

مقدمات و مبانی برنامه نویسی پایتون

آموزش مبانی برنامه نویسی به زبان پایتون PYTHON

۱۴۰۰-۱۴۰۱

2021-2022



امیرحسین خسروی



amirkho.ir

مقدمه

- امیرحسین خسروی هستم کارشناسی مهندسی نرم افزار (تا به اینجای دوره سال ۱۴۰۰-۱۴۰۱). در دوره های متعدد و معتبر بین الملئی در زمینه نرم افزار شرکت کرده ام و دارای چند گواهینامه بین المللی و گواهی مدرس رسمی از سازمان فنی و حرفه ای کشور هستم.
- سابقه تدریس در آموزشگاه های معتبر کشور و آموزش های آزاد در برترین دانشگاه های کشور دارم.
- بسیار خوشحال هستم که این دوره را برای شروع یادگیری انتخاب کرده اید چراکه در مسیر درستی جهت برنامه نویسی قرار گرفته اید.
- از ۱۲ دوازده سالگی برنامه نویسی را شروع کرده ام و حدودا تا الان ۵ سال است که به زبان پایتون برنامه نویسی میکنم و پروژه های متعددی انجام داده ام.
- در این آموزش سعی شده از بروزترین متد های آموزشی و تجربه شخصی ۵ ساله در زمینه برنامه نویسی پایتون استفاده بشود که شما عزیزان بیشترین بهره را ببرید.

• ارتباط مستقیم با مدرس:

➢ وبسایت شخصی amirkho.ir

➢ ایمیل amirpba@gmail.com

توصیه های شخصی در کنار آموزش:

✓ صبر و حوصله

✓ یادگیری زبان تخصصی کامپیوتر

✓ مبانی و مقدمات الگوریتم

✓ عدم جایی در مباحث و حوزه های تخصصی دیگر

فهرست محتوا

تعريف ساده تابع (Function)	<u>9</u>	مقدمه و سخن مدرس	<u>1</u>
كلمات كليدي و رزرو شده پايتون	<u>10</u>	تعريف برنامه (Program)	<u>2</u>
كامنت/نشانه گذاري	<u>11</u>	معرفی برخی از زبان های برنامه نویسی	<u>3</u>
متغير (Variables)	<u>12</u>	معرفی زبان Python	<u>4</u>
انواع داده در پايتون (Data types)	<u>13</u>	IDE چیست؟	<u>5</u>
داده های عددی (Numeric types)	<u>14</u>	معرفی برخی از IDE ها	<u>6</u>
تصخيص چندگانه (Multiple assignment)	<u>15</u>	نصب پايتون و Python IDLE	<u>7</u>
رشته ها (Strings)	<u>16</u>	نوشتن اولین برنامه print Hello,World! - تابع	<u>8</u>

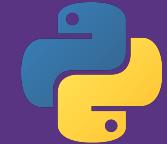
عملگرهای منطقی logical operators	<u>27</u>	قوانين نام گذاری متغیر ها Name assignment rules	<u>17</u>
عملگرهای عضویت membership operators	<u>28</u>	تبدیل نوع داده ها (Data type conversation)	<u>18</u>
عملگرهای شناسایی identify operators	<u>29</u>	تابع input	<u>19</u>
	30	پارامتر های تابع print	<u>20</u>
	31	Escape Sequences	<u>21</u>
	32	قالب بندی خروجی output formatting	<u>22</u>
	33	عملگرها operators	<u>23</u>
	34	عملگرهای حسابی arithmetic operators	<u>24</u>
	35	عملگرهای مقایسه ای comparison operators	<u>25</u>
	36	عملگرهای تخصیصی assignment operators	<u>26</u>

	47		37
	48		38
	49		39
	50		40
	51		41
	52		42
	53		43
	54		44
	55		45
	56		46

برنامه (PROGRAM) چیست؟

- مجموعه‌ای از دستورالعمل‌ها یا الگوریتم (Algorithms) است که رایانه (دسکتاپ و موبایل و...) برای انجام یک کار مشخص آن را اجرا می‌کند. خود رایانه برای انجام کارهایش به برنامه‌ها نیاز دارد و معمولاً هر برنامه را در یک واحد پردازش مرکزی (CPU) اجرا می‌کند.
- معمولاً برنامه‌های رایانه‌ای توسط یک برنامه نویس (Programmer) و تحت یک یا چند زبان برنامه نویسی نوشته می‌شوند.
- به مجموعه‌ای از برنامه‌ها، ماثول (Module) / پکیج (Package) و چهارچوب‌ها (Framework) و داده‌های (Data) مرتبط با آن‌ها نرم افزار (Software) می‌گویند

معرفی برخی از زبان های برنامه نویسی

نام	نماد	حوزه کاربردی	ویژگی ها
Python		توسعه نرم افزار های تحت دسکتاپ - هوش مصنوعی - ریاضیات و آمار - تحلیل داده - توسعه وب و علوم پزشکی و ...	متن باز (open source) - چند سکویی (Cross-Platform) - شع گرا (0.0) - تفسیری (Interpret) - سطح بالا - ساختمان داده پویا (Dynamic D.S)
C		توسعه سیستم عامل - توسعه دسکتاپ - پکیج سازی پایه و مادر برای سیستم عامل ها - برنامه نویسی سخت افزار	تابعی (Functional) - سطح میانی - کامپایلری (Compile)
JS		توسعه وب - توسعه موبایل - افزونه های مرورگر	متن باز (open source) - چند سکویی (Cross-Platform) - شع گرا (0.0) - سطح بالا - تفسیری
Dart		توسعه موبایل - توسعه وب	متن باز (open source) - چند سکویی (Cross-Platform) - شع گرا (0.0) - ساختمان داده پویا (Dynamic D.S)

دوست عزیزم عبارت های تخصصی اشاره شده صرفا جهت آشنایی و فال کردن حس کنچکاوی شما است. بنابرین اصلا نگران نباشید چون به زودی آشنا میشوید

معرفی مختصر زبان برنامه نویسی PYTHON



- در سال ۱۹۸۹ توسط آقای Guido van Rossum شروع به طراحی و در سال ۱۹۹۱ منتشر شد.
- سطح بالا است. بنابرین توسعه و خوانایی و درک آن آسان است.
- کدهای آن به صورت تفسیری (Interpreting) اجرا میشود.
- دارای ماژول‌ها و چارچوب‌های گسترده و متنوع برای حوزه‌های مختلف.
- پسوند‌های آن py - pyc - pyo - pyd
- آخرین نسخه منتشر شده آن تا این لحظه 3.10 است. (۱۴ ژانویه ۲۰۲۲)

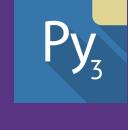


- بکار گرفته شده در :

برنامه را کجا مینویسم؟ آشنایی با IDE ها

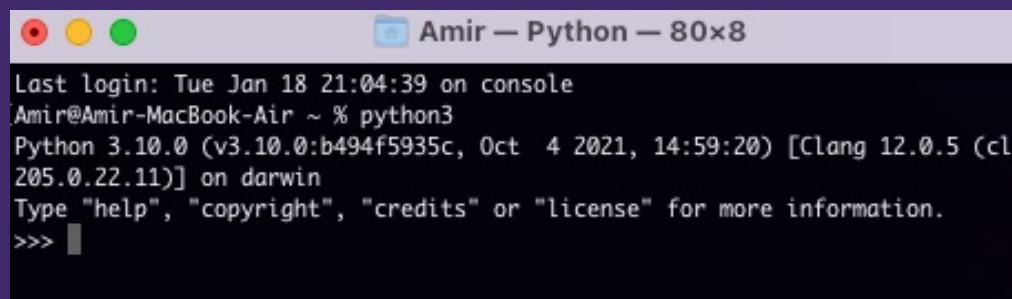
- با مفهوم برنامه و برنامه نویسی آشنا شدیم. حال کجا باید برنامه نویسی کنیم و برنامه خود را بسازیم؟
- برای برنامه نویسی نیاز به نرم افزار های محیط توسعه یکپارچه یا به اختصار IDE (integrated development environment) یا ویرایش کننده گُد Code Editor داریم.
- تفاوت IDE و Code Editor چیست؟ IDE ها محیط های توسعه کاملی هستند جدای از امکان برنامه نویسی مجموعه ای از ابزار های مفید برای برنامه نویسان را در اختیارمان قرار میدهد و درون این محیط مستقیم برنامه ما اجرا میشود . اما ویرایشگر های گُد ما فقط امکان نوشتتن کد های برنامه را میدهد و برای اجرا نیاز به ابزار های جانبی داریم. قاعدهاً ویرایشگر های کد بسیار سبکتر هستند.

معرفی برخی IDE و CODE EDITOR ها

نام	نماد	ویژگی ها
Python IDLE		رایگان - ارائه شده توسط جامعه توسعه دهندگان پایتون - صرفا پشتیبانی از زبان پایتون - بسیار سبک - عدم هوشمندی کافی در تکمیل کد
PyCharm		غیر رایگان - پشتیبانی از python , html , css , js - هوش قوی در تکمیل کد ها
Sublime Text		غیر رایگان - پشتیبانی از زبان های متعدد - بسیار سبک
VS CODE		رایگان - بسیار سبک - قابلیت پشتیبانی از حدود ۵۰ زبان برنامه نویسی - دارای افزونه های متعدد - ترمینال قوی و ...
PyDroid		رایگان - برنامه نویسی پایتون در موبایل های سیستم عامل اندروید

نصب PYTHON

- جهت نوشتن برنامه به زبان پایتون نیاز به نصب آن و همچنین نصب یکی از IDE های معرفی شده داریم.
- برای نصب مطمئن بهتر است پایتون را از سایت رسمی آن به آدرس [python.org](https://www.python.org) مراجعه کرده و از آنجا اقدام به دانلود و نصب کنیم.
- در سیستم عامل لینوکس (Linux) و مکینتاش (Mac os) پایتون به طور پیشفرض نصب است .
- جهت اطمینان از نصب بودن میبایست در یکی از محیط های CMD/Terminal با توجه به نوع سیستم عامل دستور `python3` وارد کنیم. خروجی میبایست در تمام سیستم عامل ها مطابق/مشابه تصویر زیر باشد:



```
Last login: Tue Jan 18 21:04:39 on console
Amir@Amir-MacBook-Air ~ % python3
Python 3.10.0 (v3.10.0:b494f5935c, Oct  4 2021, 14:59:20) [Clang 12.0.5 (cl
205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

PYTHON IDLE

- پس از نصب پایتون از مرجع رسمی در واقع دو نرم افزار IDLE و Python Launcher در کامپیوتر ما نصب میشود.
- همچنین به طور خودکار در ریشه (root) نیز نصب میشود و از طریق محیط های CMD/Terminal با دستور python/python3 در دسترس قرار خواهد گرفت.
- تفاوت اجرای پایتون از طریق CMD/Terminal با IDLE یا بقیه محیط های توسعه در چیست؟
- به طور خیلی ساده و خلاصه تفاوت در ذخیره شدن گذهای برنامه است. اگر محیط های کنسول CMD/Terminal گدمان را بنویسیم فقط در لحظه و همانجا میتوان نتیجه/خروجی خود را ببینیم. اما اگر در محیط های توسعه این کار را انجام دهیم میتوانیم پروژه خود را در قالب فایل های py. ذخیره کنیم و هر زمان که نیاز داشته باشیم به آن دسترسی پیدا کنیم و ویرایش انجام دهیم.

اولین برنامه

- پس از نصب و تست ابزارهای لازم جهت برنامه نویسی پایتون وقت آن است که اولین برنامه خود را بنویسیم.

```
[>>> print("Hello,World!")  
Hello,World!  
>>>
```

(1) در محیط CMD/Terminal پایتون را فراخوانی کنید

(2) این دستور را مطابق تصویر وارد کنید کنید: print("Hello,World!")

(3) نتیجه/خروجی برابر Hello,World! است.

(4) ما با استفاده تابعی (Function) به نام print این کار را انجام دادیم!

```
>>> print(Hello,World!  
  File "<stdin>", line 1  
    print(Hello,World!  
          ^  
SyntaxError: invalid syntax  
>>> |
```

(5) حالا سعی کنید نام خودتان را نمایش دهید.

(6) عدم رعایت فاصله ها و پرانتز و بقیه علامت ها برنامه به مشکل میخورد و خطای syntax error میدهد!

(7) این اولین برنامه ما بود که همچنین میتوانید در یکی از محیط های توسعه آن را بنویسید که بتوانید ذخیره کنید و ویرایش!

تعریف ساده تابع (FUNCTION)

- جهت درک مفهومی تابع عملکرد یک ماشین بستنی ساز را در نظر بگیرید:
 - (1) شیر را داخل آن میریزیم.



- print نیز یکی از توابعی است که در بسیاری از زبان های برنامه نویسی به طور پیشفرض وجود دارد (Built in Functions) و برای نمایش یک مقدار (اطلاعات/Data) در خروجی کنسول کاربرد دارد

FunctionName (Data)



- نحوه استفاده از تابع در زبان پایتون به این شکل است:

- تابع خروج برای خارج شدن از برنامه فعلی یا خروج کلی از پایتون: exit()

PYTHON KEYWORDS

- همه زبان های برنامه نویسی شامل یکسری کلمات هستند به طور ویژه برای برنامه نویسی به آن زبان تعریف و رزرو شده اند که دارای ویژگی های زیر میباشند:
 - (1) نمیتوانند به عنوان نام متغیر (Variable) استفاده شوند.
 - (2) نمیتوانند به عنوان نام توابع (Function) استفاده شوند.
 - (3) به طور خلاصه و ساده اینکه نمیتوانیم از آنها استفاده شخصی در نام گذاری ها داشته باشیم!

```
help> keywords

Here is a list of the Python keywords. Enter any keyword to get more help.

False      class      from       or
None       continue   global     pass
True       def        if         raise
and        del        import    return
as         elif       in         try
assert    else       is         while
async     except    lambda   with
await     finally   nonlocal  yield
break
```

COMMENT

- کامنت گذاری مناسب نشانه گذاشتن در کدهای برنامه است و بیشترین کاربرد آن برای درک کدها در توسعه های بعدی است.
- کامنت را صرفا برنامه نویس مینویسد و تنها خودش به آن دسترسی دارد.
- مفسر پایتون کامنت ها را به عنوان دستور یا... در نظر نمیگیرد و اجرا نمیکند.
- برای کامنت گذاشتن تک خطی در برنامه از علامت (#) استفاده میکنیم و بعد از آن پیام/متن خود را مینویسم.
- برای کامنت گذاشتن چند خطی به قبل و بعد از پیام/متن خود ۳ بار از علامت (') استفاده میکنیم.
- ✓ جهت درک برنامه زیر را در محیط IDE بنویسید و ذخیره و اجرا کنید:

```
#First Comment

print("Hello Amir !")

"""
Multiline Comment
"""

Ln: 11 Col: 0
```

همانطور که مشاهده کردید مفسر پایتون کامنت ها را در نظر نگرفت و تنها تابع print اجرا شد

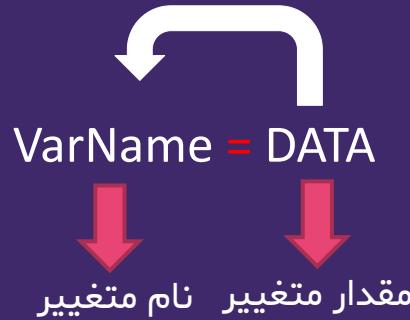
- متغیر (Variables) یک فضای ذخیره سازی موقت اطلاعات در حافظه RAM است.
- از آنجا که اطلاعات متغیر در حافظه RAM ذخیره میشود با بسته شدن یا خاموش شدن سیستم اطلاعات متغیرها حذف میشود.
- متغیرها نوع (type) های مختلفی دارند و هر کدام برای ذخیره سازی یک نوع داده کاربرد دارد.
- برای درک ساده انواع متغیر ظرف های غذا را در نظر بگیرید که هر کدام مناسب نوعی از غذا هستند برای مثال: دیس مخصوص برنج است و کاسه مخصوص آش و غذا های آبکی!
- از آنجا که زبان پایتون از نظر ساختمان داده پویا است ما نوع متغیرها را به طور پیشفرض تعریف نمیکنیم و خود مفسر پایتون نوع آن را از روی اطلاعات موجود در آن تشخیص میدهد بر خلاف بسیاری از زبان های برنامه نویسی دیگر که میباشد نوع متغیر را نیز تعریف کنیم!

VARIABLES

Part 2

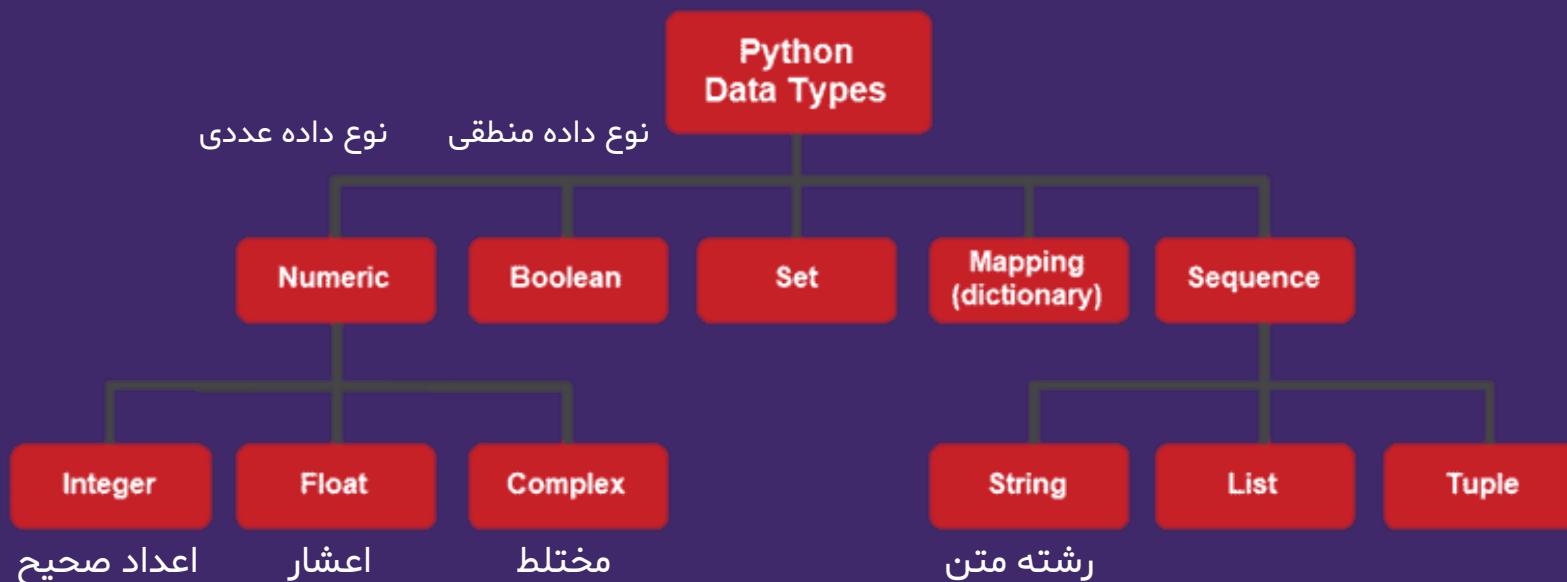
Data Types

مقدار بعد از علامت مساوی به متغیر اختصاص داده میشود

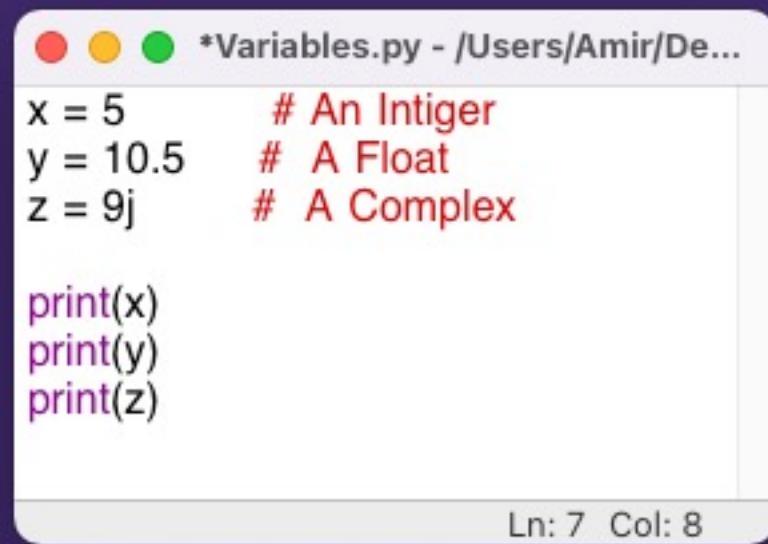


- برای تعریف متغیر به شکل مقابل عمل میکنیم:

- تصویر زیر مربوط به تمامی انواع داده (Data Types) در زبان پایتون است:



- چند مثال ساده برای تعریف و درک متغیر:
- در کد زیر سه متغیر با نام های x,y,z تعریف کردیم و آن ها را مقدار دهی کردیم.
- نام متغیر را میتوان به عنوان ورودی تابع print در نظر گرفت تا آن را چاپ/نمایش دهد.
- تمامی متغیر ها از نوع Numeric هستند.



```
*Variables.py - /Users/Amir/De...
x = 5      # An Intiger
y = 10.5    # A Float
z = 9j      # A Complex

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

Ln: 7 Col: 8
```

VARIABLES

Part 4

Multiple Assignment

```
*Variables.py - /Users/Amir/Desktop/Variabl...
x , y , z = 10 , 2.5 , 8j # Multiple Assignment
print(x)
print(y)
print(x+z)

Ln: 1 Col: 25
```

- در یک خط نیز میتوان چند متغیر تعریف کرد و آن ها را مقدار دهی کرد.

```
*Variables.py - /Users/Amir/Desktop...
# Multiple Assignment
a = b = c = 100
print(a)
print(c)
print(a,b)

>>> 100
>>> 100
>>> 100 100

Ln: 11 Col: 11
```

- همچنین میتوانیم خود متغیر ها را به یکدیگر اختصاص دهیم.

- همچنین میتوان به تابع print چند آرگومان (ورودی از نوع مقدار) داد که با علامت , از یکدیگر جدا میشوند.

- داده های رشته ای Strings (متنی/غیر محاسبه) حتما باید درون دو علامت " " یا ' ' قرار گیرند و از این حیث با داده های عددی متفاوت هستند.
- در صورت چند خطی بودن رشته آن را بین علامت های " " " " three quotes قرار میدهیم.

The screenshot shows a code editor window with the following content:

```
*Variables.py - /Users/Ami...
# String types
name = "Amir"
age = 21
print(name , age)
>>> Amir 21
Ln: 8 Col: 0
```

- رشته های به خودی خود آرایه هستند. یعنی اینکه از به هم پیوستن های کاراکتر به وجود میاید و عملیات های متعددی مثل جستجو و... روی آن انجام داد
- ✓ مقدار متنی داخل متغیر را خارج از علامت " " یا ' ' قرار دهید و برنامه را اجرا کنید.
- مقادیر رشته ای نمیتوانند خارج از استاندارد تعریف شوند.

- موارد زیر قوانین نامگذاری متغیرها ، توابع و کلاس ها هستند که باید آنها را رعایت کنیم:
 - نام ها به حروف کوچک و بزرگ حساس هستند (Case-sensitive) . متغیر age و Age دو متغیر مجزا هستند.
 - نام ها می توانند شامل حروف کوچک و بزرگ، همچنین اعداد و underscore باشند. [A-z , 0-9 , _]
 - نام ها نمی توانند شامل کarakترهای خاص مانند : !@#\$%^&* باشند.
 - نام ها نمی توانند با عدد شروع شده و یا دارای فاصله باشند.
 - نام ها نمی توانند اسمی رزرو شده در پایتون باشند، مانند : int
 - نام متغیر ها نمی تواند به underscore ختم شود.

- تبدیل نوع داده ها:
- توابعی برای تبدیل نوع داده وجود دارند که تنها برخی از آن ها به شرح جدول زیر هستند:

Function	Description
int(x)	مقدار ورودی x را به عدد صحیح تبدیل میکند. مقادیر رشته ای و اعشاری را قبول میکند.
float(x)	مقدار ورودی x را به عدد اعشاری تبدیل میکند. مقادیر رشته ای و عدد صحیح را قبول میکند.
str(x)	مقدار ورودی x را به رشته متنی تبدیل میکند.
hex(x)	مقدار ورودی x را به رشته ای بر مبنای ۱۶ تبدیل میکند. تنها عدد صحیح را قبول میکند.
oct(x)	مقدار ورودی x را به رشته ای بر مبنای ۱۶ تبدیل میکند. تنها عدد صحیح را قبول میکند.

The screenshot shows a code editor window titled "Variables.py - /Users/Amir/D...". It contains three examples of data type conversion:

```
# Data type conversation
pi = 3.1456
print(pi)

pi = int(pi)
print(pi)

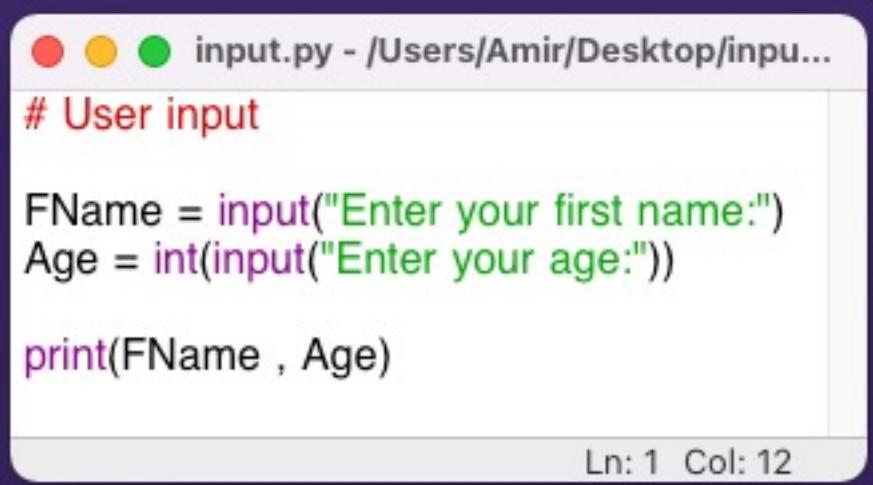
score = 20
print(float(score))
```

At the bottom of the editor, the status bar displays "Ln: 11 Col: 19".

- ✓ مقدار متغیر `pi` به نوع داده عدد صحیح تبدیل کنید.
- ✓ مقدار متغیر `score` را به نوع داده عدد صحیح تبدیل کنید.
- ✓ مقادیر پیشفرض و مقادیر نهایی تبدیل شده را نمایش دهید.
- همچنین میتوان خروجی نهایی توابع را در یک متغیر ذخیره کرد
- همچنین میتوانیم بعضی توابع را در برخی توابع دیگر به کار گیریم. مثلاً تابع `float` در `print`

INPUT FUNCTION

- از تابع `input` برای گرفتن ورودی اطلاعات از کاربر در محیط کنسول CMD/Terminal استفاده میشود.
- خروجی تابع `input` داخل متغیر ذخیره میشود.
- به طور پیشفرض میتواند یک ورودی رشته ای `string` داشته باشد که یک پیام برای کاربر و گرفتن ورودی است.
- نوع داده این تابع به طور پیشفرض رشته ای است. پس اگر قصد گرفتن عدد از کاربر را دارید حتما میبایست نوع آن را تبدیل به یکی از انواع `numeric` کنید. در غیر اینصورت اگر نیاز به پردازش و محاسبه آن عدد داشته باشیم با مشکل مواجه خواهیم شد.



```
# User input

FName = input("Enter your first name:")
Age = int(input("Enter your age:"))

print(FName , Age)
```

Ln: 1 Col: 12

PRINT PARAMETERS

- تابع print دارای چند پارامتر (ورودی اجباری یا غیر اجباری) برای نحوه نمایش میباشد

Parameters	Description
end	You can end a print statement with any character/string using this parameter.
sep	The separator between the arguments to print() function in Python is space by default (softspace feature) , which can be modified and can be made to any character, integer or string as per our choice

A screenshot of a Python code editor window titled "*print.py - /Users/Amir/Desktop/Python Course/print.py...". The code demonstrates the use of the print() function with parameters:

```
# Print Parameters

MSG = "Hello,World"
print(MSG , end = ' ! ')    # end parameters
>>> Hello,World !
print('10','09','2021', sep= ' - ')      # sep parameter
>>> Hello,World ! 10 - 09 - 2021
print('09','02','1380', sep= ' @ ', end='\\n')
>>>10 @ 09 @ 2021
```

The code shows three print statements. The first uses the 'end' parameter to append ' ! ' to the end of the string. The second uses the 'sep' parameter to separate the three arguments with a dash (' - ') instead of a space. The third uses both 'sep' and 'end' parameters to format the output as '10 @ 09 @ 2021' on a new line.

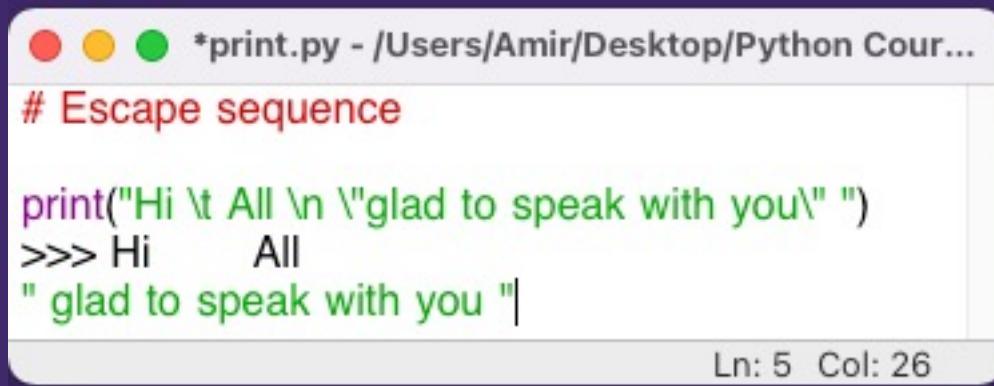
مقدار پارامتر end به طور پیشفرض \n است که باعث میشود خروجی در یک خط جدید چاپ بشود.

همچنین میتوان از این علامت در بین رشته ها نیز جهت ایجاد فاصله با خط جدید استفاده کرد.

\n یکی از ا نوع Escape sequence است.

ESCAPE SEQUENCE

- می توانید از یک یا چند کاراکتر ویژه در رشته برای قالب بندی یا اجرای یک دستور استفاده کنید. به این کاراکتر ها Escape sequence می گویند. یک دنباله Escape در پایتون با بک اسلش (\) شروع می شود.



A screenshot of a Python code editor window titled "print.py - /Users/Amir/Desktop/Python Cour...". The code contains the following:

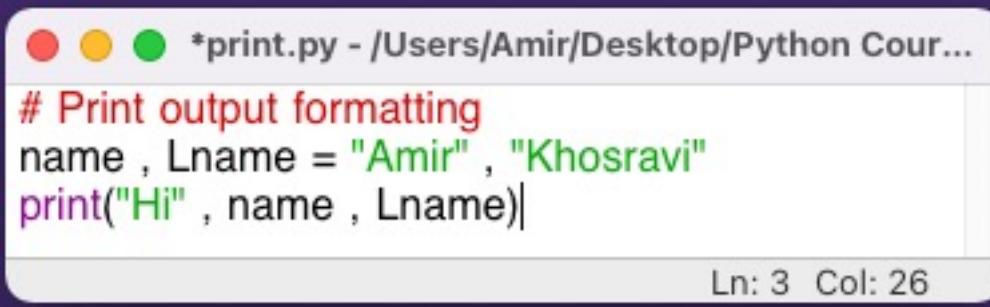
```
# Escape sequence
print("Hi \t All \n \"glad to speak with you\" ")
>>> Hi      All
" glad to speak with you "|
```

The code demonstrates the use of escape sequences: `\t` for a horizontal tab, `\n` for a new line, and `\"` for a double quote character.

Escape sequence	Description
\n	Breaks the string into a new line
\t	Adds a horizontal tab
\\	Prints a backslash
\'	Prints a single quote
\"	Prints a double quote
\a	makes a sound like a bell

OUTPUT FORMATTING

- راههای مختلفی برای ارائه خروجی وجود دارد، که برای بالا بردن خوانایی داده ها در محیط کنسول استفاده میکنیم.
- یکی از روش های قالب بندی خروجی متن در تابع print است که با علامت f قبل از محتوا تعریف میشود و اجازه میده مقادیر متغیر های دیگر را بین همان رشته از طریق صدا زدن نام متغیر مابین علامت {} نمایش دهیم.



```
*print.py - /Users/Amir/Desktop/Python Cour...
# Print output formatting
name , Lname = "Amir" , "Khosravi"
print("Hi" , name , Lname)|
```

Ln: 3 Col: 26



```
print.py - /Users/Amir/Desktop/...
# Print output formatting
name = "Amir"
Lname = "Khosravi"
print(f"Hi {name} {Lname}")
>>>Hi Amir Khosravi
```

Ln: 6 Col: 0

OPERATORS

Part 1

- عملگرها (operators) سازه هایی هستند که می توانند مقدار عملوندها (operands) (مقادیری که روی آن ها عملیات انجام میشود) را دستکاری کنند.
- عبارت $9 = 3 + 6$ را در نظر بگیرید. در اینجا 3 و 6 عملوند و + عملگر نامیده می شوند.
- زبان پایتون از انواع عملگرهای زیر پشتیبانی می کند:

Arithmetic Operators	عملگرهای حسابی
Comparison (Relational) Operators	عملگرهای مقایسه (رابطه ای)
Assignment Operators	عملگرهای تخصیصی
Logical Operators	عملگرهای منطقی
Bitwise Operators	عملگرهای بیتی
Membership Operators	عملگرهای عضویت
Identity Operators	عملگرهای هویت

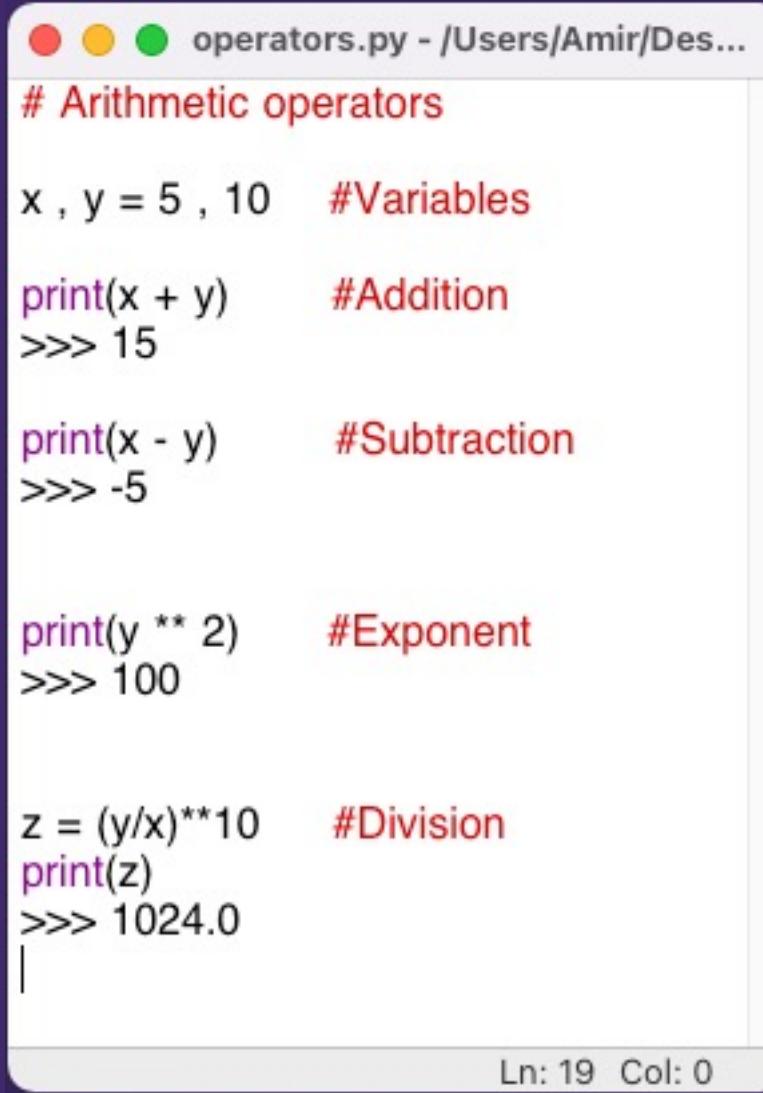
OPERATORS

Part 2

Arithmetic Operators

- عملگرهایی که عملیات ریاضی/حسابی انجام میدهند.
- دو متغیر x و y را با مقادیر 10 و 20 در نظر بگیرید:

Operators	Description	Example
+ Addition	Adds values on either side of the operator.	$x + y = 30$
- Subtraction	Subtracts right hand operand from left hand operand.	$x - y = -10$
* Multiplication	Multiplies values on either side of the operator	$x * y = 200$
/ Division	Divides left hand operand by right hand operand	$x / y = 0.5$
% Modulus	Divides left hand operand by right hand operand and returns remainder	$y \% x = 0$
** Exponent	Performs exponential (power) calculation on operators	$x ** y = 10 \text{ to the power } 20$
// Floor Division	The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity)	$7 // 2 = 3$ $-9 // 2 = -5$ $11//3 = 3$



A screenshot of a Python code editor window titled "operators.py - /Users/Amir/Desktop". The code demonstrates various arithmetic operations:

```
# Arithmetic operators

x , y = 5 , 10      #Variables
print(x + y)         #Addition
>>> 15

print(x - y)         #Subtraction
>>> -5

print(y ** 2)         #Exponent
>>> 100

z = (y/x)**10        #Division
print(z)
>>> 1024.0

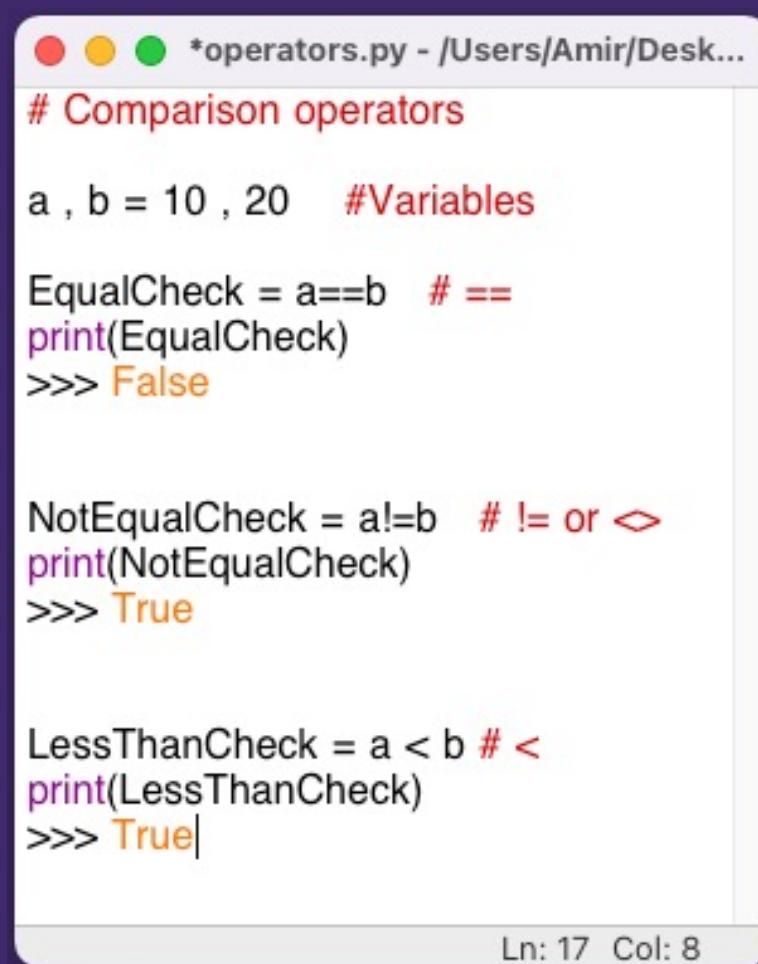
Ln: 19 Col: 0
```

- چند مثال از عملگر های حسابی:
- درون تابع print میتوانیم به شکل مستقیم محاسبات انجام دهیم.
- محاسبات را میتوان درون متغیر ذخیره کرد.
- خروجی محاسباتی که شامل بیش از یک عملگر هستند حتما نوع داده آن float خواهد بود.

- این عملگرها مقادیر دو طرف آنها را مقایسه کرده و رابطه بین آنها را تعیین می کنند. به آنها عملگرهای رابطه ای نیز گفته می شود.

`a , b = 10 , 20`

Operators	Description	Example
<code>==</code>	If the values of two operands are equal, then the condition becomes true.	<code>(a == b)</code> is not true >>> False
<code>!=</code>	If values of two operands are not equal, then condition becomes true.	<code>(a != b)</code> is true >>> True
<code><></code>	If values of two operands are not equal, then condition becomes true.	<code>(a <> b)</code> is true. This is similar to != operator
<code>></code>	If the value of left operand is greater than the value of right operand, then condition becomes true.	<code>(a > b)</code> is not true >>> False
<code><</code>	If the value of left operand is less than the value of right operand, then condition becomes true.	<code>(a < b)</code> is true >>> True
<code>>=</code>	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	<code>(a >= b)</code> is not true >>> False
<code><=</code>	If the value of left operand is less than or equal to the value of right operand, then condition becomes true	<code>(a <= b)</code> is true >>> True



The screenshot shows a Python code editor window with the following code:

```
*operators.py - /Users/Amir/Desktop
# Comparison operators

a , b = 10 , 20    #Variables

EqualCheck = a==b   # ==
print(EqualCheck)
>>> False

NotEqualCheck = a!=b  # != or <>
print(NotEqualCheck)
>>> True

LessThanCheck = a < b # <
print(LessThanCheck)
>>> True|
```

Ln: 17 Col: 8

- چند مثال از عملگر های مقایسه ای رابطه ای:

OPERATORS

Part 6

Assignment Operators

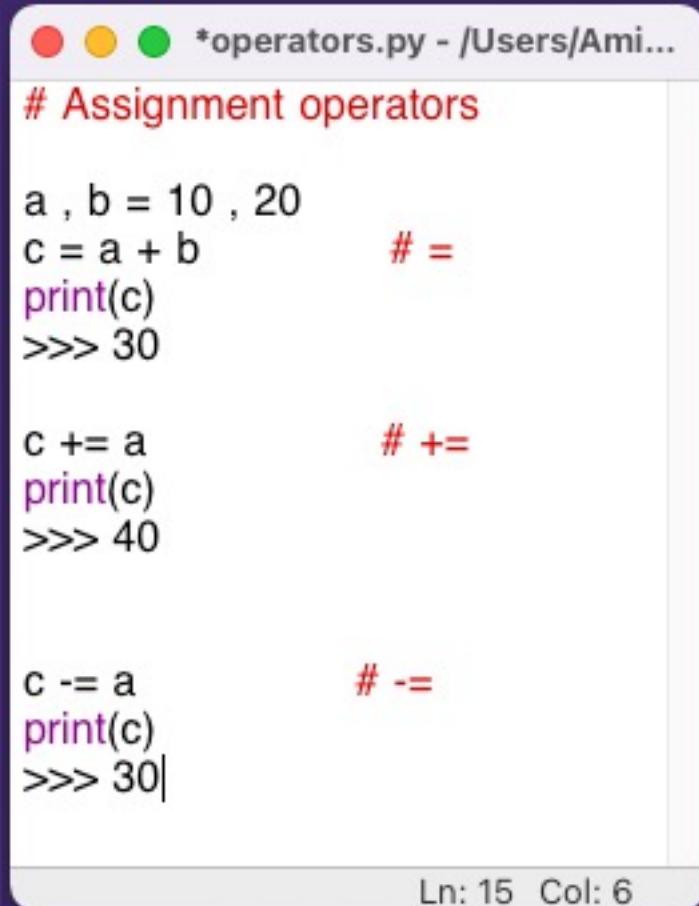
- به این عملگر ها تخصیصی یا واگذاری نیز گفته میشود

Operators	Description	Example
=	Assigns values from right side operands to left side operand	$c = a + b$ assigns value of $a + b$ into c
$+=$ ADD AND	It adds right operand to the left operand and assign the result to left operand	$c += a$ is equivalent to $c = c + a$
$-=$ Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	$c -= a$ is equivalent to $c = c - a$
$*=$ Multiply AND	It multiplies right operand with the left operand and assign the result to left operand.	$c *= a$ is equivalent to $c = c * a$
$/=$ Divide AND	It divides left operand with the right operand and assign the result to left operand	$(a < b)$ is true $>>>$ True
$\%=$ Modulus AND	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	$c \%= a$ is equivalent to $c = c \% a$
$**=$ Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	$c **= a$ is equivalent to $c = c ** a$

OPERATORS

Part 7

Assignment Operators



A screenshot of a Python code editor window titled '*operators.py - /Users/Ami...'. The code demonstrates various assignment operators:

```
# Assignment operators

a , b = 10 , 20
c = a + b      # =
print(c)
>>> 30

c += a         # +=
print(c)
>>> 40

c -= a         # -=
print(c)
>>> 30|
```

The status bar at the bottom shows 'Ln: 15 Col: 6'.

- چند مثال از عملگر های تخصیصی:

- در صورت تکرار نام متغیر و استفاده مجدد از آن همیشه مقدار آن برابر آخرین مقدار تعریف شده است

OPERATORS

Part 8

Logical Operators

عملگر های منطقی •

a , b = True , False

Operators	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

A screenshot of a Python code editor window titled "operators.py - /Us...". The code is as follows:

```
# Logical operators
|
a , b = True , False
c = a and b # AND
print(c)
>>> False
c = a or b # OR
print(c)
>>> True
```

The window shows syntax highlighting for keywords like "True", "False", "and", "or", and "not". It also highlights comments starting with "#". The output section shows the results of the print statements.

- عملگر های عضویت
- عضویت یک دنباله (String , List , Tuples) در دنباله دیگر بررسی میکند.

Operators	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a True if x is a member of sequence y.
not in	If any of the two operands are non-zero then condition becomes true.	x not in y, here not in results in a True if x is not a member of sequence y.

```

● ● ● *operators.py - /...
# Membership operators
x = "Hello"
y = "H"
c = y in x # in
print(c)
>>> True

c = y not in x # not in
print(c)
>>> False|

```

Ln: 10 Col: 9

در بخش های جلوتر به مباحث انواع داده دنباله دار (sequences) پرداخته میشود

OPERATORS

Part 10

Identify Operators

- عملگر های شناسایی
- عملگرهای هویت/شناسایی، مکان های حافظه دو شئ (شامل متغیر نیز میشود) را با هم مقایسه می کند.
- برای مشخص کردن مکان شئ/متغیر در حافظه RAM از تابع `id` میتوان استفاده کرد.
- گاهها میتوان به جای عملگر مقایسه ای `==` یا `!=` استفاده شود و با توجه به اینکه مستقیم به RAM مراجعه میکند سرعت بیشتری دارد.

The screenshot shows a code editor window with a Python script named `operators.py`. The code is as follows:

```
# Identify operators
x = 10
y = 10
c = y is x # is
print(c)
>>> True
|
print(id(x)) # id()
>>> 4481090064
```

The status bar at the bottom indicates "Ln: 8 Col: 0".

Operators	Description	Example
<code>is</code>	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	<code>x is y</code> , here <code>is</code> results in True if <code>id(x)</code> equals <code>id(y)</code>
<code>is not</code>	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	<code>x is not y</code> , here <code>is not</code> results in True if <code>id(x)</code> is not equal to <code>id(y)</code>