منبع مورد استفاده:

https://www.w۳schools.com/python/default.asp

نرم افزار مورد استفاده:

Visual Stu یک محیط یکپارچه توسعه نرم افزار که به منظور طراحی و ایجاد اپلیکیشن ها و برنامه های
و مبتنی بر فضای ابری طراحی شده است. ابزارهای ساده و در عین حال کاربردی این برنامه از نظر عملکرد
Visual Stu هستند اما دارای ویژگی های به روزتر و جامع تری می باشند و این برنامه مجموعه ای از ابزارهای
در محیط کاربری مدرن و کارآمدی ارائه می کند. یکی از ویژگی های قابل توجه این برنامه ، قابلیت های
می باشد که فرآیند تست ، ساخت و حتی گسترش انواع مختلف نرم افزارها را تسهیل می نماید. با استفاده از
کاربر می تواند چندین طراحی مختلف را ایجاد نموده و آن ها در پروژه مورد خود ذخیره نماید و به طور سریع
بت به پیکربندی آن ها اقدام کند.

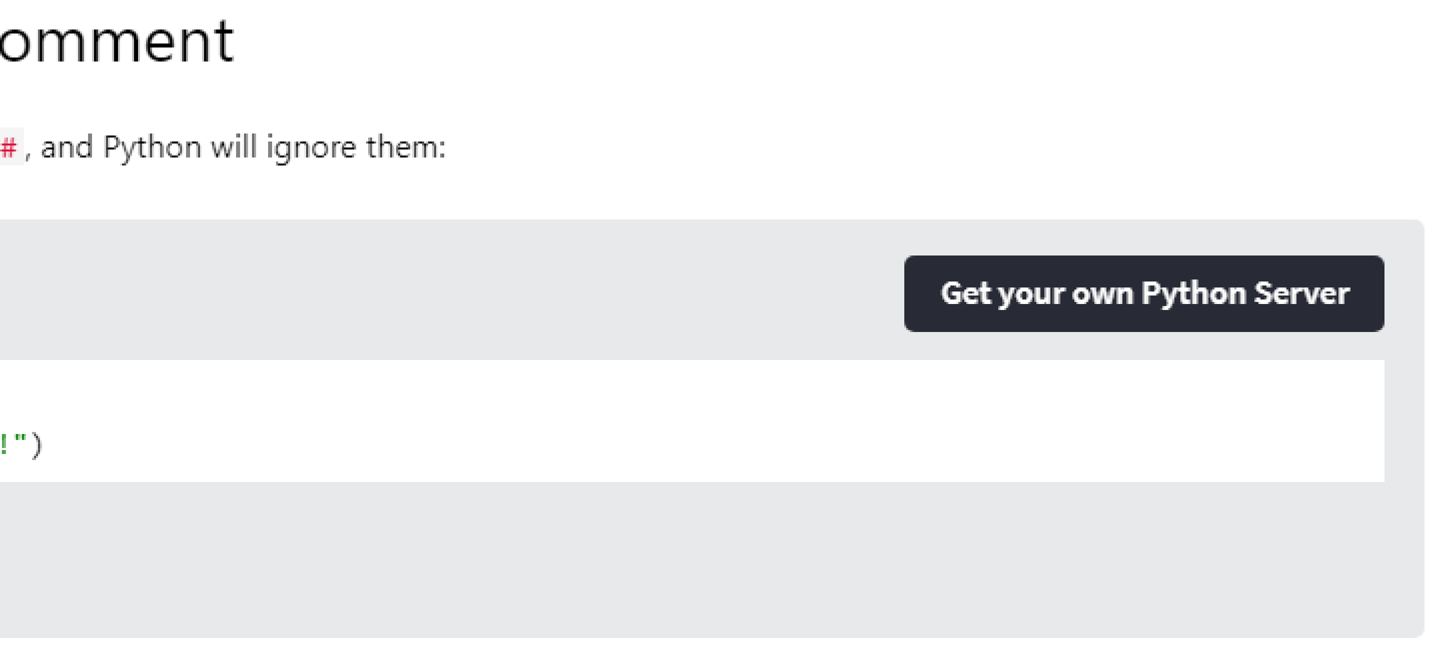
گی های برنامه Visual Studio Code می توان به امکان استفاده از اسنیپت های نمونه و همچنین امکان ایجاد و ذخیره فرگمنت ها یا قطعات کد
کاربر اشاره کرد. این برنامه قابلیت ایجاد خروجی پروژه به صورت فایل نوشتاری را دارد و از آن مهم تر از زبان های برنامه نویسی مختلف همچون
، کلوژر ، Perl ، PHP ، Lua ، JSON ، HTML ، F# ، پایتون ، SQL ، ویژوال بیسیک ، XML و برخی دیگر از زبان ها و همچنین از توسعه در
Node.js و ASP.NET پشتیبانی می کند.

https://dl۲.soft۹۸.ir/soft/u-v/Visual.Studio.Code.۱,۱۰۵,۱.x۶٤.rar?۱۷٦۱۸۱۲۳۷٥

ویژگی‌های افزونه‌های پایتون برای VSCode	نام افزونه
تکمیل کد، دیباگینگ، لینتینگ، فرمت‌بندی کد.	Python Extension
تکمیل کد پیشرفته، بررسی نوع، ناوبری کد.	Pylance
اجرای نوت‌بوک‌های Jupyter **، پشتیبانی از** Markdown **، نمایش داده‌های تعاملی.**	Jupyter

Python Docstring Generator	تولید خودکار docstring برای توابع و کلاس‌ها.
GitLens	مدیریت Git، نمایش اطلاعات commit، مقایسه فایل‌ها.
Visual Studio IntelliCode	پیشنهادهای هوشمند کدنویسی با استفاده از هوش مصنوعی.
Prettier – Code formatter	فرمت‌بندی و زیباسازی خودکار کد.
Python Test Explorer	مدیریت و اجرای تست‌های پایتونی با پشتیبانی از فریم‌ورک‌های مختلف.
Bracket Pair Colorizer	رنگی کردن پرانتزها و براکت‌ها برای بهبود خوانایی کد.
Django	تکمیل کد برای Django، پشتیبانی از دستورات manage.py، ناوبری در پروژه‌های Django.

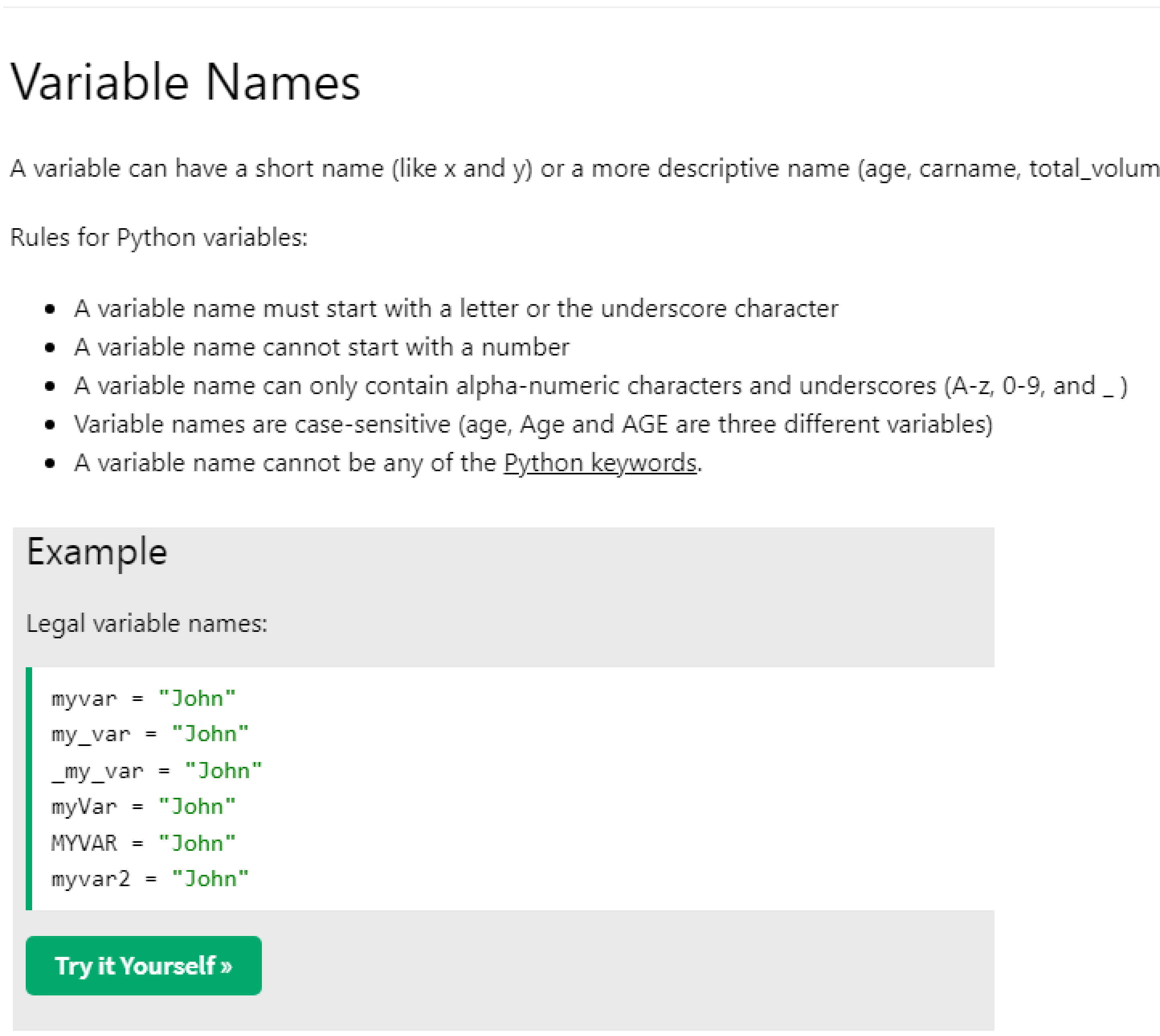
نحوه ایجاد یک کامنت


omment

#, and Python will ignore them:

```
!")
```

متغییرها



Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volum

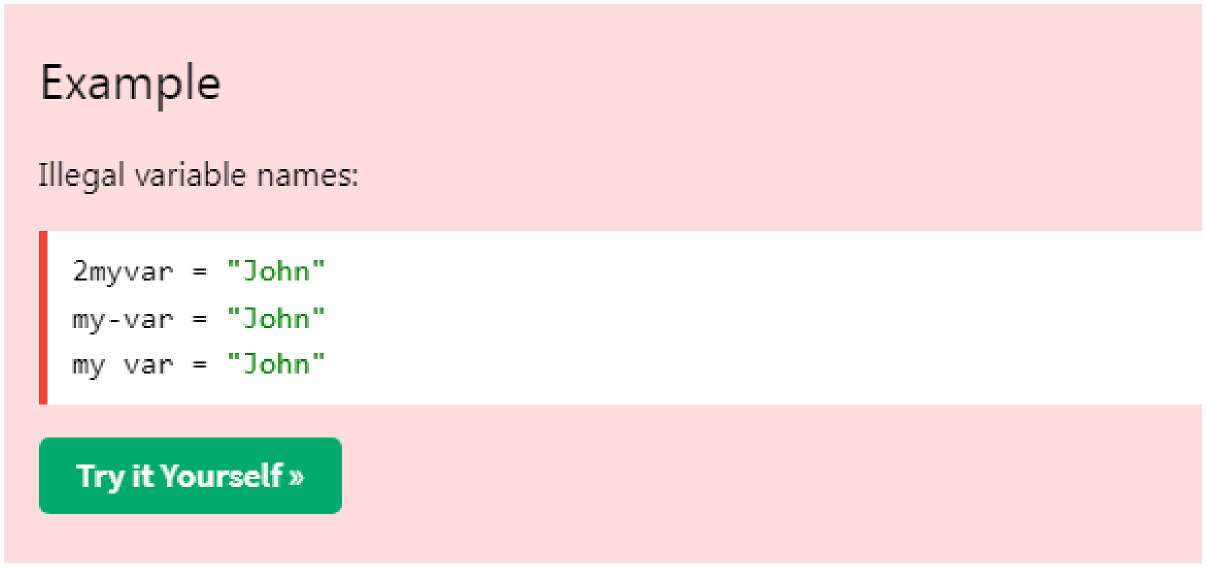
Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- A variable name cannot be any of the Python keywords.

Example

Legal variable names:

```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

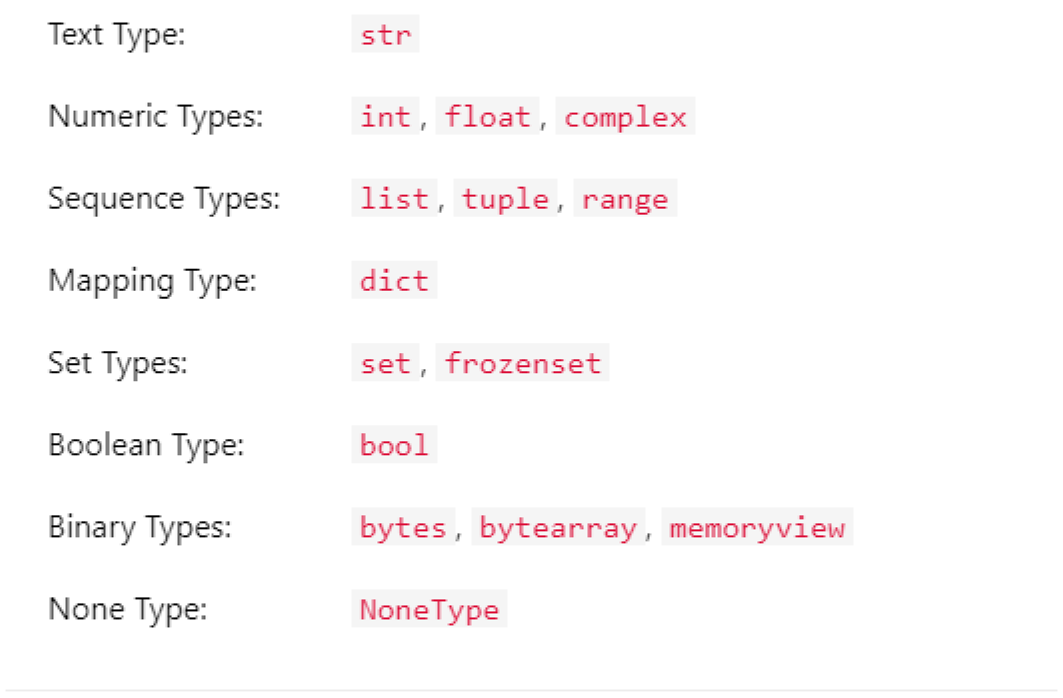


Example

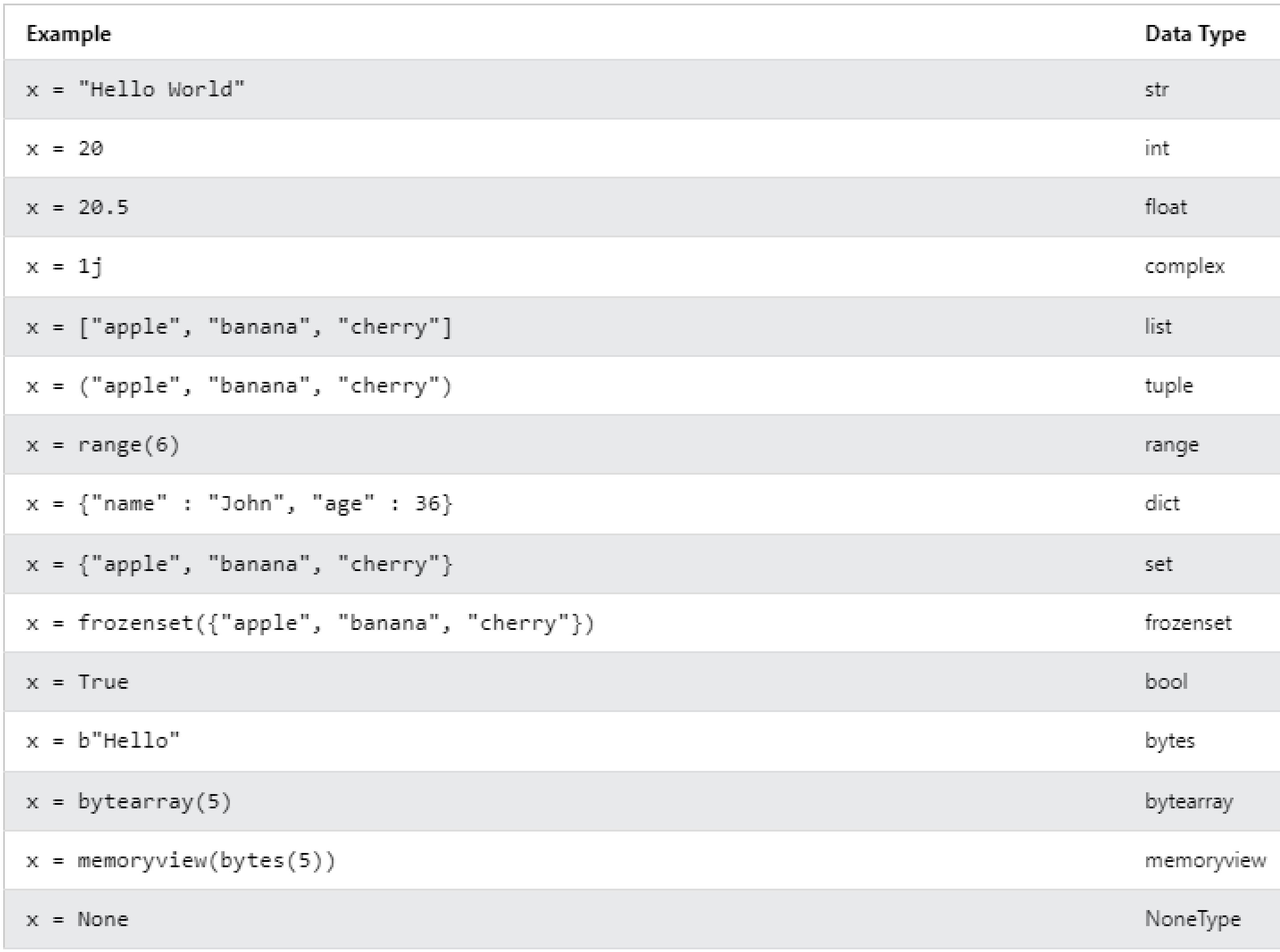
Illegal variable names:

```
2myvar = "John"
my-var = "John"
my var = "John"
```

انواع دیتا تایپ ها در پایتون

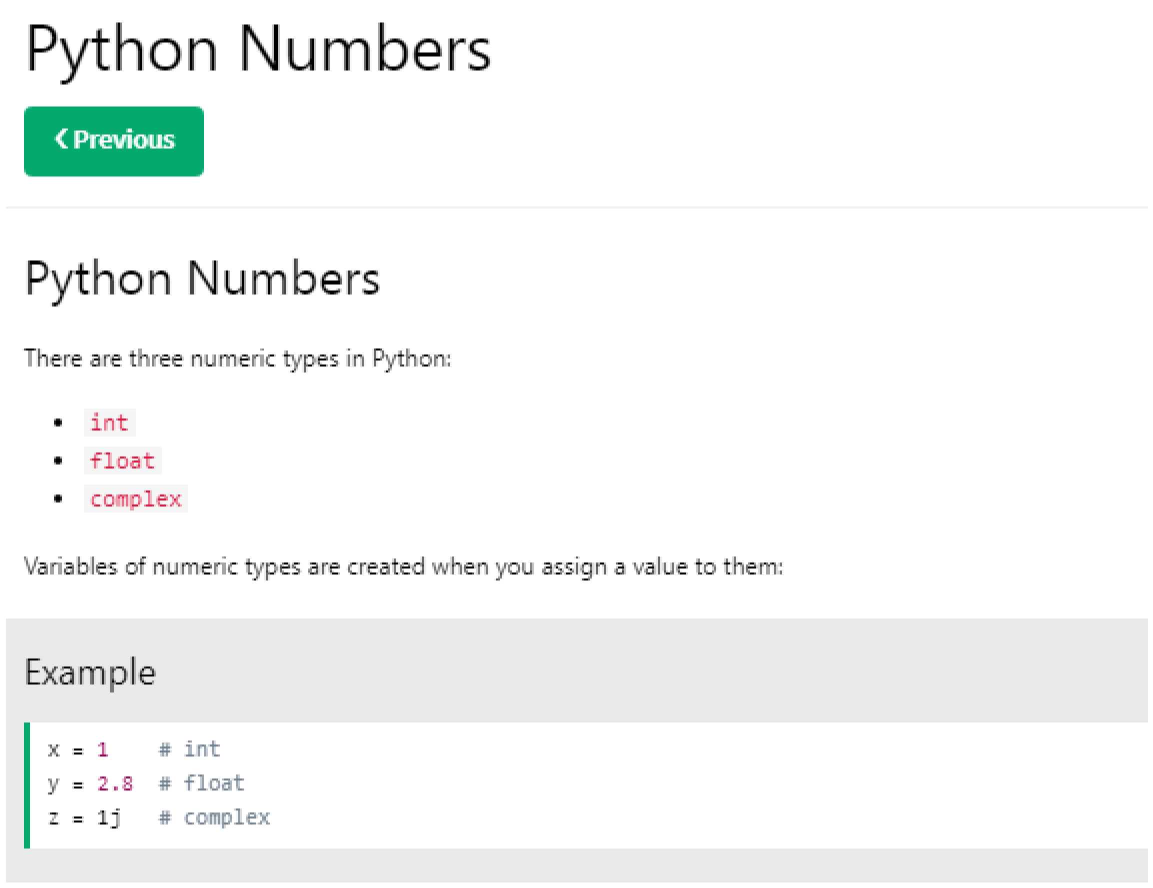


Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`
None Type:	`NoneType`



Example	Data Type
`x = "Hello World"`	str
`x = 20`	int
`x = 20.5`	float
`x = 1j`	complex
`x = ["apple", "banana", "cherry"]`	list
`x = ("apple", "banana", "cherry")`	tuple
`x = range(6)`	range
`x = {"name" : "John", "age" : 36}`	dict
`x = {"apple", "banana", "cherry"}`	set
`x = frozenset({"apple", "banana", "cherry"})`	frozenset
`x = True`	bool
`x = b"Hello"`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview
`x = None`	NoneType

اعداد در پایتون



Python Numbers

‹ Previous

Python Numbers

There are three numeric types in Python:

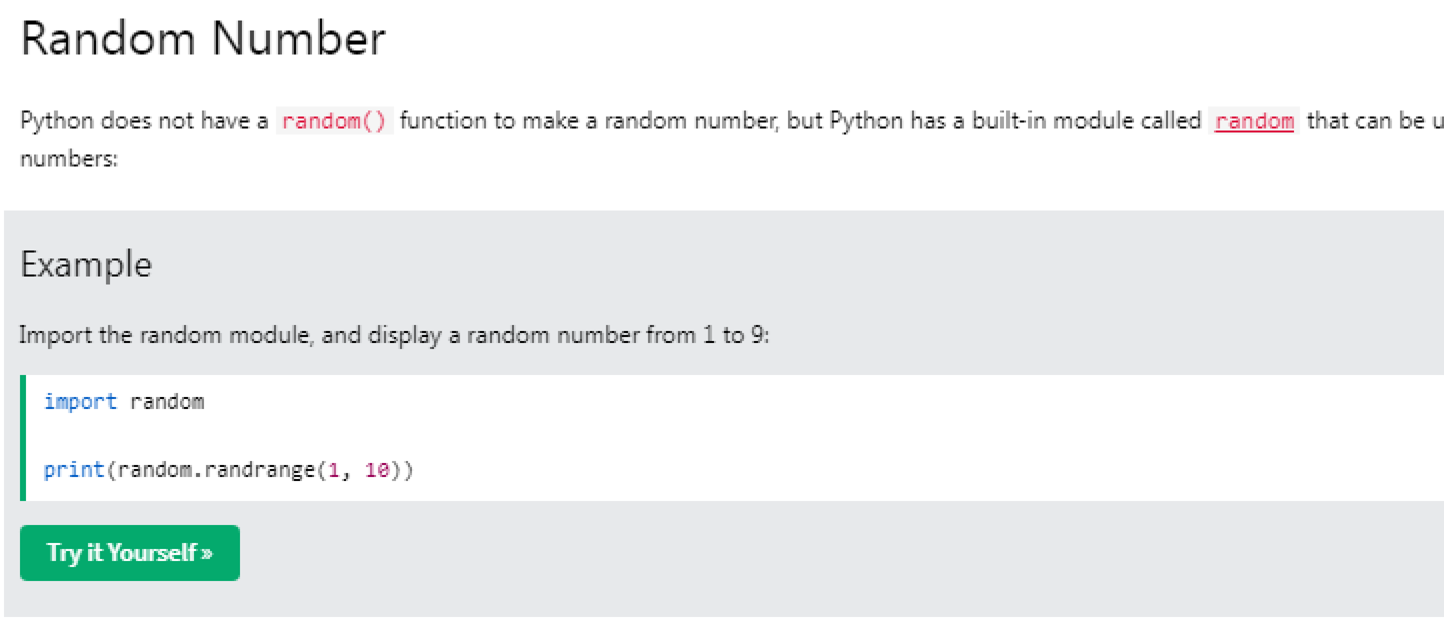
- `int`
- `float`
- `complex`

Variables of numeric types are created when you assign a value to them:

Example

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

اعداد تصادفی در پایتون



Random Number

Python does not have a `random()` function to make a random number, but Python has a built-in module called `random` that can be u
numbers:

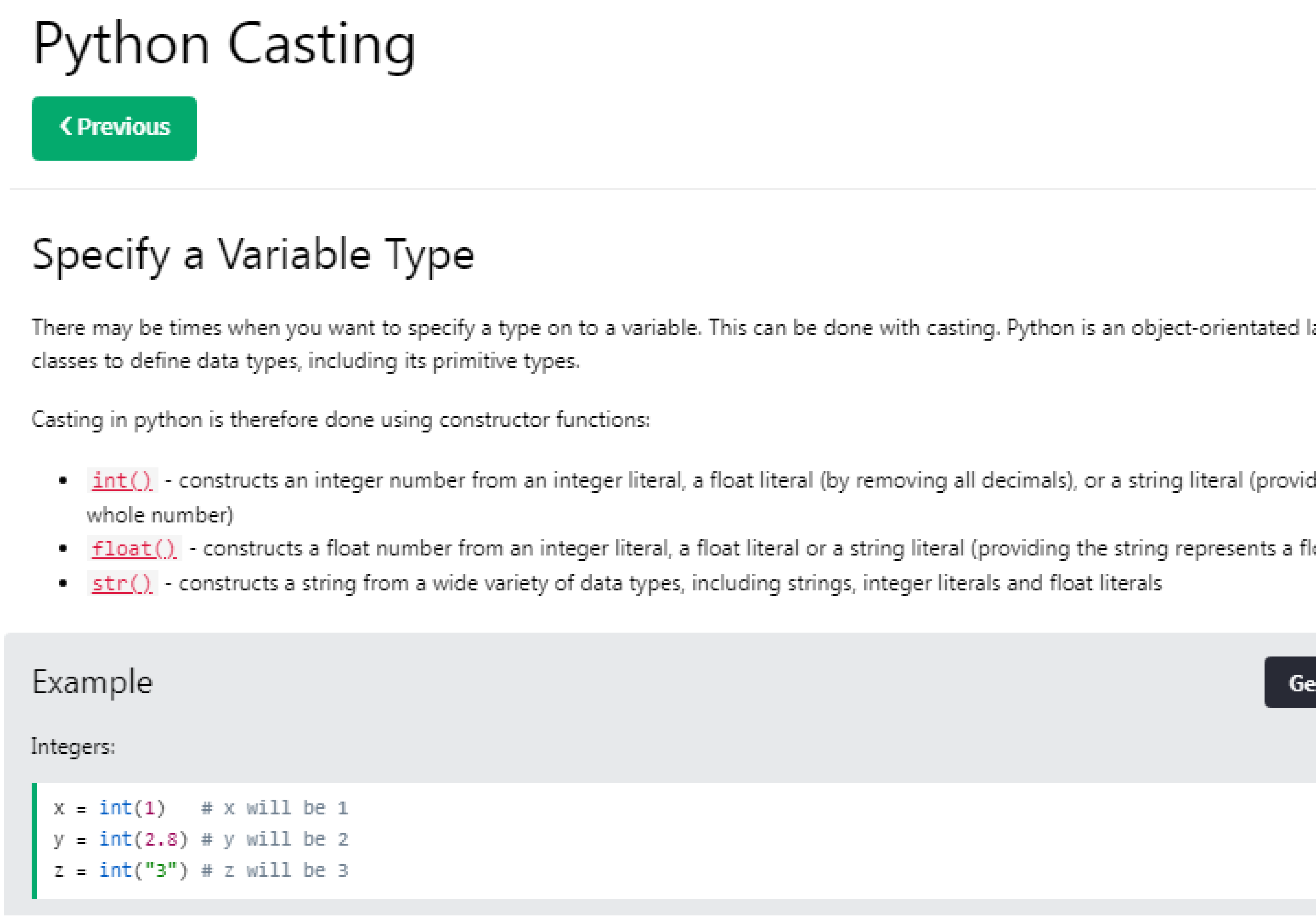
Example

Import the random module, and display a random number from 1 to 9:

```
import random

print(random.randrange(1, 10))
```

Try it Yourself »



Python Casting

Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated la
classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

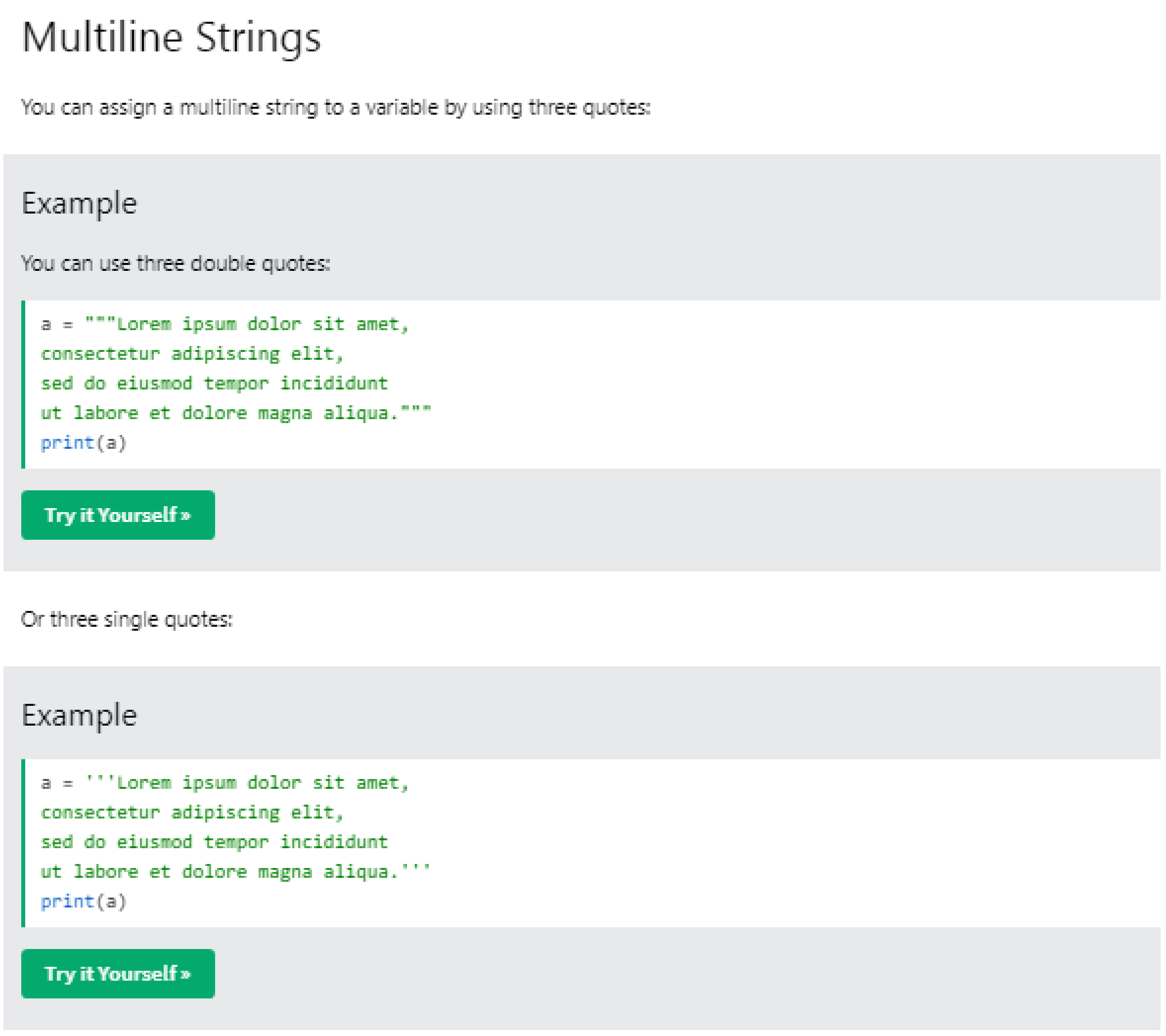
- `int()` - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (provid
whole number)
- `float()` - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a flo
- `str()` - constructs a string from a wide variety of data types, including strings, integer literals and float literals

Example

Integers:

```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

رشته ها در پایتون



Multiline Strings

You can assign a multiline string to a variable by using three quotes:

Example

You can use three double quotes:

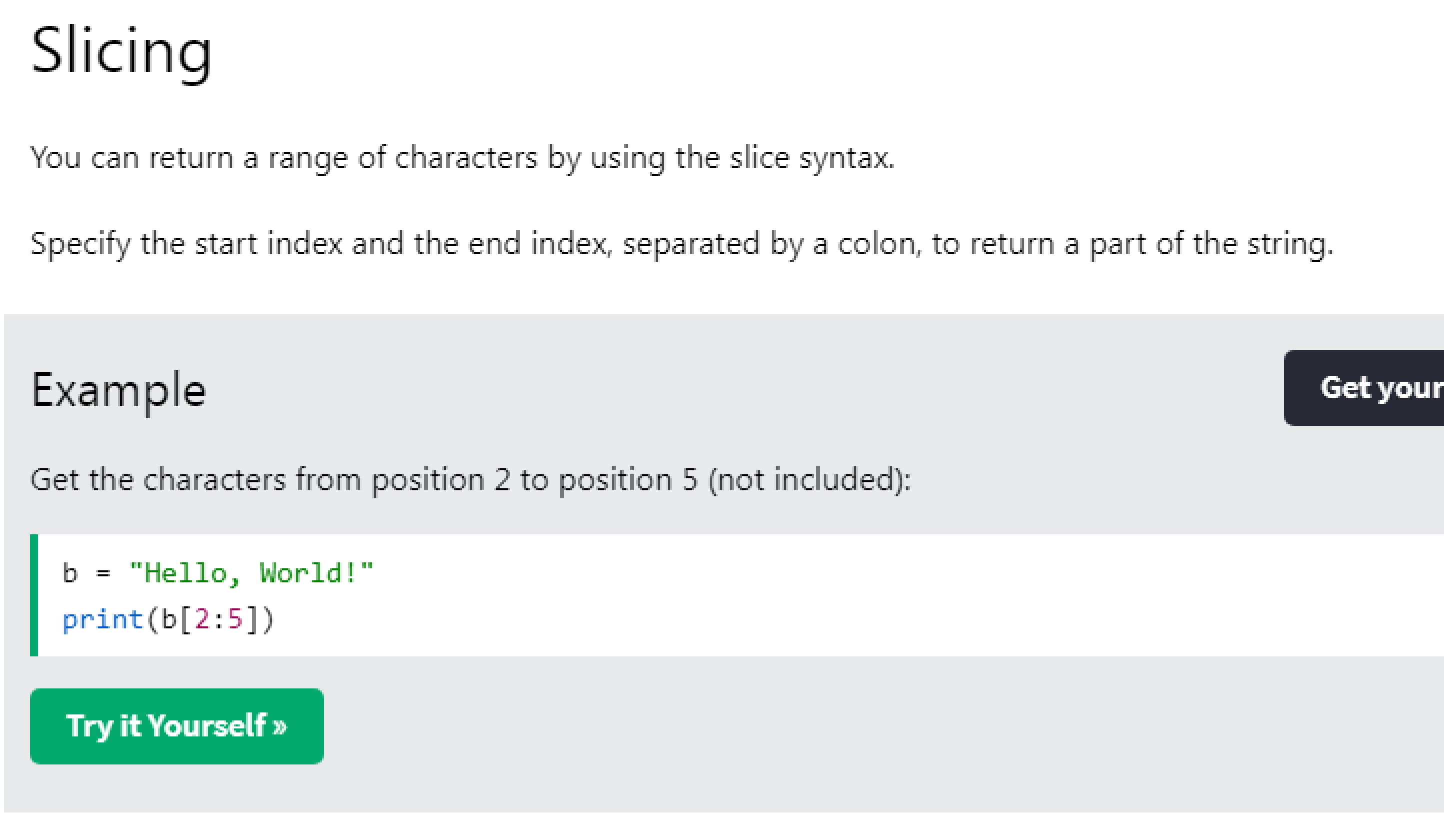
```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

Or three single quotes:

Example

```
a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
```

جدا کردن بخشی از رشته در پایتون



Slicing

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

Example

Get the characters from position 2 to position 5 (not included):

```python
b = "Hello, World!"
print(b[2:5])
```

code	answer
`b = "Hello, World!"` `print(b[:5])`	`Hello`
`b = "Hello, World!"` `print(b[2:])`	`llo, World!`
`b = "Hello, World!"` `print(b[-5:-2])`	`orl`
`b = "Hello, World!"` `print(b[2:5])`	`llo`

تغییر رشته

`a = "Hello, World!"`
`print(a.upper())`
	`a = "Hello, World!"`
	`print(a.replace("H", "J"))`

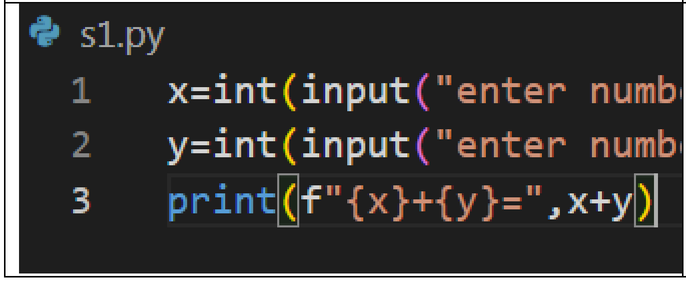
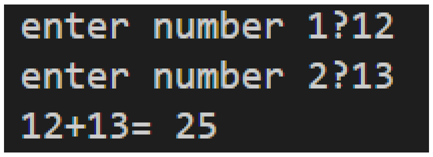
`a = "Hello, World!"`
`print(a.lower())`	

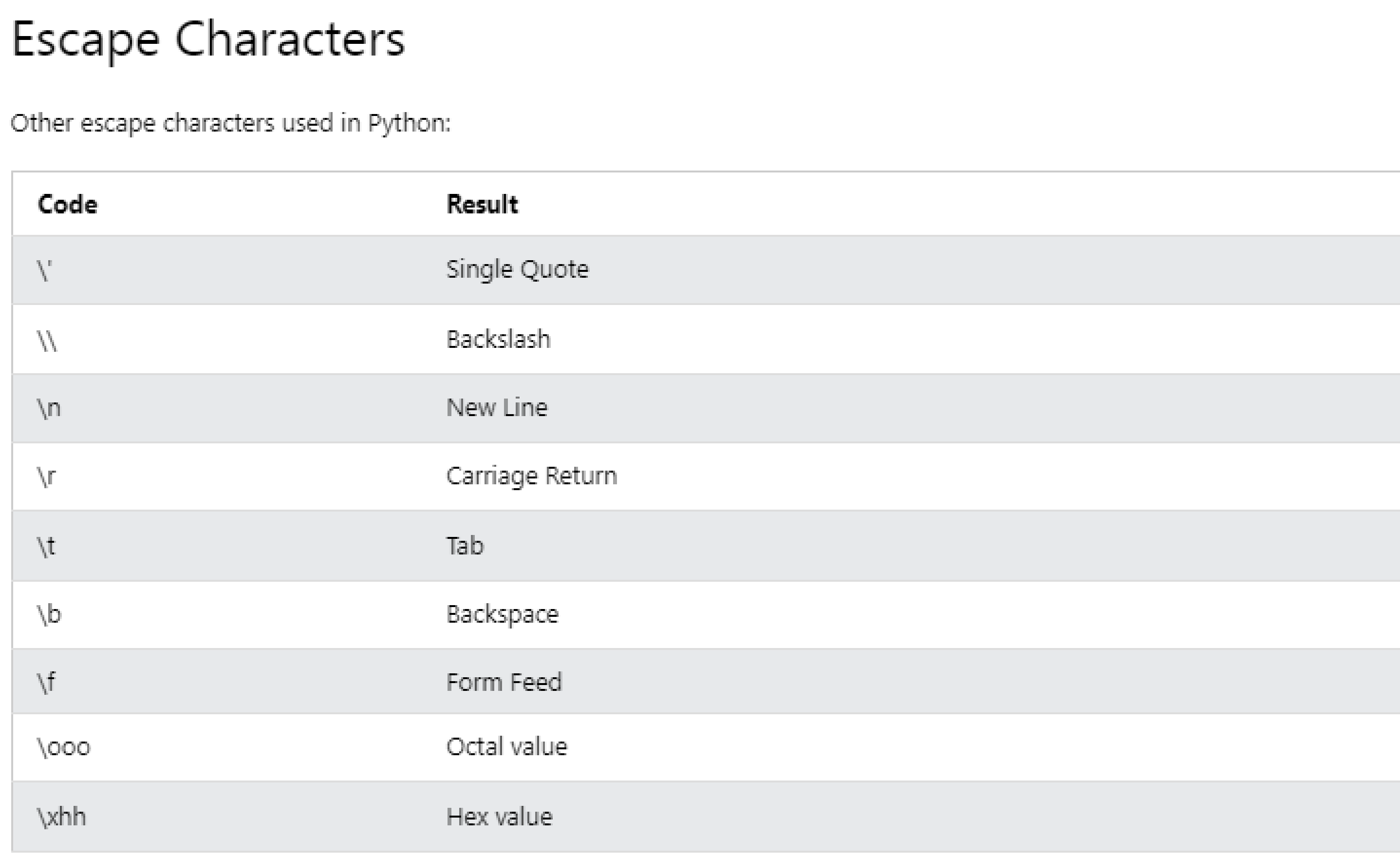
`a = " Hello, World! "`
`print(a.strip()) # returns "Hello,`
	`a = "Hello, World!"`
	`print(a.split(",")) # returns ['Hello',`

فرمت رشته

`age = 36`
`txt = f"My name is John, I am {age}"`
`print(txt)`
	`price = 59`
	`txt = f"The price is {price:.2f}`
	`print(txt)`

`txt = f"The price is {20 * 59} dollars"`
`print(txt)`
	answer

 s1.py
`x=int(input("enter numb`
`y=int(input("enter numb`
`print(f"{x}+{y}=",x+y)`
	 `enter number 1?12`
	`enter number 2?13`
	`12+13= 25`



Escape Characters

Other escape characters used in Python:

Code	Result
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

متدهای رشته

Method	**Description**
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string

encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isascii()	Returns True if all characters in the string are ascii characters
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier

islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Joins the elements of an iterable to the end of the string
ljust()	Returns a left justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value

strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case
zfill()	Fills the string with a specified number of ۰ values at the beginning
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string

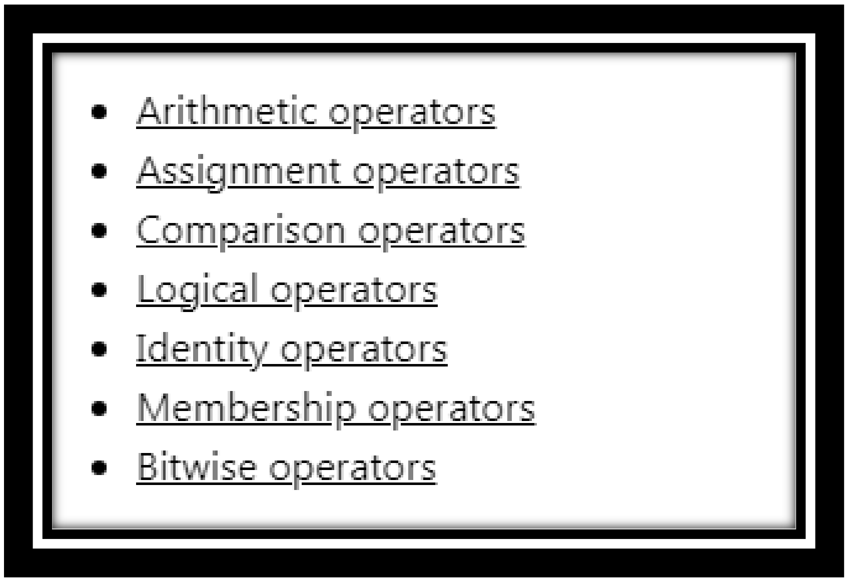
متغییر بولین در پایتون

`print(10 > 9)`
`print(10 == 9)`
`print(10 < 9)`
	`a = 200`
	`b = 33`
	
	`if b > a:`
	`  print("b is greater than a")`
	`else:`
	`  print("b is not greater than a")`

true	**false**

`bool("abc")`
`bool(123)`
`bool(["apple", "cherry", "banana"])`
	`bool(False)`
	`bool(None)`
	`bool(0)`
	`bool("")`
	`bool(())`
	`bool([])`
	`bool({})`

انواع اپراتورها در پایتون



- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	**Name**	**Example**
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Assignment Operators

Assignment operators are used to assign values to variables:

Operator	Example	Same As	Try it
=	x = 5	x = 5	
+=	x += 3	x = x + 3	
-=	x -= 3	x = x - 3	
*=	x *= 3	x = x * 3	
/=	x /= 3	x = x / 3	
%=	x %= 3	x = x % 3	
//=	x //= 3	x = x // 3	
**=	x **= 3	x = x ** 3	
&=	x &= 3	x = x & 3	
|=	x |= 3	x = x | 3	
^=	x ^= 3	x = x ^ 3	
>>=	x >>= 3	x = x >> 3	
<<=	x <<= 3	x = x << 3	

:=	print(x := 3)	x = 3
		print(x)	

Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example	Try it
==	Equal	x == y	
!=	Not equal	x != y	
>	Greater than	x > y	
<	Less than	x < y	
>=	Greater than or equal to	x >= y	
<=	Less than or equal to	x <= y	

Python Logical Operators

❮ Previous N

Logical Operators

Logical operators are used to combine conditional statements:

Operator	Description	Example	Try it
and	Returns True if both statements are true	x < 5 and x < 10	T
or	Returns True if one of the statements is true	x < 5 or x < 4	T
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)	T

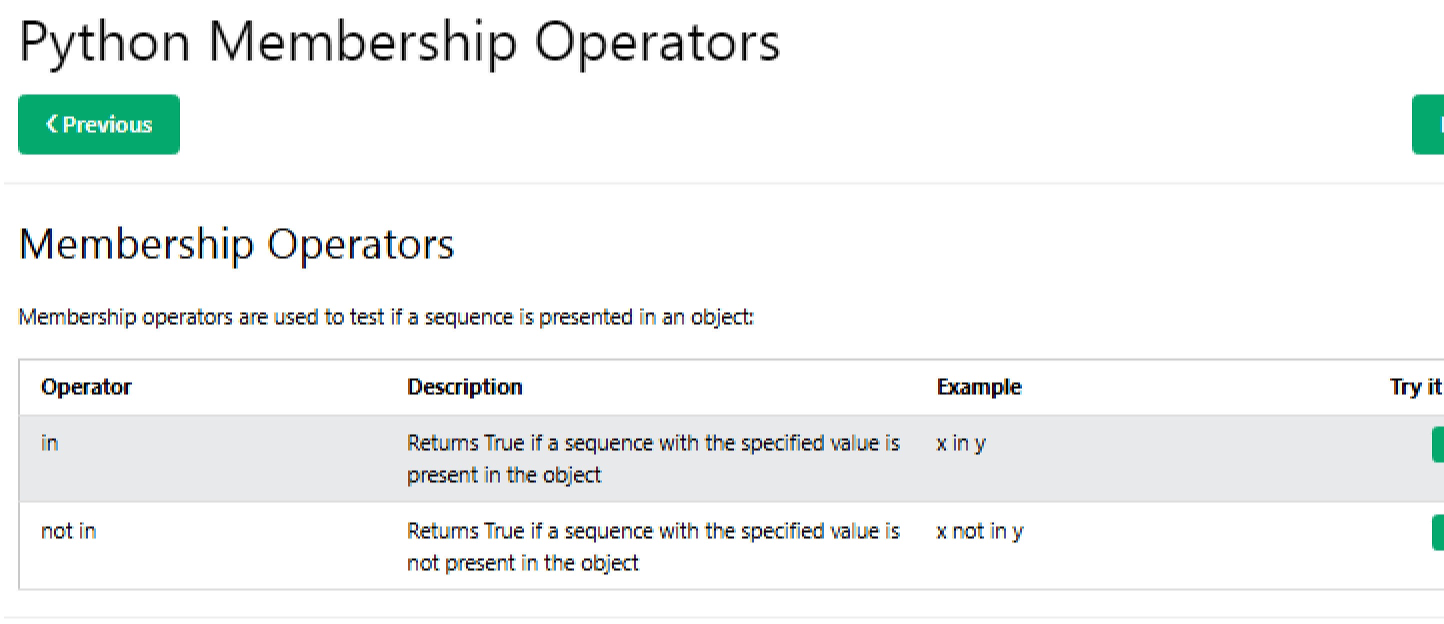
Python Identity Operators

❮ Previous

Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example	Try it
is	Returns True if both variables are the same object	x is y	
is not	Returns True if both variables are not the same object	x is not y	

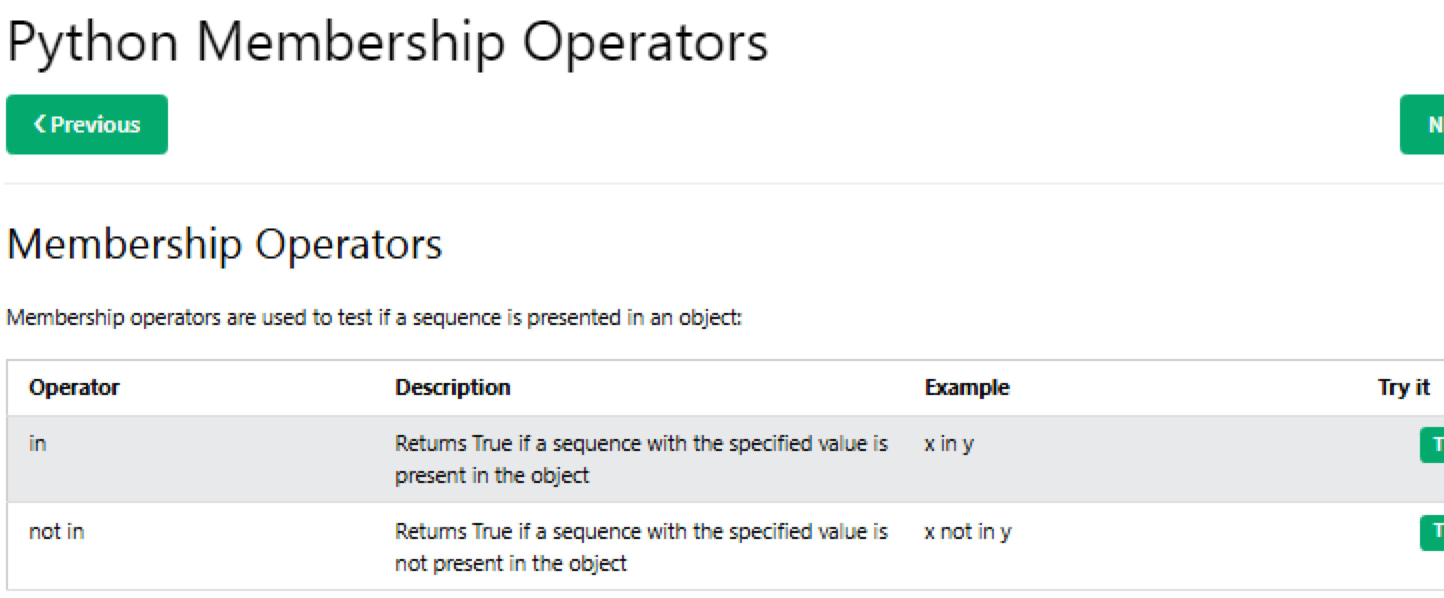


Python Membership Operators

Membership Operators

Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example	Try it
in	Returns True if a sequence with the specified value is present in the object	x in y	
not in	Returns True if a sequence with the specified value is not present in the object	x not in y	



Python Membership Operators

Membership Operators

Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example	Try it
in	Returns True if a sequence with the specified value is present in the object	x in y	
not in	Returns True if a sequence with the specified value is not present in the object	x not in y	

Python Bitwise Operators

‹ Previous

Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

Operator	Name	Description	Example	Try it
&	AND	Sets each bit to 1 if both bits are 1	x & y	
|	OR	Sets each bit to 1 if one of two bits is 1	x | y	
^	XOR	Sets each bit to 1 if only one of two bits is 1	x ^ y	
~	NOT	Inverts all the bits	~x	
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off	x << 2	
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off	x >> 2	

Precedence Order

The precedence order is described in the table below, starting with the highest precedence at the top:

Operator	Description	Try it
`()`	Parentheses	
`**`	Exponentiation	
`+x` `-x` `~x`	Unary plus, unary minus, and bitwise NOT	
`*` `/` `//` `%`	Multiplication, division, floor division, and modulus	
`+` `-`	Addition and subtraction	
`<<` `>>`	Bitwise left and right shifts	
`&`	Bitwise AND	
`^`	Bitwise XOR	
`|`	Bitwise OR	
`==` `!=` `>` `>=` `<` `<=` `is` `is not` `in` `not in`	Comparisons, identity, and membership operators	
`not`	Logical NOT	
`and`	AND	
`or`	OR	