Python Programming Language

Prepared by: Mohamed Ayman

Machine Learning Engineer

spring 2018


facebook.com/sw.eng.MohamedAyman


sw.eng.MohamedAyman@gmail.com

wuzzuf.net/me/engMohamedAyman


codeforces.com/profile/Mohamed_Ayman

Variable Types

Outline

1- Assigning Values to Variables

2- Multiple Assignment

3- Standard Data Types

4- Python Numbers

5- Python Strings

6- Python Lists

7- Python Tuples

8- Python Dictionary

9- Python Set

10- Data Type Conversion

Assigning Values to Variables

- Variables are nothing but reserved memory locations to store values. It means that when you create a variable, you reserve some space in the memory.
- Based on the data type of a variable, the interpreter allocation memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to the variables, you can store integers, decimals or characters in these variables.
- Python Variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.
- The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable.

Assigning Values to Variables


```
counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
name    = "John"       # A string
print (counter)
print (miles)
print (name)
```

```
100
1000.0
John
```

Multiple Assignment

- Python allows you to assign a single value to several variables simultaneously.

```
a = b = c = 1
```

- Here an integer object is created with the value 1, and the three variable are assigned to the same memory location. You can also assign multiple objects to multiple variables

```
a, b, c = 1, 2, "john"
```

Standard Data Types

- The data stored in memory can be of many types. For example, a person age is stored as a numeric value and his or her address is stored as alphanumeric characters.
- Every value in Python has a data type. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.
- There are various data types in Python. Some of the important types are listed below. Python has five standard data types:

Numbers — String — List — Tuple — Dictionary

Python Numbers

- Number data type store numeric values. Number objects are created when you assign a value to them.
- Integers, floating point numbers and complex numbers falls under Python numbers category. They are defined as int, float and complex class in Python.
- Python supports three different numerical types
  - int (signed integer)
  - float (floating point real value)
  - complex (complex numbers)

```
a = 5
print(a)

a = 2.0
print(a)

a = 1+2j
print(a)
```

```
5
2.0
(1+2j)
```

Python Strings

- String is sequence of Unicode characters. We can use single quotes or double quotes to represent strings. Multi-line strings can be denoted using triple quotes, ''' or """.
- String in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows either pair of single or double quotes. Subsets of strings can be taken using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 to the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

Python Strings


```
x = 'Hello Python'
print(x)
print(x[0])
print(x[3:7])
print(x[:4])
print(x[7:])
print(x[2:5]+x[7:10])
print(x+'Test')
print(x[7:10]*3)
print(x*2)
```

```
Hello Python
H
lo P
Hell
ython
lloyth
Hello PythonTest
ythythyth
Hello PythonHello Python
```

Python Strings


```
x = 'python programming'
print('x =',x)
print('x[2]      : ',x[2])
print('x[2:9]    : ',x[2:9])
print('x[:4]     : ',x[:4])
print('x[11:]    : ',x[11:])
print('x[-2]     : ',x[-2])
print('x[-4:]    : ',x[-4:])
print('x[:-7]    : ',x[:-7])
print('x[-8:-2]  : ',x[-8:-2])
print('x[4:-2]   : ',x[4:-2])
print('x[-8:14]  : ',x[-8:14])
```

```
x = python programming
x[2]      :  t
x[2:9]    :  thon pr
x[:4]     :  pyth
x[11:]    :  ramming
x[-2]     :  n
x[-4:]    :  ming
x[:-7]    :  python prog
x[-8:-2]  :  grammi
x[4:-2]   :  on programmi
x[-8:14]  :  gram
```

Python Lists

- List is an ordered sequence of items. It is one of the most used data type in Python and is very flexible. All the items in a list do not need to be of the same type.
- Declaring a list is pretty straight forward. Items separated by commas are enclosed within brackets [ ].
- To some extent, lists are similar to arrays in C. One of the differences between them is that all the items belonging to a list can be of different data types.
- The value stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Lists


```
x = [123, 'abcd', 2.23, 789, 'python']
print(x)
print(x[0])
print(x[1:3])
print(x[:4])
print(x[2:])
print(x[1:3]+x[2:4])
print(x+['Hello', 3.17])
print(x[1:4]*3)
print(x*2)
```

```
[123, 'abcd', 2.23, 789, 'python']
123
['abcd', 2.23]
[123, 'abcd', 2.23, 789]
[2.23, 789, 'python']
['abcd', 2.23, 2.23, 789]
[123, 'abcd', 2.23, 789, 'python', 'Hello', 3.17]
['abcd', 2.23, 789, 'abcd', 2.23, 789, 'abcd', 2.23, 789]
[123, 'abcd', 2.23, 789, 'python', 123, 'abcd', 2.23, 789, 'python']
```

Python Lists


```
x = ['python', 'c++', 'java', 2000, 3.2, 'math', 'physics']
print('x =',x)
print('x[2]      : ',x[2])
print('x[2:5]    : ',x[2:5])
print('x[:3]     : ',x[:3])
print('x[3:]     : ',x[3:])
print('x[-2]     : ',x[-2])
print('x[-4:]    : ',x[-4:])
print('x[:-3]    : ',x[:-3])
print('x[-5:-2]  : ',x[-5:-2])
print('x[2:-2]   : ',x[2:-2])
print('x[-5:4]   : ',x[-5:4])
```

```
x = ['python', 'c++', 'java', 2000, 3.2, 'math', 'physics']
x[2]      :  java
x[2:5]    :  ['java', 2000, 3.2]
x[:3]     :  ['python', 'c++', 'java']
x[3:]     :  [2000, 3.2, 'math', 'physics']
x[-2]     :  math
x[-4:]    :  [2000, 3.2, 'math', 'physics']
x[:-3]    :  ['python', 'c++', 'java', 2000]
x[-5:-2]  :  ['java', 2000, 3.2]
x[2:-2]   :  ['java', 2000, 3.2]
x[-5:4]   :  ['java', 2000]
```

Python Tuples

- Tuple is an ordered sequence of items same as list. The only difference is that tuples are immutable. Tuples once created cannot be modified.
- Tuples are used to write-protect data and are usually faster than list as it cannot change dynamically.
- It is defined within parentheses () where items are separated by commas.
- The main difference between lists and tuples is Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

Python Tuples


```
x = (123, 'abcd', 2.23, 789, 'python')
print(x)
print(x[0])
print(x[1:3])
print(x[:4])
print(x[2:])
print(x[1:3]+x[2:4])
print(x+('Hello', 3.17))
print(x[1:4]*3)
print(x*2)
```

```
(123, 'abcd', 2.23, 789, 'python')
123
('abcd', 2.23)
(123, 'abcd', 2.23, 789)
(2.23, 789, 'python')
('abcd', 2.23, 2.23, 789)
(123, 'abcd', 2.23, 789, 'python', 'Hello', 3.17)
('abcd', 2.23, 789, 'abcd', 2.23, 789, 'abcd', 2.23, 789)
(123, 'abcd', 2.23, 789, 'python', 123, 'abcd', 2.23, 789, 'python')
```

Python Tuples


```
x = ('python', 'c++', 'java', 2000, 3.2, 'math', 'physics')
print('x =',x)
print('x[2]      : ',x[2])
print('x[2:5]    : ',x[2:5])
print('x[:3]     : ',x[:3])
print('x[3:]     : ',x[3:])
print('x[-2]     : ',x[-2])
print('x[-4:]    : ',x[-4:])
print('x[:-3]    : ',x[:-3])
print('x[-5:-2]  : ',x[-5:-2])
print('x[2:-2]   : ',x[2:-2])
print('x[-5:4]   : ',x[-5:4])
```

```
x = ('python', 'c++', 'java', 2000, 3.2, 'math', 'physics')
x[2]      :  java
x[2:5]    :  ('java', 2000, 3.2)
x[:3]     :  ('python', 'c++', 'java')
x[3:]     :  (2000, 3.2, 'math', 'physics')
x[-2]     :  math
x[-4:]    :  (2000, 3.2, 'math', 'physics')
x[:-3]    :  ('python', 'c++', 'java', 2000)
x[-5:-2]  :  ('java', 2000, 3.2)
x[2:-2]   :  ('java', 2000, 3.2)
x[-5:4]   :  ('java', 2000)
```

Python Dictionary

- Dictionary is an unordered collection of key-value pairs.
- It is generally used when we have a huge amount of data. Dictionaries are optimized for retrieving data. We must know the key to retrieve the value.
- In Python, dictionaries are defined within braces {} with each item being a pair in the form key : value. Key and value can be of any type.
- Python dictionaries are kind of hash-table type. They work like associative arrays or hashes found in perl and consist of key-value pairs.
- A dictionary key can be almost any arbitrary Python object.
- Dictionaries are enclosed by curly braces ({}) and value can be assigned and accessed using square braces ([])

Python Dictionary


```
x = {'python':'easy', 'c++':0, 'java':'medium'}
print(x)
print(x['c++'])
x['python'] = 1
print(x)
print(x['python'])
print(x.keys())
print(x.values())
```

```
{'python': 'easy', 'c++': 0, 'java': 'medium'}
0
{'python': 1, 'c++': 0, 'java': 'medium'}
1
dict_keys(['python', 'c++', 'java'])
dict_values([1, 0, 'medium'])
```

Python Set

- Set is an unordered collection of unique items. Set is defined by values separated by comma inside braces { }. Items in a set are not ordered.

```
a = {5,2,3,1,4}

# printing set variable
print("a = ", a)
```

```
a =  {1, 2, 3, 4, 5}
```

- We can perform set operations like union, intersection on two sets. Set have unique values. They eliminate duplicates.

```
a = {1,2,2,3,3,3}

# printing set variable
print("a = ", a)
```

```
a =  {1, 2, 3}
```

Data Type Conversion

- We can convert between different data types by using different type conversion functions like int(), float(), str() etc. Type Conversion is the conversion of object from one data type to another data type.
- Implicit Type Conversion is automatically performed by the Python interpreter, but Explicit Type Conversion is also called Type Casting, the data types of object are converted using predefined function by user.
- Sometimes, you may need to perform conversions between the built-in types, you simply use the type-name as a function. There are several built-in functions to perform conversion from one data type to another. These functions return a new object representing the converted value.

```
print(float(5))
print(int(10.6))
print(int(-10.6))
print(float('2.5'))
print(str(25))
```

```
5.0
10
-10
2.5
25
```

Data Type Conversion

Function	Description
int(x [,base])	Converts x to an integer. The base specifies the base if x is a string.
float(x)	Converts x to a floating-point number.
complex(real [,imag])	Creates a complex number.
str(x)	Converts object x to a string representation.
repr(x)	Converts object x to an expression string.
eval(str)	Evaluates a string and returns an object.
tuple(s)	Converts s to a tuple.

Data Type Conversion

Function	Description
tuple(s)	Converts s to a tuple.
list(s)	Converts s to a list.
set(s)	Converts s to a set.
dict(d)	Creates a dictionary. d must be a sequence of (key,value) tuples.
chr(x)	Converts an integer to a character.
unichr(x)	Converts an integer to a Unicode character.
ord(x)	Converts a single character to its integer value.
hex(x)	Converts an integer to a hexadecimal string.
oct(x)	Converts an integer to an octal string.

Data Type Conversion

```
x = 5
print(type(x))
x = 7.3
print(type(x))
x = 'python'
print(type(x))
x = [1,2.5,'c++']
print(type(x))
x = (4.6,'java',3)
print(type(x))
x = {1:'math', 6.2:'scala', 'version':'3'}
print(type(x))
x = {4.1,7,'python'}
print(type(x))
x = True
print(type(x))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'list'>
<class 'tuple'>
<class 'dict'>
<class 'set'>
<class 'bool'>
```

Outline

- 1- Assigning Values to Variables
- 2- Multiple Assignment
- 3- Standard Data Types
- 4- Python Numbers
- 5- Python Strings
- 6- Python Lists
- 7- Python Tuples
- 8- Python Dictionary
- 9- Python Set
- 10- Data Type Conversion

Practice

Problem 1

- Take as an input name and age of user in separate lines then print:

  hello "user_name" your age is "user_age"

- Test Cases:

```
 mohamed
 20
Hello mohamed your age is 20
```

```
 amr
 17
Hello amr your age is 17
```

```
 ali
 23
Hello ali your age is 23
```

```
 mostafa
 19
Hello mostafa your age is 19
```

```
 ahmed
 25
Hello ahmed your age is 25
```

Problem 1 Solution


```
name = str(input())
age = int(input())
print('Hello',name,'your age is',age)
```

Problem 2

- Take as an input name and city of user in same line then print:

  hello "user_name" you live in "user_city"

- Test Cases:

```
 ali egypt
Hello ali you live in egypt
```

```
 amr england
Hello amr you live in england
```

```
 ahmed germany
Hello ahmed you live in germany
```

```
 mostafa france
Hello mostafa you live in france
```

```
 kareem spain
Hello kareem you live in spain
```

Problem 2 Solution


```
name,city = map(str,input().split(' '))
print('Hello',name,'you live in',city)
```

Problem 3

- Take as an input name and age and weight of user in same line then print: hello "user_name" your age is "user_age" and your weight is "user_weight"
- Test Cases:

```
 mohamed 25 80
Hello mohamed your age is 25 and your weight is 80.0
```

```
 ahmed 22 75
Hello ahmed your age is 22 and your weight is 75.0
```

```
 ali 18 73.5
Hello ali your age is 18 and your weight is 73.5
```

```
 amr 27 88.3
Hello amr your age is 27 and your weight is 88.3
```

Problem 3 Solution


```
name, age, weight = list(input().split(' '))
name = str(name)
age = int(age)
weight = float(weight)
print('Hello',name,'your age is',age,'and your weight is',weight)
```

Questions ?

References

[01]	Online Courses YouTube playlists	http://bit.ly/2EcbLvZ
[02]	Programiz Tutorial	https://bit.ly/2paX2xA
[03]	Python 3 Tutorial Point PDF	http://bit.ly/2xcqp66
[04]	Python Guru Tutorial	https://bit.ly/28TtLRr
[05]	Python for Everybody PDF	http://bit.ly/2gSWkSS
[06]	A Byte of Python PDF	http://bit.ly/1I9Yqp9
[07]	Python Course Tutorial	https://bit.ly/2I076hU
[08]	Python Basics Tutorial PDF	http://bit.ly/1jMlluB
[09]	Learning to Program Using Python	http://bit.ly/2zO4f91
[10]	Think Python PDF	http://bit.ly/2cTgLIk
[11]	Python 3 Patterns, Recipes and Idioms	http://bit.ly/2by2bsN