```

```
>>>
ful
>>>
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['fl','ful','fuul','fuuul','flang','fuuuuto']
for i in list:
    if re.match('f+?l',i):
        print i
```

```
>>>
fl
flang
>>>
```

Cəm işarəsi ({})

Bu işarə vasitəsilə isə ifadə daxilində bir hərf və ya rəqəmin neçə dənə olduğunu tam sayda qeyd edib axtarışına çıxa bilərik.Aşağıdakı misallardan bu daha aydın olacaq.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['fl','ful','fuul','fuuul','flang','fuuuuto']
for i in list:
    if re.match('fu{3}l',i):
        print i
```

```
>>>
fuuul
>>>
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['fl','ful','fuul','fuuul','flang','fuuuuto']
for i in list:
    if re.match('fu{2}l',i):
        print i

>>>
fuul
>>>
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['fl','ful','fuul','fuuul','flang','fuuuuto']
for i in list:
    if re.match('fu{1}l',i):
        print i


>>>
ful
>>>
```

hətda bu metod iki argument də ala bilir.yəni ən az və ən çox sayını qeyd edib çap axtarışa çıxa bilərik

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['fl','ful','fuul','fuuul','flang','fuuuuto']
for i in list:
```

```
    if re.match('fu{0,3}l',i):
        print i



>>>
fl
ful
fuul
fuuul
flang
fuuuuto
>>>



#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['fl','ful','fuul','fuuul','flang','fuuuuto']
for i in list:
    if re.match('fu{0,1}l',i):
        print i



fl
ful
flang
>>>


və ya

#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['fl','ful','fuul','fuuul','flang','fuuuuto']
for i in list:
    if re.match('fu{0,4}',i):
        print i


fl
```

```
ful
fuul
fuuul
flang
fuuuuto
>>>
```

Yuxarıdakı kodlarda f- ilə başlayan ən az 0 və ya 4-sayda f olan bütün ifadələri çap et komandası verdik.

Sirkumfleks işarəsi (^)

Bu işarə ifadə daxilində sadəcə ən başa baxmaqla araşdırma aparır.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['564343','fgdg454','45gfhfh','fghf34hg','fgdgd','rtyuqw']
for i in list:
    if re.search('[0-9]',i):
        print i


>>>
564343
fgdg454
45gfhfh
fghf34hg
>>>
```

Yuxarıdakı search() metodu ilə list içində rəqəm olan ifadələri axtarışa ver əmri verdik.Biz bilirikki search() metodu ifadəni bütünlüklə oxuyur.Və bəzən biz sadəcə bizə ilk rəqəmlə və ya ilk hərflə başlamayan ifadələri çap etdirmək istəyirik.Bunun üçün yuxarıdakı kodlarımıza kiçik bir dəyişiklik edib,sirkumfleks işarəsini əlavə edək

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['564343','fgdg454','45gfhfh','fghf34hg','fgdgd','rtyuqw']
for i in list:
    if re.search('^[0-9]',i):
        print i



>>>
564343
45gfhfh
>>>
```

Və gördüyümüz kimi biz sadəcə başlanğıcı rəqəmlərdən ibarət olan ifadələri axtarışa verdik.
Yuxarıdakı [0-9] 0 və 0 da daxil olmaqla 9 və doqquzda daxil olmaqla rəqəmlərin təyini üçün qısaldılmış birləşmədir.Digər qısaldılmış əlifba birləşməsi də var ki a-z və ya A-Z kimi yazılır.a-z kiçik hərflərdən ibarət əlifba,A -Z isə bütünü böyük hərflərdən ibarət birləşmədir.Bu hər üç birləşmədən irəlidə tez tez istifadə edəcik.Eləcədə yuxarıda keçdiyimiz xüsusi xarakterlərə də əlavə edə bilərsiniz.

İçində rəqəm və ya rəqəmlər([0-9]),eləcədə böyük hərflərdən([A-Z]) ibarət ifadəni axtar

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['564343','HTFFK56KD','GHDS','fghf34hg','fgdgd','rtyuqw']
for i in list:
    if re.search('[0-9]+[A-Z]',i):
        print i



>>>
```

```
HTFFK56KD
>>>
```

İçində rəqəm və ya rəqəmlər([0-9]),eləcədə kiçik hərflərdən([a-z]) ibarət ifadəni axtar

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['564343','HTFFK56KD','GHDS','fghf34hg','fgdgd','rtyuqw']
for i in list:
    if re.search('[0-9]+[a-z]',i):
        print i


>>>
fghf34hg
>>>
```

Başlanğıcı yalnız kiçik hərflərdən (^[a-z]) ibarət ifadələri axtar.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['564343','HTFFK56KD','GHDS','fghf34hg','fgdgd','rtyuqw']
for i in list:
    if re.search('^[a-z]',i):
        print i



>>>
fghf34hg
fgdgd
rtyuqw
>>>
```

Dollar işarəsi ($)

Bu işarə isə sirkumfleks işarəsinin əksinə olaraq ifadənin sonuna baxır. Aşağıda yazacağımız misallardan daha aydın olacaq.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['string','ing','fringle','fring','digh','dog','distri','sdf565']
for i in list:
    if re.search('ing$',i):
        print i
```

```
>>>
string
ing
fring
>>>
```

Yuxarıdakı kodlarda sadəcə sonu ing -lə bitən ifadələri axtarmaq üçün dollar işarəsini ing -hərf birləşməsinin ön tərəfində yazdıq və bizə həqiqətən nəticədə ing -lə bitən listdən üç ifadəni çap etdi.
Başqa işarələrdə olduğu kimi bu işarəni də digərləri ilə bərabər istifadə etmək hüququmuz var.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['string','ing','fringle','fring','digh','dog','distri','sdf565']
for i in list:
    if re.search('^ing$',i):
        print i
```

```
>>>
ing
```

```
>>>
```

Tərs əyri xətt (\)

Bundan əvvəlki dərslərdə keçdiyimiz $ və s işarələr,xüsusi işarələr sinfinə aid idi.Bəs əgər bir ifadə daxilində pul vahidlərini bildirmək istədikdə nə edəcəyik?! Məsələn

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['12$','10$','30€']
for i in list:
    if re.match('[0-9]',i):
        print i

>>>
12$
10$
30€
>>>
```

Gördüyümüz kimi içində rəqəmlə bitən ifadəni axtar əmri verdik və bizə listdə olan hər üç ifadəni çap etdi.

Əgər

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['12$','10$','30€']
for i in list:
    if re.match('[0-9]+$',i):
        print i


>>>
>>>
```

Və nəticədə heç bir cavab almayacağıq.Cünki yuxarıdakı kodlarda biz rəqəmlə bitən ([0-9]) əmr hissəsini sadəcə oxuyacaq.
Amma bizə lazım olan isə sonu $-la bitən yəni yalnız dollr pul vahidini axtarmağı lazımdır.Və burda tərs əyri xətdən istifadə edəcik (\)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['12$','10$','30€']
for i in list:
    if re.match('[0-9]+\$',i):
        print i


>>>
12$
10$
>>>
```

Və nəticədə tərs əyri xətt köməyimizə gələrək sonu $ olan ifadələri çap etdi.Çünki tərs əyri xətt yuxarıdakı kodda $ işarəsini adi işarə olaraq qeyd etdi

və ya

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['12$','10$','30€']
for i in list:
    if re.match('[0-9]+\€',i):
        print i


>>>
30€
>>>
```

Düz xətt (|)

Bu işarə bizə linuks komandalarından məlumdur.İşarənin rolu bir və ya bir neçə şəkilçiləri,sonluqları ifadədə qarşılaşdırır.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['elidor','freedom','emidor','teodor','dorelfun','dor56fgil','orellio']
for i in list:
    if re.search('or|or',i):
        print i
```

```
elidor
emidor
teodor
dorelfun
dor56fgil
orellio
>>>
```

Yuxarıda bizə ifadə daxilində or birləşməsi keçən ifadələri axtarmağı tələb etdik. Gəlin indidə əvvəli və sonunda or birləşməsi olan ifadələri axtaraq.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['elidor','freedom','emidor','teodor','dorelfun','dor56fgil','orellio']
for i in list:
    if re.search('^or|or$',i):
        print i
```

```
elidor
emidor
teodor
orellio
>>>
```

Və gördüyümüz kimi sonu və əvvəli or -ilə bitən ifadələr çap olundu.düz xətdən

savayı digər işarələr bizə tanışdır.

Mötərizə işarəsi ()

Bu işarə vasitəsilə ekrana çap olan ifadələri qruplaşdırmaq olur.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
import urllib
url='https://www.python.org/doc/'
x=urllib.urlopen(url)
for i in x:
    v=re.search('href="(.+html)">(.+)</a>',i)
    if v:
        print (v.group(1),v.group(2))
```

```
>>>
('https://docs.python.org/3/">Browse Current Python 3 Documentation</a> - <a
href="http://docs.python.org/3/py-modindex.html', '(Module Index)')
('https://docs.python.org/3/whatsnew/index.html', "What's new in Python 3")
('https://docs.python.org/3/tutorial/index.html', 'Tutorial')
('https://docs.python.org/3/library/index.html', 'Library Reference')
('https://docs.python.org/3/reference/index.html', 'Language Reference')
('https://docs.python.org/3/extending/index.html', 'Extending and Embedding')
('https://docs.python.org/3/c-api/index.html', 'Python/C API')
('https://docs.python.org/3/using/index.html', 'Using Python')
('https://docs.python.org/3/howto/index.html', 'Python HOWTOs')
('https://docs.python.org/3/search.html', 'Search the online docs')
('https://docs.python.org/3/download.html', 'Download Current Documentation')
('https://docs.python.org/2/">Browse Current Documentation</a> - <a
href="http:/
>>>
```

Xüsusi simbollar

Bu bölümdə öyrənəcəyimiz simbollar

\s -ifadələr arası boşluq axtarışında

\d -onluq sayı olan ifadələr axtarışında

\w -hərf və rəqəmlərdən ibarət ifadələrdə,eləcədə alt-tire (_)

\S -ifadələrdə boşluq olmayanlarda

\D -onluq sayılar olmayan ifadələrdə [^0-9]

\W -hərf,rəqəm və boşluq olmayan ifadələrdə [^A-Z-a-z0-9_]

istifadə olunur.
Yuxarıdakı simbolların böyük və kiçik olduqlarını yəqinki gördünüz və bunlar bir-birinin əksinə işlər görür.

Gəlin yuxarıdakı simbollar haqqında ətraflı danışaq və bir neçə misallar çəkək.

Xüsusi simbol \s

bu simbol ifadələr içində boşluqları tutur

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['5 Frank','Oktan 7','5elli']
for i in list:
    if i:
        x=re.search(".*\s[A-Za-z]+",i)
        print x


>>>
<_sre.SRE_Match object at 0x7f45e36c1a58>
None
None
>>>
```

Ekran görüntüsündə <_sre.SRE_Match object at 0x7f45e36c1a58>

5 Frank ifadəsinə düz gəlir,None görüntüləri isə yoxdur ifadəsini əvəz edir.

xüsusi simbol \d

Bu simbol ifadə daxilində onluq sayları təyin etməyə yardım edir.Hətda bu simbol daha öncə istifadə etdiyimiz [0-9] qısa simbolun yerini tutur,yəni onu əvəzetməyə qadirdir.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re

list=['467hj','56HG','d767h','dfg57dg','6']
for i in list:
    x=re.search('\d+[a-z]',i)
```

```
    if x:
        print x.group()
```

```
>>>
467h
767h
57d
>>>
```

Gördüyünüz kimi rəqəmlə başlayan və kiçik hərflərdən ibarət ifadələri çap etdirdik.amma bizə rəqəmdən əvvəl gələn və ya hərfdən sonrakı hərfləri göstərmədi
Bunun üçün

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['467hj','56HG','d767h','dfg57dg','6']
for i in list:
    x=re.search('\d+[a-z]+',i)
    if x:
        print x.group()
```

```
>>>
467hj
767h
57dg
>>>
```

hərf birləşmələrindən sonra + işarəsi əlavə etdik.yenədə çatışmayan cəhətlər varki rəqəmlərin əvvəli görsənmədi.

Bunun üçün

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['467hj','56HG','d767h','dfg57dg','6']
```

```
for i in list:
    x=re.search('.*\d+[a-z]+',i)
    if x:
        print x.group()


>>>
467hj
d767h
dfg57dg
>>>
```

Və nöqtə,ulduz işarəsi əlavə etdikdə bütünlüklə ifadələri çap edə bildik.
\d yerinə [0-9] yazıb yoxlayaq.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
list=['467hj','56HG','d767h','dfg57dg','6']
for i in list:
    x=re.search('.*[0-9]+[a-z]+',i)
    if x:
        print x.group()


>>>
467hj
d767h
dfg57dg
>>>
```

gördüyümüz kimi \d simbolu [0-9] simboluna qarşılıq simboldur və onu əvəzetməyə qadirdir.

Yuxarıda qeyd etdiyimiz əlifba simbolunu [a-z] ,hətda böyük hərflərdən ibarət bir simbol toplusu yaza bilərik. [a-zA-Z] və ya [A-Za-z]

hərf və rəqəmlərdən ibarət xüsusi simbol   \w

Bu simbol , [A-Za-z0-9_] əvəzetməyə qadirdir.

Simbola dair misallar baxaq

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
v='a23','4%6_','gf67','fghf','DFGDFDG','10J-4'
for i in v:
    if i:
        x=re.search('.*\w+',i)
        print x.group()
```

```
>>>
a23
4%6_
gf67
fghf
DFGDFDG
10J-4
>>>
```

Siniflər (class)

Bu metod pythonda əhəmiyyətli metodlardan biridir.Yadınızdadırsa biz əvvəl def ifadəsi keçmişdik,bu metod da def ifadəsinə bənzərdir.İrəlidə grafik programalarda class metodundan davamlı istifadə edəcik.Siz bu metoda internetdən yüklədiyiniz python programları içində davamlı rast gəlirsiniz.Bu metod bizə kodlarımızın aydın və başa düşülən tipə çevirir.

İlk sinifmiz

class start()

biz bu ifadəyə yaxın def ifadəsi keçmişdik.Bu da eyni qayda ilə ilk başda qeyd edirdik və sonra kodlarımız,daha sonra def ifadəsi ilə qeydlədiyimiz adı sonda kodlarımızı bağlayırdıq.

def start()

start()

Yuxarıdakı sinifmizə start () adı verdik.Bu sabit ifadə deyil,yəni siz istənilən ad verə bilərsiniz.

class ifadə_adı()

sinif ifadələrində eləcədə def ifadəsində Azərbaycan əlifbasından istifadə etməyimizi hər birimizə agahdır.
Gəlin bir mətn faylına

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
class start():
    list=['Azerbaijan','Elidor','Indonesiya']
```

yazıb,adını nə qoyursuz qoyun yaddaşa verib bağlayaq.Daha sonra pythonu açaraq(terminaldan python və ya bir python idle aça bilərsiniz) aşağıdakı komanda sətrinə from fr import* yazaq.

```
>>> from fr import*
>>>
```

Yuxarıdakı fr.py python kodlarımızı yazdığım fayldır.Və alt sətrə boş keçid etdi.Yəni kodlarımız içində yazdığım list-i ekrana çap etmədi.Nəyinsə natamam olduğuna əminik.
Biz start() adı ilə sinifmizi qeyd etdik,məsələ bu sinifin tamamlanılmasına gəldikdə isə a=start() ifadəsini sona yazaraq python əmr sətrindən programımızı bir daha açaq

```
a=start()
```

Bütünlükdə kodlarımız

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
class start():
    list=['Azerbaijan','Elidor','Indonesiya']
a=start()
```

Yuxarıdakı kodlarımız tamamlandı.Və print list ifadəsi verib pythondan listlərimizi görə bilərik

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
class start():
    list=['Azerbaijan','Elidor','Indonesiya']
    print list
a=start()
```

daha sonra bunu fr.py olaraq yaddaşa verdim və ardından python əmr sətrindən yazaraq

```
>>> from fr import*
['Azerbaijan', 'Elidor', 'Indonesiya']
>>>
```

ilk sinif programımızı hazır şəklə gətirdik.

Yuxarıda yazdığımız yeniliklər yoxdu,çünki buna bənzər def ifadəsini istifadə etmişik.

```
def __init__(self):
```

siniflərlə daima yanaşı işlənən ifadədir.self ifadəsi isə ən vacib ifadələrdən biridir.

```
#!/usr/bin/env python
```

```
# -*- coding: utf-8 -*-
class start():
    def __init__(self):
        list=[]

        a=raw_input('bir ad yazın:')
        list.append(a)
        print list

a=start()


bir ad yazın:algeria
['algeria']
>>>
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
class start():
    def __init__(self):
        list=[]

    def yaz(self):
        a=raw_input('bir ad yazın:')
        list.append(self.a)
        print list

a=start()
```

Və bu kolarımızı çalışdırdıqda heç bir nəticə əldə etmədik.Və bu anda yuxarıda bəhs etdiyimiz self ifadəsi köməyimizə gələcək.

Bütünlüklə kodlarımız

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python

class start():
    def __init__(self):
```

```
        self.list=[]
        self.yaz()
    def yaz(self):
        self.a=raw_input('bir ad yazın:')
        self.list.append(self.a)
        print self.list

a=start()
```

```
>>>
bir ad yazın:start
['start']
>>>
```

ilk öncə def ifadəsinə argument olaraq yaz() verdik.və bu yaz() ifadəsini self.yaz() olaraq def __init__(self): parçasında qeyd etdik.Çünki biz self.yaz() olaraq yazmasaydıq def yaz() ifadə kodları program daxilində çalışmayacaqdır.Biz def ifadəsi bölümündə keçdiyimiz kimi kodlarımız bitdikdən sonra def ifadəsinə atdığımız hər hansısa bir sözü sonda bağlamalı olurduq,əks halda program çalışmayacqdı

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
def start():
    print a
start()
```

class metodunun içində isə def yaz() ifadəsini yaz() ifadəsiylə bağlamağa gərəy yoxdur.

İndi isə gəlin bu fr.py faylımızı python komanda əmrindən çalışdıraq.
Yuxarıda from fr import* verdiyimiz əmrdə * -işarəsinin nə olduğuna baxaq.

Bu * - işarəsi o mənanı ifadə edirki,fr.py faylın içində sonda bağladığımız class metodunun a=start() program kodunu çalışdırmaq

```
>>> from fr import start
bir ad yazın:asddf
```

['asddf']
>>>

Gördüyümüz kimi python komanda əmrindən from fr import start metodunu istifadə edərək programı rahatlıqla terminaldan çalışdırdıq.Və ya python fr.py əmrini terminaldan verərsəniz programınız rahatlıqla çalışacaq.Amma fr.py faylı pythonu açdığınız hər hansısa bir qovluqda mütləq olmalıdır,daha alternativ variantı isə python -dan sonra fr.py faylını mousla sürüşdürərək terminala daxil edin və çalışdırın.Qovluqlar üzərində keçid metodlarını düşünürəm təkrar etməyə əsas yoxdur.os modulunu keçəndə biz bundan ətraflı danışmışdıq.

ascii

```
>>> import sys
>>> sys.getdefaultencoding()
'ascii'
```

ascii kodlaması haqqında http://www.asciitable.com/ ünvanından daha ətraflı məlumt ala bilərsiniz.Biz də bacardığımız qədər bu kodlama tipindən ətraflı danışacıq.

Ascii kodlaması latın əlifbasında standart hərflərdən başqa,başqa dillərdəki hərfləri tanımır.əlifbamızda yer alan ç,ş,ğ,ə və sairə kimi

ascii kodlamasından başqa unicode kodlama tipidə var.Bu kodlama nəticəsində dilimizdə olan yuxarıdakı hərfləri python program kodlarında rahatlıqla istifadə edə bilirik.
Bu kodlamaya (ascii)xarakter kodlama olaraq ingiliscədən (character encoding) adı verilir.
Bu kodlama bir növ morze əlifbası kimi başa düşək.Kompyuterdə yazdığımız hər bir hərflərin bir rəqəm qarşılığı var.Və hər dəfə klaviaturada basdığımız hər bir hərfi impul olaraq mikrokontroller rəqəm kimi tanıyıb onu ekrana çap edir.

Ascii kodlamanın xüsusi tabeli varki yuxarıda ilk başda qeyd etdiyim ünvandan oxuya bilərsiniz.1963-cü ildə ascii-7bitlik xarakter toplusu çıxdı.
Ascii kodlamasının rəsmi ünvanına daxil olub oradan bir neçə rəqəmlərə qarşılıq olanları yazmağa çalışaq.

```
>>> print 'salam'+chr(10)+'eldar'
salam
eldar
>>>
```

Tabeldən gördüyümüz kimi 10-rəqəminə qarşılıq,yeni sətir ifadəsi var.
Əgər

```
>>> print 'salam'+'eldar'
salameldar
>>>
```

yazsaydıq,aşağıdakı nəticəni əldə edəcəkdik.

```
salameldar
>>>
```

eləcədə cədvəldə olan 133-sayına [-uyğun gəldiyi üçün,

```
>>> print 'ex\133'
ex[
>>>
```

nəticəsini alacağıq.

Unicode

pythonda unikod dəstəyi verən,hər dəfə sətir başına `# -*- coding: utf-8 -*-` əlavə etməklə bu proglemdən yayına bilərsiniz.

Windows üçün `# -*- coding: cp1254 -*-`

format() metodu

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python

def format():
    print '{} və {} proqramlama dilləridir'.format('python','perl')
format()
```

```
>>>
python və perl proqramlama dilləridir
>>>
```

eləcədə yer təyin etmək üçün

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python

def format():
    print '{1} və {0} proqramlama dilləridir'.format('python','perl')
format()
```

```
>>>
perl və python proqramlama dilləridir
>>>
```

ilk yazdığımız python-0 a perl isə 1-ci yerdə durur və bu minvalla biz perl I önə python sözünü sonraya yerləşdirdik.

math () modulu

```
>>> import math
>>> dir(math)
['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2',
'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1',
'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isinf', 'isnan',
'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh',
'sqrt', 'tan', 'tanh', 'trunc']
```

və ya

```
>>> for i in dir(math):
        if '_' not in i:
                print I
```

modulumuzun içində yer alan pow metoduna baxaq.İngiliscədən tərcümədə üst,qüvvət mənasını ifadə edirki power sözünün qısaltmasıdır.

```
>>> math.pow(3,3)
27.0
>>>
```

pi ədədi

modul içində pi metoduna da rast gəlirik.pi ədədinin nə olduğu düşünürəm hamımıza mlumdur.standart hesablanmış pi ədədinin qiyməti, pi=3.1415926535897931 bərabərdir.

```
>>> math.pi
3.141592653589793
>>>
```

sqrt () kök çıxartma

```
>>> math.sqrt(27)
5.196152422706632
```

```
>>> math.sqrt(81)
9.0
>>>
```

Eyler sabiti (e)

e=2.718281828459045

```
>>> math.e
2.718281828459045
>>> math.sqrt(81)*math.e
24.464536456131405
>>>
```

exp ()

exp Eyler sabitini qüvvətə yüksəldən metoddur.
Məsələn

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
import math
def format():
    quvvet=input('write ineger:')
    print 'you writed:',quvvet;print math.exp(quvvet)
format()
```

```
>>>
write ineger:2
you writed: 2
7.38905609893
>>>
```

logarifma (log)

log(x,y)

```
>>> math.log(3,3)
1.0
>>>
```

degrees()

ədədi radian cinsinə çevirir

```
>>> math.degrees(2)
114.59155902616465
>>>
```

radians()

```
>>> math.radians(2)
0.03490658503988659
>>>
```

kosinus (cos)

math.radians() metodu ilə bərabər doğru cavab verir.

```
>>> math.cos(math.radians(60))
0.5000000000000001
>>>
```

cosinus cədvəlinə aşağıdakı ünvandan məlumat ala bilərsiniz

https://ro.wikipedia.org/wiki/Tabelul_valorilor_func%C8%9Biilor_sinus_%C8%99i_cosinus

Sinus (sin)

eyni qayda ilə radians metodu ilə bərabər istifadə olunur.

```
>>> math.sin(math.radians(30))
0.49999999999999994
>>>
```

tangels (tan)

```
>>> math.tan(math.radians(60))
1.7320508075688767
>>>
```

Gui(Qrafik programlama)
tkinter modulu

pythonda bir neçə qrafik programa modulları vardır.
PyGTK
PyQt
wxPython
Tkinter

Bunların içində biz bəhs edəcəyimiz Tkinter olacaq.Tkinter öyrənilməsi daha rahat və eləcədə pythonu yüklədikdə onunla bərabər modul şəklində gələnlərdəndir.
Əgər sisteminizdə tkinter yüklü deyilsə

terminalı açın və ardından sudo apt-get install python-tk yazın

```
$ sudo apt-get install python-tk
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-tk is already the newest version.
python-tk set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
panda@panda-pc:~$
```

Deməli bizdə yüklü haldadır
biz tkinter bəhsinə ilk əvvəl toplu baxış keçirdikdən sonra daha irəli kodlar yazmaqla gui programlayacıq.
Bu bəhsimizi class siniflər daxilində öyrənəcəyik.Çünki birbaşa vacib metoddan istifadə edəkki daha sonra siz sadə programları özünüz yaza biləsiniz.class sinifini öyrənmək vacibdir.Daima internetdə bu metoda rast gələcəysiniz.Biliyiniz artdıqca siz daha böyük layihələrə imza atacağınıza əminəm və bu layihələrin nə qədər kodlardan ibarət olduğunu təsəvvür edərək ona görə class sinifi üzərində bəhsi öyrənməyimizə qərar verdim.Arabir digər metodlardan da istifadə edəcik.
İlk tkinterdə pəncərə yaratmaq mövzusuna baxaq.Yəni ana pəncərəmizi açaq.

Bir idle açın və aşağıdakı kodu yazın
və ya terminaldan pythonu çağıraraq

```
>>> from Tkinter import*
>>>
```

Düşünürəm yuxarıdakı modul çağırmaq metodu bizə yad deyil.yəni Tkinter modulu içindən hər hansısa bir metodu çağır.
Və aşağı sətrə xəta vermədən keçdik,deməli Tkinter modulu bizdə yüklüdür.

Daha sonra

Tk() yazın.yəni

```
>>> Tk()
```

<Tkinter.Tk instance at 0x7f0c852c95f0>
>>>

Və gördüyümüz kimi Tkinter pəncərəsinin açıldığı haqqında məlumat aldıq.İş orasındadırki biz bunu əyani görmədik.Bunun üçün bizə mainloop() ifadəsi təklif olunur

```
>>> mainloop()
```

Və qarşımıza pəncərə açıldı

Və ya grafik program geany açaq və kodlarımızı yazaq

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
window=Tk()
mainloop()
```

def ifadəsi ilə yazsaq.

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
def window():
        window=Tk()
window()
mainloop()
```

Əgər class metodundan istifadə etsək

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
```

```
	def __init__(self):
		self.window()
	def window(self):
pencere=Tk()
a=tkinter()
mainloop()
```

Yuxarıdakı yaratdığımız pəncərə bizə çox bəsit görünür.Bunun üçün irəli gedib Tkinter modulunun içində Label metodundan istifadə edərək pəncərəmizi biraz da baxımlı hala salaq.

Label()

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
	def __init__(self):
		self.window()
		self.label #yeni əlavə etdiyimiz metod
	def window(self):

		self.label=Label(text='Salam Azərbaycan!')
		self.label.pack()

a=tkinter()
mainloop()
```

Yuxarıdakı kodlarımızda def window(self) yazmağımızın məqsədi ilk pəncərə grafik quruluşuna dair kodları def window(self) ifadəsi altında yerləşdirəcəyikki irəlidə kodlarımız artdıqca programımız daha anlayışlı olnun.Və təklif edirəmki hər

bir kodu yazdıqca qarşısından açıqlama yəni rəy bildirin(məsələn #yeni əlavə etdiyimiz metod)

Əgər fikir verdinizsə pəncərəmiz açıldıqda yuxarıda pəncərə adı tk olaraq yazılmışdır.Python bizə o tk adını yəni pəncərə başlığını dəyişmə metodu təklif edir.

a.title('ifadə')

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
                self.label #yeni əlavə etdiyimiz metod
        def window(self):

                self.label=Label(text='Salam Azərbaycan!')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
pencere=Tk()
pencere.title('yeni') #pəncərə başlıq adını dəyişmək üçün
mainloop()
```

gördüyümüz kimi pencere.title('yeni') yazaraq pəncərə başlığını dəyişdik.title() metodunu yalnız sonda istifadə etdik.

Rəng seçimi və rənglərin grafikdə istifadə qaydaları

İlk başlayacağımız metod fg() metodudur.foreground kəliməsinin qısaltmasıdır.ön plan kimi tərcümə olunur

Rəngləri yalnız ingiliscə qarşılıq adları ilə eləcədə kodlanmış rəqəmlərlə daxil edə bilərik.Aşağıda bir neçəsin yazaq

red = qırmızı
white = ağ

black = qara
yellow = sarı
blue = mavi
brown = qəhvəyi
green = yaşıl

Rənglərlə bağlı ətraflı məlumat üçün

http://www.tcl.tk/man/tcl8.3/TkCmd/colors.htm

nəzər yetirin

indi isə yuxarıda istifadə etdiyimiz Label metodunun rəngini dəyişək

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
	def __init__(self):
		self.window()
	def window(self):

		self.label=Label(text='Salam Azərbaycan!',fg='red')
		self.label.pack() #mətni pəncərədə görüntüləməy üçün
pencere=Tk()
a=tkinter()
mainloop()
```

bg, background

Yəni arxa plan kimi tərcümə olunur.background sözünün qısaltmasıdır.

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def window(self):

                self.label=Label(text='SalamAzərbaycan!',fg='red',bg='green')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
a=tkinter()
mainloop()
```

Kodlarımızın self.label=Label(text='SalamAzərbaycan!',fg='red',bg='green') içində bg='green' metodunu istifadə etdik və qarşımıza arxa planı yaşıl olan pəncərə daxilində yazı(SalamAzərbaycan) gördük.

Yazı tipləri

kodlarımıza növbəti əlavə edəcəyimiz metod yazı tipi font() olacaq

font()

bir neçə yazı tipinə nəzər yetirək

italic = yana əyri
underline = alt hissəsi xətli
bold = qalın
overstrike = ust hissəsi xətli

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
```

```
        def window(self):

                self.label=Label(text='Salam\
Azərbaycan!',fg='red',bg='green',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
a=tkinter()
mainloop()
```

```
self.label=Label(text='Salam Azərbaycan!',fg='red',bg='green',font='bold')
```

Yuxarıdakı kodlarımızda sonda istifadə etdiyimiz font='bold' metodu grafikdə olan Salam Azərbaycan! Ifadəsinin qalınlığını dəyişdi.

Pəncərə ölçüsü

Buraya qədər yaratdığımız pəncərə önümüzə açılanda çox kiçik ölçüdə olması bərbad görünür.bunun üçün python bizə Tkinter modulunda olan geometry metodundan istifadə etməyi təklif edir.

geometry()

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def window(self):
             self.label=Label(text='Salam\
zərbaycan!',fg='red',bg='green',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
pencere=Tk()
pencere.geometry('200x100+20+200')
a=tkinter()
mainloop()
```

Yuxarıdakı yazdığımız ifadədə(pencere.geometry('200x100+20+200')) ilk 200-eni,ikinci 100-uzunluğu 20-soldan pəncərənin uzunluğu,200-isə yuxarıdan pəncərənin uzunluğunu ifadə edir.

Bu ölçüləri yazdıqdan sonra fikir versəniz pəncərəni əl ilə uzatmaq,genişlətmək imkanına malikiy.Bunun qarşısını almaq almaq üçün

pencere.resizable(width=FALSE, height=FALSE) metodundan istifadə edəcik. Kodlarımıza toplu baxaq

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def window(self):
                self.label=Label(text='Salam Azərbaycan!',fg='red',bg='green',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
pencere=Tk()
pencere.geometry('200x100+200+200')#pəncərə ölçüsü
pencere.resizable(width=FALSE, height=FALSE)#pəncərəni sabit ölçüdə
a=tkinter()
mainloop()
```

pencere.resizable(width=FALSE, height=FALSE) pəncərəni sabit ölçüdə saxlamaq üçün istifadə etdik.

Button()

button metodu daxilində bir çox argument ala bilən metoddur.qrafik programalara girişdəə dediyim kimi biz hal-hazırda tkinter bəhsi metodlarına ötəri baxış keçiririk.Bütün bu metodlar bitdikdən sonra sabit metodlardan və ya yazacağımız metodlardan istifadə edərək həmin programları qrafik formada istifadəsini gerçəkləşdirəcik.

Sözlə ifadə etmək biraz qəliz olduğundan gəlin kodlarımıza button əlavə edərək pəncərəmizi daha rəngarəng edək.

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def window(self):
                self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
                self.button=Button(text='quit',fg='black',command=pencere.destroy)
pencere=Tk()
pencere.geometry('200x100+200+200')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
a=tkinter()
mainloop()
```

yeni yazdığımız kod
self.button=Button(text='quit',fg='black',command=pencere.destroy)
yuxarıdakı kodları çalışdırdıqda gördüyünüz kimi pəncərədə heç bir button görünmür.çünki biz onun işlək hala gəlməsi üçün pack() metodunu istifadə etmədik.Necəki label metodunda pack() istifadə etmişdiksə button metodunda da həmçinin istifadə edəcik.pack() metodu əlavə edək

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def window(self):
                self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
                self.button=Button(text='quit',fg='black',command=pencere.destroy)
                self.button.pack()
pencere=Tk()
pencere.geometry('200x100+200+200')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
```

```
a=tkinter()
mainloop()
```

Gördüyümüz kimi pack() metodundan istifadə edərək pəncərəmizə button əlavə etdik Sizə qaranlıq olan command ifadəsidirki,elə ingiliscədən tərcüməsi kimi komanda,yəni əmr funksiyasını yerinə yetirir.command=pencere.destroy yazmaqla biz ümumilikdə yaratdığımız pəncərəmizi bağlamaq əmrini veririk.destroy əvəzinə quit kəliməsini də istifadə edə bilərsinizki program daxilində hər ikisi eyni işi görür.Gəlin quit əlavə edək

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def window(self):
                self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
                self.button=Button(text='quit',fg='black',command=pencere.quit)
                self.button.pack()
pencere=Tk()
pencere.geometry('200x100+200+200')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
a=tkinter()
mainloop()
```

Və programı çalışdırdıqda,quit butonuna basaraq pəncərənin bağlanmasının şahidi oluruq.buton metodunda olan text bölməsinə siz çıxış adı və ya ürəyiniz istəyən ad da verə bilərsiniz(məsələn text='çıxış')

hazır button metodunu öyrənmişkən bir fayl yaradaq.Yəni qrafik programımızdan start verərək avtomatik bir fayl yaransın.
Biz bilirikki fayl=open('yenifayl.txt','w') ifadəsiylə yeni bir fayl aça bilərik.
Bunun üçün bu kodu qrafik programımızda rahat görünüşlü yazmaq üçün def ifadəsindən istifadə edərək bir komanda butonu yaratmalıyıq

kodlara baxaq

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def fayl(self):
```

```
                self.fayl=open('yenifayl.txt','w')

        def window(self):
                self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
                self.button=Button(text='quit',fg='black',command=pencere.quit)
                self.button.pack()
pencere=Tk()
pencere.geometry('200x100+200+200')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
a=tkinter()
mainloop()
```

yuxarıda gördüyümüz kimi kodlar arasına

```
def fayl(self):
        self.fayl=open('yenifayl.txt','w')
```

ifadəsini əlavə etdik,amma bunu işlək hala gətirmək üçün bu kodu quit butonu ilə əlaqələndirməliyik.Gəlin quit butonumuzun adını fayl_yarat qoyaraq def fayl(self) ifadəsiylə də əlaqələndirək.

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def fayl(self):
                self.fayl=open('yenifayl.txt','w')

        def window(self):
                self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
                self.button=Button(text='fayl_yarat',fg='black',command=self.fayl)
                self.button.pack()
pencere=Tk()
pencere.geometry('200x100+200+200')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
a=tkinter()
mainloop()
```

Kodlarımıza command=self.fayl ifadəsi artırmaqla def fayl(self) kodunu aktivləşdirdik.

Qeyd edimki əgər biz self.fayl() ifadəsini def __init__(self): kodunun altına qeyd etsəydik fayl avtomatik özü yaranacqdır.Eləcədə self.fayl.close() əmri versəydik butona basmadan program açılanda fayl özü yaranacaqdır.command metodu boş istifadə olunmur.Əgər çalışdıracağınız bir funksiyanız yoxdursa bu metodu istifadə etməyin.

Entry metodu

Entry metodu istifadəçinin tək sətirlik mətin yaza biləcəyi metodlardandır.Kodlarımıza gəlin bu metodu əlavə edək.Biz hal-hazırda bir pəncərə yaratdığımız üçün bütün metodlarımızı ora yükləyirik,irəlilədikcə pəncərə içində daha pəncərələrdə yarada biləcik.Və davam edək

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def fayl(self):
                self.fayl=open('yenifayl.txt','w')

        def window(self):
                self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
                self.button=Button(text='fayl_yarat',fg='black',command=self.fayl)
                self.button.pack()
                self.entry=Entry()
                self.entry.pack() #entry metodumuz
pencere=Tk()
pencere.geometry('200x100+200+200')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
a=tkinter()
mainloop()
```

Yuxarıda gördüyümüz kimi button.pack() metodun aşağısına self.entry=Entry() əlavə etdik.Qeyd edimki Entry ifadəsi modul içində olan sabit addır.kiçik hərflə yazılan entry isə biz yazmışıq,(məsələn self.ekran=Entry() kimi də yaza bilərsiniz)onu dəyişə də bilərsiniz.Mən entry yazmaqda məqsədim kodları siz qarışdırmayasınız.
Yaratdığımız Entry altında gəlin bir buton yaradaqki entry-ə yazdığımız ifadələri silmək olsun.

Bunun üçün bilirikki buttonun command-əmri hissəsinə bir def sil() yaradaq.
Və kodlarımız

```
def sil(self):
        self.entry.delete(0,END)
```

```
self.button_sil=Button(text='sil',command=self.sil)
self.button_sil.pack()
```

kimi olacaq.Yuxarıda yazdığımızself.entry.delete(0,END) ifadəsi entry qutusuyla əlaqədardır.Sizə qaranlıq tərəfi delete olsada bunu təkrar təkrar istifadə edəcəyimiz üçün heç narahat olmayın (0,END) isə yəni sətrin başından hara qədər yazı varsa bütünlüklə silmə əmridir.Və ya (2,5) yazmaqla yəni ikinci sətirdən 4-cü sətrə qədər ifadələri siləcəkdir.

Bütünlükdə kodlarımız

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def fayl(self):
                self.fayl=open('yenifayl.txt','w')
        def sil(self):
                self.entry.delete(0,END) #entry qutusunun silinməsi üçün
        def window(self):
                self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
                self.button=Button(text='fayl_yarat',fg='black',command=self.fayl)
                self.button_sil=Button(text='sil',command=self.sil)
                self.button_sil.pack()
                self.button.pack()
                self.entry=Entry()
                self.entry.pack()
pencere=Tk()
pencere.geometry('200x100+200+200')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
a=tkinter()
mainloop()
```

Və çalışdırdıqda sil butonunun aktiv olduğunun şahidi olduq.

Butonların düzgün yerlərdə olmamasına görə heç narahat olmayın irəlilədikcə bunu da görünüşlü formaya salacıq.

İndidə gəlin yaratdığımız entry qutusuna yazı yazaraq bütünlüklə bu mətni yenifile.txt içinə əlavə edək.Bunun üçün python bizə get ifadəsini təklif edir.get() ifadəsi entry metodunda içində nə yazı görürsə bütünlükdə oxuyub özündə saxlayır.get təkcə entry metodunda istifadə olunmur.Biz hal-hazırda bu metodu öyrəndiyimiz üçün istifadə

edirik.
Biz öncə yazdığımız def fayl(self):
self.fayl=open('yenifayl.txt','w') kodlarda artıq self ifadəsini istifadə etməyəcik,yəni sadəcə fayl=open('yenifayl.txt','w') olaraq yazacıq.çünki self ifadəsin əvvələ yazdıqca kodumuz işlək halda olaraq,program açılan kimi qovluqda yenifayl.txt faylını açacaq.Bütünlüklə kodlarımız

```
# -*- coding: utf-8 -*-
from Tkinter import*
class tkinter():
	def __init__(self):
		self.window()
	def fayl(self):
		fayl=open('yenifayl.txt','w')
		metin=self.entry.get() #entry daxil etdiyimiz mətni oxuyur
		fayl.write(metin) # oxuduğu mətni yenifayl-a köçürür

	def sil(self):
		self.entry.delete(0,END)
	def window(self):
		self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
		self.label.pack() #mətni pəncərədə görüntüləməy üçün
		self.button=Button(text='fayl_yarat',fg='black',command=self.fayl)
		self.button_sil=Button(text='sil',command=self.sil)
		self.button_sil.pack()
		self.button.pack()
		self.entry=Entry()
		self.entry.pack()
pencere=Tk()
pencere.geometry('200x100+200+200')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
a=tkinter()
mainloop()
```

Və programı çalışdırdıqda yazdığımız kodlar xətasız çalışdığına əmin olduq.

width və height

width -eni

height -uzunluğu ifadə edir

ala bildikləri argumentlər yalnız rəqəmlərdən ibarətdir.və ya rəqəmləri bir ifadəyə ataraq da istifadə etmək olur.
Yəni a=10 height =a

Gəlin bu metodların qrafikdə necə özünü göstərdiklərinə baxaq.Bu metodu

self.button_sil=Button(text='sil',command=self.sil,width=20) ifadəsinə əlavə edək və qiymətini 20 olaraq qeyd edib nə ilə nəticələndiyini test edək.

Bütünlükdə kodlar

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def fayl(self):
                fayl=open('yenifayl.txt','w')
                metin=self.entry.get()
                fayl.write(metin)

        def sil(self):
                self.entry.delete(0,END)
        def window(self):
                self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
                self.button=Button(text='fayl_yarat',fg='black',command=self.fayl)
                self.button_sil=Button(text='sil',command=self.sil,width=20)
                self.button_sil.pack()
                self.button.pack()
                self.entry=Entry()
                self.entry.pack()
pencere=Tk()
pencere.geometry('200x100+200+200')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
a=tkinter()
mainloop()
```

Gördüyünüz kimi sil butonu eninə artdı.Aşağı hissəyə doğru isə height metodu ilə

dəyişiklik verək.

```
def window(self):
        self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
        self.label.pack() #mətni pəncərədə görüntüləməy üçün
        self.button=Button(text='fayl_yarat',fg='black',command=self.fayl)
        self.button_sil=Button(text='sil',command=self.sil,width=20,height=1)
        self.button_sil.pack()
        self.button.pack()
        self.entry=Entry()
        self.entry.pack()
```

Və sil butonu aşağıya doğru uzandı.Bunu yerinə yetirən kodumuz height=1 ifadəsi oldu.

Frame ()

frame() metodu qrafik programımıza bir canlanma verəcəyinə əminəm.Biz öncə kodlarımızı çalışdırdıqda buttonların bir birinin içinə girdiyinin şahidi olduq,indi frame() dən istifadə etməklə bunlar arasında məsafələri saxlayacıq.

kodumuz
```
self.frame=Frame()
self.frame.pack()
```

ibarət olacaq.

Yuxarıdakı kodları da əlavə edərək

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def fayl(self):
                fayl=open('yenifayl.txt','w')
                metin=self.entry.get()
                fayl.write(metin)

        def sil(self):
                self.entry.delete(0,END)
        def window(self):
```

```
                self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
                self.button=Button(text='fayl_yarat',fg='black',command=self.fayl)
                self.button.pack()
                self.frame=Frame() #frame metodu məsafə üçün
                self.frame.pack()
                self.button_sil=Button(text='sil',command=self.sil)
                self.button_sil.pack()
                self.frame=Frame() #frame metodu məsafə üçün
                self.frame.pack()
                self.entry=Entry()
                self.entry.pack()
pencere=Tk()

pencere.geometry('300x300+300+300')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
a=tkinter()
mainloop()
```

Gördüyümüz kimi hər iki buton arasına və entry ifadəsinin üst hissəsinə əlavə edərək onlar arasında məsafələri təmin etdik.Hətda pack() metodunun ala bildiyi argumentlərdən əlavə edib məsafəni daha da artıra bilərik.

Kodlarımız

```
def window(self):
                self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
                self.button=Button(text='fayl_yarat',fg='black',command=self.fayl)
                self.button.pack()
                self.frame=Frame()
                self.frame.pack(pady=4)
                self.button_sil=Button(text='sil',command=self.sil)
                self.button_sil.pack()
                self.frame=Frame()
                self.frame.pack(pady=4)
                self.entry=Entry()
                self.entry.pack()
```

şəklində olacaq.

Checkbutton metodu

Bizə məlum olan qeyd etmə metodudur,sadə dildə desək quş qoyma

kodumuz
self.checkbutton=Checkbutton()
self.checkbutton.pack()

yuxarıdakı ifadəni kodlarımıza əlavə edərək

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
class tkinter():
	def __init__(self):
		self.window()
	def fayl(self):
		fayl=open('yenifayl.txt','w')
		metin=self.entry.get()
		fayl.write(metin)

	def sil(self):
		self.entry.delete(0,END)
	def window(self):
		self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
		self.label.pack() #mətni pəncərədə görüntüləməy üçün
		self.checkbutton=Checkbutton()
		self.checkbutton.pack()
		self.button=Button(text='fayl_yarat',fg='black',command=self.fayl)
		self.button.pack()
		self.frame=Frame()
		self.frame.pack(pady=4)
		self.button_sil=Button(text='sil',command=self.sil)
		self.button_sil.pack()
		self.frame=Frame()
		self.frame.pack(pady=4)
		self.entry=Entry()
		self.entry.pack()
pencere=Tk()

pencere.geometry('300x300+300+300')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
a=tkinter()
mainloop()
```

yazaq

Və fayl_yarat butonun üstündə yazmaqla xətasız programı çalışdıraq.
Indidə self.checkbutton=Checkbutton(text='seç') yazaraq bir sətir daha əlavə edək.

Checkbutton da button metodu kimi command əmri verə bilir,yəni command ala biləcəyi argumentlərdəndir.
Çünki biz bir şeyi qeyd etməklə heç bir şey əldə etmirik hal hazırda.amma bu command metodun istifadə etməklə,checkbutonunu seçdikdə bir əmri yerinə yetirdirik.
Gəlin bu qeyd etmə prosesin yerinə yetirdikdə qarşımıza bir mesaj çıxsın

bizim yeni kodlarımız

```
def tk(self):
        self.tk=Tk()
        label=Label(self.tk,text='Program work')
        label.pack()
```

və

```
self.checkbutton=Checkbutton(text='seç',command=self.tk)
self.checkbutton.pack()
```

kimi olacaq.Və bu kodları əlavə edərək

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def fayl(self):
                fayl=open('yenifayl.txt','w')
                metin=self.entry.get()
                fayl.write(metin)
        def tk(self):
                self.tk=Tk() #yeni pəncərə açılır
                label=Label(self.tk,text='Program work')
                label.pack()
        def sil(self):
                self.entry.delete(0,END)
        def window(self):
                self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
                self.checkbutton=Checkbutton(text='seç',command=self.tk)
                self.checkbutton.pack()
                self.button=Button(text='fayl_yarat',fg='black',command=self.fayl)
                self.button.pack()
```

```
                self.frame=Frame()
                self.frame.pack(pady=4)
                self.button_sil=Button(text='sil',command=self.sil)
                self.button_sil.pack()
                self.frame=Frame()
                self.frame.pack(pady=4)
                self.entry=Entry()
                self.entry.pack()
pencere=Tk()
pencere.geometry('300x300+300+300')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
a=tkinter()
mainloop()
```

Gördüyümüz kimi checkbutton program açıldığı zaman seçili halda deyil,gəlin onu biz program açılanda seçili vəziyətə gətirək.Checkbuttonun nə olduğunu daha dəqiq başa düşmək üçün yeni kodlarımızı yazaq.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import*
pencere=Tk()
a=IntVar()
a.set(0)
button=Checkbutton(text='Azərbaycan',variable=a)
button.pack()

mainloop()
```

Yuxarıda yeni pəncərə açdıq və ardından a=IntVar() a.set(0) ifadələrini yazdıq.Bu hər iki ifadə qeyd etmə qutusunda böyük rol oynayır.Əgər bir set(0) -ı yazmasaydıq python avtomatik özü 0-vəziyyətində qeydə alacaqdır.İndidə set(0) nə olduğuna dair açıqlama verək.set(0) ifadəsi yəni 0-ın qeyd etmə qutusunun program açıldığında qeydlənməməsidir,əgər 0-yerinə 1-rəqəmi yazsaq program açıldığı zaman qutu işarələnmiş olacaq.Ardından variable ifadəsi ilə a=IntVar() a.set(0) metodlarını qeyd etdikki qutu işarə sistemi nə edəcəyini bilsin.

Toplevel()

Toplevel,açdığımız pəncərədən əlavə başqa pəncərənin açılmasında böyük rol

oynayır.Biz yeni pəncərəni ana pəncərə kimi yuxarıda açmışdıq.Açdığımız pəncərə istifadəçilər tərəfindən başa düşülən olması üçün gəlin toplevel metodunu istifadə edək. Yadınızdadırsa class sinifi ilə yazdığımız kodlarımızın içində belə bir kod vardı

```
def tk(self):
        self.tk=Tk()
        label=Label(self.tk,text='Program work')
        label.pack()
```

Bu kod yeni bir pəncərənin açılmasına rol oynayır.label ifadəsi içində self.tk kodu o mənanı ifadə edirki bu label ifadəsi yalnız növbəti açılan pəncərənin daxilində olacaq.Əgər biz onu sadəcə root və ya pencere(label=Label(pencere,text='Program work'))adı ilə yazsaydıq həmin bu label mətni ana pəncərəmizin daxilində olacaqdır.

Yuxarıdakı kodumuza self.tk=Tk() deyil toplevel metodunu yazaq

yəni kodumuz

```
def toplevel(self):
                self.toplevel=Toplevel()
                label=Label(self.tk,text='Program work')
                label.pack()
```

şəklində olacaq.

Bütünlükdə kodlarımız

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
class tkinter():
        def __init__(self):
                self.window()
        def fayl(self):
                fayl=open('yenifayl.txt','w')
                metin=self.entry.get()
                fayl.write(metin)
        def toplevel(self):
                self.toplevel=Toplevel()
                label=Label(self.toplevel,text='Program work')
                label.pack()
        def sil(self):
                self.entry.delete(0,END)
        def window(self):
                self.label=Label(text='Salam Azərbaycan!',fg='red',font='bold')
                self.label.pack() #mətni pəncərədə görüntüləməy üçün
                self.checkbutton=Checkbutton(text='seç',command=self.toplevel)
```

```
                    self.checkbutton.pack()
                    self.button=Button(text='fayl_yarat',fg='black',command=self.fayl)
                    self.button.pack()
                    self.frame=Frame()
                    self.frame.pack(pady=4)
                    self.button_sil=Button(text='sil',command=self.sil)
                    self.button_sil.pack()
                    self.frame=Frame()
                    self.frame.pack(pady=4)
                    self.entry=Entry()
                    self.entry.pack()
pencere=Tk()

pencere.geometry('300x300+300+300')
pencere.resizable(width=FALSE, height=FALSE)
pencere.title('yeni')
a=tkinter()
mainloop()
```

Gördüyünüz kimi biz kodlarımıza dəyişikliklər etdik.İlk öncə def ifadəsindən sonra toplevel ifadəsi yazdıq def toplevel(self):   daha sonra self.toplevel=Toplevel() ifadəsi və ardından label mətnin bu toplevel pəncərəsində olması üçün self.toplevel ifadəsini yazdıq. label=Label(self.toplevel,text='Program work')
label.pack()

Və beləliklə programımız rahatlıqla çalışdı heç bir xəta vermədi

Listbox

Bu ifadə bizə pəncərələr içində bir list qutusu açmağa yardımçı olacaq

```
listbox=Listbox()
listbox.pack()

#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import*
pencere=Tk()
pencere.geometry('300x300+200+100')
listbox=Listbox()
listbox.pack()
mainloop()
```

Və pəncərə daxilində bir listbox yaratdıq.

Bunu def ifadəsi ilə əgər yazsaq

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import*
pencere=Tk()
pencere.geometry('300x300+200+100')
def listbox():
    listbox=Listbox()
    listbox.pack()
listbox()
mainloop()
```

yenə təkrar edimki əgər biz def listbox kod blokunu listbox() bağlamasaq sadəcə ekranda pəncərə açıldığını görəcəydiniz.

Class metodu ilə ifadə etsək

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import*
class window():
    def __init__(self):
        self.gui()
    def gui(self):
        self.listbox=Listbox()
        self.listbox.pack()

pencere=Tk()
pencere.geometry('300x300+300+300')
a=window()
mainloop()
```

Gəlin indi bu listbox qutumuzu boş qoyaraq keçməyək və bir neçə kodlar əlavə edərək onu işlək vəziyətə gətirək.

Belə bir ideya ilə başlayaraq,listbox altında bir start butonu açaq və moduldan istifadə edərək programı çalışdıraraq listbox-da məlumatların görünməsini təmin edək

import requests   from urllib import urlopen modullarından istifadə edərək,hər hansı bir ünvanın html scriptlərini oxumağa çalışaq

ilk öncə bu modullar sizdə yüklü olmalıdır.Əgər yüklü deyilsə o zaman pip metodundan istifadə edərək yükləyin.

Əvvəlsə pip -yükləmə metodunu yükləyək

sudo apt-get install python-pip   python 2 üçün

daha sonra

sudo pip install paket_adı yəni

sudo pip install requests

sudo pip install urllib

Bu modul paketlərini xətasız yükləyirik.İndidə bu modullardan istifadə qaydaları ilə tanış olaq.pythonu açırıq və ardından

```
>>> import requests
>>> from urllib import urlopen
>>> response=urlopen('http://www.facebook.com')
>>> response
<addinfourl at 140481946305888 whose fp = <socket._fileobject object at
0x7fc484189950>>
```

Modulları çağıraraq ardından facebook.com səhifəsinin html scriptlərini oxumağa çalışdıq.
Bu script sətirlərini aşkara görmək üçün isə readline() metodundan istifadə edirik

```
>>> response.readline()
'<!DOCTYPE html>\n'
>>> response.readline(20)
'<html lang="az" id="'
>>> response.readline(50)
'facebook" class="no_js">\n'
>>> response.readline(30)
'<head><meta charset="utf-8" />'
>>> response.url
'https://www.facebook.com/'
>>>
```

İndidə bunu qrafik programımıza tətbiq edək.Bunun üçün class sinifimizin içində yeni bir def ifadəsi açaraq

```
def url(self):
        i=self.entry.get()
```

```
            response=urlopen('http://www.'+i)
            self.listbox.insert(0,response.readline(50))
            self.listbox.insert(0,response.url)
```

Biz entry qutusuna sadəcə ünvan.com yazmaqla yetinəcik.

Bütünlükdə kodlarımız

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import*
import requests
from urllib import urlopen
class window():
    def __init__(self):
        self.gui()
    def url(self):
            i=self.entry.get()
            response=urlopen('http://www.'+i)
            self.listbox.delete(0,END)
            self.listbox.insert(0,response.readline(50))
            self.listbox.insert(0,response.url)
    def gui(self):
            self.listbox=Listbox()
            self.listbox.pack()
            self.entry=Entry()
            self.entry.pack()
            self.button=Button(text='start',command=self.url)
            self.button.pack()
pencere=Tk()
pencere.geometry('300x300+300+300')
a=window()
mainloop()
```

bəzən scriptlər uzun olduğundan listbox-da görünməyir,gəlin listboxu genişləndirək

```
self.listbox=Listbox(width=33,height=15)
```

```
def gui(self):
        self.listbox=Listbox(width=33,height=15)
        self.listbox.pack()
        self.entry=Entry()
        self.entry.pack()
        self.button=Button(text='start',command=self.url)
        self.button.pack()
```

və gördüyümüz kimi listbox həm eninə həmdə uzununa böyüdü.

Metodları dəqiqliklə yerləşdirmə

place(relx,rely)

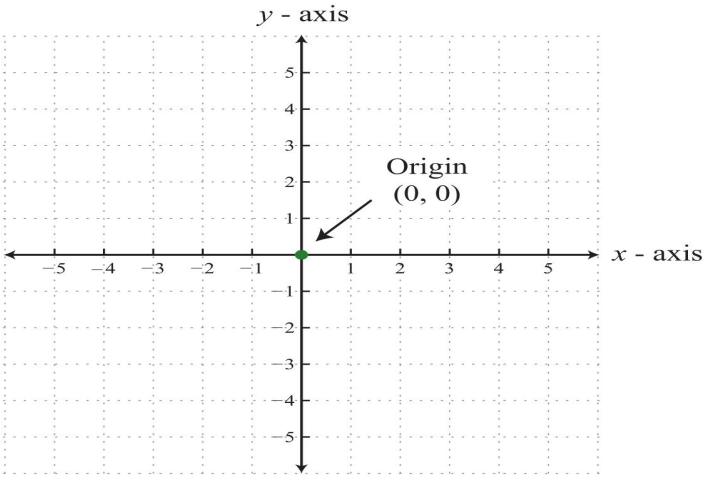
Bu yerləşdirmə metodu qarmaqarışıqlığı aradan qaldıracaq

Gəlin ana pəncərəmizi biraz da genişlədək.Bunu geometry() ilə edə biləcəyimizi bilirik

pencere.geometry('600x600+300+300')
daha sonra listboxumuzu sol tərəfdən yerləşdirək

self.listbox=Listbox()
self.listbox.place(relx=0.04,rely=0.1)

relx-yəni x oxu istiqamətində
rely-isə y oxu istiqamətində



Programı çalışdırdıqda listbox-u yerləşdirdiyimiz koordinatlarda olduğunu görürük.Amma entry qutusu və butonumuz yuxarı çıxdı.İndidə onları gəlin yerləşdirək.Və ardından biraz programımıza yeniliklər gətirək.

#!/usr/bin/python

```
# -*- coding: utf-8 -*-
from Tkinter import*
import requests,random
from urllib import urlopen
class window():
    def __init__(self):
        self.gui()
    def url(self):


                        i=self.entry.get()
                        response=urlopen('http://www.'+i)
                        self.listbox.delete(0,END)
                        self.listbox.insert(0,response.readline())
                        self.listbox.insert(0,response.url)
    def sil(self):
                        self.listbox.delete(0,END)
                        self.entry.delete(0,END)
    def gui(self):
                        self.label=Label(text='ünvanı yazın\nfor exam:python.org')
                        self.label.place(relx=0.00,rely=0.01)
                        self.listbox=Listbox(width=33,height=15)
                        self.listbox.place(relx=0.04,rely=0.1)
                        self.entry=Entry()
                        self.entry.place(relx=0.22,rely=0.01)
                        self.button=Button(text='start',command=self.url)
                        self.button.place(rely=0.05,relx=0.24)
                        self.button_sil=Button(text='sil',command=self.sil)
                        self.button_sil.place(relx=0.4,rely=0.05)
pencere=Tk()
pencere.geometry('600x600+300+300')
a=window()
mainloop()
```

Gəlin indi smtplib modulundan istifadə edərək email göndərmə programını grafik olaraq bir programa çevirək.

Smtplib modulu ilə tanış olaq

```
import smtplib
from smtplib import SMTPException
sender='sizinpoctunuz@hotmail.com'
receiver=['gondereceyinizpoct@gmail və ya hotmail.com']
message='bura ne isteseniz yaza bilersiniz'
try:
            session=smtplib.SMTP('smtp.live.com',587)
```

```
        session.ehlo()
        session.starttls()
        session.ehlo()
        session.login(sender,'sizinpoctparolunuz')
        session.sendmail(sender,receiver,message)
        session.quit()
        print'mesaj uğurla göndərildi'
except smtplib.SMTPException:
        print "mesaj göndərilmədi"
```

yuxarıda sender göndərən

receiver isə qəbul edəndir.
Email göndərmək üçün siz emailinizi sender-ə və pass yazılan yerə isə emailinizin parolunu yazmalısınız.
Smtp portları isə internetdə mövcuddur.live.com üçün 587 portu hal-hazırda istifadə olunur.Bunun üçün mən bir yeni hotmaildən poçt açıram.

Və poçtumuz pythonaz@hotmail.com

parolumuz azerbaijan1234

İndi isə yuxarıdakı modulu öncəki qrafik programımıza əlavə edək.Təbi bir def smtp(self) yaradaraq

daha sonra bir text() metodunu əlvə edirik

```
self.text=Text()
self.text.pack()
```

kodlarımızda bir çox dəyişikliklər olacaq görünüş bərbad olmasın deyə

Aşağıdakı kodlarımızda pəncərəmizi bir email göndərəcəyimiz formada yazdıq

```
def gui(self):
                self.label=Label(text='from')
                self.label.place(relx=0.01,rely=0.01)
                self.label1=Label(text='to')
                self.label1.place(relx=0.01,rely=0.09)
                self.listbox=Listbox(width=33,height=15)
                self.listbox.place(relx=0.5,rely=0.08)
                self.entry=Entry()
                self.entry.place(relx=0.08,rely=0.01)
                self.entry1=Entry()
                self.entry1.place(relx=0.08,rely=0.09)
```

```
        self.text=Text(width=25,height=13)
        self.text.place(relx=0.08,rely=0.14)
        self.button=Button(text='start')
        self.button.place(rely=0.5,relx=0.09)
        self.button_sil=Button(text='sil')
        self.button_sil.place(relx=0.25,rely=0.5)
```

İndi isə smtplib vasitəsilə kodlarımızı istifadəçidən alacaq tipdə yazaq

```
 def smtp(self):
        sender=self.entry.get()
        receiver=self.entry2.get()
        message=self.text.get(1.0,END)
        passw=self.entry1.get()
        try:
                session=smtplib.SMTP('smtp.live.com',587)
                session.ehlo()
                session.starttls()
                session.ehlo()
                session.login(sender,str(passw))
                session.sendmail(sender,receiver,message)
                self.listbox.insert(0,'Mesaj uğurla göndərildi')
                session.quit()

        except smtplib.SMTPException:
                self.listbox.delete(0,END)
                self.listbox.insert(0,'Mesaj göndərilmədi')
```

Yuxarıdakı kodlarımızda sender=self.entry.get() yazaraq ilk entry qutusundan istifadəçinin öz emailini get() ifadəsindən istifadə edərək alırıq.receiver=self.entry2.get() isə üçüncü entry qutusuna kimə göndərəcəyiniz şəxsin emailini yazıb,yenə get() ifadəsiylə alırıq.passw=self.entry1.get() isə sizin email parolunuzdur.
Daha sonra

Text() metodundan istifadə edərək istifadəçinin yazdığı mətni message=self.text.get(1.0,END) metodu ilə tamamilə aldıq.

Bütünlükdə kodlarımız

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
```

```
from Tkinter import*
import smtplib
from smtplib import SMTPException
class window():
    def __init__(self):
        self.gui()
    def smtp(self):
                        sender=self.entry.get()
                        receiver=self.entry2.get()
                        message=self.text.get(1.0,END)
                        passw=self.entry1.get()
                        try:
                                session=smtplib.SMTP('smtp.live.com',587)
                                session.ehlo()
                                session.starttls()
                                session.ehlo()
                                session.login(sender,str(passw))
                                session.sendmail(sender,receiver,message)
                                self.listbox.insert(0,'Mesaj uğurla göndərildi')
                                session.quit()

                        except smtplib.SMTPException:
                                self.listbox.delete(0,END)
                                self.listbox.insert(0,'Mesaj göndərilmədi')

    def sil(self):
                        self.listbox.delete(0,END)
                        self.entry.delete(0,END)

    def gui(self):
                        self.label=Label(text='from')
                        self.label.place(relx=0.01,rely=0.01)
                        self.label1=Label(text='passw')
                        self.label1.place(relx=0.01,rely=0.09)
                        self.label2=Label(text='to')
                        self.label2.place(relx=0.01,rely=0.14)
                        self.listbox=Listbox(width=33,height=15)
                        self.listbox.place(relx=0.5,rely=0.08)
                        self.entry=Entry()
                        self.entry.place(relx=0.08,rely=0.01)
                        self.entry1=Entry()
                        self.entry1.place(relx=0.08,rely=0.09)
                        self.entry2=Entry(show="*")
                        self.entry2.place(relx=0.08,rely=0.14)

                        self.text=Text(width=25,height=13)
                        self.text.place(relx=0.08,rely=0.2)
                        self.button=Button(text='start',command=self.smtp)
```

```
                    self.button.place(rely=0.6,relx=0.09)
                    self.button_sil=Button(text='sil')
                    self.button_sil.place(relx=0.25,rely=0.6)
pencere=Tk()
pencere.geometry('600x600+300+300')
a=window()
mainloop()
```

Və mesaj göndərmə ərəfəsində heç bir xəta ilə qarşılaşmadıq.
Əgər siz bir gmail hesabınızdan mesaj göndərmək istəsəniz o zaman session=smtplib.SMTP('smtp.live.com',587) ifadəsində smtp.live.com yerinə smtp.gmail.com,465 yazacaqsınız.İstəsəniz daha bir entry qutusu açaraq bunu da istifadəçidən ala bilərsiniz.Bunu sizin öhdənizə buraxmadan gəlin bərabər bu kodu da daxil edəkki qaranlıq tərəfi qalmasın.

Listbox üzərinə place() metodu ilə bir entry qutusu yerləşdiririk.

```
self.entry_smtp=Entry()
self.entry_smtp.place(relx=0.5,rely=0.04)
self.entry_port=Entry(width=5)
self.entry_port.place(relx=0.8,rely=0.04)
```

Və kodlarımız içində yeni bir şey olduğunu düşünmürəm.Yalnız yeni istifadə etdiyimiz

self.entry1=Entry(show="*") içindəki show="*" ifadəsidir.Bu metod parolun başqası tərəfindən görünməməsi üçün istifadə etdik.

Ulduz işarəsindən başqa bullet ifadəsi də ala bilir.Parolu yazarkən gəlin altında görünməsi üçün checkbutton yerləşdirək.

```
self.checkbutton=Checkbutton(text='parolu göstər',command=self.goster)
self.checkbutton.place(relx=0.08,rely=0.13)
```

```
def goster(self):
        showinfo('',self.entry1.get())
```

pythonda hazır dialog mesjlarından istifadə etdik.

Text() metodu

Düşünürəm bundan əvvəlki kodlarımızda text metodunun nə olduğunu başa düşdük.İndi isə bunu ayrı bir bəhs kimi öyrənək.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
pencere=Tk()
text=Text()
text.pack()
mainloop()
```

Eyni qayda ilə mətn ifadəmizi yazı tipi və rənglərlə əlavələr edə bilərik

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
pencere=Tk()
text=Text(fg='green',font='bold')
text.pack()
mainloop()
```

yazacağımız mətn yaşıl,yazı tipi isə qalın yəni bold olacaq
Digər metodlar get() ifadəsini istifadə edə bildiyi kimi text() metodu da istifadə edə bilir.

Yazdığımız mətni ala bilmək üçün

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
pencere=Tk()
text=Text(fg='green',font='bold')
text.get(1.0,END)
text.pack()
mainloop()
```

text.get(1.0,END) kodu içindəki 1.0,END -ifadəsi yəni sıra 1 sütun 0-dan başlayaraq nəqədər mətn yazılıbsa al mənasını ifadə edir.Amma bundan başqa biz yazdığımız mətni istədiyimiz aralıqda da ala bilərik.Məsələn

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
pencere=Tk()
text=Text(fg='green',font='bold')
```

```
text.get(1.0,1.9)
text.pack()
mainloop()
```

text.get(1.0,1.9) -yəni sütun bir sıra 0-dan sütun bir sıra 9-a qədər bütün mətni al.Qeyd edimki bunun içində ara boşluqları da daxildir.

Bu aldığımız mətni görmək üçün gəlin bir entry qutusu açıb ora daxil edək.

Kodlarımız

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
pencere=Tk()
def al():
        i=text.get(1.0,END)
        entry.delete(0,END)
        entry.insert(0,i)

text=Text(fg='green',font='bold')
text.pack()
entry=Entry()
entry.pack()
button=Button(text='al',command=al)
button.pack()

mainloop()
```

Hər dəfə yeni sətir yazıb al düyməsinə basanda köhnə yazının silinməsi üçün entry.delete(0,END) ifadəsindən istifadə etdik.

Scrollbar

Bu metodu dilimizdə çubuq kimi qeyd edək.Yəni çevirmə çubuğu.

Yazdığımız mətnlər,Text() metodunda hərf sayı çox olduqda aşağı yuxarı sətirlərə getmək üçün bu metod çox əlverişlidir.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

```
from Tkinter import*
pencere=Tk()

v=Scrollbar(pencere)
v.pack(side=RIGHT,fill=Y)
text=Text(yscrollcommand=v.set,fg='green',font='bold')
text.pack()

mainloop()
```

Yuxarıdakı kodlarımızda İlk öncə v=Scrollbar(pencere) ifadəsini ana pəncərəmizə tətbiq etdik,daha sonra yscrollcommand -tətbiq metodundan istifadə edərək bu scroll -u text qutuunun sağında(side=RIGHT,fill=Y köməyilə) qeyd etdik.

yscrollcommand-hələki aktiv halda deyil,bunun üçün

```
v.config(command=text.yview)
```

ifadəsini istifadə edərək text qutusuna tətbiq edirik

bütünlükdə kodlarımız

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
pencere=Tk()

v=Scrollbar(pencere)
v.pack(side=RİGHT,fill=Y)
text=Text(yscrollcommand=v.set,fg='green',font='bold')
text.pack()
v.config(command=text.yview)
mainloop()
```

əgər x-oxu üzrə yana çubuq təşkil etmək istəsək ozaman xscrollcommand-ifadəsindən istifadə edərək

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
pencere=Tk()

v=Scrollbar(pencere,orient=HORIZONTAL)
v.pack(side=BOTTOM,fill=X)
text=Text(xscrollcommand=v.set,fg='green',font='bold')
text.pack()
```

```
v.config(command=text.xview)
mainloop()
```

yazarıq

Scale

böyütmə və kiçiltmə işlərində yardımçı olur

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *
def scale(ev=None):
        label.config(font='Helvetica -%d bold' % scale.get())
root=Tk()
root.geometry('250x250')
label=Label(root,text='Salam Azərbaycan',font='Helvetica -12 bold')
label.pack(fill=Y,expand=1)
scale=Scale(root,from_=10,to=30,orient=HORIZONTAL,command=scale)
scale.set(10)
scale.pack(fill=X,expand=1)
buton=Button(root,text='quit',command=root.quit)
buton.pack()
mainloop()
```

Menu()

kompyuterdə rast gəldiyimiz yuxarı menyular başlıqları programın komanda qutularını yığcam hala salaraq daha görünüşlü etməyə yardımçı olur.Fikir vermisinizsə hər bir menu adın altında bir neçə alt-menu lar da mövcuddur.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
pencere=Tk()
menu=Menu(pencere)
pencere.config(menu=menu)
fayl=Menu(menu)
menu.add_cascade(label='file',menu=fayl)
mainloop()
```

menu.add_cascade vasitəsilə ilk file adlı menu başlığını yaratdıq.Bundan başqa əgər file adlı başlığa əlavələr etsək fayl.add_command() metodu ilə olacaq.bu metodu biz

menu.add_cascade(label='file',menu=fayl) ifadəsinə yəni file altına bağlayacağımız üçün fayl.add_command() ifadəsini istifadə edəcik

Kodlarımıza baxaq

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
pencere=Tk()
menu=Menu(pencere)
pencere.config(menu=menu)
fayl=Menu(menu)
menu.add_cascade(label='file',menu=fayl)
fayl.add_command(label='open')
fayl.add_command(label='save')
fayl.add_command(label='start')
fayl.add_command(label='exit')
mainloop()
```

Və gördüyümüz kimi file -altında open-save-start və exit ifadələri var.İndi gəlin bunları class metodunda yazaq və ardından aktiv edək bu ifadələri

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
class menu():
	def __init__(self):
		self.menu()
	def menu(self):
		menu=Menu(pencere)
		pencere.config(menu=menu)
		fayl=Menu(menu)
		menu.add_cascade(label='file',menu=fayl)
		fayl.add_command(label='open')
		fayl.add_command(label='save')
		fayl.add_command(label='start')
		fayl.add_command(label='exit')
pencere=Tk()
a=menu()
mainloop()
```

Və class sinfinə yerləşdirdik.Əgər fikir verirsinizsə file -açdıqda ifadələrin tam üstündə düz xətt görünür,bunu ləğv etmək üçün tearoff ifadəsindən istifadə edəcik.fayl=Menu(menu,tearoff=0) ifadəsi vasitəsilə

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
class menu():
	def __init__(self):
		self.menu()
	def menu(self):
		menu=Menu(pencere)
		pencere.config(menu=menu)
		fayl=Menu(menu,tearoff=0)
		menu.add_cascade(label='file',menu=fayl)
		fayl.add_command(label='open')
		fayl.add_command(label='save')
		fayl.add_command(label='start')
		fayl.add_command(label='exit',command=pencere.quit)
pencere=Tk()
a=menu()
mainloop()
```

İndidə ana menyu çubuğuna növbəti menu əlavə edək

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
class menu():
	def __init__(self):
		self.menu()
	def menu(self):
		menu=Menu(pencere)
		pencere.config(menu=menu)
		fayl=Menu(menu,tearoff=0)
		menu.add_cascade(label='file',menu=fayl)
		fayl.add_command(label='open')
		fayl.add_command(label='save')
		fayl.add_command(label='start')
		fayl.add_command(label='exit',command=pencere.quit)
		edit=Menu(menu,tearoff=0)
		menu.add_cascade(label='edit',menu=edit)
pencere=Tk()
a=menu()
mainloop()
```

Bizim ana menu çubuğumuz menu=Menu(pencere) ifadəsi ilə əlaqədardır və edit=Menu(menu,tearoff=0) menu çubuğu üçün menu.add_cascade(label='edit',menu=edit) ifadəsini istifadə etdik.

Ana pəncərəyə görünüş verək

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
import sys,os
class menu():
	def __init__(self):
		self.menu()
		self.gui()
	def gui(self):
		i=sys.version
		v=os.name
		self.label=Label(pencere,text=i)
		self.label.place(relx=0.2,rely=0.05)
		self.label1=Label(pencere,text=v,font='bold')
		self.label1.place(relx=0.4,rely=0.16)
	def menu(self):
		menu=Menu(pencere)
		pencere.config(menu=menu)
		fayl=Menu(menu,tearoff=0)
		menu.add_cascade(label='file',menu=fayl)
		fayl.add_command(label='open')
		fayl.add_command(label='save')
		fayl.add_command(label='start')
		fayl.add_command(label='exit',command=pencere.quit)
		edit=Menu(menu,tearoff=0)
		menu.add_cascade(label='edit',menu=edit)
pencere=Tk()
pencere.geometry('400x400+200+100')
a=menu()
mainloop()
```

Yuxarıdakı sys.version və os.name funksiyalarına əvvəldən tanışıq.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
import sys,os
class menu():
	def __init__(self):
		self.menu()
```

```
                self.gui()
        def toplevel(self):
                self.toplevel=Toplevel()
                self.toplevel.geometry('300x300+200+100')
                self.toplevel.title('start')
                self.label=Label(self.toplevel,text='adınız')
                self.label.place(relx=0.01,rely=0.02)

        def gui(self):
                i=sys.version
                v=os.name
                self.label=Label(pencere,text=i)
                self.label.place(relx=0.2,rely=0.05)
                self.label1=Label(pencere,text=v,font='bold')
                self.label1.place(relx=0.4,rely=0.16)
                self.text=Text(widt=40,height=10)
                self.text.place(relx=0.15,rely=0.3)
        def menu(self):
                menu=Menu(pencere)
                pencere.config(menu=menu)
                fayl=Menu(menu,tearoff=0)
                menu.add_cascade(label='file',menu=fayl)
                fayl.add_command(label='open')
                fayl.add_command(label='save')
                fayl.add_command(label='start',command=self.toplevel)
                fayl.add_command(label='exit',command=pencere.quit)
                edit=Menu(menu,tearoff=0)
                menu.add_cascade(label='edit',menu=edit)
                edit.add_command(label='font')
                edit.add_command(label='backgrount')
pencere=Tk()
pencere.geometry('400x400+200+100')
a=menu()
mainloop()
```

Geometry() ölçülər

pack()

Bu ifadə ilə artıq biz tanışıq.Bəhsdə isə onun ala bildiyi argumentlərdən danışacıq.ifadə digər metodları ölçüsüz olaraq və ardıcıllığı gözləmək şərti ilə yerləşdirir,təbiki mərkəzdən

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
root=Tk()
label=Label(text='Bu metod')
label.pack()
label1=Label(text='ikinci metod')
label1.pack()
root.title('pack')
mainloop()
```

Ardından ikinci label1 ifadəsini istifadə etdikdə pack metodu ilk labelin altından bunları yerləşdirdi.pack() metodunun ala biləcəyi ilk argument side -ı izah edəcik.

side=RİGHT
side=LEFT
side=TOP
side=BOTTOM

istifadə edərək kodlarımız

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
root=Tk()
label=Label(text='Bu metod')
label.pack(side=TOP)
label1=Label(text='ikinci metod')
label1.pack(side=LEFT)
root.title('pack')
mainloop()
```

expand

İngilis dilindən dərcüməsi genişləndirmək tərcümə oldunduğu kimi də işi yerinə genişləndirərək yetirir.
Növbəti argumentimiz expand olacaq.Bu argument daxilində YES və NO ifadələrindən istifadə edərək metodlar arasında məsafələri təşkil edir.Qeyd edimki expand=NO yazmağa ehtiyac yoxdur,çünki başdan pack() expand=NO argumenti ilə açılır.

fill

Dilimizə doldurmaq kimi tərcümə olunur.Argument daxilində

X,Y və BOTH ifadələrini istifadə edir.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
root=Tk()
label=Label(text='Bu metod')
label.pack(fill=BOTH)
label1=Label(text='ikinci metod')
label1.pack(fill=X)
root.title('pack')
root.geometry('200x200+200+100')
mainloop()
```

relief

Bu sözə coğrafiya fənnində də rast gəldiyimiz kimi relyef yəni çökəklər,qabartılar kimi mənaları ifadə edir.Argument aşağıdakı ifadələri istifadə edir.

```
relief=SOLID
relief=FLAT
relief=RIDGE
relief=SUNKEN
relief=GROOVE
relief=RAISED
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
root=Tk()
button=Button(text='relief',relief=GROOVE)
button.pack()
root.title('pack')
root.geometry('200x200+200+100')
mainloop()
```

GROOVE ifadəsindən istifadə edərək butonumuzu qabarıq göstərdik.Siz digərlərin də istifadə edib necə görünüş verdiyini test edə bilərsiniz.

grid()

Bu metod pack() metodundan fərqli olaraq ilk sətir və sıradan başlayır nizamlamağa

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
root=Tk()
button=Button(text='relief',relief=GROOVE)
button.grid()
root.title('pack')
root.geometry('200x200+200+100')
mainloop()
```

metodun ala bildiyi ən əhəmiyyətli ifadələrdən ikisi row və column

row -sətir
column – sıra mənasını ifadə edir.

Biraz place metoduna bənzəsədə amma place() metodu kimi dəqiq yerləşdirməni yerinə yetirə bilmir.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
root=Tk()
button=Button(text='relief',relief=GROOVE)
button.grid(row=1,column=0)
button1=Button(text='grid',relief=SUNKEN)
button1.grid(row=1,column=2)
root.title('pack')
root.geometry('200x200+200+100')
mainloop()
```

Kodlardan gördüyünüz kimi iki butonumuzu yanaşı düzməyi bacardıq.

Bəzi metodların argumentlərini fərqli şəkildə də yaza bilərik

```
from Tkinter import*
root=Tk()
label=Label(text='Hello world')
label.pack()
mainloop()
```

Yuxarıda bildiyimiz kodu bir başqa formada da yaza bilərik

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
root=Tk()
label=Label()
label['text']='Hello world'
label.pack()
mainloop()
```

text argumenti label ifadəsinin tam yanında məlumat mətni isə xaricdə yer aldı.
Eyni qayda ilə digər argumentlərdən məsələn command -da bu qaydada istifadə oluna bilər.Qeyd edimki bu yol uzun-uzadı yoldur,hər halda bu yolu da bilməyimizin bir ziyanı yoxdur.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import*
root=Tk()
label=Label()
label['text']='Hello world'
label.pack()
buton=Button()
buton['text']='start'
buton['command']=root.quit
buton.pack()
mainloop()
```

Standart dialoqlar (Standard Dialogs)

Biz əvvəlki dərslərdə istifadəçiyə bir məlumat vermək üçün yeni toplevel metodu ilə pəncərə açırdıqsa bu hazır dialoqlar vasitəsilə işimiz daha rahat olacaq.

Showerror()

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *
from tkMessageBox import *
def mesaj():
        showerror('error','siz xəta ilə qarşılaşdınız')
root=Tk()
button=Button(root,text='start',command=mesaj)
button.pack()
mainloop()
```

showerror('error','siz xəta ilə qarşılaşdınız') ifadəsində ilk 'error' title metodumuz kimi dialoq pəncərə adı,ikinci 'siz xəta ilə qarşılaşdınız' isə bizə məlumat verən mətndir.

Metodun type argumenti vardırki aşağıdakı ifadələri alır.

ABORTRETRYIGNORE
OK
OKCANCEL
RETRYCANCEL
YESNO
YESNOCANCEL

bundan başqa default argumenti də var.default öz növbəsində isə aşağıdakı ifadələri alır.

ABORT
RETRY
IGNORE
OK
CANCEL
YES
NO

icon argumenti isə

ERROR
INFO
QUESTION
WARNING ifadələrini alır.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *
from tkMessageBox import *
root=Tk()
def mesaj():
	i=showerror('error','quit',type=OKCANCEL)
	label['text']=i

label=Label(text='system')
label.pack()
button=Button(text='start',command=mesaj)
button.pack()

mainloop()
```

```
showinfo()
```

Məlumat göstərən dialoq pəncərəsi

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *
from tkMessageBox import *
root=Tk()
def mesaj():
	if not entry.get():
		showerror('error','empty')

	if entry.get():
		showinfo('',entry.get())
entry=Entry()
entry.pack()
button=Button(text='start',command=mesaj)
button.pack()

mainloop()
```

Kodlarımızda ilk entry qutunun boş olduğunda bizə showerror('error','empty') ifadəsi yardımı ilə empty yazısı bildiriləcək

Əgər entry qutusunu bir ifadə yazarsaq o zaman yazdığımız ifadə showinfo('',entry.get()) vasitəsilə pəncərədə bildiriləcəkdir.

Showwarning()

Həyəcan bildirən pəncərə tipi

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *
from tkMessageBox import *
root=Tk()
def mesaj():
            if not entry.get():
                        showwarning('warning','empty')

entry=Entry()
entry.pack()
button=Button(text='start',command=mesaj)
button.pack()

mainloop()
```

Mouse və klaviatura tkinter bəhsində
ev=None və ya event=None ifadəsi

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *
root=Tk()
label=Label(text='please write you name:')
label.pack()
entry=Entry()
entry.pack()
```

```
def save():
	fayl=open('python.txt','a')
	get=entry.get()
	fayl.write(entry.get()+'\n')
	fayl.close()
	entry.delete(0,END)
button=Button(text='ok',command=save)
button.pack()
mainloop()
```

Yuxarıdakı kodlarımızda bizə yad olan metod yoxdur.ok düyməsinə basdıqda def save() blokunda olan kodlar aktiv olacaq.ev=None ifadəsi əlavə edərək düymə hərəkətlərini nəzarətə götürəcəyik.event ifadəsi yerinə hər hansı bir söz yaza bilərsiniz,bu ifadə class siniflərdə olan self ifadəsinə bənzəyir.Məsələ onu argument verməkdir,yəni None argumenti düymə hərəkətlərini nəzarətə götürəcək.

Kodlarımıza baxaq

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *
root=Tk()
label=Label(text='please write you name:')
label.pack()
entry=Entry()
entry.pack()
def save(ev=None):
	fayl=open('python.txt','a')
	get=entry.get()
	fayl.write(entry.get()+'\n')
	fayl.close()
	entry.delete(0,END)
button=Button(text='ok',command=save)
button.pack()
root.bind('<Enter>',save)
mainloop()
```

def save(ev=None): ifadəsinə ev=None əlavə etdik.Sonda qeyd etdiyimiz root.bind('<Enter>',save) ifadəsi def save kod bloku ilə əlaqədə olub ok butonunun gördüyü işi yerinə yetirir,necəki ok butonuna basdıqda fayl daxilində entry qutusuna yazdığımız yazı daxil olurdusa,indi isə tək enter düyməsinə bassanız da bu əmr yerinə yetiriləcəkdir.

Başqa bir düymə əlavə edərək

```
root.bind('<Escape>',sys.exit)
```

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *
import sys
root=Tk()
label=Label(text='please write you name:')
label.pack()
entry=Entry()
entry.pack()
def save(ev=None):
        fayl=open('python.txt','a')
        get=entry.get()
        fayl.write(entry.get()+'\n')
        fayl.close()
        entry.delete(0,END)
button=Button(text='ok',command=save)
button.pack()
root.bind('<Enter>',save)
root.bind('<Escape>',sys.exit)
mainloop()
```

Escape düyməsi basılınca sistemdən çıxışı təmin etdik.

Aşağıda bəzi klaviatura ifadələrindən istifadə edə bilərsiniz

F1:
Num Lock: <Num_Lock>
Scroll Lock: <Scroll_Lock>
Backspace: <BackSpace>
Delete: <Delete>
Sol : <Left>
Sağ: <Right>
Yuxarı : <Up>
Alt: <Alt_L>
Ctrl :<Control_L>
Shift :<Shift_L>
Ctrl + s: <Control-s>
Shift + s: <Shift-s>
Alt + s: “<Alt-s>
Boşluq :<space>

Print :<Print>

Maus hərəkətləri

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *
root=Tk()
root.geometry('300x300')
label=Label(text='Hello Azerbaijan')
label.pack(anchor=NW)
label1=Label(text='X')
label1.pack(side=LEFT,anchor=N)
entry=Entry(width=5)
entry.pack(side=LEFT,padx=10,anchor=N)
label1=Label(text='Y')
label1.pack(side=LEFT,anchor=N)
entry1=Entry(width=5)
entry1.pack(side=LEFT,anchor=N)
def move(ev=None):
        entry.delete(0, END)
        entry1.delete(0, END)
        entry.insert(0, ev.x)
        entry1.insert(0, ev.y)
root.bind('<Button-1>',move)
mainloop()
```

Kodlarımızda mausun sol düyməsini ana pəncərəmizin istənilən yerinə basdıqda entry x və entry y qutularında basdığımız yerin koordinatları absis və oordinat oxları üzrə məafələri,koordinatları qeyd olunacaq.mous hərəkətlərindən aşağıdakılara da nəzər yetirə bilərsiniz.

Button-2 mausun təkəri,scroll düyməsi
Button-3 mausun sağ düyməsi
Motion basmadan maus hərəkəti
B1-Motion sol düyməyə basılı tutaraq mausun hərəkəti
B2-Motion təkərin basılı olduğu halda mausun hərəkəti
B3-Motion sağ düymənin basılı halda mausun hərəkəti
ButtonRelease-1 bir dəfə sol düyməyə basıb buraxmaqla
ButtonRelease-2 təkəri bir dəfə basıb buraxılması
Double-Button-1 iki dəfə sol düymənin basılması
Double-Button-2 təkərin iki dəfə basılması
Double-Button-3 sağ düymənin iki dəfə basılması
Leave mausun pəncərəni tərk etməsi
və sonuncu Enter ki bunu enter düyməsi kimi başa düşməyin Leave istifadə etdikdən

sonra maus geri pəncərəyə döndükdə yazılan ifadədir.

Keysum ifadəsi

Bu ifadə klaviaturadan basılan düymələri tutmaq rolunu oynayır.

Kodlarımıza baxaq.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *
root=Tk()
root.geometry('300x300')
label= Label(text='düyməs:\n')
label.pack()
def show(ev=None):
	label['text'] += ev.keysym
root.bind("<Key>",show)

mainloop()
```

label metodunda yazdığınız kəlimələri limitə də ala bilərik.yəni mətn sayını müəyyən sayla sərhədləmndirə bilərik.bunun üçün wraplength ifadəsindən istifadə edəcik

kodlarımız

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import *
root=Tk()
root.geometry('300x300')
label= Label(text='düyməs:\n',wraplength=100)
label.pack()
def show(ev=None):
	label['text'] += ev.keysym
root.bind("<Key>",show)

mainloop()
```

kodlarda gördüyünüz ev.keysym ifadəsi isə ana pəncərədə klaviatura düymələrinin basıldığını label metodu ilə görünməsini təmin edir.

Keysum vasitəsilə bir entry qutusunda olan yazını istifadəçi silə bilməmək üçün şərt də qoya bilərik.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Tkinter import*
from tkMessageBox import *
root=Tk()
root.geometry('200x200')
entry=Entry(show='salam dostlar')
entry.pack()
def clear(ev=None):
        if ev.keysym=='Delete':
                return 'break'
        elif ev.keysym =='BackSpace':
                return 'break'
entry.bind('<Key>',clear)
mainloop()
```

kodlarımızda if ev.keysym=='Delete' ifadəsi illə əgər istifadəçi delete düyməsini basarsa return 'break' metodundan istifadə edərək silmə işini dondururuq,eləcədə elif ev.keysym =='BackSpace': ifadəsinə qarşılıq təkrar olaraq return 'break' metodundan istifadə etdik.

Ardı ikinci buraxılışımızda olacaq