بنام خدا

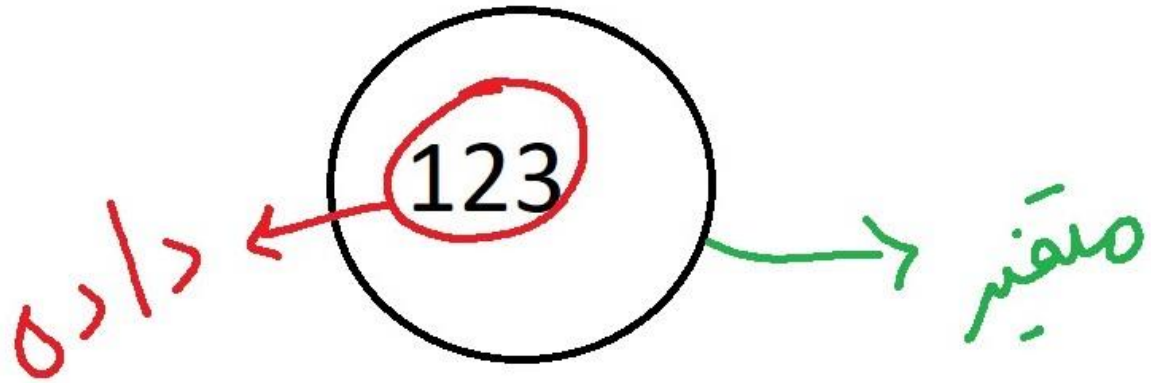
متغیر ها

متغیر ها فضایی از حافظه هستند که میتوان مقداری را در آن ها قرار داد و در طول برنامه ازشان استفاده کرد.

برای درک بهتر متغیر ها

متغیر ها را مانند یک ظرف یا لیوان در نظر بگیریم که موادی را درون ان ها ریختیم.

این مواد همان داده های ما هستند.



- هر متغیر یک نام و یک مقدار دارد.

مقدار متغیر = نام متغیر

- مقدار متغیر میتواند در طول برنامه تغییر کند.یعنی اگر اول مقدار متغیر ما 5 بوده و دوباره در یک خط بعد به همان متغیر مقدار 6 را بدهیم مقدار5 پاک میشود و مقدار 6 درون متغیر قرار میگیرد.

```
a = 5
print(a)
a=6
print(a)
```

خروجی :؟

مثال:

متغیری با نام a1تعریف کنید و مقدار 12 را به ان بدهید.
سپس در خطی دیگر مقدار a1 را برابر 32.5 قرار دهید و خروجی را چاپ کنید.

نام گذاری متغیر ها

برای نام گذاری متغیر ها قوانینی وجود دارد که باید رعایت کنیم.

1. نام متغیر باید با حروف الفبای a-z و A-Z یا _ شروع شود.
2. در نام گذاری متغیر ها نباید از کاراکتر های غیر مجاز مانند @ یا $ یا # یا % یا ! یا ؟ استفاده شود.
3. استفاده از فاصله در نام گذاری متغیر ها مجاز نیست.
4. نام متغیر ها نباید یکی از کلمات کلیدی باشد.
5. پایتون به حروف کوچک و بزرگ حساس است.یعنی متغیر a با A یکی نیستند.

دستور چاپ خروجی

چاپ خروجی به معنای نمایش خروجی برنامه است.زمانی که یک برنامه را مینویسیم نیاز داریم خروجی برنامه برایمان نمایش داده شود.به نمایش خروجی،چاپ خروجی میگوییم.

دستور چاپ در پایتون با استفاده از کلمه کلیدی print انجام میشود.

کلمه کلیدی:

کلمات کلیدی یک سری کلمات رزرو شده وتعریف شده در زبان های برنامه نویسی هستند که برای اهداف خاصی استفاده میشوند.

شکل ظاهری دستور print

```
print(مقداری که میخواهیم چاپ شود)
```

مقادیری که میتوانیم در print قرار دهیم تا چاپ شود:

- متغیر
- عدد
- متن(رشته)
- محاسبات

و غیره...

چاپ متغیر ها:

```
a=10
print(a)
```

میتوان چند مقدار را داخل یک print چاپ کرد:

```
a=10
b=20
print(a,b)
```

چاپ اعداد:

Print(3,4,5)

خروجی:345

چاپ (رشته):

Print(“hello”)

Print(‘hello’)

میتوان متن ها را هم در ‘ و هم در “ نوشت.

اما نمیتوان در یک رشته هم از ‘ و هم از “ استفاده کرد.

Print(‘hello”)

اشتباه است.

میتوان بین مقادیری که میخواهیم چاپ کنیم یک " " هم بگذاریم که در خروجی از هم فاصله بگیرند.

A=’hello’

B=’world’

Print(A , “ghfgh “ ,B,44)

خروجی:

hello ghfgh world 44

چاپ محاسبات:

میتوان درون print محاسبات را هم انجام داد:

```
Print(3+5)
```

```
A=10
Print(3+A)
```

```
A=10
B=20
Print(A+B)
```

انواع داده ها:

ما در طول برنامه با نوع داده های زیادی سر و کار داریم.

از یک سری داده ها استفاده میکنیم که به صورت متنی (رشته ای) هستند.

بعضی از داده ها عدد صحیح هستند.

بعضی از داده ها عدد اعشاری هستند و غیره...

این داده ها اسم مخصوص خودشان را دارند:

نوع داده int (عددی)

نوع داده integer به معنی اعداد صحیح میباشد.در برنامه نویسی اعداد صحیح را اعداد اینتیجر مینامیم.

مثلا:

3 , 4 , 10, 55

نوع داده float (اعشاری)

نوع داده float به معنی اعداد اعشاری میباشد.در برنامه نویسی اعداد اعشاری را اعداد فلوت مینامیم.

مثلا

5.5 , 45.2 ,3.14 , 25.5

نوع داده string (متنی)

نوع داده string به معنی رشته میباشد.رشته ها متن هایی هستند که ما به عنوان داده ازشان استفاده میکنیم و ان ها را داده های رشته ای مینامیم.

مانند:

A=‘zahra’ , ‘kiamehr’ , ‘hadi’ , ’abolfazl’

رشته ها درون ‘ یا " قرار میگیرند.

نوع داده eval

نوع داده eval همه انواع داده را در خودش دارد و بر اساس شرایط نوع خودش را مشخص میکند.

مثلا یک بار میتواند integer باشد،یک بارfloat یک بار رشته و بقیه انواع داده.

دستور گرفتن ورودی

توی مقدار دهی متغیر ها میومدیم توی کدهامون به صورت مستقیم یک مقدار مشخص رو به متغیر میدادیم.

ولی بعضی وقت ها من نمیدونم میخام درون متغیر چه مقداری قرار بگیره .کاربر باید درون خروجی مقدار رو وارد کنه برای این کار باید داده رو از ورودی بگیریم و ازش استفاده کنیم.

برای گرفتن ورودی از کلمه کلیدی input استفاده میکنیم.

Input به معنی ورودی است.

شکل ظاهری گرفتن ورودی:

نام متغیر= input(پیغام ورودی)

مثلا میخوایم یک رشته از ورودی بگیریم و ان را چاپ کنیم.

A=input('yek string vared konid:')

Print(A)

ماهیت input رشته است.یعنی هرچی که از ورودی دریافت میکنیم به عنوان یک رشته میشناسد.حتی عدد هم وارد کنیم ان را یک رشته میبیند و باهاش مانند عدد رفتار نمیکند.

ولی ما بعضی وقت ها میخوایم یک عدد از ورودی دریافت کنیم و ازش استفاده کنیم.

برای این کار باید ورودی را به نوع عددی دریافت میکنیم.

گفتیم انواع عددی مختلفی داریم:

Int float eval

برای این که داده ای با نوع int بگیریم:

```
A=int(input('enter a number'))
```

ولی من موقعی که نوع ورودی را int دادم دیگه نمیتونم عدد اعشاری وارد کنم.

برای این که داده ای با نوع float بگیریم:

```
A=float(input('enter a number'))
```

ولی ممکنه من اصلا ندانم قراره چه نوع ورودی داشته باشم میخواهم یک نوع داده ای باشد که هرچی دلم میخواد از ورودی بگیرم.

اگه عدد صحیح دادم یا اعشاری دادم یا رشته وهر نوع دیگه ای برام دریافت کنه و error نده.
برای این کار از نوع داده eval استفاده میکنم.

نوع داده eval همه نوع داده ای را میگیرد.

```
A=eval(input('enter a number'))
```

بهتر است برای گرفتن ورودی همیشه از eval استفاده کنید.

اگر از eval استفاده میکنید برای وارد کردن رشته به عنوان ورودی باید از ‘ ‘ یا " " استفاده کنید.

مثال

یک عدد از ورودی بگیرید و ان را با 300 جمع کنید و خروجی را چاپ کنید.

مثال

دو عدد از ورودی بگیرید ان هارا باهم ضرب کنید و خروجی را چاپ کنید.

مثال

نام و نام خانوادگی یک فرد را از ورودی دریافت کنید و ان ها را در کنار هم چاپ کنید.

مثال

برنامه ای بنویسید که طول و عرض مستطیلی را از ورودی دریافت کند و مساحت ان را محاسبه و چاپ کند.

مساحت مستطیل=طول*عرض

برای محاسبات یک سری اولویت ها وجود دارد:

پرانتز

ضرب و تقسیم

جمع و تفریق

مثال

```
a=3+5*6
Print(a)
```

خروجی؟

```
a =(3+5)*6
print(a)
```

دستورات شرطی

همانطور که در زندگی روز مره برای انجام کار های مختلف ممکن است در شرایطی قرار بگیرید که نیاز به تصمیم گیری داشته باشید برای حل یک مسئله هم ممکن است نیاز داشته باشید یک یا چند شرط بررسی شود.

مثلا در صورت برقرار بودن یک شرط چه دستوراتی اجرا شود و در صورت برقرار نبودن شرط دستوراتی دیگر اجرا شوند.

برای برسی شرط از کلمه کلیدی if استفاده میکنیم.و برای برسی خلاف شرط(در غیر این صورت) از else استفاده میکنیم.

شکل ظاهری دستور شرطی:

```
if شرط:
    دستورات در صورت برقراری شرط
else:
    دستورات در صورت برقرار نبودن شرط
```

If میگوید اگر شرطی که جلوی من نوشتی برقرار باشد دستورات من اجرا میشود.

else میگوید اگر شرطی که جلوی ifنوشتی برقرار نبود دستورات من اجرا میشود.

مثال

سن کاربر را از ورودی دریافت کنید اگر بالا تر از 40 بود پیغام کهنسال را چاپ کند .در غیر این صورت پیغام جوان را چاپ کند.

مثال

یک عدد از ورودی دریافت کند اگر مثبت بود پیغام مثبت را چاپ کند در غیر این صورت منفی را چاپ کند.

مثال

گاوصندوقی داریم که رمز ورود ان 1234 است .میخواهیم برای باز کردن گاوصندوق رمزش را از ورودی دریافت کنیم .اگر رمز وارد شده 1234 بود پیغام در باز شد را چاپ کند در غیر این صورت پیغام رمز نادرست است را چاپ کند.

عملگر ها

< کوچکتر

> بزرگتر

<= کوچکتر مساوی

=> بزرگتر مساوی

== مساوی/برابر

=! نامساوی/نابرابر

مثال

تصور کنید صاحب فروشگاه میوه فروشی هستید.در فروشگاه شما همه میوه ها موجود است به غیر از پرتقال.

از کاربر بخواهید نام میوه موردنظر خود را وارد کند.اگر پرتقال نبود،پیغام "خرید موفقیت امیز بود" در غیر این صورت پیغام "پرتقال موجود نیست" را چاپ کند.

انواع دستور شرطی :

1.دستور شرطی ساده (if)

2.دستور شرطی دوگانه(if else)

3.دستور شرطی چند گانه (if elif else)

4.دستور match case

1. دستور شرطی ساده:

در این دستور شرطی فقط از if استفاده میشود.یعنی فقط شرط بررسی میشود و در صورت برقرار بودن شرط دستورات if اجرا میشود و در صورت بر قرار نبودن شرط هیچ دستوری اجرا نمیشود یعنی کار خاصی انجام نمیدهد.

مثال

عددی از ورودی بگیرید اگر بیشتریا مساوی از 300 بود ان را با 300 جمع کند و خروجی را چاپ کند.

همانطور ک دیدید در دستور شرطی ساده else نداریم یعنی در صورت برقرار نبودن شرط هیچ کاری انجام نمیدهد.

2.دستور شرطی دوگانه

در دستور شرطی دوگانه هم if داریم هم else . یعنی دستورات در صورت برقراری شرط بررسی میشود هم در صورت برقرار نبودن شرط.

هم if داریم هم else

مثال

نام کاربر را از وزودی بگیرید.اگر نام وارد شده شده نام خودتان بود پیغام hello admin را چاپ کند در غیر این صورت پیغام hello user را چاپ کند.

3.دستورات شرطی چند گانه

در دستورات شرطی چند گانه چند شرط بررسی میشود.if اولین شرط را چک میکند در صورت برقراری شرط دستورات if اجرا میشود. elif میگوید اگر دستور if برقرار نبود اول شرط مرا چک کن اگر شرط من برقرار بود دستورات مرا اجرا کن در غیر این صورت برو دستور بعدی رو چک کن.Else میگوید اگر شرط اول و دوم و ... برقرار نبودن بیا دستور مرا اجرا کن.

مثال.

نام کاربر را دریافت کنید در صورت وارد کردن نام ابوالفضل پیغام " نام خانوادگی شما مرادی است"

در غیر این صورت اگر نام کیامهر را وارد کرد پیغام"نام خانوادگی شما کمالی است" و در غیر این صورت اگر نام محمد هادی را وارد کرد پیغام "نام خانوادگی شما

خوارزمی است" را چاپ کند . در غیر این صورت هر اسم دیگری را وارد کرد پیغام " نام ناشناس" را چاپ کند.

دستور شرطی match case

با این دستور میتوان شرط برقراری یک متغیر را با یک سری مقادیر مشخص برسی کرد.

شکل ساختاری دستور match case :

match نام متغیری که میخواهیم بررسی کنیم:
case مقدار اول:
دستورات
case مقدار دوم:
دستورات
case_:
دستورات

Match میگه متغیر جلوی منو برسی کن اگه با مقدار هر کدام از case های زیر برابر بود،دستور ان case را انجام بده اگه برابر نبود دستور case_ را انجام بده.

Case اولی میگه اگر متغیر جلوی match با مقدار جلوی من برابر بود،دستورات مرا انجام بده و از ساختار من بیا بیرون اگر با مقدار من برابر نبود برو case بعدی را چک کن.

Case های بعد هم همینطور

ولی اگه هیچکدام از case ها برقرار نبود،دستورات case_ را انجام دهد.

Case_ حکم else را در دستورات شرطی if دارد.

مثال

برنامه ای بنویسید که با دریافت عدد روز هفته نام روز هفته را برای ما چاپ کند.

مثال

مثال نام و نام خانوادگی

گاهی وقت ها ما نیاز داریم چند مقدار را در case ها مورد برسی قرار دهیم.

بنابراین از عملگر های منطقی | که به ان عملگر or و , که به ان عملگر and میگویند استفاده میکنیم.

And:زمانی که همه مقادیر برابر با مقدار یک case باشد دستورات ان case اجرا میشود.

Or:زمانی که یک مقدار از بین مقادیر با case برابر باشد دستورات ان case اجرا میشود.

مثلا اگر خواستیم بگوییم برابر با 3و4و5 بود :

Case 3,4,5:

و اگر بخواهیم بگوییم برابر با 3یا 4 یا 5 بود:

Case 3|4|5:

مثال

برنامه ای بنویسید که یک عدد از ورودی بگیرد اگر ذوج بود چاپ کند ذوج است اگر فرد بود چاپ کند فرد است.

اعداد ذوج تک رقمی؟

اعداد فرد تک رقمی؟

```
num=eval(input('enter a number'))
match num:
    case 1|3|5|7|9:
        print('odd')
    case 2|4|6|8:
        print('even')
    case 0:
        print('zero')
    case _:
        print('just int and 0,1,...,9')
```

نکته :

از انجایی که match case فقط دستور برابر بودن مقادیر را چک میکند اگر نیاز باشد شرط های دیگر برسی شود از if استفاده میکنیم.

مثال

برنامه ای بنویسید که یک عدد از ورودی بگیرد اگر صفر باشد چاپ کند صفر اگر مثبت باشد چاپ کند مثبت در غیر این صورت چاپ کند منفی.

```
num=eval(input('enter a number'))
match num:
    case 0:
        print('zero')
    case num if num>0:
        print('positive')
    case _:
        print('negative')
```

مثال

برنامه ای بنویسید که دو عدد و یک عملوند را دریافت کند و یک ماشین حساب ساده را با چهار عمل - + * / شبیه سازی کند.

مثال

برنامه ای بنویسید که معدل یک دانش اموز را دریافت کند اگر بالای 10 بود پیغام قبول در غیر این صورت پیغام مردود را چاپ کند.

حلقه های تکرار

گاهی وقت ها در برنامه نویسی پیش می اید که مجبوریم یک عمل تکراری را چند بار انجام دهیم .

مثلا چاپ اعداد یک تا صد به صورت جداگانه در یک خط .

برای این کار باید صد تا دستور print بنویسیم و در هرکدام از آنها یک عدد بنویسیم.

خب این کار هم وقت بیشتری از ما میگیرد هم برنامه ما را سنگین میکند و در برنامه های بزرگ سرعت اجرای برنامه ها را پایین می آورد.

برای حل این مشکل از ساختار های تکرار استفاده میکنیم.

حلقه های تکرار یک کد را اجرا میکند و بر میگردد باز کد را از اول اجرا میکند.به اندازه تعداد تکرار کد که خودمان ان را تعیین کردیم دستورات درون حلقه اجرا میشود.

در واقع مانند یک حلقه میچرخد انقد دستورات را اجرا میکند تا تعداد دفعاتی که ما گفتیم تمام شود تا دستورات از حلقه بیرون بیایند.

انواع حلقه تکرار:

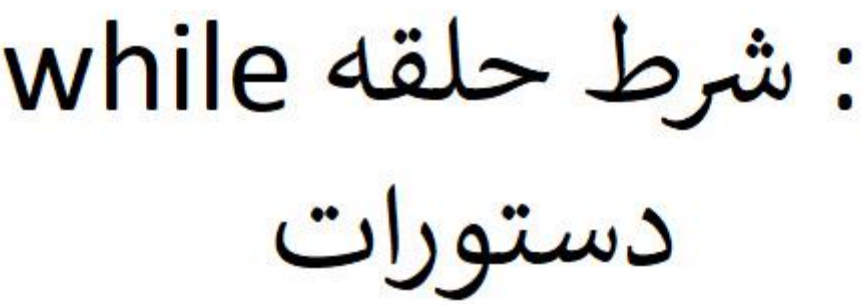
1.حلقه while

2.حلقه for

1.حلقه while

در این ساختار تا زمانی که شرط حلقه به پایان نرسیده باشد دستورات اجرا میشوند.

شکل ساختار حلقه while:



اگر تو رفتگی را رعایت نکنید دستورات داخل حلقه به اندازه شرط اجرا نمیشوند و بعد از اجرای دستورات میرود به دستورات بعد از حلقه.

مثال:

برنامه ای بنویسید تا زمانی که عدد صفر وارد نشده تعدادی عدد از کاربر دریافت کند و ان ها جمع کند و در آخر حاصل جمع اعداد گرفته شده را چاپ کند.

```
suum=0
a=eval(input('enter a number:'))
while a!=0:
    suum+=a
    a=eval(input('enter a number:'))
print('result:',suum)
```

خروجی:

```
enter a number:3
enter a number:2
enter a number:1
enter a number:0
result: 6
```

اولین عدد را دریافت کرد و درون حلقه چک کرد که ایا a صفر نیست؟اگر نیست برو ی بار دیگر شرط را برسی کن و دستورات را اجرا کرد.

اگر شرط جلوی حلقه همیشه برقرار باشد برنامه وارد حلقه بی نهایت یا loop میشود.با ctrl و cمیتوانید از حلقه خارج شوید.

دستور زیر یک حلقه loop یا بی نهایت است:

```
i=1
while i>0:
    print(i)
    i+=1
```

خروجی:

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

تا زمانی که ctrl+c را نگیریم همینطور برای ما عدد چاپ میکند.

مثال:

اعداد 0 تا 30 را چاپ کنید.

2.حلقه for:

زمانی که مجموعه مشخصی از داده ها دارید جهت کار و پیمایش آنها میتوانید از دستور for استفاده کنیم.

مجموعه مشخص :مثلا اعداد بین ده تا بیست

حرف های یک کلمه

...

شکل ساختار حلقه for:

for متغیر شمارنده حلقه in مجموعه داده ها:
دستورات

مثال:

حروف رشته داخل متغیر message را چاپ کنید:

```
message='hello'
for i in message:
    print(i)
```

خروجی:

```
h
e
l
l
o
```

استفاده از تابع range()در حلقه های تکرار

با استفاده از تابع range میتوان مجموعه ای از اعداد صحیح ترتیب دار را تولید کرد.

ورودی این تابع اگر یک عدد صحیح باشد از صفر تا ان عدد را به عنوان ورودی در نظر میگیرد:

```
For n in range(10):
```

اگر دو عدد در نظر بگیرید از عدد اول تا عدد دوم را به عنوان ورودی در نظر میگیرد.به عبارتی اعداد بین آنها را به عنوان ورودی در نظر میگیرد.

```
For n in range(1,10)
```

اگر سه عدد در نظر بگیرید عدد اول نقطه شروع و عدد دوم نفطه پایان و عدد سوم گام حرکت میباشد.

یعنی چند تا چند تا بره جلو.

```
For n in range(1,10,2):
    Print(n)
```

خروجی:

1

3

5

7

9

مثال:

اعداد 99 تا 1 را به صورت نزولی چاپ کنید.

```
for n in range(99,0,-1):
    print(n)
```

ابزار های کنترلی حلقه ها

برای کنترل اجرای دستورات در حلقه های تکرار میتوانید از سه دستور زیر استفاده کنید.

1.break

2.continue

3.pass

1.دستورbreak :

به اجرای حلقه پایان میدهد و از حلقه میاد بیرون و دستورات بعد از حلقه اجرا میشود.

2.دستورcontinue :

فقط در همان دفعه تکرار از حلقه بیرون می آید ولی باز میرود درون حلقه و دستورات را اجرا میکند.

3.دستور pass:

باعث عملیاتی میشود که کاری انجام نمیدهد.با این دستور کاری نداریم.

مثال:

تا زمانی که عدد 0 وارد نشده 10عدد از ورودی بگیرد وحاصل جمع ان هارا چاپ کند.

```
suum=0
for n in range (1,11):
    a=eval(input('enter a number:'))
    if a==0:
        break
    else:
        suum+=a
print('result=',suum)
```

خروجی با وارد کردن 0:

```
enter a number:12
enter a number:112
enter a number:0
result= 124
```

خروجی بدون وارد کردن 0:

```
enter a number:1
enter a number:2
enter a number:3
enter a number:12
enter a number:22
enter a number:125
enter a number:123
enter a number:32
enter a number:33
enter a number:555
result= 908
```

مثال:

اعداد 1 تا 100 را به جز عدد25 چاپ کند.

```
for n in range (1,101):
    if n==25:
        continue
    else:
        print(n)
```

مثال:

برنامه ای بنویسید که اعداد ذوج و فرد اعداد 10 تا 20 را با پیغام مناسب چاپ کند.

```
for n in range(10,21):
    if n%2==0:
        print(n,'is the even number')
    else:
        print(n,'is the odd number')
```

خروجی:

```
10 is the even number
11 is the odd number
12 is the even number
13 is the odd number
14 is the even number
15 is the odd number
16 is the even number
17 is the odd number
18 is the even number
19 is the odd number
20 is the even number
```

مثال:

برنامه ای بنویسید که دو عدد از ورودی دریافت کند اعداد ذوج مابین آن هارا چاپ کند.

```
num1=eval(input('enter first number:'))
num2=eval(input('enter second number:'))
for n in range(num1+1,num2):
    if n%2==0:
        print(n)
    else:
        continue
```

خروجی:

```
enter first number:10
enter second number:20
12
14
16
18
```

توابع

گاهی وقت ها ما یک دستور طولانی را مینویسیم و بعد نیاز داریم چندین و چند بار در طول برناممون از اون دستورات استفاده کنیم و هر جا از برنامه که نیازه ان دستور را از اول مینویسیم یا در بهترین حالت میایم اون دستور بالا را کپی میکنیم و اینجای برنامه ازش استفاده میکنیم.

خب این کار هم برنامه رو سنگین میکنه و زمان زیادی برای اجرای برنامه ها میگیرد هم وقت خودمان را میگیرد و ما باید یک کار تکراری را چند بار انجام دهیم.

بنابر این ما در قالب توابع دستورمان را مینویسیم و بعد در طول برنامه تابع را فراخوانی و ازش استفاده میکنیم.

انواع تابع در پایتون:

1.توابع داخلی پایتون

2.توابع تعریف شده توسط کاربر(خودمان)

1.توابع داخلی پایتون

توابعی هستند که توسط سازندگان زبان پایتون نوشته شده اند تا ما به جای نوشتن کد های طولانی از ان ها برای انجام عملیاتمون استفاده کنیم.

مثلا تابع int() برای تبدیل نوع داده های دیگر به نوع داده int استفاده میشود.

2.توابع تعریف شده توسط کاربر

تعریف توابع در زبان های برنامه نویسی،قابلیتی است که برنامه نویسی را برای برنامه نویسان اسان کرده است.

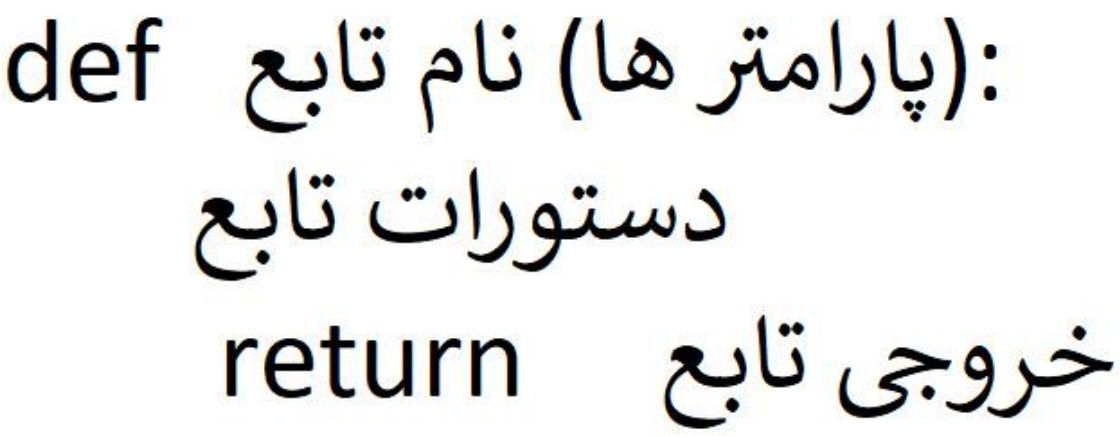
برنامه نویسان میتوانند زمانی که نیاز به نوشتن دستورات به تعداد دفعات زیاد دارند ان دستورات را طبق نیازشان بنویسند و برای دفعه بعد به جای نوشتن ان دستورات فقط ان دستورات را فراخوانی کنند.

تغریف تابع

برای تعریف تابع از کلمه کلیدی def استفاده میکنیم و برای برگرداندن نتیجه در تعریف تابع از کلمه کلیدی return استفاده میکنیم.

نکته:بعد از return دیگر دستوری درون تابع نمیتوانیم بنویسیم چون رسما کار تابع را با return تمام میکنیم.

شکل ظاهری تعریف تابع:



کار تابع زیر چاپ hello با نامی است که وارد میکنیم:

```
def sayhello(name):
    return 'hello '+name
```

فراخوانی توابع

زمانی که یک تابع نوشته میشود باید نام آن را به همراه آرگومان هایش صدا زدکه به این عمل فراخوانی تابع میگویند.

مثال

تابعی بنویسید که به نامی که تعیین میکنید سلام کند.

```
def sayhello(name):
    return 'hello '+name
print(sayhello('zahra'))
```

خروجی

```
1.py
hello zahra
```

مثال

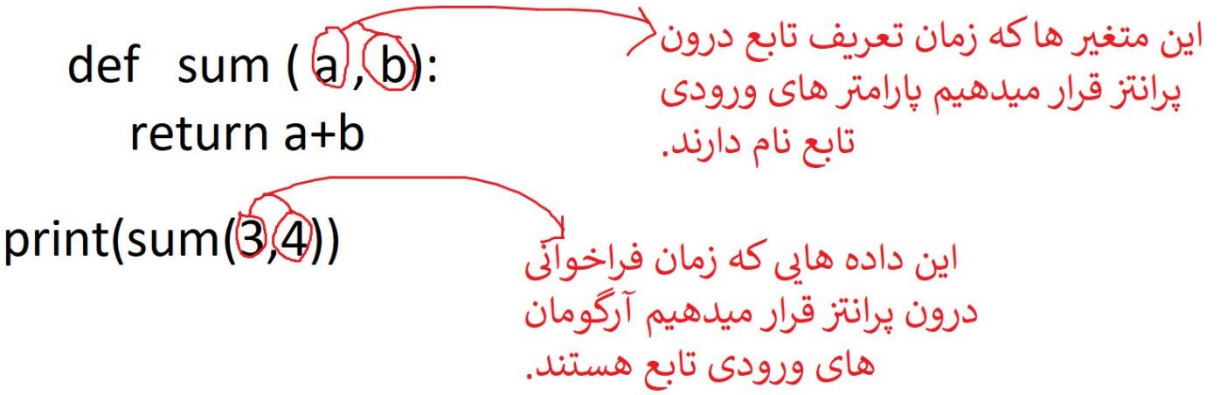
تابعی بنویسید که کارش چاپ hello word باشد.

```
def sayhello():
    print('hello word')
sayhello()
```

خروجی

```
hello word
```

تفاوت پارامتر و آرگومان



در واقع پارامتر ها متغیر های تابع هستند که آرگومان هایی ک موقع فراخوانی به تابع میدهیم درون پارامتر ها قرار میگیرند.

نکته:

زمانی که درون یک تابع از return استفاده میکنیم باید برای فراخوانی آن تابع را درون print قرار دهیم تا نمایشش بده.

ولی وقتی درون تابع از print استفاده کنیم نیازی به فراخوانی تابع با print نیست همان نام تابع با پرانتز باز و بسته جلویش کافی است.

مثال

نام کاربر را از ورودی دریافت کنید و تابعی بنویسید که چک کند اگر نامی که از ورودی گرفتید برابر Zahra بود hello admin را برگرداند در غیر این صورت هر اسمی که دادیم hello user را برگرداند.

```
username=eval(input('enter user name:'
def admincheck(name):
    if name=='zahra':
        return 'hello admin'
    else :
        return 'hello user'
print(admincheck(username))
```

خروجی

```
enter user name:'zahra'
hello admin

= RESTART: C:/Users/E-Pouya/Desktop/p
1.py
enter user name:'omid'
hello user
```

پارامتر با مقدار پیشفرض

هنگام تعریف تابع میتوان به پارامتر های ورودی مقدار پیشفرض داد که زمان فراخوانی تابع اگر مقداری به عنوان آرگومان ندادیم خودش به صورت پیشفرض مقدار مشخص خودش را داشته باشه.

به دستورات زیر توجه کنید:

```
def sum(a,b=0):
    return a+b
print('result is:',sum(9))
```

خروجی:

```
result is: 9
```

```
def sum(a=0,b=0):
    return a+b
print('result is:',sum())
```

خروجی

```
1.py
result is: 0
```

پارامتر به تعداد نامشخص

زمانی که تعداد آرگومان های ارسالی به تابع نامشخصه زمان تعریف تابع قبل نام پارامتر علامت * میگذاریم.

به دستورات زیر دقت کنید:

```
def suum(*a):
    summ=0
    for n in a:
        summ+=n
    return summ
print(suum(3,4,5,8,9,55,4,7,5,125))
```

فراخوانی یک تابع در تابع دیگر

ما میتوانیم در یک تابع از تابع های دیگر هم استفاده کنیم.

فقط کافیست آن هارا درون تابعی که میخواهیم فراخوانی کنیم.

مثال

برنامه ای بنویسید با استفاده از دو تابع در ابتدا معدل سه نمره ورودی را محاسبه کند،در تابع دوم با استفاده از تابع معدل پیغام های زیر را چاپ کند:

اگر معدل بزرگتر از 17 بود پیغام "عالی" ، در غیر این صورت اگر معدل بین 17 و 15 بود (کوچکتر مساوی 17 و بزرگتر مساوی 15) پیغام "خوب" ،در غیر این صورت اگر بین 10 تا 14 باشد(بزرگتر مساوی 10 و کوچکتر از 15) پیغام "متوسط" و در غیر این صورت اگر کوچکتر از 10 بود پیغام مردود را چاپ کند.

```
def avg(a,b,c):
    return (a+b+c)/3
def mesage():
    if avg(a,b,c)>17:
        return 'very good'
    elif  avg(a,b,c)>=15 and avg(a,b,c)<=17:
        return 'good'
    elif avg(a,b,c)<15 and avg(a,b,c)>=10:
        return 'medium'
    elif avg(a,b,c)<10:
        return 'fail'
    else :
        return 'score is not available'
a,b,c=eval(input('enter 3 score :'))
print(mesage())
```

خروجی

```
top/p1.py ====================
enter 3 score :19,18,17
very good

===== RESTART: C:/Users/E-Pouya/Desktop/p1.py =====
enter 3 score :9,7,8
fail

===== RESTART: C:/Users/E-Pouya/Desktop/p1.py =====
enter 3 score :10,11,12
medium

===== RESTART: C:/Users/E-Pouya/Desktop/p1.py =====
enter 3 score :16,17,3
medium

===== RESTART: C:/Users/E-Pouya/Desktop/p1.py =====
enter 3 score :14,15,16
good
```

انواع داده های مجموعه ای

در جلسات قبل با انواع داده ها اشنا شدیم.داده هایی مانند string int float eval

در پایتون انواع داده های دیگر هم وجود دارد که به آن ها داده های مجموعه ای میگوییم.

داده هایی که به صورت مجموعه ای از داده ها در کنار هم در قالب یک لیست یا مجموعه و... در کنار هم درون یک متغیر قرار میگیرند.

انواع داده های مجموعه ای

لیست ها

تاپل ها

دیکشنری ها

مجموعه ها

لیست ها

لیست ها انواع داده های متفاوت در کنار هم هستند که میتوان با استفاده از اندیس ها به هر یک از آن ها دسترسی داشت.

لیست ها به صورت زیر تعریف میشوند.

[..., آیتم سوم,آیتم دوم,آیتم اول]= نام لیست

نکته: آیتم های لیست میتوانند هم نوع نباشند.

```
name='zahra'
list1=[20,32.5,'python',22222,name]
```

برای دسترسی به عناصر لیست ها باید از شماره اندیس آیتم های آن استفاده کنیم.

```
name='zahra'
list1=[20,32.5,'python',22222,name]
print(list1[0])
print(list1[1])
print(list1[2])
print(list1[3])
print(list1[4])
```

خروجی:

```
20
32.5
python
22222
zahra
PS D:\project\py>
```

برای پاک کردن لیست و عناصر آن از دستور del استفاده میکنیم.

```
name='zahra'
list1=[20,32.5,'python',22222,name]
del list1[0]
print(list1)
```

خروجی:

```
[32.5, 'python', 22222, 'zahra']
PS D:\project\py>
```

انتساب چندگانه

لیست ها این امکان را دارند که بتوانند آیتم های خود را به چند متغیر جدید انتساب دهند.به این عمل unpacking میگویند.

```
list1=[20,32.5,'python']
a,b,c=list1
print(a)
print(b)
print(c)
```

خروجی:

```
20
32.5
python
PS D:\project\py>
```

معمولا تعداد متغیر ها باید برابر با تعداد آیتم های درون لیست باشد ام اگر کمتر بود با گذاشتن *قبل از متغیر آخر بقیه آیتم های لیست درون آن متغیر ریخته میشوند.

```
list1=[20,32.5,'python',2548,4458,'hello','world']
a,b,c,*d=list1
print(a)
print(b)
print(c)
print(d)
```

خروجی:

```
20
32.5
python
[2548, 4458, 'hello', 'world']
PS D:\project\py>
```

لیست های تو در تو

امکان دیگری که پایتون دارد تعریف و استفاده از لیست های تو در تو است.

لیستی که داخل لیست دیگر قرار دارد لیست تو در تو است.

```
list1=[20,32.5,'python',[2548,4458,'hello'],'world']
print(list1)
```

خروجی:

```
n.exe" d:/project/py/py.py
[20, 32.5, 'python', [2548, 4458, 'hello'], 'world']
PS D:\project\py>
```

برای دسترسی به آیتم های لیست های تو در تو باید از الگوی list1[i][j] استفاده کنیم.

i عنصر لیست اصلی و j عنصر لیست داخلی است.

```
list1=[20,32.5,'python',[2548,4458,'hello'],'world']
print(list1[3][2])
```

خروجی:

```
hello
PS D:\project\py>
```

پیمایش لیست ها

برای پیمایش لیست ها میتوانید از حلقه for یا حلقه while استفاده کنید.

پیمایش لیست ها با حلقه for

```
list1=['zahra','rezazade',21]
for i in list1:
    print(i)
```

خروجی:

```
zahra
rezazade
21
PS D:\project\py>
```

تابع len()

تابع len() تعداد آیتم های درون لیست ها را میشمارد.

```
list1=['zahra','rezazade',21]
print(len(list1))
```

خروجی:

```
3
PS D:\project\py>
```

پیمایش لیست ها با حلقه while

برای پیمایش لیست ها با حلقه while باید با استفاده از یک شمارنده (نقطه شروع) از ابتدا تا انتهای لیست را پیمایش کنیم.

```
list1=['zahra','rezazade',21]
i=0
while i<len(list1):
    print(list1[i])
    i=i+1
```

خروجی

```
zahra
rezazade
21
PS D:\project\py>
```

مثال:

اعداد ذوج درون لیست زیر را چاپ کنید.

List1=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]

با حلقه for

```
list1=[21,22,51,45,69,50,220,114,13.5,982]
for i in list1:
    if i%2==0:
        print(i)
```

خروجی:

```
22
50
220
114
982
PS D:\project\py>
```

با حلقه while

```
list1=[21,22,51,45,69,50,220,114,13.5,982]
i=0
while i<len(list1):
    if list1[i]%2==0:
        print(list1[i])
    i=i+1
```

خروجی:

```
22
50
220
114
982
PS D:\project\py>
```

توابع مهم لیست ها

تابع() len:

تعداد آیتم های درون یک لیست را بر میگرداند.

تابع()append :

با استفاده از این تابع میتوان یک آیتم را به انتهای لیست اضافه کرد .

```
list1=['hello world','python','c#']
list1.append('javascript')
print(list1)
```

خروجی:

```
['hello world', 'python', 'c#', 'javascript']
PS D:\project\py>
```

تابع insert():

این تابه یک آیتم را درشماره اندیسی که خودمان بهش میدهیم به لیست اضافه میکند.

```
list1=['hello world','python','c#']
list1.insert(0,'kotlin')
print(list1)
```

خروجی:

```
['kotlin', 'hello world', 'python', 'c#']
PS D:\project\py>
```

تابع remove()

به آن آیتمی میدهیم و ان آیتم اگر در لیست باشد آن را از لیست حذف میکند.

```
list1=['hello world','python','c#']
list1.remove('python')
print(list1)
```

خروجی:

```
['hello world', 'c#']
PS D:\project\py>
```

تابع () count:

این تابع تعداد تکرار یک ایتم را درون لیست مشخص میکند.

```
list1=['hello world','c#','c++','python','c#']
print(list1.count('c#'))
```

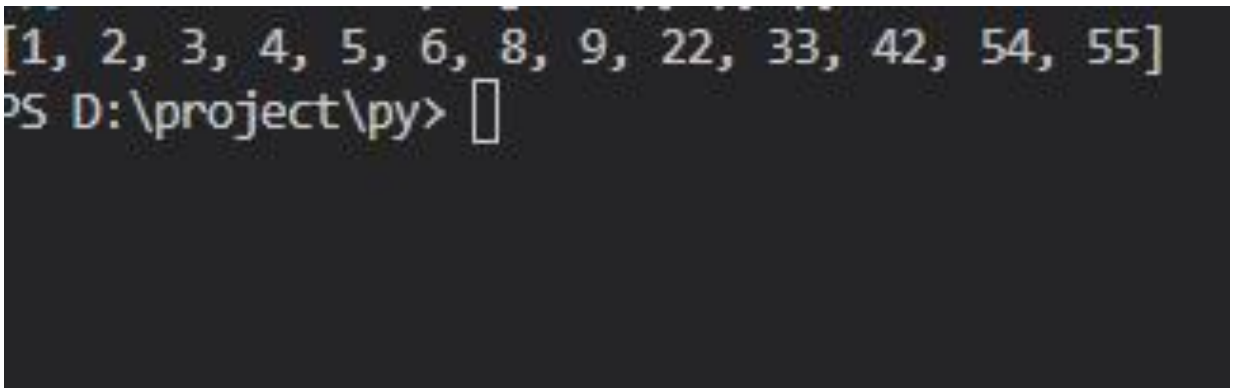
دوتا از آیتم های داخل لیست ما c# بود پس در خروجی به ما 2 را برگرداند.

تابع () sort:

آیتم های درون لیست را به ترتیب خاصی مرتب میکند .اگر عدد باشند به ترتیب اعدا و اگر string باشند به ترتیب حروف الفبا مرتبشان میکند.

```
list1=[22,54,8,5,4,6,9,3,2,55,1,42,33]
list1.sort( )
print(list1)
```

خروجی:



تابع () reverse:

این تابع آیتم های درون لیست را براساس ترتیبشان بر عکس میکند.

```
list1=[22,54,8,5,4,6,9,3,2,55,1,42,33]
list1.reverse()
print(list1)
```

خروجی:

```
[33, 42, 1, 55, 2, 3, 9, 6, 4, 5, 8, 54, 22]
PS D:\project\py>
```

تابع index ():

این تابع یک آیتم میگیرد و شماره اندیس آن را برمیگرداند.

```
list1=['python','heml','css','javascript','c#']
print(list1.index('c#'))
```

خروجی:

```
4
PS D:\project\py>
```

تابع min():

این تابع از بین اعداد درون لیست کوچکترینش را بر میگرداند و اگر درون لیست ما string باشد آیتمی را با کوچکترین حروف الفبا بر میگرداند.

```
list1=['python','heml','css','javascript','c#']
list2=[3,6,3,2,5,8,2258,4,1,52,5556]
print(min(list1))
print(min(list2))
```

خروجی:

```
c#
1
PS D:\project\py>
```

تابع() max:

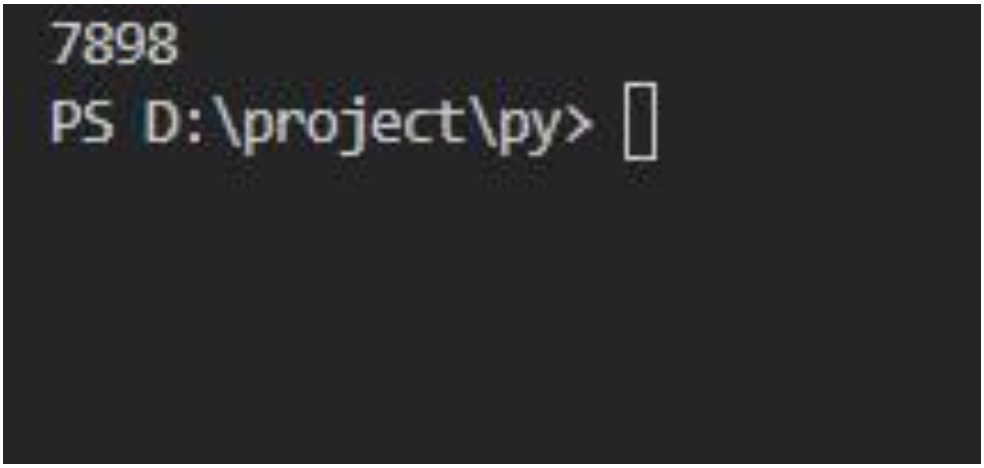
از بین اعداد درون لیست بزرگترینش را بر میگرداند اگر آیتم های ما stringبودند آن آیتم با بزرگ ترین حروف الفبا را برمیگرداند.

تابع ()sum:

این تابع آیتم های درون یک لیست را باهم جمع میکند و حاصل را برمیگرداند.

```
list1=['python','heml','css','javascript','c#']
list2=[3,6,3,2,5,8,2258,4,1,52,5556]
print(sum(list2))
```

خروجی:



تاپل ها tuple

یکی دیگر از انواع داده های مجموعه ای تاپل ها هستند.مانند لیست ها هستند با تفاوت های زیر:

1.تاپل ها با () تعریف میشوند.

2.آیتم های لیست را میتوانستیم تغییر دهیم اما تاپل ها را نمیتوانیم تغییر دهیم(به داده های تغییر نا پذیر داده های فقط خواندنی یا read only میگویند.)

شکل ظاهری تعریف کردن تاپل

Mytpl=(item1,item2,...)

```
mynum=eval(input('enter a number:'))
mytpl=('zahra',21,'kerman',mynum)
print(mytpl)
```

خروجی:

```
enter a number:25
('zahra', 21, 'kerman', 25)
PS D:\project\py>
```

البته آیتم های تاپل میتوانند بدون پرانتز تعریف شوند اما باید بین آن ها کاما باشد:

```
mynum=eval(input('enter a number:'))
mytpl='zahra',21,'kerman',mynum
print(mytpl)
```

خروجی:

```
enter a number:25
('zahra', 21, 'kerman', 25)
PS D:\project\py>
```

نکته:

اگر تاپل شما تک عضوی است حتما بعد از آیتم آن کاما بگذارید وگرنه پایتون آن را با نوع داده ای دیگر اشتباه میگیرد.

با گذاشتن کاما:

```
mytpl=(32,)
print(type(mytpl))
```

خروجی:

```
PS D:\project\py> & "C:/Program Files/Pyt
/python.exe" d:/project/py/py.py
<class 'tuple'>
PS D:\project\py>
```

بدون گذاشتن کاما:

```
mytpl=(32)
print(type(mytpl))
```

خروجی:

```
<class 'int'>
PS D:\project\py>
```

برای دسترسی به عناصر تاپل ها مانند لیست ها از اندیس ها درون [] استفاده میشود:

```
mytpl= ('zahra','kerman','python',21)
print(mytpl[0])
```

خروجی:

دیکشنری ها

دیکشنری ها نوع دیگر از انواع داده های مجموعه ای هستندکه از دو بخش کلید (key) و مقدار (value) تشکیل شدند.

میتوانیم به هر مقدار از طریق کلید آن دسترسی داشته باشیم.

توجه داشته باشید که هر مقدار تنها با کلید منحصر به فرد خودش قابل دسترسی است.

در تعریف یک دیکشنری باید دانست که کلید یک نام منحصر به فرد است که نباید تکراری باشد ولی مقدار آن میتواند از هر نوع داده باشد و اشکال ندارد که تکراری باشد.

برای تعریف دیکشنری از علامت {} استفاده میکنیم.

شکل ظاهری دیکشنری به شکل زیر است:

```
نام دیکشنری={
        کلید اول:مقدار اول,
        کلید دوم:مقدار دوم,
        ...
        }
```

```
person1={
      'name':'zahra',
      'family':'rezazade',
      'age':21
}
person2={
     'name':'ehsan',
      'family':'rezazade',
      'age':22
}
```

دسترسی به عناصر دیکشنری

برای دسترسی به عناصر دیکشنری از نام دیکشنری به همراه [] استفاده میکنیم که داخل [] نام کلید مان را مینویسیم.

مثال :

نام person1 و person2 را چاپ کنید.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
   'name':'ehsan',
    'family':'rezazade',
    'age':22
}
print('person1-name:',person1['name'])
print('person2-name:',person2['name'])
```

خروجی:

```
person1-name: zahra
person2-name: ehsan
PS D:\project\py>
```

همچنین میتوانید میقدار یک کلید را تغییر دهید:

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
   'name':'ehsan',
    'family':'rezazade',
    'age':22
}
person1['age']=22
print(person1['age'])
```

خروجی:

حذف عناصر دیکشنری

برای حذف عناصر دیکشنری میتوانید از کلمه کلیدی del استفاده کنید.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
   'name':'ehsan',
    'family':'rezazade',
    'age':22
}
del person1['name']
print(person1)
```

خروجی:

```
{'family': 'rezazade', 'age': 21}
PS D:\project\py>
```

با نوشتن دستور بالا کلید name با مقدارش حذف شد.

با استفاده از تابع clear() میتوانید همه کلید ها و مقدار شان را حذف کنید در واقع با این کار دیکشنری خالی میشود اما حذف نمیشود.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
    'name':'ehsan',
    'family':'rezazade',
    'age':22
}
person1.clear()
print(person1)
```

برای این که یک دیکشنری کاملا حذف شود از کلمه کلیدی del استفاده کنید.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
   'name':'ehsan',
    'family':'rezazade',
    'age':22
}
del person1
print(person1)
```

خروجی:

```
Traceback (most recent call last):
  File "d:\project\py\py.py", line 12, in <module>
    print(person1)
NameError: name 'person1' is not defined. Did you mean: 'person2'?
PS D:\project\py>
```

همانطور که دیدید به ما error داد و میگوید person1 وجود ندارد.

توابع و عملگر ها در دیکشنری ها

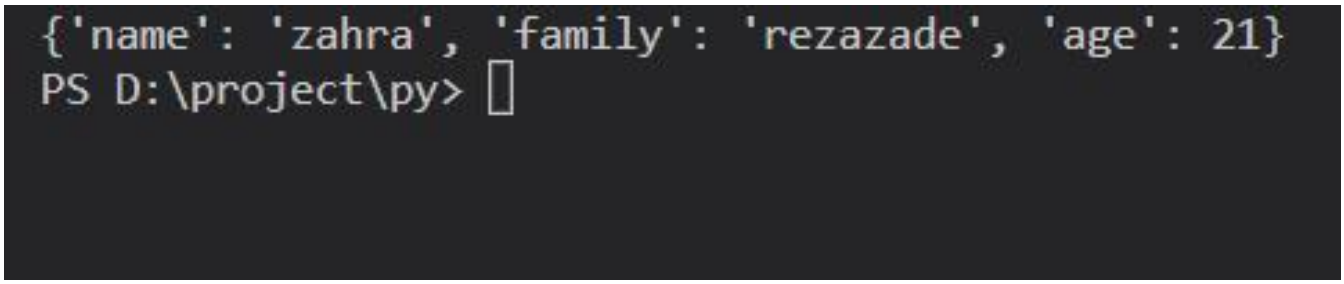
علاوه بر توابع len و type وin و * که در قسمت های قبل توضیح دادیم تابع های دیگری هم وجود دارند که برای استفاده از دیکشنری ها از آن ها استفاده میکنیم.

تابع str()

این تابع یک دیکشنری را میگیرد و به string تبدیلش میکند.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
   'name':'ehsan',
    'family':'rezazade',
    'age':22
}
print(str(person1))
```

خروجی:



تابع ()copy

ب استفاده از این تابع میتوانید یک کپی از دیکشنری تان بگیرید و درون یک متغیر دیگر بریزید.

تابع ()get

این تابع یک کلید را میگیرد و چک میکند در صورت وجود کلید آن را بر میگرداند در غیر این صورت پارامتر دوم این تابع که یک پیغام است به جای آن تابع بر میگردد.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
   'name':'ehsan',
    'family':'rezazade',
    'age':22
}
print(person1.get('city','not found'))
```

خروجی:

```
not found
PS D:\project\py>
```

از آنجایی که ما کلیدی به نام city در person1 نداشتیم به ما not foundرا برگرداند.

تابع keys()

این تابع لیستی از کلید های درون یک دیکشنری را برمیگرداند.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
   'name':'ehsan',
    'family':'rezazade',
    'age':22
}
print(person1.keys())
print(person2.keys())
```

خروجی:

```
dict_keys(['name', 'family', 'age'])
dict_keys(['name', 'family', 'age'])
PS D:\project\py>
```

تابع values()

این تابع لیستی از مقدار ها را برایتان بر میگرداند.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
    'name':'ehsan',
    'family':'rezazade',
    'age':22
}
print(person1.values())
print(person2.values())
```

خروجی:

```
dict_values(['zahra', 'rezazade', 21])
dict_values(['ehsan', 'rezazade', 22])
PS D:\project\py>
```

تابع items()

این تابع لیستی از کلید و مقدار را برایتان برمیگرداند.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
    'name':'ehsan',
    'family':'rezazade',
    'age':22
}
print(person1.items())
print(person2.items())
```

خروجی:

```
dict_items([('name', 'zahra'), ('family', 'rezazade'), ('age', 21)])
dict_items([('name', 'ehsan'), ('family', 'rezazade'), ('age', 22)])
PS D:\project\py>
```

تابع update()

این تابع مقادیر دیکشنری که میگیرد را درون دیکشنری اول میریزد.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
   'name':'ehsan',
    'family':'rezazade',
    'age':22
}
person1.update(person2)
print(person1)
```

خروجی:

```
py.py
{'name': 'ehsan', 'family': 'rezazade', 'age': 22}
PS D:\project\py>
```

تابع pop()

با استفاده از این تابع میتوان مقدار کلید مشخص شده را پاک کرد.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
   'name':'ehsan',
    'family':'rezazade',
    'age':22
}
person1.pop('name')
print(person1)
```

خروجی:

```
{'family': 'rezazade', 'age': 21}
PS D:\project\py>
```

پیمایش دیکشنری ها

به سه روش میتوان دیکشنری ها را پیمایش کرد.

1.فقط پیمایش کلید ها

2.فقط پیمایش مقدار ها

3.هم پیمایش کلید ها و هم پیمایش مقدار ها

پیمایش کلید ها

برای پیمایش کلید ها از حلقه for و تابع ()keys استفاده میکنیم.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
   'name':'ehsan',
    'family':'rezazade',
    'age':22
}
for i in person1.keys():
    print(i)
```

خروجی:

```
name
family
age
PS D:\project\py>
```

پیمایش مقدار ها

برای پیمایش مقدار ها از حلقه for و تابع ()values استفاده میکنیم.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
   'name':'ehsan',
    'family':'rezazade',
    'age':22
}
for i in person1.values():
    print(i)
```

خروجی:

```
zahra
rezazade
21
PS D:\project\py>
```

پیمایش کلید ها و مقادیر

برای پیمایش کلید ها و مقادیر میتوان از حلقه for و تابع items() استفاده کنید.

```
person1={
    'name':'zahra',
    'family':'rezazade',
    'age':21
}
person2={
   'name':'ehsan',
    'family':'rezazade',
    'age':22
}
for i,j in person1.items()
    print(i,j)
```

تابع type()

در زبان پایتون برای فهمیدن نوع داده یک متغیر از تابع type استفاده میکنیم.

```
name='zahra'
age=21
fav=['python','c#','HTML','CSS','Java script']
print(type(name))
print(type(age))
print(type(fav))
```

خروجی

```
PS D:\project\py> & "C:/Program Files/Python310/pyt
ct/py/p1.py
<class 'str'>
<class 'int'>
<class 'list'>
PS D:\project\py>
```

برای تبدیل انواع داده ها به هم :

```
age='21'
age=int(age)
print(type(age))
```

خروجی:

```
<class 'int'>
PS D:\project\py>
```

جدول انواع داده ها

داده های عددی	اعداد صحیح integer	850 30
	اعداد اعشاری float	12.5 33.75
	اعداد مختلط complex	3.14j
داده های منطقی	boolean	13>5 true 13<5 false
داده های رشته ای	string	‘zahra’ “Zahra”
لیست ها	List	[‘zahra’,21]
چند تایی	Tuple	(‘zahra’,21)
مجموعه ها	set	{2,4,5,8,7} {‘zahra’,21}
واژه نامه ها	dictionary	{‘name’:’zahra’, ‘age’:21}