**Python**

Python is a general purpose high level programming language. If you want to develop data science application, machine learning, web application, AI, Robotics, you can go with python.

**Guido Van Rossam** developed Python in 1989 in National Research Institute in Netherland and officially if was out in 1991.

>>> print ("Hello World")

>>> a, b = 10, 20

>>> print (a + b)

>>> type(a)

<class ‘int’>

c, java are **Statically Typed Programming** as you have to define the type of each variable while defining.

Python is **Dynamically Typed language**. a is int type as above but if you will assign true to a, it will be treated as boolean. In case of java, c if you try to do same you will get the compilt time error.

>>> a = True

>>> type(a)

<class ‘bool’>

1. Python borrowed functional programming from C language
2. Python borrowed OOP from C++.
3. Scripting features from Perl and Shell script.
4. Modular programming from Modula-3.

Hence, Python can be all kinds of programming language.

>>> def myFunction(): print("Hello World");

>>> myFunction()

Hello World

Python can be used:

1. Desktop applications
2. Web applications
3. DJango is the framework to develop web apps
4. Database application
5. Networking applications
6. Games
7. Data Analysis
8. AI
9. IOT

Companies which are using pyhthon:

1. Google
2. Youtube
3. Dropbox
4. NASA

**Features**

1. Freeware and Opensource
2. High Level Programming language
3. Platform Independent
4. Portability
5. Dynamically Typed
6. Procedural as well as OOP
7. Extensible
8. Embedable
9. Extensive Libraries

Rich library support is there in Python

from random import \*

for i in range(10):

print(randint(0, 9))

**Limitations:**

1. Performance
2. Mobile Applications

**Identifiers**

1. Alphabet symbols (upper case/ lower case)
2. Digits(0-9) , but should not start with digits
3. \_
4. \_variable : private
5. \_\_variable : strongly private
6. \_\_name\_\_ : python specific, so not recommended

**Reserved Words**

1. 33

True, False, None

and, or, is, not

if, else, elif

while, for, break, continue, return, in, yield

try, except, finally, raise, assert

import, from, as, class, def, pass, global, nonlocal, lambda, del, with

**Data type:**

1. int
2. float
3. complex = 2+1j
4. bool
5. str
6. bytes
7. bytearray
8. range
9. list
10. tuple
11. set
12. frozenset
13. dict
14. none

NOTE: In python everything is an object.

**Forms:**

1. **Decimal Form(Base 10):** 0-9
2. **Binary(Base 2**): 0-1 :

* It should start with 0b or 0B , eg: 0b1111

0B1111 = 15

1. **Octal(Base 8):**

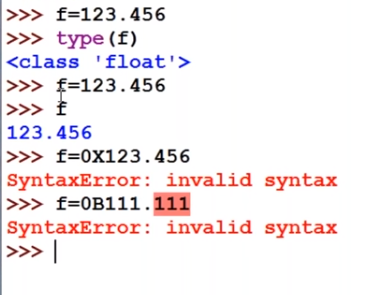
* It should start with 0o or 0O , eg: 0O1111

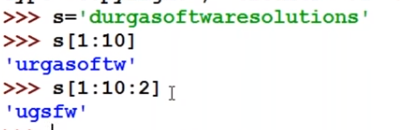
0O1111 = 585

1. **HexaDecimal(Base 16):** 0-9,a-f, A-F

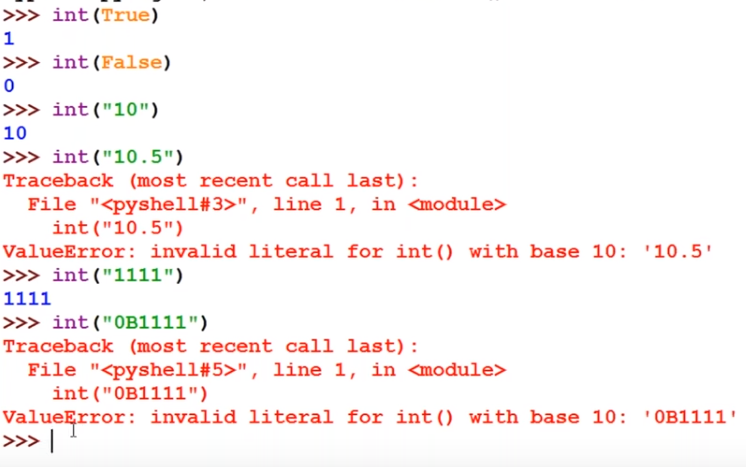
* It should start with 0x or 0X , eg: 0XBeef

0Xbeef = 48879





**Type Conversion/Coersion**



**Immutability**

>>> x=10

>>> x=20

Above, X will point to 20 now and 10 will be eligible for the **garbage collection**.

>>> x = 10

>>> y = 10

>>> z = 10

Here, Object 10 will be created and will be reused by the y and z.

Now if,

>>> z=78

Object 78 will be created and z will point towards 78. Hence 10 is not get deleted or eligible for garbage collection as object 10 is shared by other refrences.

>>> x=10

>>> y=10

>>> x is y

True

>>> x=257

>>> y=257

>>> x is y

False

Reusabilty of the object ranges up to 256, Hence the above is false.

>>> x=10.0

>>> y=10.9

>>> x is y

False

For floating point, complex number reusability is not valid.

For boolean and str case reusability concept will be all there.

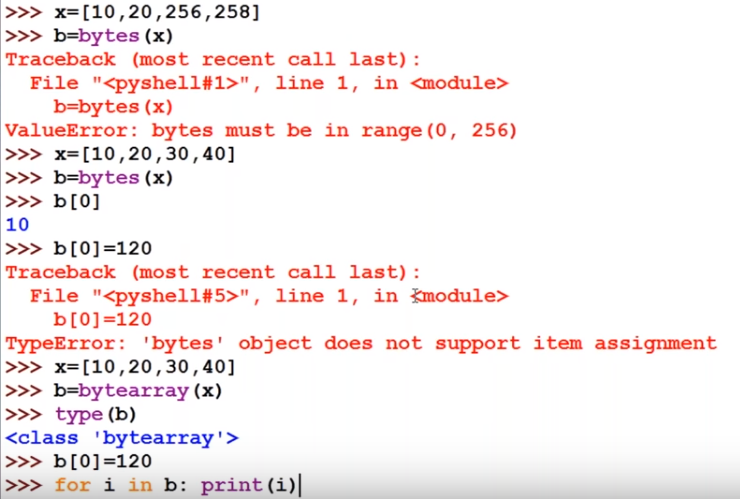
At the time of beginning i.e, when python interpreter starts, Python will create 0 to 256 integers. So when you will define integer between 0 to 256, it will reuse already created integer.

**bytes/bytearray**

bytes is immutable

bytearray is mutable

byte should be in the range 0 to 256



Above, when you do b[0] = 120, it will give error as bytes is muutable.

But when you do the same in the case of bytearray, it is possible.

**List**

>>> l = []

>>> type(l)

<class ‘list’>

>>> l.append(10)

>>> l.append(20)

>>> l.append(30)

>>> print(l)

[10, 20 30]

**Tuple**

Tuple is same as that of list but it is immutable.

t = (10, 20, 30)

>>> r = range(5)

>>> type(r)

>>> <class ‘range’>

>>> r

range(0,4)

>>> for i in r: print(i)

0

1

2

3

4

**Set**

Set is defined by curly ( {} ) brackets.

>>> s = {10, 20, 30, 10, 20, 30}

>>> s

{10, 20, 30}

>>> s[0]

Traceback (most recent call last):

File “<pyshell#3>”, line 1, in <module>

s[0]

TypeError: ‘set’ object does not support indexing

>>> s[1:]

Traceback (most recent call last):

File “<pyshell#4>”, line 1, in <module>

s[0]

TypeError: ‘set’ object is not subscriptable

Frozenset is immutable as the name suggests.

>>> s = {10, 20, 30, 10, 20, 30}

>>> fs = frozenset(s)

>>> type(fs)

<class ‘frozenset’>

>>> fs.add(50)

ERROR

**Dictionary**

>>> d = {100: ‘A’, 200: ‘B’}

>>> d1 = {}

>>> type(d1)

<class ‘dict’>

{}, is treated as empty dictionary and not empty set.

Dictionary is mutable.

NOTE: upto above 11 no video is a short desc.

**Operaters:**

+

-

\*

/

%

// : floor

\*\* :

**Ternary Operations**

>>> max = 10 if 3>2 else 5

10

>>> max = 10 if 3<2 else 5 if 4>2 else 1

5

**Special Operators**

1. Identity Operators
2. Membership operators

Identity Operators:

* is
* is not

Membership Operators:

list = [10, 20, 30]

>>> print(10 in list) // True

>>> print(70 not in list) // False

>>> s = “Hello World”

>>> print(“H” in s) // True

>>> print(“h” in s) // False due to case sensitive

>>> print(“ “ in s) // True

**Importing Libraries**

import math as m

print(m.sqrt(25))

from math import sqrt, pi

print(sqrt(25))

from math import \*

NOTES

>>> x = input(“Enter something”)

Enter something: hello

>>> type(x)

<class ‘str’>

>>> x = input(“Enter something”)

Enter something: 10

>>> type(x)

<class ‘str’>

No matter you type string or integer, it will be treated as str in python. You have to type cast it.

**Multiple Input**

Step 1: input(“Enter two numbers”)

Step 2: input(“Enter two numbers”).split() // by default split by space “ “

Step 3: for x in input(“Enter two numbers”).split() // for each x in

Step 4: int(x) for x in input(“Enter two numbers”).split() // typecase in int

Step 4: [int(x) for x in input(“Enter two numbers”).split()] //in list now usng [] brackets

a, b = [int(x) for x in input(“Enter two numbers”).split()]

Q. Read 2 float values, separate by , and print sum

>>> a,b = [float(x) for x in float(“Enter two Float numbers”).split(“,”)]

>>> print(“SUM IS: ” + (a + b))

**eval()**

>>> x = eval(“10 + 20 + 30”)

print(x)

>>> ex = input(“Enter some expression”) // 20+30

>>> result = eval(ex)

>>> print(result) // 50

How to read input from the keyboard.?

Python 2

* raw\_input()
* input()

Python 3

* input()

Whenever we use input(), as default it is considered as str.

So, we need to perform type casting.

**CommandLine argument**

We need argv from sys if we want to pass commandline arguments in python

from sys import argv

print(argv)

>>> py test.py 10 20 30 40

Output

[test.py 10 20 30 40]

As we can see output we get as a list.

Q. Read group of int through cmd and print sum of it

from sys import argv

args = argv[1:]

sum= 0

for x in args:

n = int(x)

sum +=n

print(sum)

If you want your command line argumants should be treated as single entity, you have to use “”

>>> pt test.py “my name is tanuj”

from sys import argv

print(argv[1])

Output

my name is tanuj

If you want to access the element with its index which is not present, you will get an error.

from sys import argv

print(argv[7])

>>> pt test.py 1 2 3 4

IndexError: list index out of range

However, you will get empty list in the below case.

from sys import argv

print(argv[7:1000])

>>> pt test.py 1 2 3 4

[]

Key things to remember

When we use + operator

+ 🡺 both arguments should be same

\* 🡺 one of the argument should be int

print(“tanuj” + ”tripathi”) 🡺 tanujtripathi

print(“tanuj”, ”tripathi”) 🡺 tanuj tripathi

a,b,c = 10, 20, 30

print(a, b, c) 🡺 10 20 30

print(a, b, c, sep=’,’) 🡺 10, 20, 30

print(a, b, c, sep=’:’) 🡺 10: 20: 30

print(“Hello”, end=’ ’)

print(“How”)

print(“are”)

🡺Hello How

are

**Formatting**

print(“one number is %i and second number is %i and my name is %s ” %(1, 2, “tanuj”))

print(“one number is {} and second number is {} and my name is {}”.format(1, 2, “tanuj”))

print(“one number is {x} and second number is {y} and my name is {z} ”

.format(x = 1, y = 2, z = “tanuj”))

**Slice examples**

>>> S = “0123456789”

>>> s[0:7:1]

‘0123456’

>>> s[0:7:2]

‘0246’

>>> s[0:7]

‘0123456’

>>> s[0:]

‘0123456789’

>>> s[::]

‘0123456789’

>>> s[::-1]

‘9876543210’