زبان پایتون (Python)

زبان پایتون از اعداد 0…9 حروف A…Z و a…z و علائم ویژه ای مانند : { } [ ] \ /) (_ و... تشکیل شده است.

مقدمه ای بر متغیر (variable)

یک از اصلی ترین مباحث در زبان های برنامه نویسی متغیرهاست، متغیرها محفظه هایی برای ذخیره مقادیر داده ها هستند. به بیان دیگر متغیرها **ظرفهایی از خانه های حافظه** هستند. در واقع متغیر نامی است برای یک مکان از حافظه که ممکن است در طول اجرای برنامه مقدار آن تغییر کند.

مثال

```
# variable name = value
y = "Hello, World!"
x = 5
```

مقدمه ای بر تابع print()

برای نمایش اطلاعات روی صفحه نمایشگر از تابعی به نام تابع print استفاده میکنیم.لازم به ذکر است که در پایتون دستورات از بالا به پایین آن هم بصورت خط به خط بررسی میشوند.

مثال

```
y = "Hello, World!"
x = 5
Print("Hello, World!")     #print(Something to be printed)
Print(5)
Print(y)
Print(x)
```

RUN

```
Hello, World!
5
Hello, World!
5
```

مقدمه ای بر رشته(string) :

رشته توالی از کاراکترها است. اگر بخواهیم یک جمله ای (رشته ای) بنویسیم و آنرا در یک متغیر ذخیره کنیم باید از رشته استفاده کنیم. بطور کلی هر زمان که از رشته ها در هرجایی که استفاده میکنیم باید آن رشته را در داخل دبل کوتیشن یا تک کوتیشن (single or double quotes) قرار دهیم. یعنی داخل " " یا داخل ' ' قرار دهیم.(هیچ تفاوتی ندارد که از دبل کوتیشن یا تک کوتیشن استفاده کنیم).

مثال

```
x = "Amir"          #variable name = "string"
print(x)
y = 'Amir'
print(y)
print("x")
```

RUN

```
Amir
Amir
x
```

ایجاد متغیرها

پایتون هیچ دستوری (command) برای اعلان(declaring) متغیر ندارد.یک متغیر در لحظه ای ایجاد می شود که برای اولین بار یک مقدار به آن اختصاص می دهید.برای ایجاد یک متغیر کافی است یک نام انتخاب کنیم (با استفاده از قوانین نام گذاری که بعدا خواهیم آموخت) و سپس آنرا مقداردهی کنیم.

مثال

```
x = 5               #variable name = value
y = "Amir"
print(x)
print(y)
```

RUN

```
5
Amir
```

متغیرها نیاز به مشخص کردن datatype (نوع داده، که بعدا راجع به این موضوع مفصلا بحث می کینم) ندارند. و حتی یک متغیر در طول یک برنامه می تواند برحسب داده ای که در آن قرار می گیرد datatype های مختلفی داشته باشد.

مثال

```
x = 4
x = "Sally"
print(x)
```

RUN

```
Sally
```

حساسیت به حروف کوچک و بزرگ (case-sensitive)

پایتون به حروف بزرگ و کوچک حساس است.

مثال

```
a = 4
A = " Sally "
print(a)
print(A)
```

RUN

```
4
Sally
```

نام متغیرها(variable names)

نام متغیرها

یک متغیر می تواند یک نام کوتاه مانند x و y یا یک نام بلندتر (age, carname, total_volume) داشته باشد. (اهمیتی ندارد که اسم متغیر بامعنی یا بی معنی باشد اما اگر بامعنی باشد بهتر است)

قوانین برای متغیرهای پایتون :

- نام متغیر فقط می‌تواند شامل کاراکتر های حروف ، اعداد و underscore باشد(a-z ، 0-9 ، _).
- نام متغیر باید با یک حرف یا کاراکتر underscore شروع شود.
- نام متغیر نمی تواند با یک عدد شروع شود.

- نام متغیر به حروف بزرگ و کوچک حساس است (age, Age , AGE) سه متغیر متفاوت هستند. (case-sensitive)
- در بین نام یک متغیر نمی توان از فاصله (space ، tab و...) استفاده کرد
- نام متغیر نمی تواند هیچ یک از کلمات کلیدی (کلماتی که در زبان برنامه نویسی دارای مفهوم خاصی هستند) پایتون باشد.

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

مثال

نمونه هایی از نام های مجاز برای متغیر:

```
myvar = "Amir"
my_var = "Amir"
_my_var = "Amir"
myVar = "Amir"
MYVAR = "Amir"
myvar2 = "Amir"
print(myvar)
print(my_var)
print(_my_var)
```

```
print(myVar)
print(MYVAR)
print(myvar2)
```

RUN

```
Amir
Amir
Amir
Amir
Amir
Amir
```

مثال

نمونه هایی از نام های غیرمجاز برای متغیر:

```
2myvar = "Amir"
my-var = "Amir"
my var = "Amir"
if = "Amir"
```

یک نام متشکل از چند کلمه برای متغییر:

نام متغیرهایی با بیش از یک کلمه ممکن است به سختی خوانده شوند.چند تکنیک وجود دارد که می توانید برای خوانایی بیشتر آنها استفاده کنید:

کیس مار(snake case)

هر کلمه با یک کاراکتر underscore جدا می شود:

```
my_variable_name = "Amir"
```

کیس پاسکال(pascal case)

هر کلمه با یک حرف بزرگ شروع می شود:

```
MyVariableName = "Amir"
```

کیس شتر (camel case)

هر کلمه، به جز کلمه اول، با یک حرف بزرگ شروع می شود:

```
myVariableName = "Amir"
```

نظرات (comments)

گاهی اوقات میخواهید توضیح یا نظری یا جمله ای در برنامه بنویسید اما نمیخواهید آن توضیح اجرا شود. به این توضیح، کامنت گویند. پایتون نیز دارای قابلیت اظهارنظر(commenting capability)است. کامنت ها با # شروع می شوند و پایتون بقیه خط را به عنوان کامنت نمایش می دهد در واقع پایتون کامنت ها را نادیده می گیرد:

مثال

```
#This is a comment.
print("Hello, World!")
```

RUN

```
Hello, World!
```

- از کامنت ها می توان برای توضیح کد پایتون استفاده کرد.
- برای خوانایی بیشتر کد می توان از کامنت ها استفاده کرد.
- از کامنت ها می توان برای جلوگیری از اجرا در هنگام تست کد استفاده کرد.

مثال

```
#tozihe code : salam donya
print("Hello, World!")
```

RUN

```
Hello,World!
```

کامنت ها را می توان در هرجایی از یک خط کد قرار داد و پایتون بقیه آن خط کد را نادیده می گیرد:

مثال

```
print("Hello, World!")     #az inja be baad comment ast
```

RUN

```
Hello,World!
```

یک کامنت لازم نیست متنی باشد که کد را توضیح می دهد، همچنین می تواند برای جلوگیری از اجرای بخشی از کد استفاده شود:

مثال

```
# baraye jlogiri az ejra ye code
```

```
#print("Hello, World!")
print("Checkmate!")
```

RUN

```
Checkmate!
```

کامنت های چند خطی(Multiline Comments)

پایتون سینتکسی برای کامنت های چند خطی ندارد.برای افزودن یک کامنت چند خطی می توانید یک # برای هر خط یک کامنت درج کنید:

مثال

```
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```

RUN

```
Hello,World!
```

همچنین برای کامنت چند خطی می توانید از یک رشته چند خطی استفاده کنید.

برای درج یک رشته چند خطی از triple quotes (یعنی ''' یا """) استفاده می کنیم و رشته را درون آن قرار میدهیم.

از آنجایی که پایتون رشته هایی را که به یک متغیر اختصاص داده نمی شود نادیده می گیرد؛ می توانید یک رشته چند خطی (با استفاده از علامت نقل قول سه گانه (triple quotes)) را در کد اضافه کنید و کامنت خود را در آن قراردهید:

مثال

```
"""
This is a comment
written in
more than just one line
"""

print("Hello, World!")
'''
This is a comment
written in
```

```
more than just one line
'''
```

RUN

```
Hello, World!
```

تا زمانی که رشته به متغیری اختصاص داده نشده باشد، پایتون کد(رشته) را می خواند، اما آن را نادیده می گیرد و شما یک کامنت چند خطی نوشته اید.

اختصاص چندین مقدار(Assign Multiple Values)

در مثال های قبل برای مقداردهی هر متغیر از یک خط جداگانه استفاده میکردیم اما گاهی نیاز است با استفاده از یک دستور چند متغیر را مقداردهی کنیم.

اختصاص چند مقدار به چند متغییر

این امکان وجود دارد که در یک خط به **چندین متغیر** ، **چند مقدار** را اختصاص دهید:

مثال

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

RUN

```
Orange
Banana
Cherry
```

باید توجه داشته باشید که تعداد متغیرها با تعداد مقادیرمطابقت داشته باشد، در غیر این صورت با خطا مواجه خواهید شد.

اختصاص یک مقدار به چندین متغیر

همچنین این امکان وجود دارد که **یک مقدار** را به **چند متغیر** در یک خط اختصاص دهید:

مثال

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

RUN
Orange
Orange
Orange

Unpacking a Collection

اگر مجموعه ای از مقادیر در یک لیست، تاپل و غیره دارید، میتوانید مقادیر را در متغیرها استخراج کنید. به این کار unpacking می گویند .

مثال

```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits        # x, y, z = ["apple", "banana", "cherry"]
print(x)
print(y)
print(z)
```

RUN
apple
banana
cherry

خروجی متغیرها(Output Variables)

خروجی متغییرها

تابع print() برای نمایش خروجی استفاده می شود.

مثال

```
y="hello"
x = "Python is awesome"
z=256
print(x)
print(y)
print(z)
print("     ")
print("z")
```

RUN
hello

Python is awesome

256

z

در مثال بالا هر تابع print() را در یک خط فراخوانی کرده ایم برای همین درهنگام نمایش، خروجی مربوط به هرکدام در یک خط چاپ شده است. اما این امکان وجود دارد که در یک تابع print() در یک سطر ، چند متغیر(یا مقدار) را با کاما (,) از هم جدا کنیم، در اینصورت همه در یک سطر چاپ میشوند. (البته چون کاما قرار داده ایم با فاصله چاپ میشوند)

مثال

```
x = "Python"
y = "is"
z = "awesome"
print(x, y, z)
```

RUN
Python is awesome

همچنین می توانید از عملگر + برای خروجی چندین متغیر استفاده کنید:

مثال

```
x = "Python"
y = "is"
z = "awesome"
print(x + y + z)
```

RUN
Pythonisawesome

مثال

```
x = "Python "
y = "is "
z = "awesome"
print(x + y + z)
```

RUN
Python is awesome

به کاراکتر space بعد از " Python "و " is"، توجه کنید بدون آنها نتیجه"Pythonisawesome" خواهد بود.

برای اعداد، کاراکتر+ به عنوان یک عملگر ریاضی عمل می کند:

مثال

```
x = 5
y = 10
print(x + y)
```

RUN
15

در تابع()print ، وقتی می خواهید یک رشته و یک عدد را با عملگر + ترکیب کنید، پایتون به شما خطا می دهد:

مثال

```
x = 5
y = "Amir"
print(x + y)
```

بهترین راه برای خروجی چند متغیر در تابع ()print جدا کردن آنها با کاما است که حتی از انواع داده های مختلف پشتیبانی می کند:

مثال

```
x = 5
y = "Amir"
print(x, y)
```

RUN
5 Amir

مثال

```
name="ali"
Lname="ahmadi"
#age=29
```

```
print("my name is:" , name , Lname , "and I am" , 29 , "years old.")
```

RUN

```
my name is ali ahmadi and I am 29 years old.
```

سینتکس پایتون(syntax)

در واقع به مجموعه ی قواعد و قوانین نوشتاری حاکم بر یک زبان سینتکس آن زبان گویند. به عبارت دیگر سینتکس یک زبان نحوه ی کنارهم قرار گرفتن صحیح اجزای یک زبان برنامه نویسی می باشد.

فرورفتگی پایتون (indentation)

یک از مهمترین مباحث در سینتکس پایتون، فرورفتگی ها میباشد. فرورفتگی به فضاهای خالی (white space) **ابتدای** یک خط کد اشاره دارد.در سایر زبان های برنامه نویسی فرورفتگی در کد فقط برای خوانایی است، اما فرورفتگی در پایتون بسیار بسیار مهم است.پایتون از فرورفتگی برای نشان دادن یک بلاک کد استفاده می کند.

مثال

```
if 5 > 2:
    print("Five is greater than two!")
```

RUN

```
Five is greater than two!
```

اگر در اعمال فرورفتگی اشتباهی مرتکب بشوید، پایتون به شما خطا می دهد یا ممکن است به نتیجه مطلوب نرسید:

مثال

```
if 5 > 2:
print("Five is greater than two!")
```

RUN

```
 File "demo_indentation_test.py", line 2
   print("Five is greater than two!")
IndentationError: expected an indented block
```

میزان فضای خالی به شما به عنوان یک برنامه نویس بستگی دارد، اما معمولا 4 عدد اسپیس برای فرورفتگی استفاده میشود،(4 تا space معادل یک tab) اما باید حداقل یک عدد اسپیس باشد.

مثال

```
if 5 > 2:
    print("Five is greater than two!"
if 10 > 4:
        print("ten is greater than four!")
```

RUN

Five is greater than two!
ten is greater than four!

شما باید از تعداد فاصله مشخصی در یک بلاک کد استفاده کنید، در غیر این صورت پایتون به شما خطا می دهد:

مثال

```
if 5 > 2:
    print("Five is greater than two!")
        print("Five is greater than two!")
```

RUN

ERROR

مثال

```
if 5 > 2:
    print("Five is greater than two!")
    print("that's right")
    if 3 < 4:
        print("four is greater than three!")
        print("congratulations!")
    print("4-3=" , 4-3)
    print("have a nice time")
print("bye")
```

RUN

Five is greater than two!

that's right

four is greater than three!

congratulations!

4-3= 1

have a nice time

bye

نوع داده (Data Type)

انواع داده های داخلی (از پیش تعریف شده یا ساخته شده) (Built-in Data Types)

در برنامه نویسی، نوع داده (data type) یک مفهوم مهم است.متغیرها می توانند داده هایی از انواع مختلف را ذخیره کنند .

پایتون **انواع داده های داخلی** (data types built-in) زیر را دارد:

str	نوع رشته	مثال→ 2 , -123132545 , 0
int	نوع عدد صحیح	مثال→ 0.0 , 1.0 , 12.35 , - 0.12255285412
float	نوع عدد اعشاری	مثال→ "ali" , "-2.333" , "a", "4.0" ,"c"
complex	نوع عدد مختلط	مثال→ 3j , 1+2j , 0+4j , 2+ 0j
list	نوع لیست	مثال→ [2 , True , 2.3 , [2 ,'ali']
tuple	نوع تاپل	مثال→ (1 , 2 , 3)
dict	نوع دیکشنری	مثال→ {'ali':19.5 , 'reza':18}
set	نوع ست	مثال→ {1 , 2 ,3 , 4 ,5 }
range	نوع range	مثال→ range(1 , 100 , 3)
bool	نوع بولین	مثال→ True , False

در پایتون، زمانی که مقداری را به یک متغیر اختصاص می دهید، نوع داده تنظیم می شود:

مثال

مثال	دیتاتایپ	کد	اجرا
x = "Hello World"	str	x = "Hello World" print(x)	Hello World
x = 20	int	x = 20 print(x)	20
x = 20.5	float	x = 20.5 print(x)	20.5
x = 1j	complex	x = 1j print(x)	1j

x = ["apple", "banana", "cherry"]	list	x = ["apple", "banana", "cherry"] print(x)	['apple', 'banana', 'cherry']
x = ("apple", "banana", "cherry")	tuple	x = ("apple", "banana", "cherry") print(x)	('apple', 'banana', 'cherry')
x = range(6)	range	x = range(6) print(x)	range(0, 6)
x = {"name" : "Amir", "age" : 36}	dict	x = {"name" : "Amir", "age" : 36} print(x)	{'name': 'Amir', 'age': 36}
x = {"apple", "banana", "cherry"}	set	x = {"apple", "banana", "cherry"} print(x)	{'banana', 'apple', 'cherry'}
x = True	bool	x = True print(x)	True

بدست آوردن نوع داده یک شی

با استفاده از تابع type() می توانید دیتا تایپ هر شی (object) را بدست آورید:

مثال

```
x = 11

print(11)
print(type(x))
```

RUN

```
11
<class 'int'>
```

مثال

```
print(5)

print(type(5))
```

RUN

```
5
<class 'int'>
```

مثال

```
x = 5
y = "Amir"
print(type(x))
print(type(y))
```

RUN

<class 'int'>
<class 'str'>

تبدیل نوع : type conversion

2 صورت تبدیل نوع (type conversion) داریم: ضمنی و صریح

تبدیل نوع ضمنی (Implicit Type Conversion): پایتون خودش بصورت خودکار تبدیل نوع را انجام میدهد.

مثال

```
a = 50
b = 6.258
c = a + b
print("datatype of a:",type(a))
print("datatype of b:",type(b))
print("Value of c:",c)
print("datatype of c:",type(c))
```

RUN

datatype of a: <class 'int'>
datatype of b: <class 'float'>
Value of c: 56.258
datatype of c: <class 'float'>

مثال

```
Print(3/0.75)
```

RUN

4.0

مثال:

```
a = 50
b = "ali"
c = a + b
```

```
print("datatype of a:",type(a))
print("datatype of b:",type(b))
print("Value of c:",c)
print("datatype of c:",type(c))
```

RUN

```
Traceback (most recent call last):
  File "C:\Users\user\Desktop\test.py", line 3, in <module>
    c = a + b
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

تبدیل نوع صریح (Explicit Type Conversion) یا (type casting): ما (برنامه نویس) تبدیل نوع انجام می دهیم.در مثال بالا پایتون تبدیل نوع انجام نداد، برای رفع ارور بالا باید خودمان به عنوان برنامه نویس از ابتدا تبدیل نوع انجام دهیم که به این صورت تبدیل نوع ، تبدیل نوع صریح یا casting نیزگویند.

Type casting:

پس اگر می خواهید نوع داده خاصی را روی مقداری (یا متغییری) اعمال کنید ، می توانید از تابع سازنده آن نوع داده (constructor function) زیر استفاده کنید. یا به عبارت دیگر اگر می خواهید نوع داده را مشخص کنید ، می توانید از توابع سازنده زیر استفاده کنید:

دیتاتایپ	مثال
Str	x = str("Hello World")
Int	x = int(20)
Float	x = float(20.5)
Complex	x = complex(1j)
List	x = list(("apple", "banana", "cherry"))
Tuple	x = tuple(("apple", "banana", "cherry"))
Range	x = range(6)
Dict	x = dict(name="Amir", age=36)
Set	x = set(("apple", "banana", "cherry"))
Bool	x = bool(5)

پس اگر می خواهید نوع داده یک متغیر را مشخص کنید، این کار را می توان با casting انجام داد.

مثال

```
x = str(3)
y = int(3)
z = float(3)
print(x)
print(y)
print(z)
```

RUN

```
3
3
3.0
```

مثال

```
a = 50
b = "fifty"
a=str(50)
c = a + b
print("datatype of a:",type(a))
print("datatype of b:",type(b))
print("Value of c:",c)
print("datatype of c:",type(c))
```

Run

```
datatype of a: <class 'str'>
datatype of b: <class 'str'>
Value of c: 50fifty
datatype of c: <class 'str'>
```

اعداد پایتون(Python Numbers)

اعداد پایتون

سه نوع عددی (numeric types) در پایتون وجود دارد:

- int
- float
- complex

مثال

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

همانگونه که دیدیم برای پی بردن به نوع هر شی (object) در پایتون، از تابع type() استفاده کردیم:

مثال

```
x = 1

y = 2.8

z = 1j

print(type(x))

print(type(y))

print(type(z))
```

RUN

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

Int

Int یا integer، یک عدد صحیح، مثبت یا منفی، بدون اعشار، با طول نامحدود است.

مثال

```
x = 1
y = 35656222554887711
z = -3255522
print(type(x))
```

```
print(type(y))
print(type(z))
```

RUN

```
<class 'int'>
<class 'int'>
<class 'int'>
```

Float (اعشاری)

float یک عدد مثبت یا منفی است که شامل یک یا چند رقم اعشار است.

مثال

```
x = 1.10
y = 1.0
z = -35.59
c=0.0
print(type(x))
print(type(y))
print(type(z))
print(type(c))
```

RUN

```
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
```

Floatهمچنین می تواند بصورت نماد علمی با کاراکتر "e" باشد

مثال

```
x = 35e3
y = 12E4
z = -87.7e100
m = 87.7e-3
print(x)
print(y)
print(z)
print(type(x))
```

```
print(type(y))
print(type(z))
print(type(m))
```

RUN

```
35000.0
120000.0
-8.77e+101
0.0877
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
```

complex

اعداد مختلط با یک "j" به عنوان قسمت موهومی نوشته می شوند:

مثال

```
x = 3+5j
y = 5j
z = -5j
print(type(x))
print(type(y))
print(type(z))
```

```
<class 'complex'>
<class 'complex'>
<class 'complex'>
```

همانگونه که بحث شد شما می توانید از یک نوع به نوع دیگر با توابع int() و float() وcomplex() تبدیل کنید:

مثال

تبدیل از یک نوع به نوع دیگر:

```
x = float(1)
```

```
y = int(2.8)
z = complex(1)
print(x)
print(y)
print(z)
print(type(x))
print(type(y))
print(type(z))
```

RUN

```
1.0
2
(1+0j)
<class 'float'>
<class 'int'>
<class 'complex'>
```

توجه :نمی توانید اعداد مختلط را به نوع اعداد دیگری تبدیل کنید.(اما برعکس آن امکان پذیر است)

بررسی type casting اعداد با رشته ها

- int(): یک عدد صحیح از انواع دیتاتایپ های مختلف مانند نوع صحیح، نوع اعشاری (با حذف تمام اعشار)، یا نوع رشته (به شرط اینکه رشته فقط یک عدد صحیح را نشان دهد) می سازد.
- float(): یک عدد اعشاری از انواع دیتاتایپ های مختلف مانند یک عدد صحیح ، یک عدد اعشاری یا یک رشته (به شرط اینکه رشته نشان دهنده یک اعشاری یا یک عدد صحیح باشد) می سازد.
- str(): رشته ای را از طیف گسترده ای از انواع داده ها، از جمله رشته ها، اعداد صحیح و اعداد اعشاری و...می سازد.

مثال

اعداد صحیح:

```
x = int(1)
y = int(2.8)
```

```
z = int("3")
print(x)
print(y)
print(z)
```

RUN

```
1
2
3
```

مثال

اعداد اعشاری:

```
x = float(1)
y = float(2.8)
z = float("3")
w = float("4.2")
print(x)
print(y)
print(z)
print(w)
```

RUN

```
1.0
2.8
3.0
4.2
```

مثال

رشته ها:

```
x = str("s1")
y = str(2)
z = str(3.0)
```

```
print(x)
print(y)
print(z)
```

RUN

```
s1
2
3.0
```

بولین ها

مقادیر بولی

در برنامه نویسی اغلب باید بدانید که یک عبارت True یا False است؟ می توانید هر عبارتی را در پایتون ارزیابی کنید و یکی از دو پاسخ True یا False دریافت کنید.وقتی دو مقدار را با هم مقایسه می کنید، عبارت مورد ارزیابی قرار می گیرد و پایتون پاسخ بولی را برمی گرداند:

مثال

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

RUN

```
True
False
False
```

وقتی شرطی را در دستور if اجرا می کنید، پایتون True یا False برمیگرداند

مثال

یک پیام چاپ کنید بر اساس اینکه آیا این شرطها درست است یا نادرست:

```
a = 200
b = 33
if b > a:
        print("b is greater than a")
else:
        print("b is not greater than a")
```

RUN

```
b is not greater than a
```

تابع bool()

این تابع به شما امکان می دهد هر مقداری را ارزیابی کنید و به شما True یا False برمیگرداند.

برخی از مقادیر False هستند

در واقع، مقادیر زیادی وجود ندارد که false ارزیابی شوند ، به جز مقادیرخالی(empty values) مانند ( ), "", { }, [ ] و عدد صفر و مقدار خالی (value None) و خود واژه ی False به صورت False ارزیابی میشود.

مثال

موارد زیر False را برمی گرداند:

```
Print(bool(False))
Print(bool(None))
Print(bool(0))
Print(bool(""))
Print(bool( ( ) ))
Print(bool([ ]) )
Print(bool({ }) )
```

RUN

```
False
False
False
False
False
False
False

False
```

بیشتر مقادیر (VALUES)درست (True) هستند

تقریباً هر مقداری(value) True ارزیابی می شود در صورتیکه دارای نوعی مقدار(محتوا) باشد.

- هر رشته ای True است ، به جز رشته های خالی.
- هر عددی True است به جز 0
- هر list، tuple، set و dictionary ، True هستند ، به جز آنهایی که خالی هستند.

مثال

موارد زیر True را برمی گرداند:

```
Print(bool("abc") )
print(bool(123) )
print(bool(["apple", "cherry", "banana"]) )
```

RUN

```
True
True
True
```

مثال

```
x = "Hello"
y = 15

z=0
print(bool(x))
print(bool(y))

print(bool(z))
```

RUN

```
True
True
```

تابع Input()

بعضی مواقع نیاز داریم اطلاعاتی را از کاربر دریافت کنیم به همین منظور از تابع input استفاده میکنیم (تابع input() را فراخوانی میکنیم). سینتکس تابع input بصورت زیر است:

```
Variable=input( 'string' )
```

بعد از ورود اطلاعات کاربر کلید enter را فشار میدهد. و رشته وارد شده کاربر به متغیر مربوطه انتقال داده میشود. وقتی نوبت به تابع input() می‌رسد، اجرای برنامه را متوقف میشود و زمانی که کاربر مقداری ورودی داده است، ادامه می‌یابد.

مثال زیر نام کاربری را می‌پرسد و وقتی نام کاربری را وارد کردید، روی صفحه نمایش داده می‌شود:

مثال

```
a = input("Enter username:")
print("Username is: " + a)
```

RUN

```
Enter username: ali
Username is: ali
```

پایتون مقادیر وارد شده را بصورت رشته می خواند اگر عددی بخواهیم بگیریم باید نوع آن را از رشته به نوع عددی دلخواه تغییر دهیم.(یعنی تبدیل نوع انجام دهیم)

مثال

```
x=input('enter x:')
y=input('enter y:')
x=int(x)
y=int(y)
print('your result is:' , 3*x**2+-3*y+2)
```

RUN

```
enter x:2
enter y:3
your result is: 5
```

میتوانستیم تبدیل نوع را همان اول انجام دهیم :

مثال

```
x=int(input('enter x:'))
y=int(input('enter y:'))
#x=int(x)
#y=int(y)
print('your result is:' , 3*x**2+-3*y+2)
```

RUN

```
enter x:2
enter y:3
your result is: 5
```

در مثال های بالا اگر تبدیل نوع انجام ندهیم چون مقدار ذخیره شده در متغیر بصورت رشته بود است اما عملیات ما ، یک عملیات ریاضی بوده است پایتون ارور میدهد.

عملگرهای پایتون

عملگرها(Operators) برای انجام عملیات روی متغیرها و مقادیر استفاده می شوند.در مثال زیر از عملگر +برای جمع کردن دو مقدار(دو عملوند) استفاده می کنیم:

مثال

```
print(10 + 5)
```

RUN

15

پایتون عملگرها را به گروه های زیر تقسیم می کند:

- عملگرهای محاسباتی (Arithmetic)
- عملگرهای انتساب(Assignment)
- عملگرهای مقایسه (Comparison)
- عملگرهای منطقی (Logical)
- اپراتورهای همانی(Identity)
- اپراتورهای عضویت(Membership)
- عملگرهای بیتی(Bitwise)

عملگرهای محاسباتی پایتون

عملگرهای محاسباتی برای انجام عملیات ریاضی روی مقادیر عددی استفاده می شوند:

علامت اپراتور	اسم اپراتور	مثال	قطعه کد	اجرا
+	Addition	x + y	x = 5 y = 3 print(x + y)	8
-	Subtraction	x – y	x = 5 y = 3 print(x - y)	2
*	Multiplication	x * y	x = 5 y = 3 print(x * y)	15
/	Division	x / y	x = 10 y = 3 print(x / y)	3.33333333
%	Modulus	x % y	x = 5 y = 2 print(x % y)	1
**	Exponentiation	x ** y	x = 5 y = 3 print(x ** y)	125
//	Floor division	x // y	x = 15 y = 2 print(x // y)	7

عملگرهای انتساب پایتون(عملگرهای ترکیبی)

عملگرهای انتساب برای تخصیص مقادیر به متغیرها استفاده می شوند(این عملگرها معمولا ترکیببی از عملگرهای محاسباتی یا بیتی و عملگر = هستند)

علامت اپراتور	مثال	مثال	قطعه کد	اجرا
=	x = 5	x = 5	x = 5 print(x)	5
+=	x += 3	x = x + 3	x = 5 x += 3 print(x)	8
-=	x -= 3	x = x - 3	x = 5 x -= 3 print(x)	2
*=	x *= 3	x = x * 3	x = 5 x *= 3 print(x)	15
/=	x /= 3	x = x / 3	x = 5 x /= 3 print(x)	1.666666
%=	x %= 3	x = x % 3	x = 5 x%=3 print(x)	2
//=	x //= 3	x = x // 3	x = 5 x//=3 print(x)	1
**=	x **= 3	x = x ** 3	x = 5 x **= 3 print(x)	125

عملگرهای مقایسه ای پایتون

عملگرهای مقایسه برای مقایسه دو مقدار استفاده می شوند و نتیجه True یا Flase برمیگردانند.

علامت اپراتور	اسم اپراتور	مثال	قطعه کد	اجرا
==	Equal	x == y	x = 5 y = 3 print(x == y)	False
!=	Not equal	x != y	x = 5 y = 3 print(x != y)	True
>	Greater than	x > y	x = 5 y = 3 print(x > y)	True
<	Less than	x < y	x = 5 y = 3 print(x < y)	False
>=	Greater than or equal to	x >= y	x = 5 y = 3 print(x >= y)	True
<=	Less than or equal to	x <= y	x = 5 y = 3 print(x <= y)	False

عملگرهای منطقی پایتون

عملگرهای منطقی بر روی عبارت منطقی (True و False)عمل می‌کنند

علامت اپراتور	اسم اپراتور	مثال	قطعه کد	اجرا
and	زمانی true است که تمامی ورودی ها true باشند	x >3 and x < 10	x = 5 print(x > 3 and x < 10)	True
or	زمانی true است که حداقل یکی از ورودی ها true باشند	x > 3 or x < 4	x = 5 print(x > 3 or x < 4)	True
not	Trueرا به false و برعکس	not(x >3 and x < 10)	x = 5 print(not(x > 3 and x < 10))	False

عملگرهای هویت (همانی) پایتون

عملگرهای هویت برای مقایسه اشیاء استفاده می‌شوند، نه این که فقط از لحاظ مقداربرابر باشند، بلکه آیا از لحاظ حافظه ای (آدرس محل ذخیره) هم به مکان یکسانی اشاره دارند یا خیر؟

علامت اپراتور	مفهوم	نمایش	قطعه کد	اجرا
is	اگر دو شی دقیقا مثل هم باشند	x is y	x = ["apple", "banana"] y = ["apple", "banana"] z = x print(x is z) print(x is y) print(x == y)	True False True
is not	اگر دو شی دقیقا مثل هم نباشند	x is not y	x = ["apple", "banana"] y = ["apple", "banana"] z = x print(x is not z) print(x is not y) print(x != y)	False True False

اپراتورهای عضویت پایتون

عملگرهای عضویت برای بررسی اینکه آیا یک شی در یک iterable (یک مقدار یا متغیر در یک توالی) وجود دارد یا خیر ، استفاده می شود:

Operator	Description	Example	Code	preview
in	زمانی true برمیگرداند که دقیقا شی اول عضوی از شی دوم باشد.	x in y	x = ["apple", "banana"] print("banana" in x)	True
not in	زمانیtrue برمیگرداند که دقیقا شی اول عضوی از شی دوم نباشد.	x not in y	x = ["apple", "banana"] print("pineapple" not in x)	True

اولویت (precedence) اپراتورها

اولویت عملگرها ترتیب انجام عملیات را توصیف می کند.ترتیب اولویت در جدول زیر توضیح داده شده است که با بالاترین اولویت در بالا شروع می شود:

اپراتور	قطعه کد	اجرا
()	print((3 + 6) - (3 + 6))	0
**	print(3 ** 3 - 100)	-73
* / // %	print(3 * 5 + 100)	115
+ -	print(3 * 5 - 100)	-85
<< >>	Print(8>>4-2)	2
== != > >= < <= is Isnot in notin	print(5 == 4 + 1)	true
not	print(not 5 == 5)	false
and		
or		

اگر دو عملگر دارای اولویت یکسان باشند، عبارت از چپ به راست بررسی می شود.

مثال

ضرب و تقسیم اولویت یکسانی دارند و بنابراین عبارت را از چپ به راست ارزیابی می کنیم:

```
print(24/8*3)
```

RUN

```
9
```

مثال

برنامه ای بنویسید که شعاع یک دایره را از کاربر بگیرد و محیط و مساحت آنرا محاسبه کند.

```
radius = float(input("enter the radius: "))
perimeter = radius * 2 * 3.14
area = 3.14 * radius**2
print("The perimeter is ", perimeter)
print("The area is ", area)
```

RUN

```
enter the radius: 3
The perimeter is  18.84
The area is  28.26
```

مثال

برنامه ای بنویسید که سن شما را به سال بگیرد و بگوید به طور تقریبی چند ماه و چند روز و چند ساعت و چند دقیقه و چند ثانیه عمر کرده اید.

```
year = float(input("enter your age: "))
month=year*12
day=year*365
hour=day*24
minute=hour*60
second=minute*60
print('month: ' , month)
print('day: ' , day)
print('hour: ' , hour)
print('minute: ' , minute)
print('second: ' , second)
```

RUN

```
enter your age: 32
month:  384.0
```

day: 11680.0

hour: 280320.0

minute: 16819200.0

second: 1009152000.0

مثال

برنامه ای بنویسید پارامتر های لازم را از کاربر دریافت کند و حاصل چند جمله ای زیر را به ازای آن پارامتر ها بدست آورد.

H=$12 * x^4 - 5 * y^3 $ * x+3 * $(xy)^3$-34

```
x = float(input("enter x: "))
y = float(input("enter y: "))
h = 12 * x ** 4 - 5 * y ** 3 * x + 3 * (x*y)**3 - 34
print("h = ", h)
```

RUN

enter x: 3

enter y: 2

h = 1466.0

مثال

فرض کنید خریداری به فروشنده ای مراجعه میکند و 3 کالا میخرد. برنامه ای بنویسید که ابتدا تعداد و قیمت 3 کالا را بگیرد و هزینه ی خریدار را اعلام کند و به خریدار بگوید چند اسکناس 200 هزار تومانی و چند اسکناس 100 هزار تومانی و چند اسکناس 50 هزار تومانی و چند اسکناس 10 هزار تومانی و چند اسکناس 5 هزار تومانی و چند اسکناس 2 هزار تومانی و چند اسکناس هزار تومانی باید به فروشند بپردازد.

```
num1 = float(input("Enter the number of the first item:"))
price1 = float(input("Enter the price of the first item:"))
num2 = float(input("Enter the number of the second item:"))
price2 = float(input("Enter the price of the second item:"))
num3 = float(input("Enter the number of the third item:"))
price3 = float(input("Enter the price of the third item:"))
sum_cost = num1*price1+num2*price2+num3*price3
```

```
print('your cost is: ' , sum_cost)
banknote200 = sum_cost//200000
temp = sum_cost % 200000
banknote100 = temp//100000
temp = temp % 100000
banknote50 = temp//50000
temp = temp % 50000
banknote10 = temp//10000
temp = temp % 10000
banknote5 = temp//5000
temp = temp % 5000
banknote2 = temp//2000
temp = temp % 2000
banknote1 = temp//1000
print('bank note 200000 tomani : ', banknote200)
print('bank note 100000 tomani : ', banknote100)
print('bank note 50000 tomani : ', banknote50)
print('bank note 10000 tomani : ', banknote10)
print('bank note 5000 tomani : ', banknote5)
print('bank note 2000 tomani : ', banknote2)
print('bank note 1000 tomani : ', banknote1)
```

RUN

```
Enter the number of the first item:3
Enter the price of the first item:150000
Enter the number of the second item:6
Enter the price of the second item:199000
Enter the number of the third item:3
```

Enter the price of the third item:33000

'your cost is: 1743000.0

bank note 200000 tomani : 8.0

bank note 100000 tomani : 1.0

bank note 50000 tomani : 0.0

bank note 10000 tomani : 4.0

bank note 5000 tomani : 0.0

bank note 2000 tomani : 1.0

bank note 1000 tomani : 1.0

مثال

برنامه ای بنویسید که یک عدد دو رقمی را از کاربر بگیرد و سپس ارقام این عدد را برعکس کند.

```
num = int(input("Enter a two-digit number: "))
digit1 = num // 10
digit2 = num % 10
print("The reverse of a two-digit number is :", digit2 * 10 + digit1)
```

RUN

Enter a two-digit number: 34

The reverse of a two-digit number is : 43

ساختار های کنترلی در پایتون : **ساختار های تصمیم گیری و ساختار های تکرار**

الف)ساختار های تصمیم گیری: if…elif…else

شرط ها در پایتون

گاهی پیش می آید که بخواهید در برنامه شرطی قرار دهید برای این منظور از ساختار if…elif…else استفاده می کنیم.(البته وجود elif و else اختیاری است)

سینتکس ساختار شرطی بصورت زیر است:

```
if  condition :
        statements
elif    condition :
        statements
elif    condition :
        statements
.
.
.
else:
        statements
```

ساختار بالا می گوید که اگر شرط جلوی if ، True بود وارد بلاک دستورات مربوطه اش میشویم. اما اگر شرط جلوی if، False بود ابتدا به سراغ elif ها (به ترتیب) می رویم ، اگر شرط جلوی elif ، True بود وارد بلاک دستورات مربوطه اش میشویم. اما اگر شرط جلوی elif ها هم False بود سراغ else می رویم و دستورات مربوطه اش را بررسی میکنیم.

همانطور که در فصل قبل دیدیم، عملگرهای مقایسه ای ، منطقی و هویت و عضویت مقادیر True یا False برمیگرداندند که از آنها میتوان در شرطها استفاده نمود.

استفاده از عملگرهای مقایسه ای شرطها

مثال

```
a = 33
b = 200
```

```
if  b > a :
        print("b is greater than a")
```

RUN

b is greater than a

در این مثال از if برای بررسی اینکه آیا b بزرگتر از a است یا خیر استفاده می شود . از آنجایی که a= 33 است و b= 200 است ، b is greater than a را چاپ می کنیم.

فرورفتگی(indentation)

همانطور که در فصل اول هم بررسی کردیم پایتون برای تعریف محدوده در کد به فرورفتگی (فاصله خالی در ابتدای خط) متکی است .سایر زبان های برنامه نویسی از آکولاد برای این منظور استفاده می کنند.

مثال

```
a = 33
b = 200
if b > a:
print("b is greater than a")
```

RUN

Error

Elif

اگر شرط جلوی if ،False بود ابتدا به سراغ elif ها (به ترتیب) می رویم ، اگر شرط جلوی elif ، True بود وارد بلاک دستورات مربوطه اش میشویم. اما اگر شرط جلوی elif ها هم False بود به سراغ else می رویم.

مثال

```
a = 33
b = 33
if b > a:
        print("b is greater than a")
elif a == b:
        print("a and b are equal")
```

RUN

a and b are equal

در این مثال a برابر با b است ، بنابراین شرط اول درست نیست، اما شرط elif درست است، بنابراین چاپ می کنیم که a و b برابر هستند.

Else

در ساختار شرطی اگر شروط مربوط به if و elif های بالا False باشد ، وارد بلاک دستورات مربوط به else میشویم.

مثال

```
a = 200
b = 33
if b > a:
        print("b is greater than a")
elif a == b:
        print("a and b are equal")
else:
        print("a is greater than b")
```

RUN

a is greater than b

در این مثال a بزرگتر ازb است ، بنابراین شرط اول درست نیست، همچنین شرط elif درست نیست، بنابراین به سراغ else می رویم .

میتوان از else ، بدون elif هم استفاده کرد:

مثال

```
a = 200
b = 33
if b > a:
        print("b is greater than a")
else:
        print("b is not greater than a")
```

RUN

b is not greater than a

if کوتاه (short hand if)

اگر فقط یک دستور برای اجرا دارید، می توانید آن را در همان خط دستور if قرار دهید.

مثال

عبارت if یک خطی:

```
if a > b : print("a is greater than b")
```

RUN
a is greater than b

If...else کوتاه (short hand if...else)

اگر فقط یک دستور برای اجرا دارید، یکی برای if و یکی برای else ، می توانید همه آن را در یک خط قرار دهید:

مثال

```
a = 2
b = 330
print("yes")  if   a > b  else  print("no")
```

RUN
no

استفاده از عملگرهای منطقی در شرطها

And

مثال

```
a = 200
b = 33
c = 500
if a > b and c > a:
        print("Both conditions are True")
```

RUN
Both conditions are True

OR

مثال

```
a = 200
b = 33
c = 500
if a > b or a > c:
        print("At least one of the conditions is True")
```

RUN
At least one of the conditions is True

NOT

مثال

```
a = 33
b = 200
if not a > b:
        print("a is NOT greater than b")
```

RUN

```
a is NOT greater than b
```

if های تودرتو (nested if)

می توانید شرطهایی داخل شرطهای دیگر داشته باشید، به این ساختار if های تودرتو می گویند .

مثال

```
x = 41
if x > 10:
        print("Above ten,")
        if x > 20:
                print("and also above 20!")
        else:
                print("but not above 20.")
```

RUN

```
Above ten,
and also above 20!
```

Pass

if نمی تواند خالی باشند، اما اگر به هر دلیلی if بدون بلاک دستور دارید، برای جلوگیری از ارور، کلمه کلیدی pass را در دستور if قرار دهید.

مثال

```
a = 33
b = 200
if b > a:
        pass
```

RUN

حلقه های پایتون

پایتون دو ساختار برای حلقه ها دارد:

- حلقه while
- حلقه for

while loops

سینتکس حلقه while بصورت زیر است:

```
while  condition :
    statements
```

با حلقه while می‌توانیم مجموعه‌ای از دستورات را تا زمانی که شرط درست است اجرا کنیم.

مثال

```
i = 1
while i < 6:
    print(i)
    i += 1
```

RUN

```
1
2
3
4
5
```

حلقه while به تعریف متغیرها و مقداردهی اولیه ی آنها (به عنوان شمارنده) نیاز دارد در این مثال متغییر i را به عنوان شمارنده تعریف کردیم و مقدار اولیه ی 1 به آن دادیم.

به یاد داشته باشید متغیر هایی مانند i را که تعریف کرده اید باید افزایش یا کاهش دهید(برحسب شرطی که تعریف کرده اید)، در غیر این صورت حلقه برای همیشه ادامه خواهد داشت.

مثال

```
i = 1
while i < 6:
    print(i , end="\t")
    i += 1
```

RUN

```
1    2    3    4    5
```

Break

حتی اگر شرط حلقه درست باشد با دستور break حلقه به اتمام میرسد.

مثال

```
i = 1
while i < 6:
        print(i , end="    ")
        if i == 3:
                break
        i += 1
```

RUN

```
1    2    3
```

Continue

با دستور continue می‌توانیم تکرار فعلی را متوقف کرده و به ابتدای حلقه بازگردیم و ادامه دهیم:

مثال

```
i = 0
while i < 6:
        i += 1
        if i == 3:
                continue
        print(i , end="    ")
```

RUN

```
1    2    4    5    6
```

While و Else

با استفاده از else بعد از آنکه شرط while دیگر صادق نبود ، میتوانیم **یکبار** دستورات بلاک مربوط به else را اجرا نماییم.

مثال

```
i = 1
while i < 6:
        print(i ,end="    ")
        i += 1
```

```
else:
        print("i is no longer less than 6")
```

RUN

```
1   2   3   4   5   i is no longer less than 6
```

توجه :اگر حلقه while توسط دستور break متوقف شود، بلاک else اجرا نخواهد شد.

مثال

```
i = 1
while i < 6:
        print(i,end="    ")
        if i==4:
                break
        i += 1
else:
   print("i is no longer less than 6")
```

RUN

```
1   2   3   4
```

حلقه for (for loops)

حلقه for برای پیمایش (تکرار) در یک دنباله (که یک لیست، یک تاپل، یک دیکشنری، یک ست یا یک رشته) استفاده می شود. For در پایتون خیلی شبیه به for در سایر زبان های برنامه نویسی نیست. با حلقه for می‌توانیم مجموعه‌ای از دستورات را، برای **هر عنصر(آیتم)** در یک لیست، تاپل، ست و ... اجرا کنیم.

مثال

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x , end="    ")
```

RUN

```
apple    banana    cherry
```

حلقه در یک رشته

حتی رشته ها نیز اشیای تکرارپذیرهستند، آنها حاوی یک دنباله از کاراکترها هستند.

مثال

```
for x in "banana":
    print(x , end="    ")
```

RUN

```
b    a    n    a    n    a
```

Break

با دستور break می‌توانیم حلقه را قبل از اینکه دستورات را طی کند ،متوقف کنیم.

مثال

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x , end="    ")
    if x == "banana":
        break
```

RUN

```
apple    banana
```

مثال

```
fruits = ["apple", "banana", "cherry"]
for   x   in   fruits:
        if x == "banana":
                break
        print(x)
```

RUN

```
apple
```

Continue

با دستور continue می‌توانیم تکرار فعلی حلقه را متوقف کرده و به ابتدای حلقه برگردیم وباز هم ادامه دهیم.

مثال

```
fruits = ["apple", "banana", "cherry"]
for   x   in   fruits:
         if x == "banana":
                continue
        print(x , end="    ")
```

RUN

```
apple    cherry
```

تابع range()

برای حلقه در مجموعه ای از کدها به تعداد مشخص، می توانیم از تابع range() استفاده کنیم .

تابع range() دنباله ای از اعداد را برمی گرداند که به طور پیش فرض از 0 شروع می شود و به صورت پیش فرض 1 واحد افزایش می یابد و به یک عدد کمتر از عدد پایانی ختم می شود.

```
Range(start , stop , step)
```

مثال

```
for  x  in  range(6):
        print(x , end="    ")
```

RUN

```
0   1   2   3   4   5
```

range(2, 6) : به معنای مقادیر از 2 تا 6 است (اما شامل 6 نمی‌شود).

مثال

```
for  x  in  range(2, 6):
        print(x , end="    ")
```

RUN

```
2   3   4   5
```

تابع range() پیش‌فرض دنباله را 1 واحد افزایش می‌دهد، اما می‌توان مقدار افزایش را با افزودن پارامتر سوم تعیین کرد:

مثال

```
for   x   in   range(2, 30, 3):
        print(x ,end="    ")
```

RUN

```
2   5   8   11   14   17   20   23   26   29
```

For و Else

else بعد از for ، یک بلاک از کد را ایجاد می کند که باید پس از اتمام حلقه for فقط یکباراجرا شود.

مثال

```
for   x   in   range(6):
        print(x ,end="    ")
else:
        print("Finally finished!")
```

RUN

```
0   1   2   3   4   5   Finally finished!
```

توجه :اگر حلقه for توسط یک دستور break متوقف شود، بلاک else اجرا نخواهد شد.

مثال

```
for x in range(6):
        if x == 3: break
        print(x ,end="    ")
else:
        print("Finally finished!")
```

RUN

```
0   1   2
```

حلقه for تودرتو(nested)

حلقه تو در تو ، حلقه ای در داخل یک حلقه است. حلقه داخلی هر بار برای هر تکرار حلقه بیرونی اجرا می شود:

مثال

```
adj = ["red", "good", "tasty"]
fruits = ["apple", "banana", "cherry"]
for x in adj:
    for y in fruits:
        print(x, y)
```

RUN

```
red apple
red banana
red cherry
good apple
good banana
good cherry
tasty apple
tasty banana
tasty cherry
```

pass

for ها نیز نمی‌توانند خالی باشند، اما اگر هر دلیلی حلقه ی for بدون دستوری دارید، برای جلوگیری از خطا، عبارت pass را قرار دهید.

مثال

```
for x in [0, 1, 2]:
    pass
```

RUN

مثال

```
for x in  [0 , 1 , 2] :
    pass
    print(x , end="    ")
```

RUN
0 1 2

مثال

برنامه ای بنویسد که یک عدد دلخواه از کاربر بگیرد و قدر مطلق آنرا چاپ کند.

```
num = float ( input ( "please enter a number:" ) )
if   num >= 0  :
        print(num)
else:
        print(-num)
```

RUN

```
please enter a number: -12.25
12.25
```

مثال

برنامه ای بنویسید که یک عدد دلخواه از کاربر بگیرد و تشخیص دهد آن عدد زوج است یا فرد.

```
num=int(input ( "please enter a number:") )
if num % 2 == 0 :
        print("your number is even")
else:
        print("your number is odd")
```

RUN

```
please enter a number:5
your number is odd
```

مثال

برنامه ای بنویسید که ضرایب معادله درجه 2 به فرم ax^2+bx+c را از کاربر دریافت کند و ریشه های معادله را بدهد.

```
a = float(input("please Enter a:"))
b = float(input("please Enter b:"))
c = float(input("please Enter c:"))
```

```
delta = b ** 2 - 4 * a * c
if delta < 0:
        print("The equation has no real roots")
elif delta == 0:
        print(" x1 , x2 = " , -b/ (2.0 * a))
else:
        print(" x1 = " , (-b - delta ** 0.5) / (2.0 * a))
        print(" x2 = " , (-b + delta ** 0.5) / (2.0 * a))
```

RUN

```
please Enter a:1
please Enter b:3
please Enter c:2
 x1 =  -2.0
 x2 =  -1.0
```

مثال

برنامه ای بنویسید که سه ضلع یک مثلث را از کاربر بگیرد و تشخیص دهد که آیا مثلث است یا خیر؟ و اگر مثلث بود، تشخیص دهد که آیا مثلث قائم الزاویه است یا خیر؟

```
a = float(input("Enter a:"))
b = float(input("Enter b:"))
c = float(input("Enter c:"))
if (a+b > c) and (a+c > b) and (c+b > a) == True:
        print("Yes, it is a triangle")
        if  (a ** 2 + b ** 2 == c ** 2) or \
            (a ** 2 + c ** 2 == b ** 2) or (c ** 2 + b ** 2 == a ** 2) == True:
                print("It is also a right triangle")
        else:
                print("But the triangle is not right angled")
else:
        print("No, it is not a triangle")
```

RUN

```
Enter a:3
Enter b:4
Enter c:5
Yes, it is a triangle
It is also a right triangle
```

مثال

برنامه ای بنویسید که یک عدد را خوانده و مضارب طبیعی 7 را تا آن عدد نمایش دهد.

```
num = int(input("Enter a number:"))
for i in range(7, num+1, 7):
        print(i, end = '    ')
```

RUN

```
Enter a number:42
7    14    21    28    35    42
```

مثال

برنامه ای بنویسید که یک عدد از کاربر بگیرد و فاکتوریل آن عدد را بدهد.

```
num = int(input("Enter num:"))
fact = 1
for i in range(1, num+1):
   fact = fact * i
print("the factorial is", fact)
```

RUN

```
Enter num:6
the factorial is 720
```

مثال

برنامه ای بنویسید که x و y را از کاربر بگیرد و عبارت زیر را محاسبه کند

$x^2 - x^5 + x^8 - \ldots + x^y$

```
x = int(input("Enter x:"))
y = int(input("Enter y:"))
sum = 0
sign = 1
for i in range(2, y+1 , 3):
   sum = sum + x ** i * sign
   sign = sign*-1
print("the sum is" , sum )
```

RUN

```
Enter x:3
Enter y:12
the sum is -170820
```

مثال

رشته ای را از کاربر بگیرید و حروف صدادار و نیز تعداد حروف صدادار آن را چاپ کنید

```
string = input("Enter a string:")
```

```
vowel_sound ="aeiouAEIOU"
counter = 0
for x in string:
        if x in vowel_sound:
                print(x , end="    ")
                counter = counter+1
print()
print("Count is ", counter)
```

RUN

```
Enter a string:hello world , this is A test
e    o    o    i    i    A    e
Count is  7
```

مثال

برنامه ای بنویسید که یک عدد از کاربر بگیرد و تشخیص دهد که آیا این عدد با مقلوبش برابر است یا خیر؟

```
num = int(input("Enter a number:"))
main_num = num
reverse = 0
while num > 0:
        reverse = reverse*10 + num % 10
        num = num // 10
if main_num == reverse:
        print("The number is invertible")
else:
        print("The number is not invertible")
```

RUN

```
Enter a number:563365
The number is invertible
```

مثال

برنامه ای بنویسید که چند عدد از کاربر بگیرد و مجموع و میانگین و حاصل ضرب آنها را بدهد.

```
count = 0
sum = 0
mul=1
while True:
   num = float(input("Enter a number :"))
   Q=input("Do you want to continue? y/n: ")
   sum += num
   mul *= num
```

```
    count = count + 1
    if Q == "y":
            continue
    elif Q == "n":
            break
print("The multiplication is ", mul)
print("The result of Sum is ", sum )
print("The average is ", sum / count)
```

RUN

```
Enter a number :2
Do you want to continue? y/n: y
Enter a number :3
Do you want to continue? y/n: y
Enter a number :4
Do you want to continue? y/n: n
The multiplication is  24.0
The result of Sum is  9.0
The average is  3.0
```

مثال

برنامه ای بنویسید که نمایش دهد یک اسکناس 5000 تومانی را به چند طریق میتوان با اسکناس های 2000 و 1000 تومانی خرد کرد؟

```
for y in range(0, 2+1):
        for z in range(0, 5+1):
                if y * 2000 + z * 1000 == 5000:
                        print(y , " * 2000 + ", z , "* 1000 = 5000")
```

RUN

```
0  * 2000 +  5 * 1000 = 5000
1  * 2000 +  3 * 1000 = 5000
2  * 2000 +  1 * 1000 = 5000
```

مثال

یک عدد از کاربر بگیرید و کوچکترین رقم آنرا چاپ کنید

```
num = int(input("Enter a number:"))
min = num % 10
while num > 0:
        if min > num % 10 :
                min = num % 10
        num = num // 10
print("The minimum is: ", min)
```

RUN

```
Enter a number:956325826
The minimum is:  2
```

مثال

برنامه ای بنویسید که شماره جمله را دریافت کند و سری فیبوناچی آنرا تحویل دهد.

```
#1      1       2       3       5       8       13      21
num = int(input("Enter num:"))
if num == 1:
   print ( 1)
elif num == 2:
   print(1, '\t', 1)
else:
   a1 , a2 = 1, 1
   print(a1, '\t', a2, end ='\t')
   for x in range(3, num + 1):
      a3 = a1 + a2
      print(a3, end = '\t')
      a1 = a2
      a2 = a3
```

RUN

```
Enter num:10
1       1       2       3       5       8       13      21      34      55
```

مثال

برنامه ای بنویسید که یک عدد بگیرد و اعداد اول ماقبل آن را نمایش دهد.

```
num = int(input("enter: "))
for i in range(2, num+1):
   temp = int(i**0.5 + 1)
   for j in range(2, temp):
      if i % j == 0:
         break
      else:
         continue
   else:
      print(i, end="\t")
```

RUN

```
enter: 20
2       3       5       7       11      13      17      19
```

مثال

برنامه ای بنویسید که یک عدد بگیرد و مشخص کند که آیا این عدد تام(کامل) است یا خیر؟

```
num = int(input("enter: "))
sum1 = 0
for j in range(1, num):
    if num % j == 0:
        sum1 = sum1 + j
if sum1 == num:
    print("yes")
else:
    print("no")
```

RUN

```
enter: 28
yes
```

مثال

برنامه ای بنویسید که خروجی زیر را چاپ کند.

```
n = int(input("Enter n:"))
for i in range (1 , n+1):
    for j in range(1, n+ 1):
        print(i , end =" ")
    print()
```

RUN

```
Enter n:5
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
```

مثال

برنامه ای بنویسید که خروجی زیر را چاپ کند.

```
n = int(input("Enter n:"))
for i in range (1 , n+1):
    for j in range(1, n+ 1):
        print(j , end =" ")
    print()
```

RUN

```
Enter n:5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

مثال

برنامه ای بنویسید که خروجی زیر را چاپ کند.

```
n = int(input("Enter n:"))
for i in range (1 , n+1):
    for j in range(1, n+ 1):
        print(end ="*")
    print()
```

RUN

```
Enter n:4
****
****
****
****
```

مثال

برنامه ای بنویسید که خروجی زیر را چاپ کند.

```
n = int(input("Enter n:"))
for i in range (1 , n+1):
    for j in range(1, n+ 1):
        if i == 1 or j == 1 or i == n or j == n or i == j or i+j==n+1:
            print(end ="*")
        else:
            print (end =" ");
    print()
```

RUN

Enter n:10

مثال

برنامه ای بنویسید که خروجی زیر را چاپ کند.

```
*
**
***
****
*****
```

```
n = int(input("Enter n:"))
for i in range (1 , n+1):
    for j in range(1, n+ 1):
        if i == n or j == 1 or j <= i:          # if  j <= i :
            print(end ="*")
        else:
            print (end =" ");
    print()
```

RUN

```
Enter n:5
*
**
***
****
*****
```

set, dictionary, tuple, list, string

String (رشته)

دنباله ای کاراکتر ها که در داخل ' یا " یا ''' یا """ قرار می گیرد.

رشته s را درنظر بگیرید این رشته اندیس گذاری (شماره گذاری یا index گذاری) شده است. این شماره گذاری از سمت چپ ، از صفر شروع می شود. اما از سمت راست از منفی یک شروع میشود.

-9	-8	-7	-6	-5	-4	-3	-2	-1
0	1	2	3	4	5	6	7	8

s='my python'

دسترسی به هر کاراکتر:

[اندیس کاراکتر موردنظر] نام رشته

مثال

```
's='my python
print(s[0])
print(s[5])
print(s[-2])
```

RUN

```
m
t
o
```

گاهی اوقات میخواهیم قسمتی از یک رشته را بدست آوریم به این منظور بصورت زیر عمل می کنیم:

String name[start : stop : step]

[گام : اندیس پایان : اندیس شروع] نام رشته

اندیسی که به عنوان شماره پایان در بالا قرار می گیرد ، شامل آن نمی شود.

اگر دو اندیس درج کنیم ، یعنی اندیس های شروع و پایان را نوشته ایم و بصورت پیشفرض ، گام را یک درنظر میگیریم.

اگر اندیس شروع را **خالی** بگذاریم از ابتدا شروع می شود ، اگر اندیس پایان را **خالی** بگذاریم تا پایان رشته می رود. و اگر جایگاه گام را خالی بگذاریم ، 1+ درنظر می گیرد.

برای حرکت معکوس (یعنی از انتها به ابتدا) باید گام را منفی درنظر بگیریم.

میتوانیم یکی در میان حرکت کنیم که در اینصورت گام 2+ یا 2- می شود. یا میتوانیم دوتا در میان حرکت کنیم که در اینصورت گام 3+ یا 3- می شود و ...

میتوان از اندیس های مثبت و منفی همزمان برای اندیس های شروع یا پایان استفاده کرد.

مثال

-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
0	1	2	3	4	5	6	7	8	9	10

s = "abcdefghijk"

```
print(s[2:5])       #cde
print(s[0:9:2])     #acegi
print(s[5:2:-1])    #fed
print(s[-1:-5:-1])  #kjih
print(s[-5:-1])     #ghij
print(s[2:-1])      #cdefghij
print(s[-1:2:-1])   #kjihgfed
print(s[-3: :-1])   #ihgfedcba
print(s[  :-5:-1])  #kjih
print(s[ : :-1])    #kjihgfedcba
```

برای بدست آوردن طول یک رشته از تابع len() استفاده می شود.

```
s = "abcdefghijk"
print(len(s))       #11
print(len("abcdefghijk"))       #11
```

متدها و توابع مختلفی برای رشته ها وجود دارد که در فصل مربوطه به آن خواهیم پرداخت.

الحاق دو رشته

برای الحاق دو رشته از عملگر + استفاده می کنیم

```
a='python'
b=' is the best'
print(a+b)
```

RUN

```
python is the best
```

همچنین برای تکرار یک رشته به تعداد دلخواه می توان از عملگر * استفاده نمود.

```
a='python  '
print(a*2)
print(2*a)
```

RUN

```
python  python
python  python
```

نمی توان در یک رشته یک یا تعدادی کاراکتر را جایگزین نمود.
مثال

```
a='python'
a[0]='j'
RUN
ERROR
```

list

یکی از انواع داده ساختمان داده های مخصوص پایتون، لیست ها هستند، لیست ها با [ ] احاطه شده اند.
به هر یک از اعضای لیست که با کاما جدا شده اند عنصر یا آیتم می گوییم.
لیست ها مانند ست و دیکشنری و تاپل برای ذخیره چندین مقدار در یک متغیر استفاده می شوند.
در لیست می توان انواع داده های مختلفی قرار داد.(یعنی در یک لیست، عناصر میتوانند از دیتاتایپ های مختلفی باشند) حتی می توان یک لیست یا تاپل و... را در داخل یک لیست دیگر قرار داد.
مانند رشته ها وتاپلها شماره گذاری شده اند (اندیس دارند) ، مانند دیکشنری تغییرپذیرند، مانند تاپلها میتوانند مقادیر تکراری داشته باشند.

نحوه ساخت لیست:

متغیر =[عناصر لیست]

-5	-4	-3	-2	-1
0	1	2	3	4

```
a=[12 , 'ali' , False , [1 , 'hello'] , 2.5 ]
```

مثال

```
a=[12 , 'ali' , False , [1 , 'hello'] , 2.5 ]
print(a)
```

RUN

```
[12, 'ali', False, [1, 'hello'], 2.5]
```

نحوه دسترسی به عناصر لیست:

نام لیست[اندیس عنصر موردنظر]

گاهی اوقات میخواهیم قسمتی از یک لیست را بدست آوریم به این منظور بصورت زیر عمل می کنیم:

```
list name[start : stop : step]
```

نام لیست[گام : اندیس پایان : اندیس شروع]

اندیسی که به عنوان شماره پایان در بالا قرار می گیرد ، شامل آن نمی شود.
اگر دو اندیس درج کنیم ، یعنی اندیس های شروع و پایان را نوشته ایم و بصورت پیشفرض ، گام را یک درنظر میگیریم.
اگر اندیس شروع را **خالی** بگذاریم از ابتدا شروع می شود ، اگر اندیس پایان را **خالی** بگذاریم تا پایان لیست می رود. و اگر جایگاه گام را خالی بگذاریم ، 1+ درنظر می گیرد.
برای حرکت معکوس (یعنی از انتها به ابتدا) باید گام را منفی درنظر بگیریم.

میتوانیم یکی در میان حرکت کنیم که در اینصورت گام 2+ یا 2- می شود. یا میتوانیم دوتا در میان حرکت کنیم که در اینصورت گام 3+ یا 3- می شود و ...
میتوان از اندیس های مثبت و منفی همزمان برای اندیس های شروع یا پایان استفاده کرد.

مثال

```
a=[12 , 'ali' , False , [1 , 'hello'], 2.5 ]
print(a[2])        #False
print(a[3][1])     #hello
print(a[1:3])      #['ali', False]
print(a[0:4:2])    #[12, False]
print(a[4:1:-1])   #[2.5, [1, 'hello'], False]
print(a[-1:-5:-1]) #[2.5, [1, 'hello'], False, 'ali']
print(a[-5:-1])    #[12, 'ali', False, [1, 'hello']]
print(a[2:-1])     #[False, [1, 'hello']]
print(a[-1:2:-1])  #[2.5, [1, 'hello']]
print(a[-3: :-1])  #[False, 'ali', 12]
print(a[  :-4:-1]) #[2.5, [1, 'hello'], False]
print(a[ : :-1])   #[2.5, [1, 'hello'], False, 'ali', 12]
```

لیست ها قابل تغییرند

مثال

```
a=[12 , 'ali' , False , [1 , 'hello'], 2.5 ]
a[2]=1526   #یعنی عدد 1526 را در جایگاه سوم قرار بده
print(a)       # [12, 'ali', 1526, [1, 'hello'], 2.5]
a[1:4]=['bye']
print(a)       #[12, 'bye', 2.5]
```

برای بدست آوردن طول یک لیست از تابع len() استفاده می شود.

مثال

```
a=[12 , 'ali' , False , [1 , 'hello'], 2.5 ]
print(len(a))
print(len([12 , 'ali' , False , [1 , 'hello'], 2.5 ]))
```

RUN

```
5
5
```

همچنین میتوان از تابع list() برای ساخت لیست مانند مثال زیر استفاده نمود.

مثال

```
thislist = list(("apple", "banana", "cherry"))
print(thislist)
```

RUN

```
['apple', 'banana', 'cherry']
```

برای الحاق دو لیست از عملگر + استفاده میکنیم؛ در اینصورت لیست دوم به انتهای لیست اول الحاق می شود.

مثال

```
a=[1 , 2]
b=[3 , 4]
print(a+b)
print([1 , 2] + [3 , 4])
```

RUN

```
[1, 2, 3, 4]
[1, 2, 3, 4]
```

اگر از عملگر ضرب استفاده کنیم به تعداد مورد نظر تمام عناصر لیست را بصورت زیر تکرار می کند:

مثال

```
a=[1 , 2]
b=[3 , 4]
print(a*4)
print([1 , 2] * 4 )
```

RUN

```
[1, 2, 1, 2, 1, 2, 1, 2]
[1, 2, 1, 2, 1, 2, 1, 2]
```

Tuple

تاپل(چندتایی) ساختمان داده ای است که با پرانتز احاطه شده است و مشابه لیست می باشد.

عناصر تاپل اندیس دارند، عضو تکراری می پذیرد ، انواع داده های مختلف را میتوان همزمان در یک تاپل قرار داد اما برخلاف لیست تغییر پذیر نیست.

عناصر تاپل مانند لیست اندیس دارند و دسترسی به عناصر آن مانند لیست از طریق اندیس عنصر مورد نظر می باشد.

مثال

دسترسی به عناصر در تاپلها:

```
    -5    -4      -3          -2          -1
     0     1       2           3           4
a=(12 , 'ali' , False , [1 , 'hello'], 2.5 )
print(a[2])        #False
print(a[3][1])     #hello
print(a[1:3])      #('ali', False)
print(a[0:4:2])    #(12, False)
print(a[4:1:-1])   #(2.5, [1, 'hello'], False)
print(a[-1:-5:-1]) #(2.5, [1, 'hello'], False, 'ali')
print(a[-5:-1])    #(12, 'ali', False, [1, 'hello'])
print(a[2:-1])     #(False, [1, 'hello'])
print(a[-1:2:-1])  #(2.5, [1, 'hello'])
print(a[-3: :-1])  #(False, 'ali', 12)
print(a[  :-4:-1]) #(2.5, [1, 'hello'], False)
print(a[ : :-1])   #(2.5, [1, 'hello'], False, 'ali', 12)
```

مثال

تاپلها قابل تغییر نیستند:

```
a=(12 , 'ali' , False , [1 , 'hello'], 2.5 )
a[0]=10
print(a)
```

RUN

ERROR

برای بدست آوردن طول یک تاپل از تابع len() استفاده می شود.

مثال

```
a=(12 , 'ali' , False , [1 , 'hello'], 2.5 )
print(len(a))
```

```
print(len((12 , 'ali' , False , [1 , 'hello'], 2.5 )))
```
RUN
```
5
5
```
همچنین میتوان از تابع tuple() برای ساخت تاپل مانند مثال زیر استفاده نمود.

مثال

```
thistuple = tuple(("apple", "banana", "cherry"))

print(thistuple)
```
RUN
```
('apple', 'banana', 'cherry')
```
برای الحاق دو تاپل از عملگر + استفاده میکنیم؛ در اینصورت تاپل دوم به انتهای تاپل اول اضافه می شود.

مثال

```
a=(1 , 2)
b=(3 , 4)
print(a+b)
print((1 , 2) + (3 , 4))
```
RUN
```
(1, 2, 3, 4)
(1, 2, 3, 4)
```
اگر از عملگر ضرب استفاده کنیم به تعداد مورد نظر تمام عناصر تاپل بصورت زیر تکرار می شوند:

مثال

```
a=(1 , 2)
print(a*3)
print((1 , 2) * 3)
```
RUN
```
(1, 2, 1, 2, 1, 2)

(1, 2, 1, 2, 1, 2)
```

Set

ست ها مشابه مجموعه ها در ریاضی اند و با { } احاطه شده اند.

عناصر ست اندیس مخصوص به خود ندارند(شماره گذاری نشده است) ، ست تغییرپذیر نیست و نیز داده های تکراری نمی پذیرد. اما انواع داده(دیتا تایپ ها) مختلف می پذیرد.

مثال

```
a={1 ,1 ,1,'ali' , 0 , False , 4 ,True, 2.3}
print(a)
print({1 ,1 ,1,'ali' , 0 , False , 4 ,True, 2.3})
```

RUN

```
{0, 1, 2.3, 4, 'ali'}
{0, 1, 2.3, 4, 'ali'}
```

نکته : مقادیر True و 1 در ست ها مشابه هم هستند

نکته : مقادیر False و 0 در ست ها مشابه هم هستند

همانطور که گفته شد ست مانند لیست و تاپل نیست که بتوان با استفاده از اندیس به عناصر آن دسترسی داشت:

مثال

```
a={1 ,1 ,1,'ali' , 0 , False , 4 ,True, 2.3}
print(a[0])   #ERROR
a[5]=4    #ERROR
print(a)
```

با استفاده از تابع len() میتوان طول یک ست را بدست آورد.

مثال

```
a={1 ,1 ,1,'ali' , 0 , False , 4 ,True, 2.3}
print(len(a))
```

RUN

```
5
```

از تابع set() میتوان برای ساخت ست استفاده کرد.

مثال

```
thisset = set(("apple", "banana", "cherry"))
print(thisset)
```

RUN

```
{'cherry', 'apple', 'banana'}
```

از عملگر های زیر میتوان برای ست ها استفاده کرد:

عملگر "-" برای تفاضل دو ست

عملگر "&" برای اشتراک دو ست

عملگر "|" برای اجتماع دو ست

مثال

```
a={1,2,3,4}
b={1,10,20,30}
print(a-b)
print(a&b)
print(a|b)
```

RUN

```
{2, 3, 4}
{1}
{1, 2, 3, 4, 10, 20, 30}
```

برای دسترسی به تمام عناصر یک ست میتوان از یک حلقه استفاده نمود:

مثال

```
a={1,2,3,4}
for i in a:
    print(i , end="    ")
```

RUN

```
1    2    3    4
```

Dictionary

دیکشنری (dict) هم مانند لیست و ست و تاپل یک ساختمان داده است.

یک عنصر در دیکشنری بصورت مقدار : کلید یا key: value می باشد.
عناصر با کاما از هم تفکیک می شوند و با { } احاطه شده اند.
دیکشنری اندیس گذاری نشده است (اما برای دسترسی به مقدار هر عنصر از کلید آن استفاده میکنیم که بعدا توضیح داده خواهد شد) همچنین داده های تکرار نمی پذیرد(یعنی اگر کلید و مقدار دو عنصر عینا یکسان باشد ، یکی از آنها را درنظر خواهد گرفت و نیز اگر کلید عناصری مشابه باشد اما مقدار های آنها متفاوت باشد ، آن عنصری که آخر آمده مورد قبول است.

مثال

```
a = { "ali": 19 ,"ali": 17 , "azhra": 18.5 , "azhra": 18.5  , "farid": 20 ,  }
print(a)
```

RUN

```
{'ali': 17, 'azhra': 18.5, 'farid': 20}
```

دسترسی به مقدار یک عنصر:
برای دسترسی به مقدار یک عنصر و یا تغییر مقدار یک عنصر، از کلید آن استفاده میکنیم:

مثال

```
d={'ali': 17, 'azhra': 18.5, 'farid': 20}
print(d['ali'])
d['ali']=19.5
print(d)
```

RUN

```
17
{'ali': 19.5, 'azhra': 18.5, 'farid': 20}
```

با استفاده از تابع len() میتوان طول یک دیکشنری را بدست آورد.

مثال

```
d={'ali': 17, 'azhra': 18.5, 'farid': 20}
print(len(d))
```

RUN

```
3
```

می توان با استفاده از تابع dic() دیکشنری ساخت.

مثال

```
d=dict(ali=19 , reza=18 , hossein=20)
print(d)
```
RUN
```
{'ali': 19, 'reza': 18, 'hossein': 20}
```
حلقه در دیکشنری:

اگر در یک دیکشنری بصورت زیر حلقه تشکیل دهیم ، کلید ها را به ما می دهد، اما بعد از تدریس متدهای دیکشنری می توانیم با استفاده از متدهای values() و items() به مقدار و کلید و مقدار دسترسی داشت.

مثال

```
d=dict(ali=19 , reza=18 , hossein=20)
for i in d:
    print(i ,end="    ")
```
RUN
```
ali    reza    hossein
```
مانند مثال زیر می توان یک عنصر را به یک دیکشنری بدون استفاده از متدی اضافه نمود(به آخر دیکشنری اضافه میشود):

مثال

```
d={'ali': 19, 'reza': 18, 'hossein': 20}
d['sara']=17
print(d)
```
RUN
```
{'ali': 19, 'reza': 18, 'hossein': 20, 'sara': 17}
```

تابع (function)

تابع یک بلاک از کد است که فقط زمانی که فراخوانی می شود، اجرا می شود. می توان داده ها را به عنوان آرگومان ، به یک تابع منتقل کرد.یک تابع می تواند نتیجه عملیات روی داده ها را برگرداند.
انواع تابع: 1- کتابخانه ای 2- توابعی که برنامه نویس می نویسد.

توابعی که برنامه نویس می نویسد:

ایجاد یک تابع

در پایتون یک تابع با استفاده از کلمه کلیدی def تعریف می شود:

```
def نام تابع (پارامترها) :
        بدنه تابع
```

مثال

```
def my_function():
    print("Hello from a function")
```

فراخوانی یک تابع(call)

```
نام تابع (آرگومانها)
```

برای فراخوانی یک تابع، نام تابع به همراه پرانتز و آرگومان های آن را می آوریم. (**توجه**: فراخوانی تابع ، خارج از بلاک مربوط به تابع است)

مثال

```
def my_function():
    print("Hello from a function")
my_function()
```

RUN

```
Hello from a function
```

آرگومانها

اطلاعات را می توان به عنوان آرگومان به توابع منتقل کرد. در هنگام فراخوانی، آرگومان ها بعد از نام تابع در داخل پرانتز مشخص می شوند .آرگومانها با کاما از هم جدا میشوند.

مثال

```
def my_function(fname):
    print(fname + " amad")
```

```
my_function("ali")
my_function("reza")
my_function("zahra")
```

RUN

```
ali amad
reaza amad
zahra amad
```

پارامترها و آرگومان ها:

پارامتر متغیری است که در داخل پرانتز در تعریف تابع مشخص شده است.
آرگومان مقداری است که هنگام فراخوانی تابع به آن ارسال می شود.

تعداد آرگومانها

الف) آرگومانهای اجباری(positional): یک تابع باید با تعداد آرگومان های مشخص فراخوانی شود .به این معنی که اگر تابع شما 2 پارامتر دارد، باید تابع را با 2 آرگومان فراخوانی کنید، نه بیشتر و نه کمتر.

مثال

این تابع 2 پارمتر دارد پس 2 آرگومان دریافت می کند:

```
def my_function(fname, lname):
        print(fname + " " + lname)
my_function("ali", "ahmadi")
```

RUN

```
ali ahmadi
```

اگر بخواهید تابع بالا را با مثلا 1 یا 3 آرگومان فراخوانی کنید، با ارور مواجه خواهید شد :

مثال

```
def my_function(fname, lname):
        print(fname + " " + lname)
my_function("ali")
```

RUN

```
Error
```

مثال

تابعی بنویسید که اعدادی را بگیرد و کوچکترین آنها را نمایش دهد

```
def comp(a,b):
   if a < b :
      return a
   else:
      return b
result = comp(20,10)
print(result)
def other_comp(x,y,z,f):  #برای مقایسه 4 عدد بجای اینکه تابع جداگانه بنویسیم از تابع مقایسه دو عدد استفاده میکنیم
   return comp(comp(x,y), comp(z,f))
print(other_comp(10,12,2,14))
```

RUN

10

2

ب)آرگومانهای با طول متغیر: *args

اگر نمی دانید چه تعداد آرگومان به تابع شما ارسال می شود یک * قبل از نام پارامتر در تعریف تابع اضافه کنید.

به این ترتیب تابع تاپلی از آرگومانها دریافت می کند و می تواند به آیتمها(عناصر) دسترسی داشته باشد:

مثال

```
def my_function(*kids):
      print("The youngest child is " + kids[2])
my_function("ali", "reza", "zahra")
```

RUN

The youngest child is zahra

مثال

```
def sum_all(*s):
    sum1 =0
    for i in range(0, len(s)):
        sum1 = sum1 + s[i]
```

```
    return sum1
print(sum_all(1,2,3,4,5,6))
```

RUN
21

ج) آرگومانهای کلمه کلیدی: Keyword Arguments

همچنین می توانید آرگومانها را با دستور key = value ارسال کنید.
در این حالت ترتیب آرگومان ها مهم نیست.

مثال

```
def my_function(child3, child2, child1):
    print("The youngest child is " + child3)
my_function(child1 = "reza", child2 = "zahra", child3 = "ali")
```

RUN
The youngest child is ali

د)آرگومان های کلیدی دلخواه، **kwargs

اگر نمی دانید چند keyword arguments به تابع شما ارسال می شود، قبل از نام پارامتر در تعریف تابع دو ستاره ** وارد نمایید.
به این ترتیب تابع یک دیکشنری از آرگومان ها را دریافت می کند و می تواند به آیتمها دسترسی داشته باشد:

مثال

```
def my_function(**kid):
    print("His last name is " + kid["lname"])
my_function(fname = "ali", lname = "ahmadi")
```

RUN
His last name is ahmadi

آرگومانهای با مقدار پیش فرض پارامتر

اگر تابع را بدون آرگومان فراخوانی کنیم، از مقدار پیش فرض استفاده می کند:

مثال

```
def my_function(country = "Norway"):
    print("I am from " + country)
my_function("Sweden")
```

```
my_function("India")
my_function()
my_function("Brazil")
```

RUN

```
I am from Sweden
I am from India
I am from Norway
I am from Brazil
```

ارسال یک لیست به عنوان یک آرگومان

شما می توانید هر نوع داده ای از آرگومانها را به یک تابع ارسال کنید (رشته، عدد، لیست، دیکشنری و غیره)، و به عنوان همان نوع داده در داخل تابع در نظر گرفته می شود.

به عنوان مثال، اگر لیستی را به عنوان آرگومان ارسال کنید، زمانی که به تابع برسد، همچنان یک لیست خواهد بود:

مثال

```
def my_function(food):
        for x in food:
                print(x)
fruits = ["apple", "banana", "cherry"]
my_function(fruits)
```

RUN

```
apple
banana
cherry
```

مقدار های بازگشتی

برای اینکه یک تابع مقداری را برگرداند، از عبارت return استفاده می کنیم.

مثال

```
def my_function(x):
        return 5 * x
print(my_function(3))
print(my_function(5))
print(my_function(9))
```

RUN
15
25
45

عبارت pass

تابع در هنگام تعریف بدنه نمی توانند خالی باشند، اما اگر به دلایلی تعریف تابعی بدون بدنه دستورات دارید ، برای جلوگیری از خطا، عبارت pass را جای دستورات آن قرار دهید.

مثال

```
def myfunction():
        pass
```

توابع بازگشتی

یعنی یک تابع خودش را فراخوانی می کند .مزیت این است که می توانید برای رسیدن به نتیجه در داده ها حلقه تشکیل دهید.

مثال

فاکتوریل با استفاده از توابع بازگشتی

```
def fact(num):
            if num==0 : return 1
            else: return num * fact(num-1)
s=int(input("enter number:"))
print(fact(s))
```

RUN

```
enter number:4
24
```

مثال

دنباله فیبوناچی را با استفاده از توابع بازگشتی پیاده سازی نمایید

```
def fibona(n):
   if n == 1 : return 1
   elif n == 2 : return 1
   else : return fibona(n-1) + fibona(n-2)
print(fibona(8))
```

RUN

```
21
```

متغیرهای سراسری (global)

متغیرهای Global می توانند برای همه جا استفاده شوند، هم در داخل توابع و هم در خارج.

مثال

یک متغیر خارج از یک تابع ایجاد کنید و از آن در داخل تابع استفاده کنید

```
x = "awesome"
def   myfunc() :
        print("Python is " + x)
myfunc()

print("Python is " + x)
```

RUN

```
Python is awesome

Python is awesome
```

متغیر محلی (local)

اگر متغیری با همین نام در داخل یک تابع ایجاد کنید، این متغیر محلی (local) خواهد بود و فقط در داخل تابع قابل استفاده است .

متغیر سراسری با همان نام و به همان طوری که بود، سراسری و با مقدار اصلی باقی می ماند.

مثال

یک متغیر در داخل یک تابع، با همان نام متغیر سراسری ایجاد کنید:

```
x = "awesome"
def myfunc():
        x = "fantastic"
        print("Python is " + x)
myfunc()
print("Python is " + x)
```

RUN

```
Python is fantastic
Python is awesome
```

کلمه کلیدی global

به طور معمول، وقتی یک متغیر را در داخل یک تابع ایجاد می کنید، آن متغیر محلی است و فقط می تواند در داخل آن تابع استفاده شود.

برای ایجاد یک متغیر سراسری در داخل یک تابع، می توانید از کلمه کلیدی global استفاده کنید.

مثال

```
def myfunc():
    global x
    x = "fantastic"
myfunc()
print("Python is " + x)
```

RUN

Python is fantastic

همچنین اگر می خواهید متغیر سراسری را در داخل یک تابع تغییر دهید از کلمه کلیدی global استفاده کنید.

مثال

```
x = "awesome"
def myfunc():
    global x
    x = "fantastic"
myfunc()
print("Python is " + x)
```

RUN

Python is fantastic

مثال

برنامه ای بنویسید که یک عدد خوانده و مربع و مکعب آنرا نمایش دهد

```
def myfunction():
   n=int(input('enter a number:'))
   return  n**2  ,  n**3
print(myfunction())
```

RUN

```
enter a number:10
(100, 1000)
```

مثال

برنامه ای بنویسید که یک عدد از کاربر بگیرد و تعیین کند که آیا این عدد تام است یا خیر؟

```
def justsum(num):
      sum1 = 0
      for j in range(1, num):
         if num % j == 0:
            sum1 = sum1 + j
      return sum1
def istaam(num):
      if justsum(num) == num:
         return True
      else:
         return False
num = int(input("enter a number: "))
if istaam(num) == True:
      print('the number is perfect')
else:
```

```
    print('the number is not perfect')
```

RUN

```
enter a number: 6
the number is perfect
```

مثال

لیست ها تغییر پذیرند ؛ در مثال زیرتابع return ندارد و مستقیما هم در داخل تابع پرینت نشده است اما تغییرات در لیست اعمال شده است.

```
mylist = [10, 2 , 10]
def myfunction(a):
   a[0] *=2
myfunction(mylist)
print(mylist)
```

RUN

```
[20, 2, 10]
```

مثال

برای درک بهتر آرگومانهای positional و keyword و مقدار پیشفرض:

```
def myfunction(x,  y=5 , z=7) :
   print(x,y,z)
myfunction(10)
myfunction(5, 15)
myfunction(20, 30, 40)
myfunction(12,  z=20)
myfunction(z=10,  y=12  ,x=17)
```

RUN

```
10 5 7
5 15 7
20 30 40
12 5 20
17 12 10
```

مثال

ماشین حساب ساده:

برنامه ای بنویسید که دو عدد و عملگر موردنظر را از کاربر بگیرد و عملیات ریاضی مورد نظر را انجام دهید.اگر کاربر عملگری وارد نکرد آن دو عدد را در هم ضرب کند!

```
def calculator(number1, number2, operator="*"):
```

```
    if operator == "+":
        print(number1, operator, number2, "=", number1 + number2)
    elif operator == "-":
        print(number1, operator, number2, "=", number1 - number2)
    elif operator == "*":
        print(number1, operator, number2, "=", number1 * number2)
    elif operator == "/":
        if number2 == 0:print('the second number cannot be zero!')
        print(number1, operator, number2, "=", number1 / number2)
    elif operator == "%":
        print(number1, operator, number2, "=", number1 % number2)
    elif operator == "//":
        print(number1, operator, number2, "=", number1 // number2)
    elif operator == "&":
        print(number1, operator, number2, "=", number1 & number2)
    elif operator == "|":
        print(number1, operator, number2, "=", number1 | number2)
    elif operator == "^":
        print(number1, operator, number2, "=", number1 ^ number2)
    else:
        print('please enter a valid operator')

number1 = int(input('please enter a first number: '))
number2 = int(input('please enter a second number: '))
operator = input('please enter a operator: ')
calculator(number1, number2, operator)
```

RUN

```
please enter a first number: 2
please enter a second number: 3
please enter a operator: &
2 & 3 = 2
```

مثال

برنامه ای بنویسید که شماره جمله سری فیبوناچی را بگیرد و از ابتدا تا آن شماره جمله ، جملات سری فیبوناچی را چپ کند.(با استفاده از تابع بازگشتی)

```
def fibona(num):
   if num == 1: return 1
   elif num == 2: return 1
   elif num >= 3: return fibona(num - 2) + fibona(num - 1)
num = int(input("Enter a number :"))
for i in range(1, num+1):
   print(fibona(i) , end="    ")
```

RUN

```
Enter a number :7
1    1    2    3    5    8    13
```

مثال

تابعی بنویسید که نمرات میانترم و پایانترم و ضریب درس مربوط به یک دانشجو برای دروس مختلف را دریافت کند و در پایان معدل او را محاسبه نماید.(میانترم 30 درصد و پایانترم 70 درصد ارزشیابی را شامل می شوند همچنین نمره زیر 10 در معدل حساب نشود.)

```
def average_stu():
      sum1 = 0
      count = 0
      while True:
            mid = float(input("mid term : "))
            final = float(input("final : "))
            factor = float(input("factor : "))
            question = input("do you want to continue? (y/n): ")
            nomre_dars = (mid*0.3+final*0.7)
            if nomre_dars < 10:
                  continue
            sum1 = sum1 + nomre_dars*factor
            count += factor
            if question == 'y':
```

```
                continue
        elif question == 'n':
                return (sum1/count)
print(average_stu())
```

RUN

```
mid term : 10
final : 10
factor : 2
do you want to continue? (y/n): y
mid term : 12
final : 8
factor : 3
do you want to continue? (y/n): y
mid term : 20
final : 18
factor : 3
do you want to continue? (y/n): n
15.16
```