Python

Clase 2

Father of Python

- Guido Van Rossam

DOB - Feb 20th 1991

How Python name selected for language ?

BBC fun show

The complete Monty Python Circus

Python: - Functional programming language. (C)

- Object Oriented (C++)
- Scripting language (Perl + shell Script)

Why we should learn & use Python?

1. Simple and easy to learn.

ex:

```
for i in range (10):
    print (i)
```

↳ exactly like english language.

↳ Best for Non Technical people.

in Java:

```
for (int i=0 ; i<10; i++)
{
    System.out.println(i);
}
```

In Java 53 Keywords.

In Python 30 Keywords.

2. Free ware and Open Source:

Free completely

Can view source code

If internal application is not good → we can modify the source code.

3. High level programming language:

→ Easily understandable, like english language.

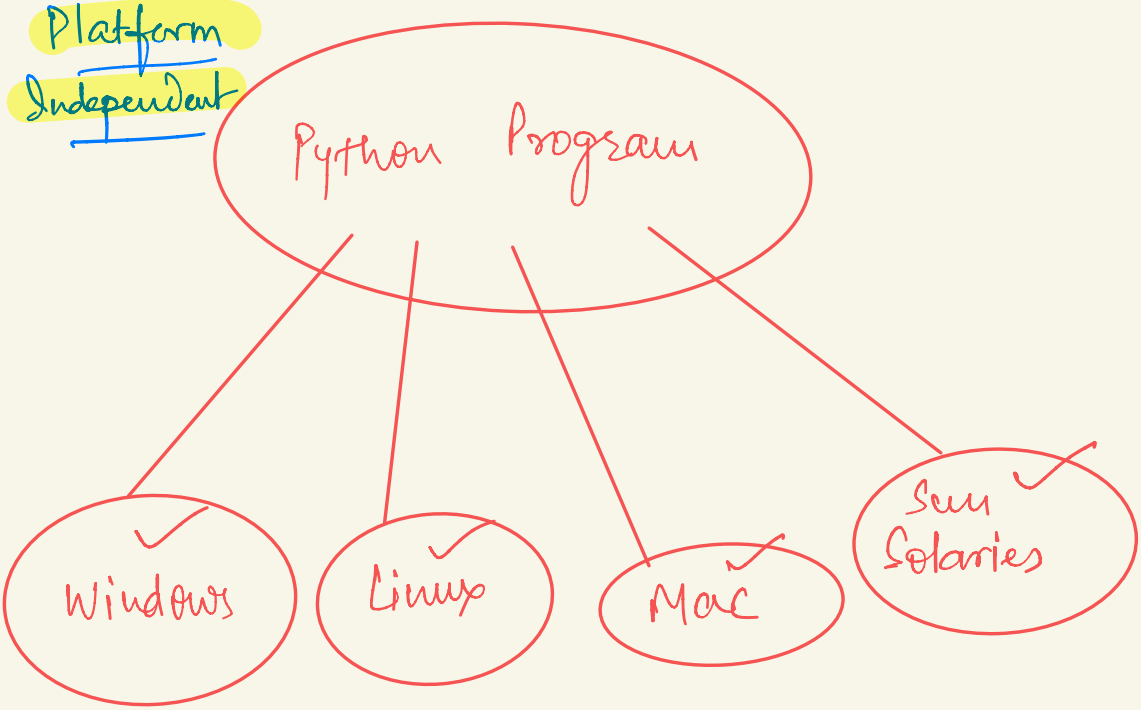
4. Python is Platform independent.

→ Write Once & Use anywhere.

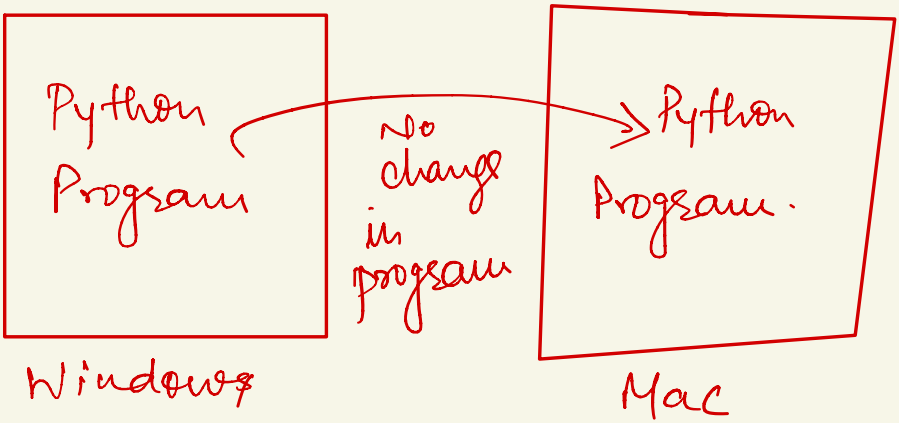
5. Portability

→ Moving Python program from one machine to other is very easy without changing the code.

Platform Independent



Portability:



6. Dynamically Typed programming language:

→ No Variable type declaration is required.

```
>>> x=10
    print(type(x)) ↵
>>> INT.
```

```
>>> x=10.5
    print(type(x)) ↵
>>> float.
```

→ Based on value provided, variable type is provided by Python.

7. Python is both Procedure Oriented and Object Oriented language —

```
x=10
def f1():
print(" Don't disturb ....", x)
f1()
```

→ No class → We called function f1() → procedure based.

8. Interpreted Programming language

→ No compilation is required.

like javac.

9. Extensible Programming language:

Already existing other languages code can be run in python.

10. **Embedded:**

We can use Python programs in other languages.

→ Reverse of Extensible.

11. **Extensive Libraries:**

→ Already developed libraries are available in python.

```
import math
print(math.sqrt(4))
```

```
from random import *
print(randint(0,9))
```

Six digit OTP random —

```
from random import *
print (randint(0,9), randint(0,9), randint(0,9),
       randint(0,9), randint(0,9), randint(0,9),
       sep = '')
```

→ Six digit random OTP will be generated.

Limitation of Python:

1. Performance
2. Not much penetration in mobile app development.

Class 3

Flavours of Python :

- Jython or JPython (Java appl^ns)
- Iron Python
- PyPy → PVM → JIT (Just In Time Compiler)
- Ruby Python (Ruby appl^ns)
- AnacondaPython (Big Data)
- Stackless (Python for Concurrency)

Python versions :

Python 1.0 introduced in Jan 1994

Python 2.0 " " Oct 2000

Python 3.0 " " Dec 2008.

Current version:

Python 3.6 ⟶ 2016.

- No Backward support / Compatibility in Python.

(Prog. written in version 2 donot run in version 3 as version 3 is developed differently to have several new features).

Identifiers:

Just as we use names to identify people in real world, names are used to identify keywords in python. These names are called identifiers.

```
x=10          ---> variable
def f1():     }
    pass      }   function
```

class Test (Exception) → class.

- It may be variable name, function name, class name or method name.

Rules to define identifiers in Python

1. alphabet symbols (both upper case and lower case)
   digits (0 to 9)
   underscore (_)

ex - cash = 10 ↵ → valid

ca$h = 10 ↵ → Invalid

2. total123 = 10 ↵ → valid

123total = 10 ↵ → Invalid.

⇒ identifier should not start with number / digits.

3. Identifiers are case sensitive

total = 10 ↵ valid

TOTAL = 20 ↵ valid.

4. Predefined keywords are not allowed as variables.

if = 10 ↵ → Invalid.

5. Max^m length allowed for identifier in Python is no length.

CLASS 4

Reserved Words: Keywords.

The words reserved to represent specific entity are called Reserved Words.

like apple is reserved in english to represent a specific fruit.

Python - 33 reserved Words.

- True, False, None
- and, or, not, is
- 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.

Note: - All reserved words contain only alphabet symbols.

- Except first three, all other keywords are in lower case.

```
>>> import keyword
    keyword.kwlist ↵
```

All keywords will be shown.

Data Type in Python:

Python → Dynamically typed programming language.

⇒ Based on a value provided to a variable, the data type will be automatically provided by python.

- int (10, 20, 100 etc.)
- float (10.50, 11.12, 100.99 etc)
- complex (10 + 2i → Scientific calculations)
- bool
- str
- bytes
- bytearray

- range
- list
- tuple
- set
- frozenset
- dict
- None.

Python provides some inbuilt functions:

print()
type()
id()

1. int :

integral values

10, 20, 30, 100.

Note: long data type is available in Python 2 but not in Python 3.

int values can be represented as

1. Decimal form (Base 10)

   0 to 9

   ex- a = 7879

2. Binary (Base 2)

   0 and 1

   ex- a = 1111 (Base 10 by default)

for binary

a = 0b1111 / 0B1111

Starts with zero b / zeroB

3. Octal form: (Base 8)

0 to 7

eg - a = 0o777 / 00777.

zero small O

4. Hexadecimal form (Base 16)

0 to 9, a to f / A to F

(Here python is not case sensitive)

a = 0xface ↵ / 0Xface

64206.

Note: Python output is always Decimal form.

CLASS 5

Base Conversions:

bin()

oct()

hex()

Decimal to binary

```
x = 10
bin(x) ↵
```

- Similarly any base to binary bin() is used
- Any base to Octal oct() is used
- " " Hexadecimal hex()

float datatype :

in python, double type is not there.

ex- salary = 7025 6.99

Note: float can only be represented in decimal form.

f = 1.2e3
↳ exponential
↳ valid in Python

```
>>> f=1.2e3 ↵
>>> f ↵
>>> 1200.0
```

complex data type :-

format for complex no :-

$a + bj$

a = real part

b = imaginary part

$j^2 = -1$

$\Rightarrow j \wedge 2 = -1$

$\Rightarrow j = \sqrt{-1}$

$\Rightarrow$ Python provides support for complex number.

```
a = 10 + 20j
a.real ↵
a.imag ↵
```

bool :

True and False.

```
>>> a = True
>>> type(a) ↵
>>> True
```

```
>>> a = 10 ↵
>>> b = 20
>>> c = 10 < 20 ↵
>>> True.
```

True = 1
False = 0
} at python memory.

str: String data type —

s = 'amit' / "amit" ↵

s ↵

amit

type(s) ↵

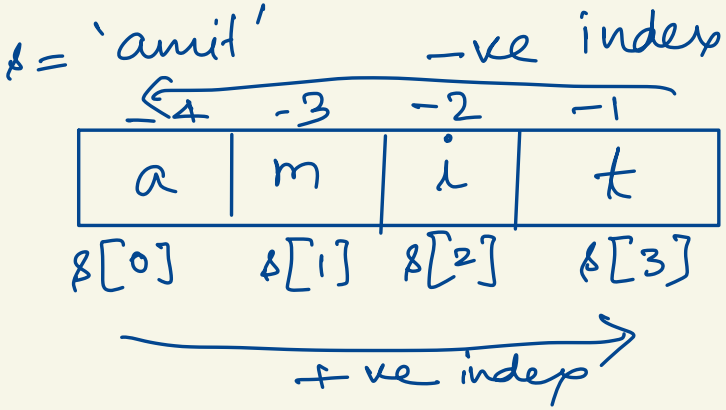
'' / " " → Single line string.

''' ''' → Multi line string.

CLASS 6

Slice Operator:

- means cutting a string into parts.
- substring.




```
>>> s[0] ↵
'a'
```

```
>>> s[-1] ↵
't'
```

Syntax of slice :

s = 'amit'

s[begin : end]
  index index

>>> s[1:3] ↵

'mi'

⇒ returns substring from begin index to end -1 index.

s[1:] → Begin to end

s[:3] → Begin to end-1

s[:] → Total string.

s[-1:-3] → '' Blank O/P

s[-3:-1] → 'mit'

int
float
complex
bool
str

} Python's fundamental data types.

Type Casting:

Converting one type to another also called type coersion.

int() → Any other type to int type
float() → " " " " float "
complex()
bool()
str()

CLASS 7

int():

int (10.123) ==> 10

int (10+20j) ==> TypeError

int (True) ==> 1

int (False) ==> 0

int ("10") ==> 10

int ("10.5") ==> ValueError

float():

float(10) => 10.0
float (10+5j) => TypeError
float (True) => 1.0
float ("10") => 10.0
float ("10.5") => 10.5
float ("ten") => ValueError.
float ("0B1111") => " "

complex():

Other data types to complex type.

Form-1 : complex(x) ==> x+0j

Form-2 : complex(x,y) ==> x+yj

Form-1 : complex(x) ==> x+0j

```
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") => ValueError
complex("0B1111") => ValueError.
```

Form-2 : complex(x,y) ==> x+yj

complex(10,20) ⇒ (10+20j)

complex(True,False) ⇒ (1+0j)

complex(10,20.5) ⇒ (10+20.5j)

complex("10","12") ⇒ TypeError.

---

bool():

bool(0) ⇒ False

bool(10) ⇒ True (Non Zero means True in Python)

bool(0+0j) ⇒ False

bool(10+20j) ⇒ False

bool('') ⇒ False

bool("amit") ⇒ True.

str() : Any Data type → String type.

str(10.5) ==> '10.5'

str(10+20j) ==> '10+20j'

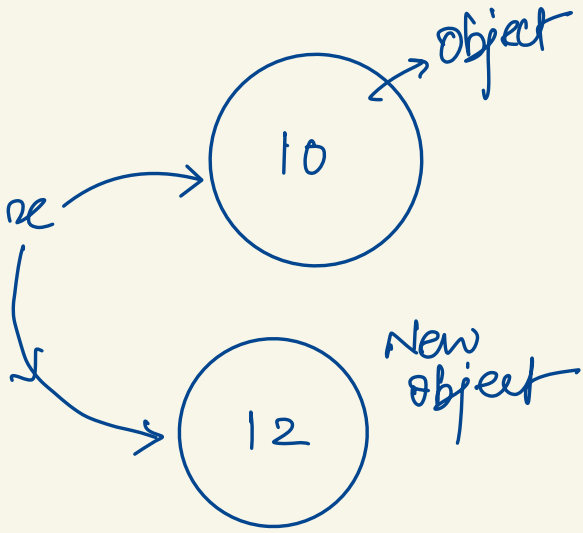
Immutable : Unchangeable

Immutable vs Fundamental Data Type:

All fundamental data types are immutable.

x = 10

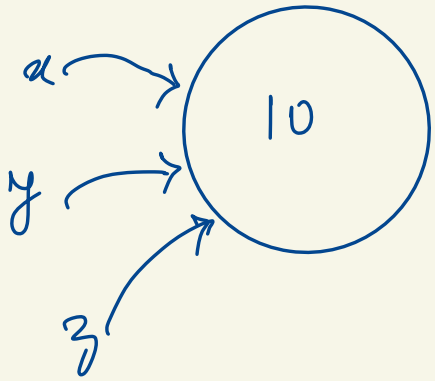
x = 12



x = 10

y = 10

z = 10



⇒ No new object will be created for same value

id(x) ↵ => - - - -

id(y) ↵ => - - - - -

id(z) ↵ => - - - - -

if value of one variable is changed, new object will be created

z = 12 ⇒

id(z) ⇒ - - - - - only id of z will be changed.

To check whether two variables are pointing to same Object

is

x is y => True

y is x => True

=> memory address of x & y is same.

available only in the following ranges:

int => 0 to 256

bool => always

str => always

float, complex => Always another object will be created.

Class 8

Various Type Casting functions in Python:

float value ⇒ int value == int()

bool " ⇒ int " == int()

int value ⇒ str " == str()

complex " ⇒ int " == ~~int()~~ not possible

Complex Nos can't be converted into int / float type.

any type ⇒ str type ⇒ str()

---

bytes data type: To represent a group of byte numbers just like an array

```
>>> x=[10,20,30,40]
>>> b=bytes(x)
>>> type(b)↵ ⇒ 'bytes'
>>> b[0]
10
>>> b[1]
20
```

To get all entries in bytes

```
>>> for x in b: print(x)↵
10
20
30
40
```

Conclusion:

1. In bytes data type every value must be in the range of 0 to 256.

```
>>> x=[10,20,256,258]
>>> b=bytes(x) ↵
```

ValueError

2. Bytes data type is immutable

```
>>> x=[10,20,30,40]
>>> b=bytes(x)
>>> b[0] ↵
10
>>> b[0]=120.
TypeError: ....
```

bytearray data type:

Only diff. between bytes data type & bytearray data type is

→ bytearray data type is mutable.

```
>>> x=[10,20,30,40]
>>> b = bytearray(x)
>>> type(b)
class 'bytearray'
>>> b[0] = 120
>>> for i in b: print(i)
120
20
30
40
```

but compulsory between 0 to 256 range.

list datatype: If we want to represent a group of values as a single entity where insertion order is preserved and duplicates are allowed.

```
>>> l=[]
>>> type(l)
 class 'list'

>>> l.append(10)
>>> l.append(20)
 "  "     "  (30)
 "  "     "  (10)
>>> print(l)
[10,20,30,10]
>>> l.append('Amit')
```

[10, 20, 30, 10, 'Amit']

```
>>> l[1:5]
[20, 30, 10, 'Amit']
```

Conclusion: list is

- contains order.
- duplicates allowed
- heterogeneous (diff. data types)
- growable.
- values should be enclosed with [].

CLASS 9

Tuple data type: Same as list but immutable.

representation:

list

```
l = [10, 20, 30, 40]
```

tuple

```
t = (10, 20, 30, 40)
```

tuple is declared in ( ) and list is " " [ ]

```
>>> t[0] = 100 ↵
```

TypeError -
as tuple is immutable and data cannot be assigned.

Range data type:

range ():

→ represent a sequence of values

→ immutable.

Form-1 : range (10)

it represents values from 0 to 9.

⇒ 0 to end-1

```
>>> r = range (10)
>>> type (r) ↵
'range'
>>> r ↵
range (0, 10)
>>> for i in r : print(i) ↵
0
1
2
3
4
5
6
7
8
9
```

to represent numbers from 10 to 29

form-2 : range (10, 30)

---

step → increment by a specific number

form-3 : range (10, 30, 2)

10, 12, 14, ... 28

```
>>> for i in range (10,50,5) : print(i)
10
15
20
25
30
35
40
45
```

set data type:

If we don't want duplicate and order to be preserved, instead of list, we can go for set data type.

→ set is represented in { }

list	set
- Duplicates	- no duplicates
- order preserved	- no order preserved.
- ( )	- { }

```
>>> s = {10,20,30,10,20,30}
>>> s ↵
{10,20,30} ⇒ No duplicates.
```

set is mutable.

s.add ('Amit')

s.remove (10) ↵

But indexing and slicing operations are not allowed in set.

frozenset data type:

same as set but no change in data is allowed.

```
>>> s={10,20,30,40}
>>> fs = frozenset(s)
>>> type(fs) ↵
    frozenset
>>> fs ↵
frozenset({40,10,20,30})
```

add and remove not allowed.

dict data type:

dictionary.

all other data types represent individual objects.

sometimes, we want to represent a group of objects as

key - value pairs.

we should use dict data type.

Key	Value
100	Anil
200	Sunil
300	shashi

representation {}

```
>>> d = {100: 'Amit', 200: 'Sumit',
        300: 'Shashi'}
>>> type(d) ↵
    dict
```

{} → open {} by default represent dictionary.

for representing set we have to use set() set function.

```
>>> d1[100] = 'Sunny'
>>> d1 ↵
  {100: 'Sunny'}
>>> d1[200] = 'Bunny' ↵
>>> d1
{100: 'Sunny', 200: 'Bunny'}.
```

CLASS 10

In Python:

- Fundamental Data types
  - int, float, complex, bool, str
- bytes and bytearray
  - ↳ to represent Binary data like images, video files and audio
- long data type
  - Python 2 → available
  - Python 3 → Not available
- In python char data type is not there
  - char can be represented by str only.
  - ' ' " " ''' '''

Summary of Python data types.

1. **int:**

   To represent whole numbers
   integral numbers

   - immutable.

2. **float:**

   To represent decimal point numbers.

   - immutable.

3. **complex:**

   $a + bj$

   To represent complex numbers.

   - immutable.

4. bool:

To represent boolean.

allowed values

true → 1

false → 0

- immutable.

5. str:

Most commonly used data type.

To represent a sequence of characters.

- single quote
- double quote
- Triple quote.

- immutable.

6. **bytes:**

To represent group of byte values in the range 0 to 256.

representation [ ].

— immutable

7. **bytearray:**

Same as bytes but mutable.

8. **range:**

To represent a range of values.

0 to 9.

range (10)

10 to 29

range (10, 30)

10 to 29, increment by 5

range (10, 30, 5) immutable.

9. **list:**

To represent a group of objects as a single entity.

insertion order preserved
duplicates allowed
heterogeneous objects
growable
[]
mutable.

l = [10, 20, 30, 40].

10. **Tuple:**

Same as list but immutable.

()
t = (10, 20, 30, 40)

11. **set**:

$s = \{1, 2, 3, 4\}$

- No duplicates.
- order not applicable.
- index concept not applicable
- slicing " " ".
- mutable.

Empty set

$s = set()$

12. **frozenset**:

$s = \{10, 20, 30, 40\}$

$fs = frozenset(s)$

everything is frozen ⇒ immutable.

13. dict:

Key-value pair

d={101:'Amit'}

14. None Data Type:

To represent null value

```
>>> def f1():
        a=10
>>> def f2("Hello Don't be like None")
>>> f2() ↵
"Hello . . . . . "
>>> print(f1()) ↵
None.
```

Escape characters:

\n

\t

\r

\'

\"

\v

Constants:

constant concept not applicable in python.