Python

31 January 2022 14:33

© RAJAT KUMAR
https://www.linkedin.com/in/imRajat/
https://github.com/im-Rajat

Following : https://www.w3schools.com/python/default.asp

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

Execute Python Syntax

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

Python Indentation

Indentation refers to the spaces at the beginning of a code line.
Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

Comments

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

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

Python Variables

```
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)

x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0

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

Single or Double Quotes?

```
x = "John"
# is the same as
x = 'John'
```

Case-Sensitive

Variable names are case-sensitive.

```
a = 4
A = "Sally"
#A will not overwrite a
```

Python Variables - Assign Multiple Values

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

One Value to Multiple Variables

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

Unpack a Collection

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

Output Variables

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

Global Variables

```
x = "awesome"

def myfunc():
  print("Python is " + x)

myfunc()
```

The global Keyword

```
def myfunc():
  global x
  x = "fantastic"

myfunc()

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

Built-in Data Types

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

```
x = 5
```

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

Setting the 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

Setting the Specific Data Type

x = str("Hello World")	str
x = int(20)	int
x = float(20.5)	float
x = complex(1j)	complex
x = list(("apple", "banana", "cherry"))	list
x = tuple(("apple", "banana", "cherry"))	tuple
x = range(6)	range
x = dict(name="John", age=36)	dict
x = set(("apple", "banana", "cherry"))	set
x = frozenset(("apple", "banana", "cherry"))	frozenset
x = bool(5)	bool
x = bytes(5)	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

Python Numbers

There are three numeric types in Python:

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

#convert from int to float:
a = float(x)

#convert from float to int:
b = int(y)
```

```
#convert from int to complex:
c = complex(x)
```

Random Number

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

Python Casting

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

x = float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2

x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

Python Strings

```
a = "Hello"
print(a)

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

Strings are Arrays

```
a = "Hello, World!"
print(a[1])

for x in "banana":
  print(x)
```

String Length

```
a = "Hello, World!"
print(len(a))
```

Check String

```
txt = "The best things in life are free!"
print("free" in txt)   # return true
```

Check if NOT

```
txt = "The best things in life are free!"
print("expensive" not in txt)
```

Slicing Strings

```
b = "Hello, World!"

print(b[2:5])    # Get the characters from position 2 to position 5 (not included):
print(b[:5])   # Get the characters from the start to position 5 (not included):
print(b[2:])    # Get the characters from position 2, and all the way to the end:
print(b[-5:-2])   # Use negative indexes to start the slice from the end of the string:
```

Modify Strings

```
a = "Hello, World!"

print(a.upper())
print(a.lower())
```

The strip() method removes any whitespace from the beginning or the end:

```
print(a.strip()) # returns "Hello, World!"
```

The replace() method replaces a string with another string:

```
print(a.replace("H", "J"))
```

The split() method splits the string into substrings if it finds instances of the separator:

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

More about String methods:
https://www.w3schools.com/python/python_ref_string.asp

String Concatenation

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

String Format

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))

quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

Escape Characters

The escape character allows you to use double quotes when you normally would not be allowed:

```
txt = "We are the so-called \"Vikings\" from the north."
```

Python Operators

Python divides the operators in the following groups:

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

Python Lists

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data,

the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

```
mylist = ["apple", "banana", "cherry"]
print(thislist)

print(len(thislist))

list1 = ["abc", 34, True, 40, "male"]
```

The list() Constructor

```
thislist = list(("apple", "banana", "cherry")) # note the double round-brackets
print(thislist)

thislist = ["apple", "banana", "cherry"]
thislist.insert(2, "watermelon")
print(thislist)

thislist = ["apple", "banana", "cherry"]
thislist.append("orange")    # at the end
print(thislist)

thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)
print(thislist)

thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)


thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)

thislist = ["apple", "banana", "cherry"]
thislist.pop()    # Remove the last item:
print(thislist)

thislist = ["apple", "banana", "cherry"]
del thislist[0]    # Remove the first item:
print(thislist)

thislist = ["apple", "banana", "cherry"]
del thislist    # Delete the entire list:

thislist = ["apple", "banana", "cherry"]
thislist.clear()    # The clear() method empties the list.
print(thislist)
```

Loop Lists

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

for i in range(len(thislist)):
  print(thislist[i])
```

```
i = 0
while i < len(thislist):
  print(thislist[i])
  i = i + 1

thislist = ["apple", "banana", "cherry"]
[print(x) for x in thislist]
```

Python Collections (Arrays)

There are four collection data types in the Python programming language:

- List is a collection which is ordered and changeable. Allows duplicate members.
- Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
- Set is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
- Dictionary is a collection which is ordered** and changeable. No duplicate members.