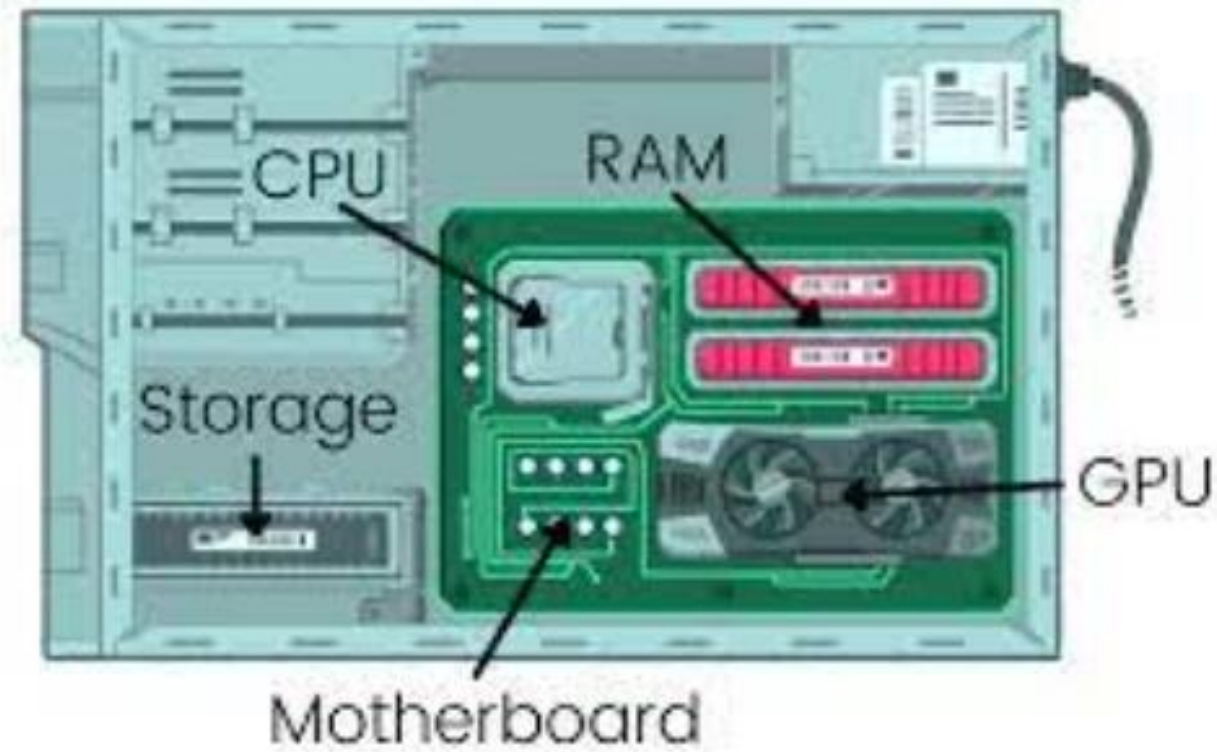


پایتون مقدماتی

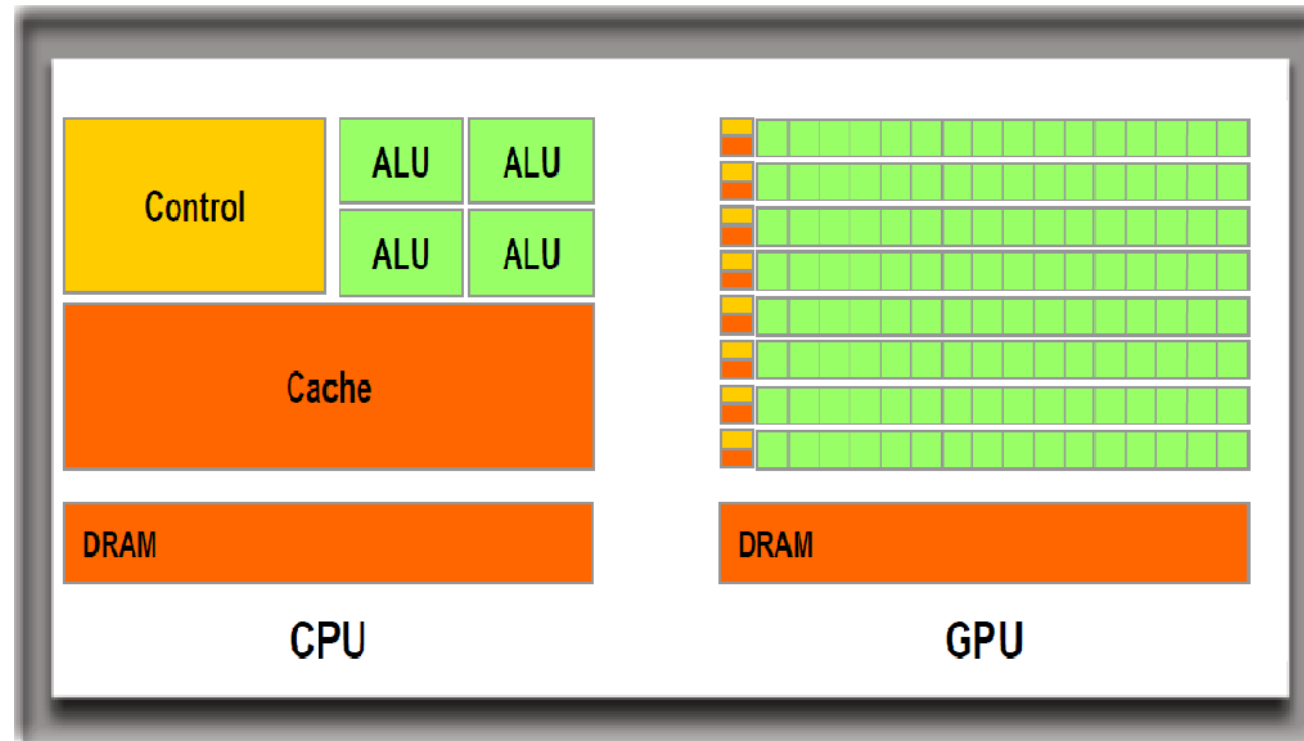
مرتضی مالکی

زمستان ۱۴۰۱

کامپیوتر چیست؟



تفاوت CPU و GPU



تفاوت CPU و GPU



[Guido van Rossum](#)

[Guido van Rossum](#) began working on Python in the late 1980s as a successor to the [ABC programming language](#) and first released it in 1991 as Python 0.9.0

Python 2.0 was released in 2000

تفاوت CPU و GPU



[Guido van Rossum](#)

[Guido van Rossum](#) began working on Python in the late 1980s as a successor to the [ABC programming language](#) and first released it in 1991 as Python 0.9.0

Python 2.0 was released in 2000

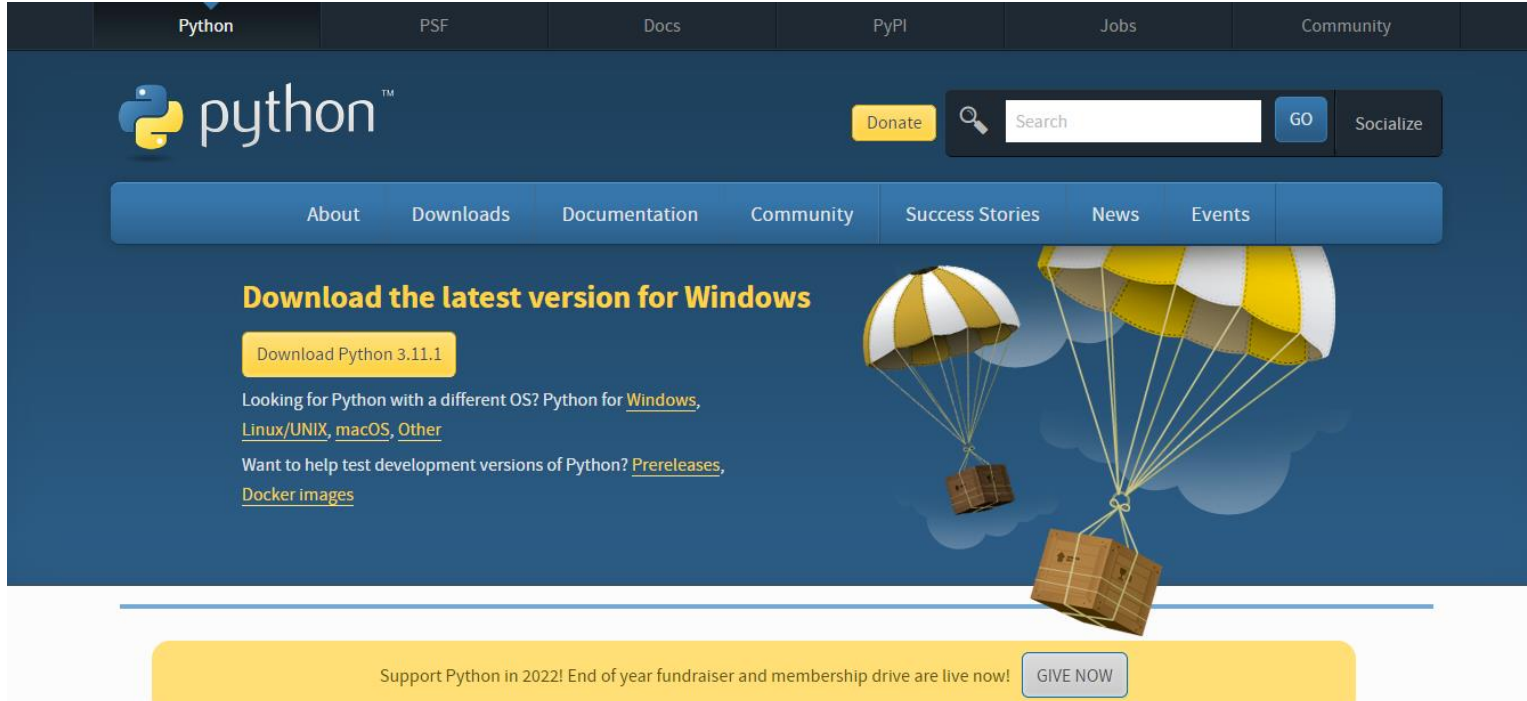
اینا چی هستن؟

ورژن

۱.۰.۹.۱۷.۴۴.۴۲۳

نصب پایتون

<https://www.python.org/downloads/>



خط فرمان Command line Terminal

```
john@ubuntu: ~  
john@ubuntu:~$ ls  
john_directory john_file  
john@ubuntu:~$ ls -l  
total 8  
drwxrwxr-x 2 john john 40 Oct 1 11:10 john_directory  
-rw-rw-r-- 1 john john 5120 Oct 1 11:17 john_file  
john@ubuntu:~$ ls -l -h  
total 8.0K  
drwxrwxr-x 2 john john 40 Oct 1 11:10 john_directory  
-rw-rw-r-- 1 john john 5.0K Oct 1 11:17 john_file  
john@ubuntu:~$ ls -lh john_file  
-rw-rw-r-- 1 john john 5.0K Oct 1 11:17 john_file  
john@ubuntu:~$ ls -l --human-readable john_file  
-rw-rw-r-- 1 john john 5.0K Oct 1 11:17 john_file  
john@ubuntu:~$
```

```
Command Prompt  
12/31/2020 05:33 PM 26,284 win98pro.htm  
12/31/2020 05:33 PM 32,586 windows-10.htm  
12/31/2020 05:33 PM 30,989 windows7.htm  
12/31/2020 05:33 PM 30,165 windows8.htm  
12/31/2020 05:33 PM 10,927 winipcfg.htm  
12/31/2020 05:33 PM 23,545 winmega.htm  
12/31/2020 05:33 PM 21,640 winntqa.htm  
12/31/2020 05:33 PM 32,706 winxpqa.htm  
12/31/2020 05:33 PM 62,952 wmic.htm  
05/27/2010 03:53 PM 745 wp.jpg  
12/31/2020 05:33 PM 23,328 wqanda.htm  
12/31/2020 05:33 PM 22,675 xcopyhlp.htm  
12/31/2020 05:33 PM 18,154 xdoseror.htm  
12/31/2020 05:33 PM 15,148 xext.htm  
12/31/2020 05:33 PM 12,703 yext.htm  
12/31/2020 05:33 PM 11,784 youtube.htm  
12/31/2020 05:33 PM 12,264 zext.htm  
10/12/2020 12:09 PM <DIR> _notes  
432 File(s) 7,679,571 bytes  
38 Dir(s) 254,861,524,992 bytes free  
  
C:\ch>echo computerhope.com  
computerhope.com  
  
C:\ch>
```


خط فرمان

mkdir

rmdir

Ctrl + C

cd

IDLE

IDLE is an integrated development environment for Python

مفسر تعاملی

>>>

اجرای اسکریپت

۱. جایگذاری عبارت و انواع داده:

عناوین:

- مود تعاملی پایتون
- متغیرها
- عبارت ها
- جایگذاری
- رشته، اعداد صحیح و اعشاری

شِل تعاملی پایتون

پایتون را می توان به عنوان ماشین حسابی قدرتمند در نظر گرفت.
کافیه در شل سیستم خودتون **python** را تایپ کنید.
اگر پایتون نصب باشد با **prompt** به شکل زیر مواجه خواهید شد:

```
>>>
```

بیایید مساحت دایره را حساب کنیم:

```
>>> r = 10  
>>> A = 3.14*r*r  
>>> print(A)  
314.0
```

تفاوت دستورات پایتون با ریاضی

```
>>> r = 10  
>>> A = 3.14*r*r  
>>> print(A)  
314.0
```

در پایتون نمی‌توان به شکل $A = 3.14xr \times r$ نوشت.

تفاوت دستورات پایتون با ریاضی

```
>>> r = 10
>>> A = 3.14*r*r
>>> print(A)
314.0
```

در پایتون نمی توان به شکل $A = 3.14xr \times r$ نوشت.

تفاوت دستورات پایتون با ریاضی

```
>>> r = 10
>>> A = 3.14*r**2
>>> print(A)
314.0
```

در پایتون توان را به صورت ****** می نویسیم.

تفاوت دستورات پایتون با ریاضی

```
>>> r = 10
>>> A = 3.14*r**2
>>> print(A)
314.0
```

r و A متغیر هستند، در ریاضیات هم متغیر داریم ولی متغیر در برنامه نویسی مقداری تفاوت دارد.

متغیرها

```
>>> r = 10  
>>> A = 3.14*r**2
```

r -> 10

A -> 314.0

یک **متغیر** نام یک مکان در حافظه RAM می باشد، متغیر را مانند یک جعبه در نظر بگیرید.

متغیر **مقداری** را در خود جای می دهد، مقدار را مانند محتویات جعبه در نظر بگیرید.

عبارت جایگذاری

```
>>> r = 10
```

علامت **=** نشان دهنده ی جایگذاری است؛ در عبارت `r=10` متغیر `r` ساخته می شود و مقدار `10` برای آن جایگذاری می شود.

عبارت جایگذاری

```
>>> r = 10  
>>> A = 3.14*r**2
```

$r \rightarrow 10$

$A \rightarrow 314.0$

متغیر را می توان در یک عبارت استفاده کرد، مانند $3.14*r**2$.
با این کار عبارت حساب می شود و سپس ذخیره می شود.

ترتیب مهم است

```
>>> A = 3.14*r**2
```

```
>>> r = 10
```

```
NameError: name 'r' is not defined
```

r -> 10

A -> 314.0

در تعریف و استفاده از متغیرها ترتیب مهم است و قبل از استفاده از یک متغیر حتما قبلش باید تعریف شود.

جایگذاری در مقابل برابری

```
>>> r = 10
```

```
>>> 3.14*r**2 = A
```

```
SyntaxError: can't assign to an operator
```

در ریاضیات $=$ به معنای برابری دو سمت مساوی است،
در پایتون این علامت یک عملیات را نشان می دهد، به این معنا که
عبارت سمت راست را حساب کن و در متغیر سمت چپ قرار بده

عبارت جایگذاری

>>> r = 10	r -> 10
>>> A = 3.14*r**2	A -> 314.0
>>> S = A/2	S -> 157.0

در اینجا ما متغیر S را با مساحت نیم دایره مقداردهی می کنیم.

عبارت جایگذاری

>>> r = 10	r -> 10
>>> A = 3.14*r**2	A -> 314.0
>>> A = A/2	A -> 157.0

در اینجا ما متغیر A را با مساحت نیم دایره مقداردهی می کنیم.
در خط آخر کد ما نصف A را به A دادیم.

دنبال کردن به روزرسانی

```
>>> y = 100
```

در اینجا ما متغیر A را با مساحت نیم دایره مقداردهی می کنیم.
در خط آخر کد ما نصف A را به A دادیم.

دنبال کردن به روزرسانی

```
>>> y = 100
```

```
>>> t = 10
```

```
y -> 100
```

```
t -> 10
```

در اینجا ما متغیر A را با مساحت نیم دایره مقداردهی می کنیم.
در خط آخر کد ما نصف A را به A دادیم.

دنبال کردن به روزرسانی

```
>>> y = 100
```

```
>>> t = 10
```

```
>>> y = y+t
```

y -> 100

t -> 10

y -> 110

دنبال کردن به روزرسانی

```
>>> y = 100  
>>> t = 10  
>>> y = y+t  
>>> t = t+10
```

y -> 100

t -> 10

y -> 110

t -> 20

دنبال کردن به روزرسانی

```
>>> y = 100  
>>> t = 10  
>>> y = y+t  
>>> t = t+10  
>>> y = y+t
```

y -> 100

t -> 10

y -> 110

t -> 20

y -> 130

معادله در مقابل جایگذاری

در ریاضیات،

$$t = t + 10$$

در ریاضیات این عبارت درست نیست.

معادله در مقابل جایگذاری

در ریاضیات،

$$t = t + 10$$

در ریاضیات این عبارت درست نیست.

در پایتون،

$$t = t + 10$$

به معنای افزودن مقدار ۱۰ به t است و ذخیره نتیجه در t مقداردهی می شود.

جایگذاری به طور کلی

< variable name > = < expression >

1. عبارت سمت راست محاسبه می شود.
2. نتیجه را در نام متغیر سمت راست ذخیره می کنیم.

نام گذاری متغیرها

```
>>> radius = 10  
>>> Area = 3.14*radius**2
```

radius -> 10

Area -> 314.0

قوانین نام گذاری:

1. نام باید متشکل از اعداد، حروف بزرگ، حروف کوچک و علامت زیر خط _ باشد.
2. نامها باید با حروف یا _ شروع شوند.

تقدم محاسبات

ترتیب عملیات محاسباتی به چه شکل است؟

در محاسبات ترتیب عملیات به شکل زیر است:

این عبارت

$$A + B * C$$

$$-A ** 2 / 4$$

$$A * B / C * D$$

همانند این عبارت است

$$A + (B * C)$$

$$- (A ** 2) / 4$$

$$((A * B) / C) * D$$

پراتنز

توان

ضرب

تقسیم

جمع

تفریق

(1

(2

(3

(4

(5

(6

بهتر است تا جایی که ممکن است از پراتنز استفاده کنیم.

اعداد صحیح و اعشاری

در ریاضیات بین اعداد صحیح و اعشاری تفاوتی قائل هستیم.

اعداد صحیح:

100, 0, -89, 1234567

اعداد اعشاری:

-2.1, 100.01, 100.0, 12.345

int و float

در پایتون اعداد نوع دارند.

اعداد صحیح با نوع **int** شناخته می شوند.

اعداد اعشاری با نوع **float** شناخته می شوند.

int

```
>>> x = 30  
>>> y = 8  
>>> q = x/y  
>>> print(q)  
3.75
```

float

```
>>> x = 30.  
>>> y = 8.  
>>> q = x/y  
>>> print(q)  
3.75
```


توابع از پیش تعریف شده

همواره پرانتز باز و بسته را دارند \longrightarrow **type()** \longleftarrow توابع معمولاً با حروف کوچک نوشته می شوند

```
>>> x = 30.0  
>>> type(x)  
<class float>
```

تبدیل نوع داده به صورت صریح

```
>>> x = 30.0  
>>> y = 8.0  
>>> q = int(x)/int(y)  
>>> print(q)  
3
```

تفاوت محاسبات `float` و `int`

محاسبات `int` دقیق هستند.

محاسبات `float` دقیق نیستند.

تفاوت محاسبات `int` و `float`

محاسبات `int` دقیق هستند.

محاسبات `float` (معمولا) دقیق نیستند.

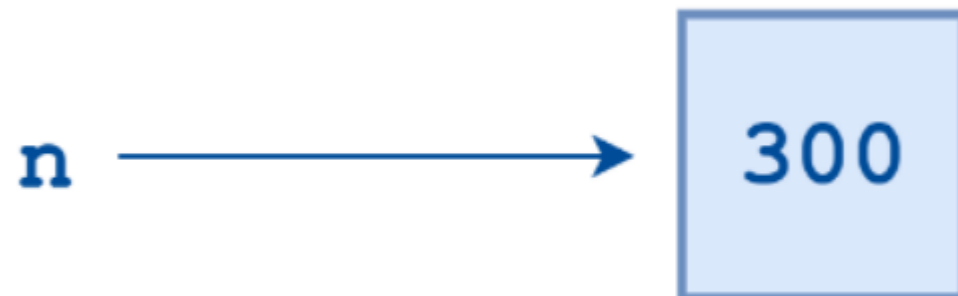
```
>>> x = 1.0/3.0  
>>> print(x)  
.3333333333333333
```

ارجاع به اشيا

شيء

ارجاع

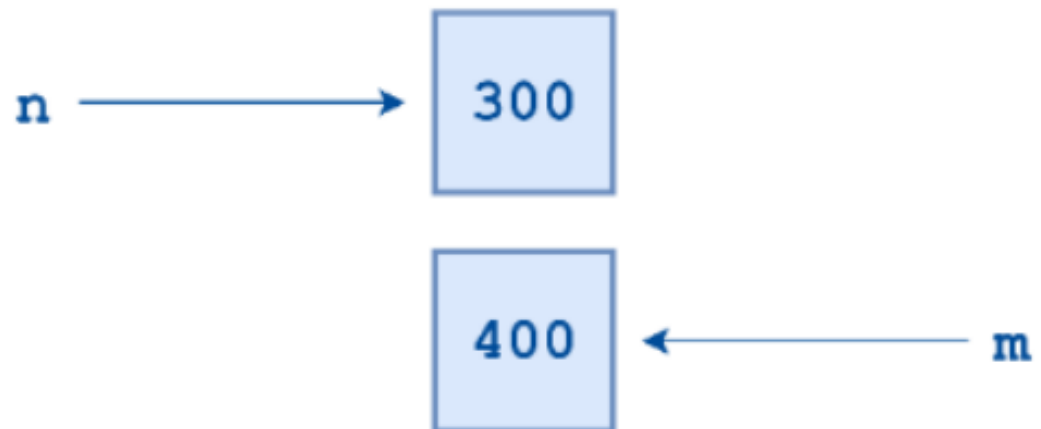
ارجاع به اشیا



ارجاع به اشیا



ارجاع به اشیا



رشته‌ها `strings`

تا الان محاسبات اعداد را بحث کردیم.

حال محاسبات بر روی نوشته (`text`) را بررسی می‌کنیم، ما از رشته یا `string` برای نمایش نوشته استفاده می‌کنیم.

رشته

```
>>> s1 = 'abc'  
>>> s2 = 'ABC'  
>>> s3 = ' A B C '
```

```
>>> s1 = "abc"  
>>> s2 = "ABC"  
>>> s3 = " A B C "
```

s1 و s2 و s3 متغیرهای با مقدار رشته هستند

رشته

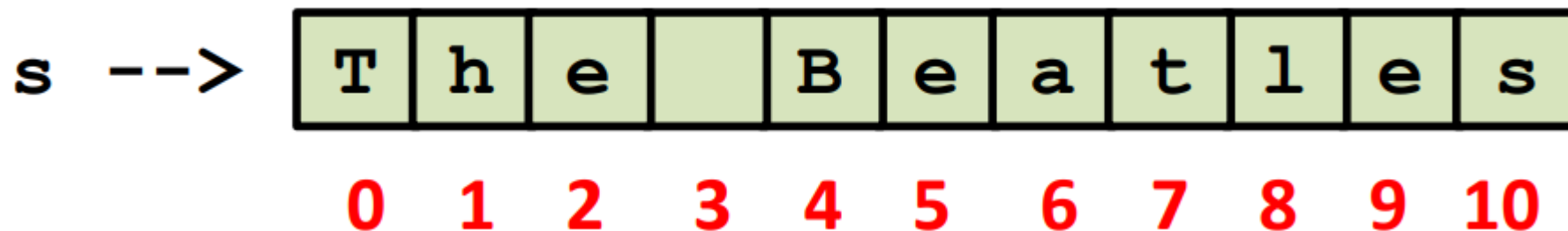
```
>>> s1 = 'abc'  
>>> s2 = 'ABC'  
>>> s3 = ' A B C '
```

```
>>> s1 = "abc"  
>>> s2 = "ABC"  
>>> s3 = " A B C "
```

هر سه متغیر بالا متفاوت هستند، چون فاصله، کوچک یا بزرگ بودن حروف مهم است.

رشته‌ها اندیس دارند

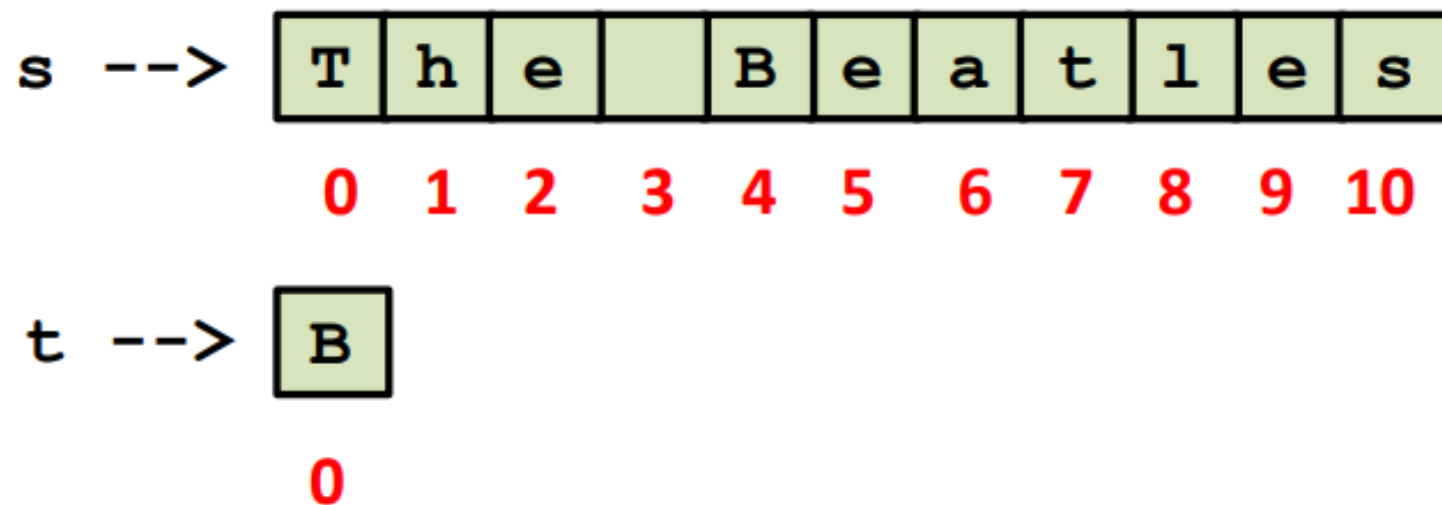
```
>>> s = 'The Beatles'
```



می‌توان عناصر یک رشته را با اندیسشان صدا زد، به این کار subscripting می‌گویند.

رشته‌ها اندیس دارند

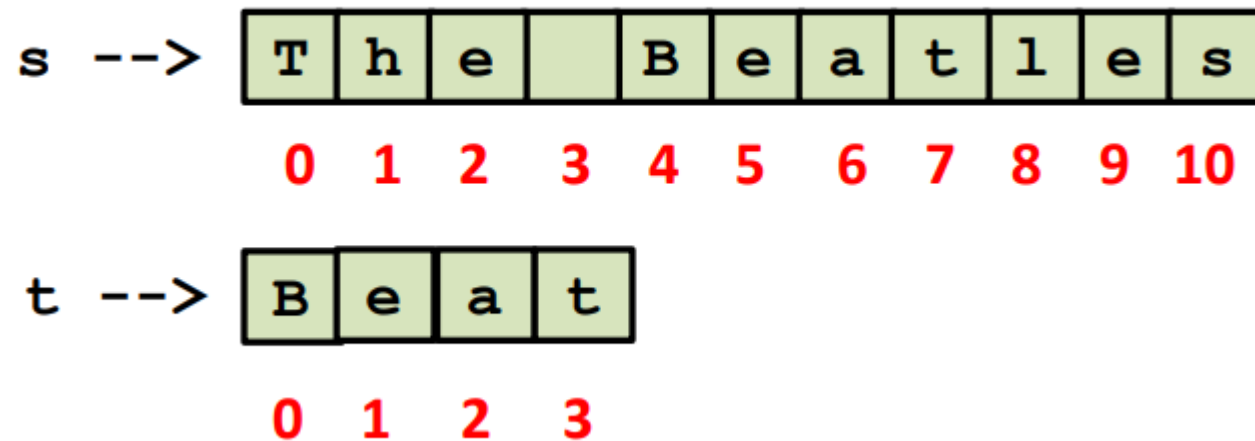
```
>>> s = 'The Beatles'  
>>> t = s[4]
```



برای دسترسی به عنصری از رشته از `[]` استفاده کردیم، بعدتر می بینیم که خیلی استفاده دارد.
توجه: یک کاراکتر خود یک رشته است.

برش رشته‌ها

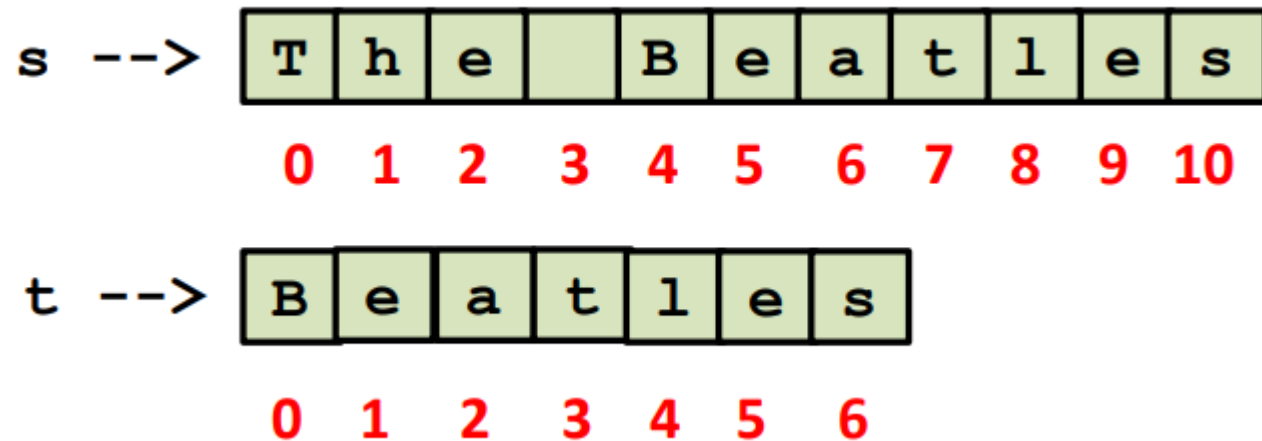
```
>>> s = 'The Beatles'  
>>> t = s[4:8]
```



ما می‌گوییم که t برشی از s است.

برش رشته‌ها

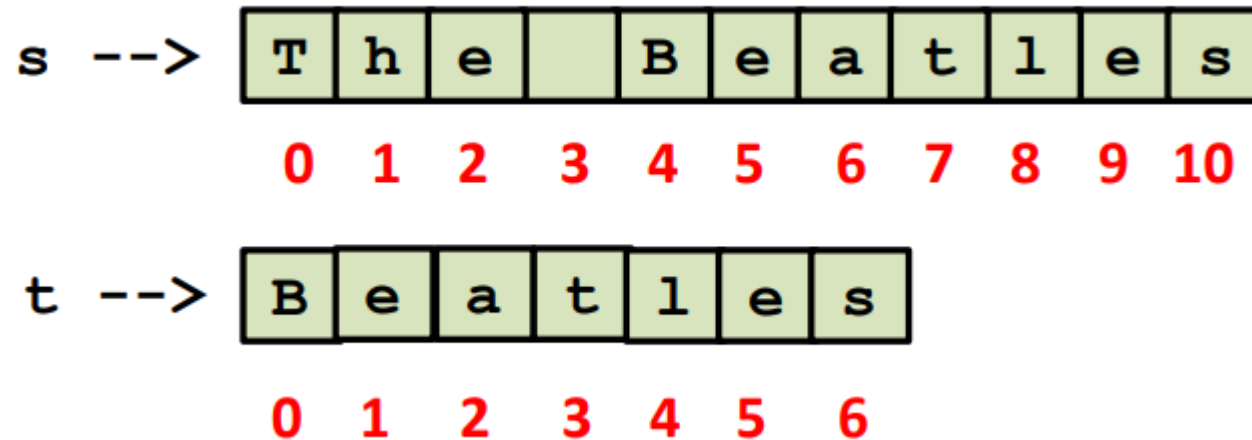
```
>>> s = 'The Beatles'  
>>> t = s[4:]
```



مثل حالت `s[4:11]` است، ولی این موقعی که تا انتها را می‌خواهیم راحت‌تر است.

برش رشته‌ها

```
>>> s = 'The Beatles'  
>>> t = s[:4]
```



مثل حالت `s[0:4]` است، ولی این موقعی که از ابتدا را می‌خواهیم راحت‌تر است.

برش رشته‌ها

```
>>> s = 'The Beatles'
```

```
>>> t = s[11]
```

IndexError: string index out of range

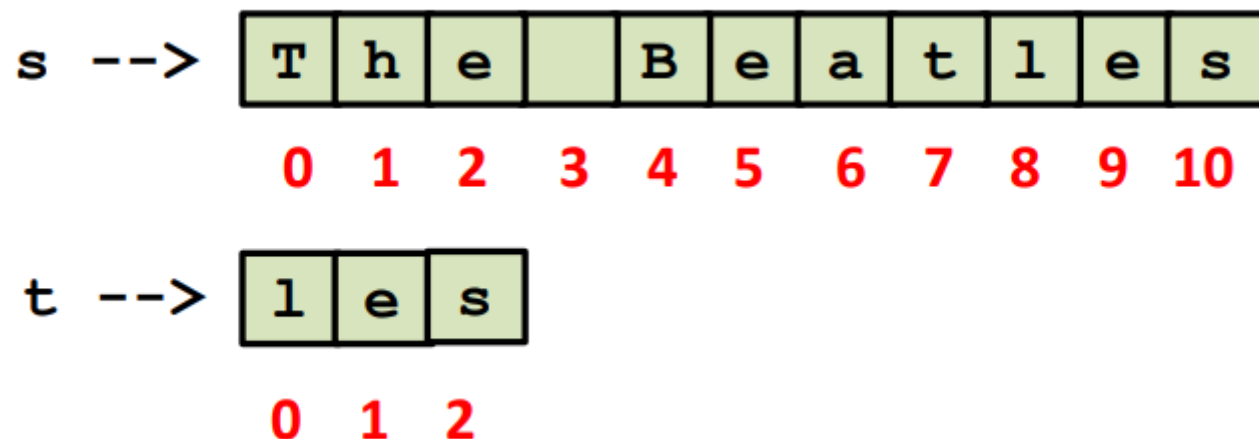
s -->

T	h	e		B	e	a	t	l	e	s
0	1	2	3	4	5	6	7	8	9	10

ما `s[11]` نداریم، برای همین این کاری غیر قانونی بود.

برش رشته‌ها

```
>>> s = 'The Beatles'  
>>> t = s[8:20]
```



انتخاب اندیس با بیش از تعداد کاراکترهای موجود مشکلی ندارد.

رشته‌ها را می‌توان ترکیب کرد

```
>>> s1 = 'The'  
>>> s2 = 'Beatles'  
>>> s = s1+s2
```

s -->

T	h	e	B	e	a	t	l	e	s
---	---	---	---	---	---	---	---	---	---

به این کار اتصال یا **concatenation** می‌گویند.

رشته‌ها را می‌توان ترکیب کرد

```
>>> s1 = 'The'  
>>> s2 = 'Beatles'  
>>> s = s1 + ' ' + s2
```

s -->

T	h	e		B	e	a	t	l	e	s
---	---	---	--	---	---	---	---	---	---	---

ما یک فاصله افزودیم.

هیچ محدودیتی برای جمع رشته‌ها وجود ندارد. $s = s2+s2+s2+s2+s2$

رشته‌ها را می‌توان ترکیب کرد

```
>>> s1 = 'The'  
>>> s2 = 'Beatles'  
>>> s = s1 + ' ' + s2
```

s -->

T	h	e		B	e	a	t	l	e	s
---	---	---	--	---	---	---	---	---	---	---

ما یک فاصله افزودیم.

هیچ محدودیتی برای جمع رشته‌ها وجود ندارد. $s = s2+s2+s2+s2+s2$

نوع داده

رشته یک نوع است: str

بنابراین تا الان سه نوع داده را معرفی کردیم:

int اعداد صحیح

-12

float اعداد اعشاری

9.12, -12.0

str رشته‌ها

'abc', '12.0'

پایتون انواع دیگری نیز دارد، بعداً خودمان هم یک مقادیری برای خودمان خواهیم ساخت

نوع داده ترکیبی از مقادیر و عملیات روی آنها است.

int 123, -123, 0

float 1.0, -.00123, -12.3e-5

str 'abcde', '123.0'

نماد e برای توان استفاده می‌شود، برای توان های بزرگ بهتر از توان عادی است.

تبدیل نوع

```
>>> s = '123.45'  
>>> x = 2*float(s)  
>>> print(x)  
246.90
```

نوشته‌ی حاوی عدد اعشاری را می‌توان به **float** تبدیل کرد.

تبدیل نوع

```
>>> s = '-123'  
>>> x = 2*int(s)  
>>> print(x)  
-246
```

نوشته‌ی حاوی عدد صحیح را می‌توان به **int** تبدیل کرد.

تبدیل نوع

```
>>> x = -123.45  
>>> s = str(x)  
>>> print(x)  
'-123.45'
```

اعداد را می‌توان به **str** تبدیل کرد.

تبدیل اتوماتیک نوع داده

```
>>> x = 1/2.0  
>>> y = 2*x
```

عملیات میان float و int مقداری float را خواهد داد.

داده های پایتون پویا هستند (Dynamic)

یک متغیر می تواند نوع داده های متفاوت را به خود بگیرد.

```
>>> x = 'abcde'  
>>> x = 1.0  
>>> x = 32
```

در سایر زبان های برنامه نویسی نوع داده ها ثابت است.

۲. ماژول، اسکریپت، و IO

عنوان‌ها:

مود اسکریپت

ماژول

عبارت‌های print و input

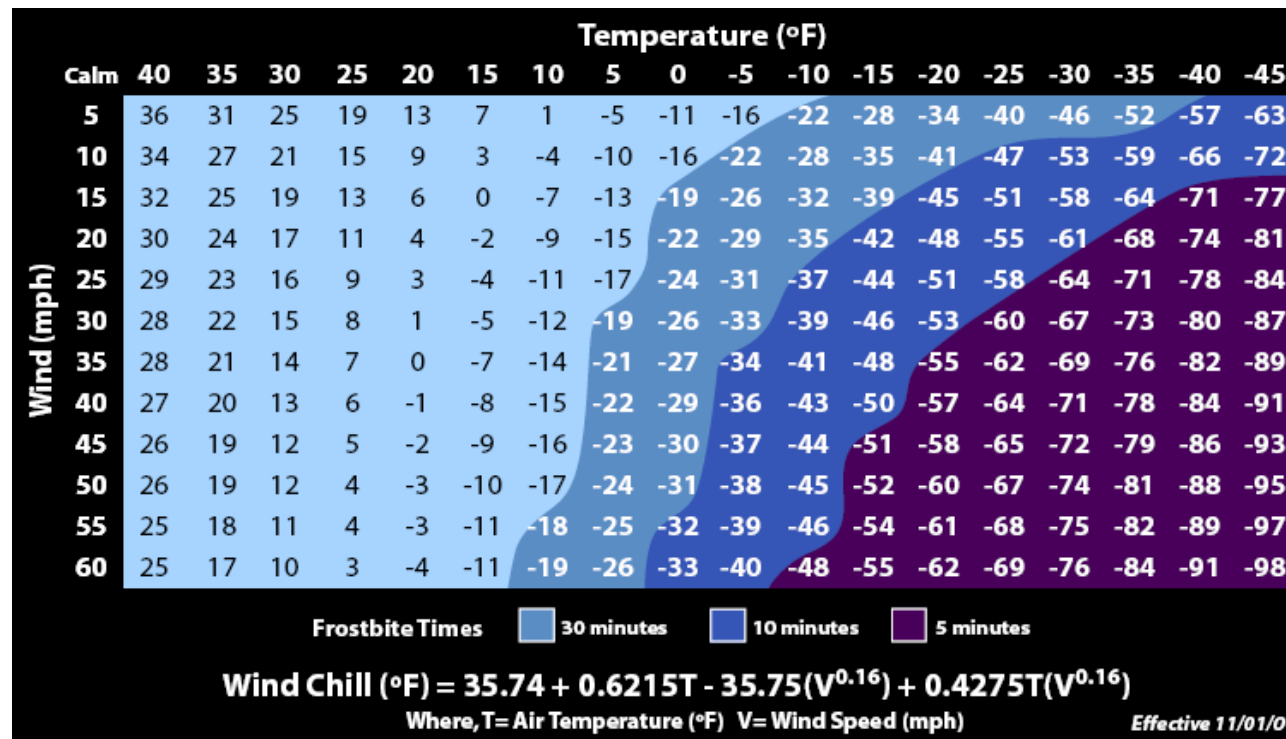
Formattting

نگاهی به وارد کردن ماژول‌ها از دیگر کتابخانه‌ها

محاسبه‌ی سردی هوا wind chill

مقیاسی از سردی هوا با توجه به دما و سرعت باد

$$W_{chill} = (35.74 + 0.6215 * T) + (-35.75 + 0.4275 * T) * W^{.16}$$



محاسبه‌ی سردی هوا wind chill

حالت تعاملی

```
>>> Temp = 32
>>> Wind = 20
>>> A = 35.74
>>> B = .6215
>>> C = -35.75
>>> D = .4275
>>> e = .16
>>> WC = (A+B*Temp) + (C+D*Temp)*Wind**e
>>> print(WC)
19.9855841878
```

نگاهی به `print`

```
>>> print(WC)
```

نتیجه‌ی کد بالا نمایش مقدار `X` روی صفحه می‌باشد.

انگیزه‌ای برای مود اسکریپت

پشت سر هم زیاده!
اگر سرعت باد بیشتر بشه چه باید کرد؟
بهتر نیست یک جا باشن؟

1. مود اسکرپیت

1. به جای مفسر تعاملی می‌توان از مود اسکرپیت استفاده کرد.
2. کدی که نیاز است اجرا شود (اسکرپیت) به صورت فایل (ماژول) وارد خواهد شد.
3. بنابراین ما از پایتون می‌خواهیم یک اسکرپیت را اجرا کند

ماژول چیست؟

ماژول فایل `.py` است که در خود کد پایتون جای داده است.

ماژول WindChill.py

WindChill.py

```
Temp = 32
Wind = 20
A = 35.74
B = .6215
C = -35.74
D = .4275
e = .16
WC = (A+B*Temp) + (C+D*Temp) *Wind**e
print WC
```

اجرای مازول

1. در خط فرمان به آدرس فایل بروید.
2. بنویسید: `python WindChill.py`

آدرس دهی و اجرا در خط فرمان

```
C:\Users\cv\Desktop\TODAY> python WindChill.py  
19.6975841877955
```

نوشتن چند خطی

WindChill.py

```
Temp = 32
Wind = 20
A=35.74;B=.6215;C=-35.74;D=.4275;e=.16
WC = (A+B*Temp)+(C+D*Temp)*Wind**e
Print(WC)
```

گاهی اوقات برای صرفه جویی در مکان از این روش خواهیم رفت.

خوانایی مازول: کامنت

کامنت‌ها با # شروع می‌شوند.

WindChill.py

```
Temp = 32
Wind = 20
# Model Parameters
A=35.74;B=.6215;C=-35.74;D=.4275;e=.16
# Compute and display the windchill
WC = (A+B*Temp)+(C+D*Temp)*Wind**e
Print(WC)
```


کامنت: روش استفاده

کامنت‌ها را می‌توان در همان خط آورد

```
Wind = 20 # wind speed in miles-per-hour
```

هر چیزی پس از # جزو کامنت خواهد بود

کامنت و خوانایی

هر اسکریپت را با یک کامنت توضیحی شروع کنید.
هر متغیر یا ثابت را تعریف کنید.
یک بخش از کد با جزئیات فراوان حتما باید با کامنت شروع شود.

خوانایی مازول: کامنت

WindChill.py

```
"""Computes windchill as a function of  
wind(mph)and temp (Fahrenheit)."""  
Temp = 32  
Wind = 20  
# Model Parameters  
A=35.74;B=.6215;C=-35.74;D=.4275;e=.16  
# Compute and display the windchill  
WC = (A+B*Temp)+(C+D*Temp)*Wind**e  
print(WC)
```

داکسترینگ docstring

- داکسترینگ ها کامنت های چند خطی هستند که با سه نقل قول مشخص می شوند.
- عموماً در اول یک قسمت مهم از کد قرار می گیرند.
- وظیفه ی آنها توصیف عملکرد کد می باشد.

داکسترینگ docstring

- داکسترینگ ها کامنت های چند خطی هستند که با سه نقل قول مشخص می شوند.
- عموماً در اول یک قسمت مهم از کد قرار می گیرند.
- وظیفه ی آنها توصیف عملکرد کد می باشد.

بررسی ورودی‌های متفاوت

WindChill.py

```
"""Computes windchill as a function of  
wind(mph) and temp (Fahrenheit)."""
```

```
Temp = 32
```

```
Wind = 20
```

آیا می‌توان اینجا را تغییر داد؟

```
# Model Parameters
```

```
A=35.74; B=.6215; C=-35.74; D=.4275; e=.16
```

```
# Compute and display the windchill
```

```
WC = (A+B*Temp) + (C+D*Temp)*Wind**e
```

```
print(WC)
```

به صورت دستی؟؟

می‌توان هر قسمت را به صورت دستی تغییر داد!

عبارت `input`

```
input( < string that serves as a prompt > )
```


بررسی ورودی‌های متفاوت

WindChill.py

```
"""Computes windchill as a function of  
wind(mph) and temp (Fahrenheit)."""  
Temp = input('Enter temp (Fahrenheit):')  
Wind = input('Enter wind speed (mph):')  
# Model Parameters  
A=35.74; B=.6215; C=-35.74; D=.4275; e=.16  
# Compute and display the windchill  
WC = (A+B*Temp) + (C+D*Temp)*Wind**e  
print(WC)
```

بررسی ورودی‌های متفاوت

خط فرمان به شکل زیر خواهد بود:

```
> Enter temp (Fahrenheit) :
```

پاسخ ما:

```
> Enter temp (Fahrenheit) : 15
```

Formatting

```
BlahBlah> WindChill
Enter temp (Fahrenheit) : 15
Enter wind speed (mph) : 50
-9.79781580448
```

```
BlahBlah> WindChill
Enter temp (Fahrenheit) : 15
Enter wind speed (mph) : 50
                        Windchill : -10
```

پرینت بدون قالب‌دهی

Script:

```
x = 2./5.  
print (x)  
x = 1./3.  
print (x)  
x = 1234.5678901234  
print (x)
```

Output:

```
0.4  
0.3333333333333333  
1234.56789012
```

پرینت بدون قالب‌دهی

Script:

```
x = 1234  
y = 12345678  
print(x,y)
```

Output:

```
1234 12345678
```

قالبدهی %f

```
x = 1234.123456789  
print ('x = %16.3f' %x)  
print ('x = %16.6f' %x)  
print ('x = %16.9f' %x)
```

```
x =          1234.123  
x =        1234.123457  
x =    1234.123456789
```

قالب دهی %e

```
x = 1234.123456789  
print ('x = %16.3e' %x)  
print ('x = %16.6e' %x)  
print ('x = %16.9e' %x)
```

```
x =          1.234e+03  
x =       1.234123e+03  
x = 1.234123456e+03
```

قالبدهی %d

```
x = 1234
print('x = %4d' %x )
print('x = %7d' %x )
print('x = %10d' %x )
```

```
x = 1234
x =      1234
x =           1234
```


قالبدهی %S

Script:

```
Band = 'The Beatles'  
print ('%s in 1964' % Band)
```

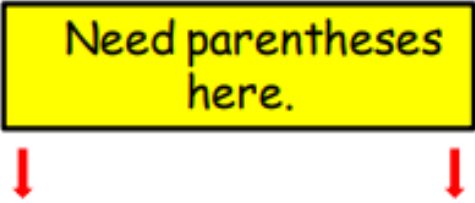
Output:

```
The Beatles in 1964
```

قالب‌دهی بیش از یک عبارت

Script:

```
y1 = 1964
y2 = 1971
Band = 'The Beatles'
print('%s in %4d and %4d' % (Band, y1, y2))
```



Output:

```
The Beatles in 1964 and 1971
```

چرا خوانایی مهم است؟

نکته‌ای دیگر:

WindChill.py

```
# WindChill.py  
# Xavier Zanzibar (xz3)  
# January 1, 1903
```

etc

Name of module

Your name and netid

Date

مثال:

برنامه‌ای بنویسید که مساحت دایره را بگیرد و شعاع دایره را حساب کند.

مثال:

Radius.py

```
A = input('Enter the circle area: ')
r = sqrt(A/3.14)
print(r)
```

خطا می‌گیریم!

```
A = input('Enter the circle area: ')  
r = sqrt(A/3.14)  
print('The radius is %6.3f' % r)
```

```
r = sqrt(A/3.14)  
NameError: name 'sqrt' is not defined
```

sqrt تابع از پیش تعریف شده نمی‌باشد.

نتیجہ‌ی نهایی:

Radius.py

```
from math import sqrt
A = input('Enter the circle area: ')
r = sqrt(A/3.14)
print('The radius is %6.3f' % r)
```

ما تابع sqrt را از ماژول math وارد می‌کنیم.

ایده‌ی کلی وارد کردن ماژول

```
from math import sqrt
```

یک روش بهتر

Radius.py

```
from math import sqrt, pi
A = input('Enter the circle area: ')
r = sqrt(A/pi)
print('The radius is %6.3f' % r)
```

خلاصه

1. متغیرها جایی برای نگهداری مقادیر است.
2. عملگر جایگذاری = مقادیر را به متغیرها می‌دهد.
3. داده‌های عددی را می‌توان با **int** و **float** نمایش داد.
4. داده‌ی نوشته را می‌توان توسط **str** نمایش داد.

نوعی دیگر، بولی **boolean**

True , False

عملگرها

- عملگرهای حسابی
- عملگرهای مقایسه
- عملگرهای منطقی
- عملگرهای بیتی
- عملگرهای تخصیص
- عملگرهای خاص

نماد e برای توان استفاده می‌شود، برای توان های بزرگ بهتر از توان عادی است.

عملگرها

<operand> <operator> <operand>

عملوند

عملگر

عملوند

عملگرها

<operand> <operator> <operand>

عملوند

عملگر

عملوند

عملگرهای حسابی

+

-

*

/

%

//

**

عملگرهای مقایسه ای

>

<

==

!=

>=

<=

عملگرهای منطقی

and
or
not

عملگرهای رشته

+

*

بک اپ

نشل در مقابل سكریپت