PYTHON 3.X

Pratik Joshi
Data Engineer
Cloud Engineer (AWS Certified)
Android Developer (Google Certified)

Contents (Core Python)

Contents (Advance Python)

- PYTHON OOP
- PYTHON RegEx
- PANDAS
- NUMPY
- JSON
- Databases
- Scheduling
- Caching
- PYTHON GUI
- Mail Sending
- Automation in Python

INTRODUCTION

What is Python?

- Python is **High Level**, **Interpreted, Interactive** and **Object-Oriented** programing language.
- Python in copyrighted [General Public License (GPL)]
- Created by Guido van Rossum in 1991.
- Derived from ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.
- Initial Version: 1.4 (1996)
- Latest Version: 3.9 (2021)

Why Python?

- Used for Web development, software development, data analytics and Automation.
- Able to handle Big data and Complex Mathematics
- Platform Independent
- Allows developer to write fewer lines of codes
- Can be connected to multiple databases
- Can be used as both scripting language and compiled languages.
- Simple syntax : Easy to learn, read and maintain

Python 2 Vs Python 3

- For beginner there is no difference between version 2 and 3 except print syntax.

Python 2	Python 3
It is the more stable and transparent version of the Python programming language.	It is the future of Python designed to address the design flaws in the previous versions.
The print-syntax is treated as a statement rather than a function which requires text to be wrapped in parenthesis.	The print is explicitly treated as a function and replaced by the print( ) function in Python 3 which requires an extra pair of parenthesis.
ASCII string type is used by default to store strings.	Unicode is the implicit string type by default.
It simply returns an integer to the nearest whole number when dividing two integers.	It makes integer division more intuitive by using true division for integers and floats.
xrange function reconstructs the sequence every time.	xrange is replaced by range( ) function in Python 3.

Python Setup (DIY)

- https://www.python.org/downloads/
- Setup environment variable
- python --version

Hello World with Python

- print(“Hello World”) #Interactive mode
- python hello.py #scripting mode
- Three Golden words to be remember:
  - Indentation : To indicate block of code and scope (Whitespace or tab)
  - Working Directory
  - Colon (:) : To define a block
- Comment:
  - Single line : #
  - Multi line : “”” “”””
- Multiple statement in a single line : use ;

  E.g : statement 1 ; statement 2; statement3

Variables

- No need of explicit declaration, a variable is declared when assigned.

```
x = 5
y = "Hi"
print(x)
```

- Multiple Assignment

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

- Casting

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
```

- Type() : returns type of variable

```
x = 5
print(type(x))
```

Variable Names and Global Variable

- 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

Global Variable:

- Variable created outside function scope.
- To create **global** variable inside function global keyword is used.

Data Types

Built-in datatypes:

- Text Type: str
- Numeric Types: int, float, complex
- Sequence Types: list, tuple
- Mapping Type: dict
- Set Types: set
- Boolean Type: bool
- Binary Types: bytes

Numbers

```
a = 5
print(a, "is of type", type(a))  #int
a = 2.0
print(a, "is of type", type(a))  #float
a = 1+2j
print(a, "is complex number?", isinstance(1+2j,complex))    #True
```

Casting:

```
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("4.5") # w will be 4.5
int(10.6)        # 10
str(25)          #'25'
int('1s')        # Error
set([1,2,3])      ## {1, 2, 3}
tuple({5,6,7})    ## (5, 6, 7)
list('hello')     ## ['h', 'e', 'l', 'l', 'o']
dict([(3,26),(4,44)])  ## {3: 26, 4: 44}
```

Python Output

Output : print(“Hello World”)

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

Objects: Value to be printed
Sep : separator, default: space
File: the object where the values are printed and its default value is **sys.stdout**

Output Formatting:

```
>>> x = 5; y = 10
>>> print('The value of x is {} and y is {}'.format(x,y))
The value of x is 5 and y is 10

>>> print('Hello {name}, {greeting}'.format(greeting = 'Goodmorning', name = 'Pratik'))
Hello John, Goodmorning

>>>print('{0} is better then {1}'.format('Modi','Rahul'))
Modi is better then Rahul
```

Python Input

```
input([prompt])
```

```
>>> num = input('Enter a number: ')
Enter a number: 10
>>> num
'10'
```

Python Operators

Arithmetic Operators : +,-,*,/,%,//,**

Comparison Operators : <,>,==,!=,>=,<=

Assignment Operators: =,+=,-=,*=,/=,%=,//=,**=

Logical Operators: and, or, not

Membership Operator: in, not in

Bitwise Operators: &,|,~,^,<<,>>

FLOW CONTROL AND LOOP

Decision Control

If Statement:

```
if condition:
   Statement1 #executed if condition True
Statement2
```

If..else:

```
if condition:
   Statement1 #executed if condition True
else:
   Statement2
```

Nested-if:

```
if condition1:
   Statement1 #executed if condition1 True
      if condition2:
          Statement2 #Executed if condition2 True
```

Decision Control (Cont…)

if-elif-else ladder:

```
if (condition):
    statement
elif (condition):
    statement
else:
    Statement
```

Short Hand if statement

```
if a > b: print("a is greater than b")
```

Short Hand if...else

```
i = 10
print(True) if i < 15 else print(False)
```

Pass:

```
if b > a:
  pass
```

For Loop

Syntax:

```
for var in iterable:
      Statements
```

E.g

```
lst = ["Raju", "Shyam", "Babu"]
for i in lst:
    print(i)
```

Continue:

```
for letter in 'pratik':
  if letter == 'a' or letter == 'i':
      continue
  print('Without Vowel :', letter)
```

Break : Brings control out of the loop

Pass: Used to write empty loop

```
for letter in 'INDIA':
    pass
print('Last Letter :', letter)
```

range()

range(start, stop,step_size)
Step_size = 1 by default

- Lazy Evaluation
- Doesn’t store all values in memory

E.g:

```
print(list(range(10)))
print(list(range(2, 8)))
print(list(range(2, 20, 3)))

for x in range(2, 30, 3):
  print(x)

l=[“Sachin”,”Ramesh”,”Tendulkar”]
for i in range(len(l)):
    print(l[i], end=" ")
```

for-else loop

```
for i in range(1, 5):
    print(i)
else: # Executed because no break in for
    print("Done without BreakUp :)")

for i in range(1, 4):
    print(i)
    break
else: # Not executed as there is a break
    print("Break-up :(" )
```

- The else block will NOT be executed if the loop is stopped by a break statement.

Hands-On 1

1. Print your name separated with *
2. Print Sum of first 10 number
3. Print odd/even number between 1-100.
4. Write a Python program to count numbers which are divisible by 7 and multiple of 5, between 500 and 1400 (both included).
5. Write a Python program that prints all the numbers from 0 to 101 except the numbers divisible by 5.
6. Write a Python program which iterates the integers from 1 to 50. For multiples of three print "Oppo" instead of the number and for the multiples of five print "Vivo". For numbers which are multiples of both three and five print "OppoVivo".
7. Guess Game
8. Password Strength Checker