# Python-Notes

## Author: - Sameer Shrivastava

- Developed By Guido van Rossum

**What is Python: -** Python is a general purpose and high level programming language, its combination of functional, object oriented, scripting and modular languages.

- ✓ Interpreter language
- ✓ Dynamic typing

**Note**: - No braces, no semicolons, everything here is space

**Types: -**
1. Cpython (Python)
2. Jython
3. Iron Python

**Applications of python:-**

- ✓ Web
- ✓ Database Application
- ✓ Data crawling
- ✓ Data analysing
- ✓ Data science
- ✓ Machine learning
- ✓ Artificial Intelligence
- ✓ Network Programming
- ✓ IOT
- ✓ Develop Games

**Features of Python:-**

- ➢ Simple and easy
- ➢ Open source
- ➢ High level programing language
- ➢ Platform independent
- ➢ Portability
- ➢ Interpreted
- ➢ Extensive Library

**Awesome Python :-** https://github.com/sameer2425/Awesome-Python

**Python version :-**

- ✓ Python 2 (python 2.x.x)
- ✓ Python 3 (python 3.x.x)

**Downloading python**:-

- ✓ https://www.python.org/downloads/windows/
- ✓ https://www.ics.uci.edu/~pattis/common/handouts/pythoneclipsejava/python.html

**First program on python interpreter (Hello World):-**

- ✓ Print “hello world” (python 2)
- ✓ Print (“Hello World”) (python 3)

**Variables in python (Dynamic typing):-**

- a = 1

```
>>> a=1
>>> type(a)
<class 'int'>
>>>
```

- b = “some word” => string needs to be enclosed in a double or single quotes

```
>>> b = "some word"
>>> type(b)
<class 'str'>
>>>
```

type (variable name) => Give the data type of the variable

**Memory allocation in python :-**

```
>>> a=1
>>> b=1
>>> id(a)
1382860928
>>> id(b)
1382860928
```

id(variable name) => provide memory address

**Examples**:-

- Add two variable

```
>>> a=1
>>> b=2
>>> c=a+b
>>> print(c)
3
```

- Swapping

```
>>> a=1
>>> b=a
>>> print(b)
1
```

- Strings and integer

```
>>> a = 1
>>> b = 1
>>> a=1
>>> b=1
>>> c=a+b
>>> print(c)
2
>>> a="some"
>>> b="word"
>>> c=a+b
>>> print(c)
someword
```

**Execute the Script:-**

- Python <filename.py>
- Interactive Mode: Python interpreter, ipython, IDLE

**Python File Extensions:-**

- .py => Normal python file
- .pyc => compiled bytecode
- .pyw => Python script for windows
- .pyd => python script made as a windows dll

**Python Comments:-**

- single line comment

```
>>> #single line comment
...
>>>
```

- Multiline comment

```
>>> """
... Multiline
... Comments
... """
'\nMultiline\nComments\n'
>>>
```

**Shell as a Simple Calculator:-**

```
In [1]: 2+2
Out[1]: 4

In [2]: 8/2
Out[2]: 4.0

In [3]: 2*3
Out[3]: 6

In [4]:
```

**Order of operations:-**

() => come first Multiplication second
(2+3) * 2 = 5 *2

```
In [7]: (2+3) * 2
Out[7]: 10

In [8]:
```

**Python IDE :-**

- **Eclipse** : https://www.eclipse.org/downloads/
- **Pycharm** : https://www.jetbrains.com/pycharm-edu/download
- **Spyder** : https://www.spyder-ide.org/
- Can use Jupyter notebook also

**Jupyter Notebook :-** web application to execute codes

- https://jupyter.readthedocs.io/en/latest/install.html

**Anaconda:-** Provides everything we need for data science and machine learning

- https://www.anaconda.com/distribution/#download-section
- Contain core python
- 100+ python libraries
- Spyder and Jupyter Notebook come with package
- Package manager :- conda
- conda install <package_name>

**PIP :-** Pip is a package manager for python to install its libraries

- pip install <package_name>

**Identifiers**: - Allowed characters

- Alphabet symbol
- Digit (0 to 9)
- Underscore ( _ )

```
>>> cash=10
>>> ca$h=10
  File "<stdin>", line 1
    ca$h=10
      ^
SyntaxError: invalid syntax
>>> 123total=12
  File "<stdin>", line 1
    123total=12
           ^
SyntaxError: invalid syntax
>>> total123=12
```

**Reserved Words:-**

```
In [1]: import keyword

In [2]: keyword.kwlist
```

**Data Types: -** Type of data (Dynamic type)

- Type :- to check type of variable
- Id :- to get address of object

- In Python everything is object
- In python2 we have long data type but in python3 only int type
- Long data type is available in python2 but not in python3
- Use char value as string
- Int, float, complex, bool, str, bytes, bytearray, range, list, tuple, set, frozenset, dict, None

**Int:-**

```
>>> a=10
>>> type(a)
<class 'int'>
>>> a=0B1111
>>> print(a)
15
>>> type(a)
<class 'int'>
>>> a=0o123
>>> print(a)
83
>>> type(a)
<class 'int'>
>>> a=0XFACE
>>> print(a)
64206
>>> type(a)
<class 'int'>
```

```
>>> bin(15)
'0b1111'
>>> oct(10)
'0o12'
>>> hex(100)
'0x64'
```

**Float :-**

```
>>> f=1.23
>>> type(f)
<class 'float'>
>>> f=1.2e3
>>> type(f)
<class 'float'>
>>> print(f)
1200.0
```

**Complex Type :-**

```
>>> a=10+1.5j
>>> type(a)
<class 'complex'>
>>> print(a)
(10+1.5j)
```

**Bool Type :-**

```
>>> a=True
>>> type(a)
<class 'bool'>
>>> b=False
>>> type(b)
<class 'bool'>
```

**Multiple Assignment:-**

```
In [8]: a,b = 1,2

In [9]: print(a)
1

In [10]: print(b)
2

In [11]: a,b,c = 1,2,'someword'

In [12]: print(c)
someword
```

**String Handling: -** Strings => ‘singlequote’ or ”doublequote”

**String operations: -** len () => to find length of a string

```
In [9]: mystring="someword"

In [10]: len(mystring)
Out[10]: 8
```

**Access by index:-**

```
In [12]: myword='someword'

In [13]: myword[0]
Out[13]: 's'

In [14]: myword[1]
Out[14]: 'o'

In [15]: myemail='some@gmail.com'

In [16]: myemail[4]
Out[16]: '@'
```

**String slicing :-**

```
In [18]: mystring='someword'

In [19]: mystring[0]
Out[19]: 's'

In [20]: mystring[0:4]
Out[20]: 'some'

In [21]: mystring[4:0]
Out[21]: ''

In [22]: mystring[-1]
Out[22]: 'd'

In [23]: mystring[-4]
Out[23]: 'w'

In [24]: mystring[-4:-1]
Out[24]: 'wor'

In [25]: mystring[-1:-4]
Out[25]: ''
```

**String operations:-** upper(), lower(), isupper(), islower(), find()

```
In [37]: mystring="someword"

In [38]: mystring.upper()
Out[38]: 'SOMEWORD'

In [39]: chage_val = mystring.upper()

In [40]: print(chage_val)
SOMEWORD

In [41]: chage_val
Out[41]: 'SOMEWORD'

In [42]: change_lower = chage_val.lower()

In [43]: print(change_lower)
someword
```

```
In [45]: mystr = 'SOMEWORD'

In [46]: mystr.isupper()
Out[46]: True

In [47]: mystr = 'Someword'

In [48]: mystr.isupper()
Out[48]: False

In [49]: mystr = 'myword'

In [50]: mystr.islower()
Out[50]: True

In [51]: mystr = 'Myword'

In [52]: mystr.islower()
Out[52]: False
```

```
In [56]: mystr = 'some1'

In [57]: mystr.isalnum()
Out[57]: True

In [58]: mystr = 'someword'

In [59]: mystr.isalpha()
Out[59]: True

In [60]: mystr = 'someword1'

In [61]: mystr.isalpha()
Out[61]: False

In [62]: mystr = '12345'

In [63]: mystr.isdigit()
Out[63]: True

In [64]: mystr = '12345a'

In [65]: mystr.isdigit()
Out[65]: False
```

```
In [67]: mystr = 'some word'

In [68]: mystr.isspace()
Out[68]: False

In [69]: mystr = ' some '

In [70]: mystr.isspace()
Out[70]: False

In [71]: mystr = ' '

In [72]: mystr.isspace()
Out[72]: True
```

```
In [77]: mystr = 'someword'

In [78]: mystr.startswith('m')
Out[78]: False

In [79]: mystr.startswith('s')
Out[79]: True

In [80]: mystr.endswith('g')
Out[80]: False

In [81]: mystr.endswith('d')
Out[81]: True
```

```
In [20]: mystr='sameer'

In [21]: mystr[1:]
Out[21]: 'ameer'

In [22]: mystr[:4]
Out[22]: 'same'

In [23]: mystr[:]
Out[23]: 'sameer'

In [24]: mystr*3
Out[24]: 'sameersameersameer'
```

```
In [26]: mychar = 'c'

In [27]: type(c)
Out[27]: str
```

**Type Conversion:-**

**Int :-**

```
In [29]: int(123.98)
Out[29]: 123

In [30]: int(10+5j)
---------------------------------------------------------------
t call last)<ipython-input-30-aeeb86d69c41> in <module>
----> 1 int(10+5j)

TypeError: can't convert complex to int

In [31]: int(True)
Out[31]: 1

In [32]: int(False)
Out[32]: 0

In [33]: int("10")
Out[33]: 10

In [34]: int("10.5")
---------------------------------------------------------------
t call last)<ipython-input-34-54dd49a25c21> in <module>
----> 1 int("10.5")

ValueError: invalid literal for int() with base 10: '10.5'

In [35]: int("ten")
---------------------------------------------------------------
ValueError                                Traceback (most
<ipython-input-35-846b3cf4a083> in <module>
----> 1 int("ten")

ValueError: invalid literal for int() with base 10: 'ten'
```

**Float :-**

```
In [37]: float(10)
Out[37]: 10.0

In [38]: float(10+5j)
---------------------------------------------------------------
TypeError                                 Traceback (most r
<ipython-input-38-d2f956539d9b> in <module>
----> 1 float(10+5j)

TypeError: can't convert complex to float

In [39]: float(True)
Out[39]: 1.0

In [40]: float(False)
Out[40]: 0.0

In [41]: float("10.0")
Out[41]: 10.0

In [42]: float("10")
Out[42]: 10.0

In [43]: float("ten")
---------------------------------------------------------------
ValueError                                Traceback (most r
<ipython-input-43-c8abde1341af> in <module>
----> 1 float("ten")

ValueError: could not convert string to float: 'ten'
```

**Complex:-**

```
>>> complex(10)
(10+0j)
>>> complex(10.5)
(10.5+0j)
>>> complex(True)
(1+0j)
>>> complex(False)
0j
>>> complex("10")
(10+0j)
>>> complex("10.5")
(10.5+0j)
>>> complex("ten")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: complex() arg is a malformed string
```

**Bool :-**

```
>>> bool(1)
True
>>> bool(0)
False
>>> bool(10)
True
>>> bool(10.5)
True
>>> bool(0.0)
False
>>> bool(0+1.5j)
True
>>> bool("True")
True
>>> bool("False")
True
>>> bool("")
False
```

**String:-**

```
>>> str(10)
'10'
>>> str(10.5)
'10.5'
>>> str(10+5j)
'(10+5j)'
>>> str(True)
'True'
```

**Mutable vs Immutable Objects :-**

- Mutable :- List, Dict, Set
- Immutable :- Tuple

```
>>> a = 10
>>> b = 10
>>> id(a)
1458555152
>>> id(b)
1458555152
>>> a is b
True
>>> a = 10+5j
>>> b = 10+5j
>>> id(a)
51174600
>>> id(b)
51174624
>>> a is b
False
```

**Bytes Data Type: -**

- Allowed values are 0 to 256

```
>>> x=[10,20,30,40]
>>> b = bytes(x)
>>> type(b)
<class 'bytes'>
>>> print(b[0])
10
>>> print(b[1])
20
>>> print(b[-1])
40
>>> for i in b:
...     print(i)
...
10
20
30
40
```

- Can't change its value (Immutable)

```
>>> x=[10,20,30,40]
>>> b=bytes(x)
>>> b[0]=100
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'bytes' object does not support item assignment
```

**Bytearray Data Type:**- Same as bytes but its Mutable

```
>>> x=[10,20,30,40]
>>> b=bytearray(x)
>>> for i in b:
...     print(i)
...
10
20
30
40
>>> b[0]=100
>>> for i in b:
...     print(i)
...
100
20
30
40
```

**List Data Type: -** Its represent group of values as single entity

- Insertion order same
- Duplicate allowed
- Represent :- [ ]

```
>>> list=[10,20,30,40]
>>> list[0]
10
>>> list[-1]
40
>>> list[1:3]
[20, 30]
>>> list[0] = 100
>>> for i in list:
...     print(i)
...
100
20
30
40
```

- Growable

```
>>> list=[10,20,30]
>>> list.append('sameer')
>>> list
[10, 20, 30, 'sameer']
>>> list.remove(20)
>>> list
[10, 30, 'sameer']
>>> list2 = list*2
>>> list2
[10, 30, 'sameer', 10, 30, 'sameer']
```

**Tuple Data Type:-** same as list but its immutable

- Represent :- ( )

```
>>> t=(10,20,30,40)
>>> type(t)
<class 'tuple'>
>>> t[0]=100
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignmen
>>> t.append('sameer')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
```

**Range Data Type: -**

- Sequence of number
- Its immutable

```
>>> range(10)
range(0, 10)
>>> r=range(10)
>>> for i in r:
...     print(i)
...
```

```
>>> r=range(10,20)
>>> for i in r:
...     print(i)
...
```

```
>>> r=range(10,20,2)
>>> for i in r:
...     print(i)
...
```

```
>>> r=range(10,20,2)
>>> r[0]
10
>>> r[15]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: range object index out of range
>>> r[0]=100
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'range' object does not support item assignment
```

- Create list using range

```
>>> l=list(range(10))
>>> l
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

**Set Data Type:-**

- No Duplicate
- Order not important
- Mutable
- No index concept
- Growable
- Represent :- { }

```
>>> s={100,0,10,200,10,'sameer'}
>>> type(s)
<class 'set'>
>>> s[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object does not support indexing
>>> s.add(60)
>>> s
{0, 100, 200, 10, 'sameer', 60}
>>> s.remove(100)
>>> s
{0, 200, 10, 'sameer', 60}
```

**Frozenset Data Type: -** same as set except that it is immutable

```
>>> s={10,20,30,40}
>>> fs=frozenset(s)
>>> type(fs)
<class 'frozenset'>
>>> fs
frozenset({40, 10, 20, 30})
>>> fs.add(70)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'frozenset' object has no attribute 'add'
>>> fs.remove(10)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'frozenset' object has no attribute 'remove'
```

**Dict Data Type:-** its represent group of value as key – value pairs

- Duplicate keys not allowed but value can be duplicate
- If tying to insert duplicate key then old value replace with new value
- Mutable and order not maintain
- Represent :- { }

```
>>> d={101:'sameer',102:'prashantha',103:'muzammil'}
>>> d
{101: 'sameer', 102: 'prashantha', 103: 'muzammil'}
>>> d[101]
'sameer'
```

```
>>> d={}
>>> type(d)
<class 'dict'>
>>> d['a']='apple'
>>> d['b']='banana'
>>> print(d)
{'a': 'apple', 'b': 'banana'}
```

**None Data Type :-** None means nothing or no value

```
>>> def m1():
...     a=10
...
>>> print(m1())
None
```

- Constants concept is not in python

**Operators:-**

- Arithmetic
- Relational
- Logical
- Bitwise
- Assignment
- Special