Python

Introduction

----

1) Python is a general purpose high level programming language
2) Machine language (0's and 1's) and Assembly language (Pseudo codes like 8085 , 8086)
   are lower level languages
3) C and C++ are middle level languages
4) Java and Python are higher level languages
5) 'Guido van Rossam� has designed python in 1989 but released in the market in 1991
6) Python is derived from 'c' and ABC languages
7) Pyhton has some of the features of 'c' language and 'ABC' language

What does Python support

----

1) Python is a functional programming language (like 'C' language)
   i.e. Python supports functions

```
Eg: def f1():
             statements
       def f2():
             statements
       def f3():
statements
```

2) Python is an Object oriented language (like C++ and Java)
   i.e. Python supports classes and objects

```
Eg: class c1:
        def m1(self):
         statements
        def m2(self):
         statements
       # End of the class
a = c1() ---> 'a' is c1 class object
```

3) Python is a scripting language (like Perl and Shell script)
   It is possible to write a python program without using functions and classes

```
Eg: stmt1
        stmt2
        stmt3
 and so on
```

4) Python is a modular programming language (like Modula 3)

In other words, pyhton is an allrounder

Where is Python used

Python can be used to design:

1) Desktop applications (Stand alone applications) like Notepad , Calculator , Paint Brush , ...

2) Web applications
3) Database applications becoz python supports PDBC(Python Data Base Connectivity)
4) Networking applications
5) Games
*6) Data science applications
*7) Machine learning applications
*8) Artificial Intelligence
*9) IOT(Internet Of Things) applictions

Features of Python
----
1) Simple and Easy to learn
2) Free Ware(i.e. Free of cost) and
   Open Source(i.e. We can see source code behind Python and modify it)
3) High Level programming language
4) Platform independent language (like Java)
   i.e. Compile anywhere ('X' O.S. and 'Y' Processor) and run any where else
   This is in contrast to C and C++ where program must be executed on the same system
   where it is compiled
5) Portability
   i.e. Migrate from one system to another system without making any major changes
*6) Dynamically typed language
   i.e. Objects can be used without any prior declaration
   Eg: x = 25 ---> since 25 is int, 'x' is automatically int (Don't write int x)
         print(type(x)) ---> <class 'int'>
7) Procedure oriented and object oriented (like C++)
   i.e. Python program can be designed with and without class (like C++)
   Java program must contain class but it is not mandatory in Python
8) Interpreter language
   i.e. Line by line translation and execution
   stmt1 ---> Translated and executed
   stmt2 ---> Translated and executed
   stmt3 ---> Translated and executed
   .....
   Translation and Execution are alternate
   (C and C++ are compiler languages,
   Python is interpreter language and
   Java is both compiler and interpreter language)
9) Extensible
   i.e. Pyhton program can use other language functions and methods and
   thereby performance of the application is improved
10) Embedded
   i.e. Python code can be used in other language programs
11) In other words, extensible and embedded are quite opposite
12) Extensive library
   There are too many libraries(predefined functions and classes) in python

Limitations

----

1) Performance of python program is low becoz python is an interpreter language
   i.e. Python program is translated every time program is executed
   Therefore Python program execution is slow due to repeated translation
   This is in contrast to C and C++ where program execution is fast becoz they are compiler languages
2) Python is not suitable for mobile applications

Different Flavors of Python

Since Python is open source, There are several flavors of python

1) Cpython(Original Python)
2) Jython (or) JPython
3) IronPython
4) PyPy
5) RubyPython
6) Anaconda Python
7) Stackless

Python Versions

----

1) Python 1(Designed in 1994)
2) Python 2(Designed in 2000)
3) Python 3(Desgined in 2008)
   a) Python 3.8 (2019)
   b) Python 3.9 (2020)
   c) Python 3.10(2021)
   d) Python 3.11(2023)

Python Objects

There are 11 objects in Python

1) int
2) float
3) complex
4) bool
5) NoneType
      The above 5 are called non-sequences
6) str
7) range
8) list
9) tuple
10) set
11) dict
      The above 6 are called sequences

1) What is a sequence ? ---> A group of elements
2) What is a non-sequence ? ---> Single element

3) Does python support char, long and double objects ? ---> No

4) What is another name of sequence ? ---> Iterable (or) collection

Note:

1) What does 'c' language support ? ---> Variable , pointer , array , structure and union

2) What does java support ? ---> Variable and object

3) What does python support ? ---> Only object

4) In other words, python does not support variable , pointer , array , structure and union

```
# int object demo program
a = 25 # Ref 'a' points to object 25
print(a) # Value of object 'a' i.e. 25
print(type(a)) # <class 'int'>
print(id(a)) # Address of object 'a' (may be 1000)
a = 99999999999999999999999999999999999999999999999999999999999999999999999999999999999 # Valid
print(b) # 99999999999999999999999999999999999999999999999999999999999999999999999999999999999
#c = 75$ # Error
'''
1) What does a = 25 do ? ---> Assigns reference 'a' to int object 25
2) a = 25
   What is 25 called ? ---> Object i.e. int class object
   What is 'a' called ? ---> Reference
3) a = 25
   What does object 'a' contain ? ---> Integer number 25
   What does reference contain ? ---> id of the object (i.e. Address)
4) What is the name of object ? ---> Reference name is nothing but object name
    If reference name is 'a' , what is the object name ? ---> 'a' itself
5) int x = 25
   Is the above statement valid ? ---> No becoz there is no declaration in python
   Is x = 25 valid ? ---> Yes
6) What does print(object) do ? ---> Prints content (or) value of the object
   What does type(object) do ? ---> Returns type of the object
   What does id(object) do ? ---> Returns address of the object
7) What is the extension to python program file ? ---> .py
8) How to run a python program ? ---> py filename . py (or) python filename . py
9) There is no compilation in python becoz it is an interpreter language
10) In other words, run python program directly without any prior compilation
'''
```

str object
----

1) What can a str object hold ? --->
String
2) What is a string ? --->
A group of characters in single , double (or) triple quotes
3) Is 'Rama Rao' a string ? --->
Yes due to single quotes
   Is "9247" a string ? --->
   Yes due to double quotes
   Is '''+-$''' a string ? --->
   Yes due to triple quotes
   Is """A2#""" a string ? --->
   Yes due to triple double quotes
4) What is another name of string ? --->

Alphanumeric becoz string can have alphabets , digits and special characters

5) Which quotes are used for multi-line string ? --->

Triple quotes only

Which quotes are used for single-line string ? --->

Single , double (or) triple quotes

6) Can single (or) double quotes be used for multi-line string ? --->

No

7) Is string a sequence ? --->

Yes becoz it is a group of characters

8) Can str object be modified ? --->

No becoz it is an immutable object

9) Is str object indexed ? --->

Yes

10) What are the indexes of characters from left to right ? --->

0 , 1 , 2 , ... length - 1

What are the indexes of characters from right to left ? --->

-1 , -2 , -3 , ... -length

11) What is the index of 10th character from left to right ? --->

9

What is the index of 10th character from right to left ? --->

-10

12) What is the result of 'Hyd'[0] ? --->

Character at index 0 i.e. 'H'

What is the result of 'Hyd'[1] ? --->

Character at index 1 i.e. 'y'

What is the result of 'Hyd'[2] ? --->

Character at index 2 i.e. 'd'

13) What is the result of 'Hyd'[-1] ? --->

Character at index -1 i.e. 'd'

What is the result of 'Hyd'[-2] ? --->

Character at index -2 i.e. 'y'

What is the result of 'Hyd'[-3] ? --->

Character at index -3 i.e. 'H'

14) What is the advantage of indexes ? --->

Random access

15) What is random access ? --->

It is possible to access 10th character of the string directly without accessing first nine characters

16) Can str object be repeated ? --->

Yes with * operator

What does 'Hyd' * 3 do ? --->

Repeats 'Hyd' thrice

i.e. 'HydHydHyd'

17) What does len('Hyd') do ? --->

Returns number of characters in 'Hyd' i.e. 3

Note:

1) Are non-sequences indexed ? --->

No due to single element

2) Why are sequences indexed ? --->

Since they have got a group of elements

3) Are non-sequences immutable (or) mutable ? --->

Immutable objects

4) What are the three mutable objects in python ? --->

List , set and dictionary

5) Every object in python is immutable except the above three

6) Does python string end with '\0' ? --->

No (unlike 'c' string)

```
# Index demo program (Home work)
a = 'Hyd'
print(a[0]) # How to print 'H' of object 'a' ---> H
print(a[1]) # How to print 'y' of object 'a' ---> y
print(a[2]) # How to print 'd' of object 'a' ---> d
#print(a[3]) # Error becoz there is no index 3 in 'Hyd'
print(a[-1]) # How to print 'd' of object 'a' with -ve index ---> d
print(a[-2]) # How to print 'y' of object 'a' with -ve index ---> y
print(a[-3])# How to print 'H' of object 'a' with -ve index ---> H
#print(a[-4]) # Error becoz there is no index -4 in 'Hyd'
print(a[0] == a[-3]) # 'H' == 'H' is True
#a[2] = 'c' # Error becoz str object is immutable
#print(25[0]) # Error becoz non-sequence (such as int) is not indexed
print('25'[0]) # Char at index 0 i.e. '2'
#print(True[1]) # Error becoz non-sequence (such as bool) is not indexed
print('True'[1]) # Char at index 1 i.e. 'r'
'''
# Find outputs (Home work)
a = 'Hyd'
print(a * 3) # Repeat object 'a' thrice i.e. HydHydHyd
print(a * 2) # HydHyd
print(a * 1) # Hyd
print(a * 0) # Empty string
print(a * -1) # Empty string
print(25 * 3) # 75
print('25' * 3) # 252525
#print('25' * 4.0)# Error due to float operand 4.0
print(3 * 'Hyd') # HydHydHyd
print('25' * True) # 25




'''
1) What does non-sequence * integer do ? ---> Multiplication
   What does sequence * integer do ? ---> Repetition

2) Is * operator overloaded ? ---> Yes becoz * operator does both multiplication and repetition

3) Are 'Hyd' * 3 and 3 * 'Hyd' same ? ---> Yes
'''1) What is another name of index ? ---> Subscript
2) What does == operator do ? ---> Compares objects
   What does = operator do ? ---> Assigns refernce to an object
'''
```

```python
# Find outputs (Home work)
a = 'Hyd'
print(a * 3) # Repeat object 'a' thrice i.e. HydHydHyd
print(a * 2) # HydHyd
print(a * 1) # Hyd
print(a * 0) # Empty string
print(a * -1) # Empty string
print(25 * 3) # 75
print('25' * 3) # 252525
#print('25' * 4.0)# Error due to float operand 4.0
print(3 * 'Hyd') # HydHydHyd
print('25' * True) # 25
'''
1) What does non-sequence * integer do ? ---> Multiplication
   What does sequence * integer do ? ---> Repetition
2) Is * operator overloaded ? ---> Yes becoz * operator does both multiplication and repetition
3) Are 'Hyd' * 3 and 3 * 'Hyd' same ? ---> Yes
'''
# len() function (Home work)
print(len('Hyd')) # 3
print(len('Rama Rao')) # 8
print(len('9247')) # 4
print(len('')) # 0 due to empty strrring
print(len(' ')) # 1 due to space
#print(len(689)) # Error becoz 689 is not a sequence
'''
len() function
-----------------
1) What does len(string) do ? ---> Returns number of characters in the string
2) What is the argument of len() function ? ---> Any sequence such as string
3) Is len(non-sequence) valid ? ---> No
'''
```

```
# Find outputs (Home work)
a = """"Hyd""" # Excess opening quote is char of the string
print(a) # "Hyd
print(len(a)) # 4
print(a[0]) # "
#print("""Hyd"""") # Error due to excess closing quotes
b = """""Hyd""" # Excess opening quotes are chars of the string
print(b) # ""Hyd
print(len(b)) # 5
'''
1) What happens to excess opening quotes in the string ? ---> They are treated as characters of the string
2) What happens to excess closing quotes in the string ? ---> Throws error
'''
Slice
------
1) What is obtained when string is sliced ? --->
Substring
2) What is the syntax of slice ? --->
string[begin : end : step]
3) string[begin : end : 0]
   Is the above statement valid ? --->
   No becoz step cannot be 0
4) In other words, step can be positive (or) negative but not 0
5) What is the result of string[x : y : z] ? --->
String from indexes x to y - 1 in steps of z
    What is the result of string[x : y : -z] ? --->
    String from indexes x to y + 1 in steps of -z
6) string[begin : end]
   What is the default step ? --->
   1
7) string[ : : +ve step]
   What is the default begin ? --->
   0 becoz index of 1st character is 0
   What is the default end ? --->
   String length becoz index of last char is length - 1
8) string[ : : -ve step]
   What is the default begin ? --->
   -1 becoz index of first character is -1 from right to left
   What is the default end ? --->
   -string length - 1 becoz index of last char is -length
```

```
# Find outputs
a = 'Sankar Dayal Sarma'
print(a[7 : 12]) # a[7 : 12 : 1] ---> string from indexes 7 to 11 in steps of 1 ---> Dayal
print(a[7 : ]) # a[7 : 18 : 1] ---> string from indexes 7 to 17 in steps of 1 ---> Dayal Sarma
print(a[ : 6]) # a[0 : 6 : 1] ---> string from indexes 0 to 5 in steps of 1 ---> Sankar
print(a[ : ]) # a[ 0 : 18 : 1] ---> string from indexes 0 to 17 in steps of 1 ---> Sankar Dayal Sarma
print(a[: : ]) # a[ 0 : 18 : 1] ---> string from indexes 0 to 17 in steps of 1 ---> Sankar Dayal Sarma
print(a[1 : 10 : 2]) # string from indexes 1 to 9 in steps of 2 ---> akrDy
print(a[0 : : 2]) # a[ 0 : 18 : 2] ---> string from indexes 0 to 17 in steps of 2 ---> Sna<space>aa<space>am
print(a[1 : : 2]) # a[1 : 18 : 2] ---> string from indexes 1 to 17 in steps of 2 ---> akrDylSra
print(a[-5 : -1]) # a[ -5 : -1 : 1] ---> string from indexes -5 to -2 in steps of 1 ---> Sarm
print(a[::-1]) # a[-1 : -19 : -1] ---> string from indexes 1- to 18 in steps of -1 ---> Reverse string
print(a[-1:-5:-1]) # string from indexes -1 to -4 in steps of -1 ---> amra
print(a[ : : -2]) # a[ -1 : -19 : -2] ---> string from indexes -1 to -18 in steps of -2 ---> arSlyDrka
print(a[3 : -3]) # a[ 3 : -3 : 1] --->  string from indexes 3 to -4 in steps of 1 ---> kar<space>Dayal<space>Sa
print(a[2 : -5]) # a[2 : -5 : 1] ---> string from indexes 2 to -6 in steps of 1 ---> nkar<space>Dayal<space>
print(a[-1:-5]) # a[-1 : -5 : 1] --->
 Empty string becoz -1 >= -5
print(a[3 : 3]) # a[3 : 3 : 1] --->
 Empty string becoz 3 >= 3
# 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
# S a n k a r D a y a l S a r m a
# -18 -17 -16 -15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
'''
1) string[x : y: +ve step]
   When is the result empty string ? --->
 When x >= y
2) string[x : y: -ve step]
   When is the result empty string ? --->
When x <= y
'''
# Find outputs (Home work)
a = 'A'
#print(a[1]) # Error becoz there is ni index 1 in 'A'
print(a[1:]) # a[1 : 1 : 1 ] ---> empty string becz 1 >= 1
'''
Indexing throws error when the index is invalid but
slice never throws error even when indexes are invalid and result is empty string when indexes are invalid
'''
```

Comments
----
1) How to write a single line comment ? ---> With # operator
2) How to write a multi-line comment ? ---> '''

```
              Line 1
              Line 2
              Line 3
         '''
```

3) Are comments executed ? ---> No and they are ignored
4) What is the advantage of comments ? ---> More clarity and better readability

Type-casting (or) Type-coersion functions
----
1) What is typecasting ? ---> Conversion of an object to a different class object
2) Conversion of int object to float object ,
   float object to int object ,
   int object to string object and so on is called typecasting
3) What are the different typecasting functions ? ---> int() , float() , complex() , bool() , str() , bin() , oct() , hex()

complex() function
----
1) What does complex(3 , 4) do ? ---> Returns 3 + 4j
2) What does complex(3.8) do ? ---> Returns 3.8 + 0j
3) What does complex('9.5') do ? ---> Returns 9.5 + 0j
4) Is complex(3 , '4') valid ? ---> No becoz 2nd arg can not be a string
5) In other words, arg1 can be a string but not arg2
6) Is complex('3' , 4) valid ? ---> No becoz arg2 is not permitted when arg1 is a string

```
# bin() function demo program
print(bin(25)) # Converts decimal number to binary i.e. 0B11001
print(bin(0O6247)) # Converts octal number to binary i.e. 0B110 010 100 111
print(bin(0XA7B9)) # Converts hexa decimal number to binary i.e. 0B1010 0111 1011 1001
'''
```

bin() function
----
1) What does bin(x) do ? ---> Converts object 'x' to binary number where
   'x' can be decimal / octal / hexa-decimal number
2) Conversion of decimal number to binary number

```
        16 8 4 2 1 ---> Weights
      1 1 0 0 1
```

3) Conversion of octal number to binary number (2 ^ 3 = 8)

```
   4 2 1 4 2 1 4 2 1 4 2 1 ---> Weights
   1 1 0 0 1 0 1 0 0 1 1 1
```

4) Conversion of hexa-decimal number to binary number (2 ^ 4 = 16)

```
  8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 ---> Weights
    1 0 1 0 0 1 1 1 1 0 1 1 1 0 0 1
'''
```

```
# int() function demo program
print(int(10.8)) # Converts 10.8 to 10
print(int(True)) # Converts True to 1
print(int(False)) # Converts False to 0
print(int('25')) # Converts '25' to 25
print(int('0075')) # 75
print(int(0B11010)) # Converts binary number to decimal number i.e. 16 + 8 + 2 = 26
print(0B11010) # Converts binary number to decimal number i.e. 16 + 8 + 2 = 26
print(int(0O6247)) # Converts octal number to decimal number i.e. 6 * 8 ^ 3 + 2 * 8 ^ 2 + 4 * 8 ^ 1 + 7 * 8 ^ 0 =
3239
print(0O6247) # Converts octal number to decimal number i.e. 6 * 8 ^ 3 + 2 * 8 ^ 2 + 4 * 8 ^ 1 + 7 * 8 ^ 0 = 3239
print(int(0XA7B9)) # Converts hexa decimal number to decimal number i.e. 10 * 16 ^ 3 + 7 * 16 ^ 2 + 11 * 16 ^
1 + 9 * 16 ^ 0 = 42937
print(0XA7B9) # Converts hexa decimal number to decimal number i.e. 10 * 16 ^ 3 + 7 * 16 ^ 2 + 11 * 16 ^ 1 + 9
* 16 ^ 0 = 42937
#print(int(3 + 4j)) # Error becoz complex number can not be converted to integer
#print(int('25.4')) # Error due to string float
#print(int('Ten')) # Error becoz 'Ten' can not be converted to integer
'''
int() function
----------------
1) What does int(x) do ? ---> Converts object 'x' to integer
2) Conversion of binary number to decimal number
    -------------------------------------------------------
         16 8 4 2 1 ---> Weights
       1 1 0 1 0 ---> 16 + 8 + 2 = 26
3) Conversion of octal number to decimal number
    ------------------------------------------------------
       512 64 8 1 ---> Weights
      6 2 4 7 ---> 6 * 512 + 2 * 64 + 4 * 8 + 7 * 1 = 3239
4) Conversion of hexa-decimal number to decimal number
    ---------------------------------------------------------------
       4096 256 16 1 ---> Weights
      A 7 B 9 ---> 10 * 4096 + 7 * 256 + 11 * 16 + 9 * 1 = 42937
'''
# float() function demo program
print(float(25)) # Converts 25 to 25.0
print(float(True)) # Converts True to 1.0
print(float(False)) # Converts False to 0.0
print(float('92')) # Converts '92' to 92.0
print(float('36.4')) # Converts '36.4' to 36.4
print(float('0075')) # Converts '0075' to 75.0
print(float(0B1010101)) # Converts binary number to decimal number i.e. 64 + 16 + 4 + 1 = 85.0
print(float(0O6247)) # Converts octal number to decimal number i.e. 6 * 8 ^ 3 + 2 * 8 ^ 2 + 4 * 8 ^ 1 + 7 * 8 ^ 0 =
3239.0
```

```
print(float(0XA7B9)) # Converts hexa decimal number to decimal number i.e. 10 * 16 ^ 3 + 7 * 16 ^ 2 + 11 * 16 ^ 1 + 9 * 16 ^ 0 = 42937.0

#print(float(3 + 4j)) # Error becoz complex number can not be converted to float

#print(float('Ten')) # Error becoz 'Ten' can not be converted to float

'''

float() function

--------------------

1) What does float(x) do ? ---> Converts object 'x' to float

2) Conversion of binary number to decimal number

    -----------------------------------------------------------

          64 32 16 8 4 2 1 ---> Weights

        1 0 1 0 1 0 1 ----> 64 + 16 + 4 + 1 = 85.0

3) Conversion of octal number to decimal number

     ----------------------------------------------------------

         512 64 8 1 ---> Weights

       6 2 4 7 ---> 6 * 512 + 2 * 64 + 4 * 8 + 7 * 1 = 3239.0

4) Conversion of hexa decimal number to decimal number

     -------------------------------------------------------------------

          4096 256 16 1 ---> Weights

            A 7 B 9 ---> 10 * 4096 + 7 * 256 + 11 * 16 + 9 * 1 = 42937.0

5) How to convert '25.8' to 25 ? ---> int(float('25.8'))

6) Is int('25.8') valid ? ---> No becoz string float can not be converted to integer

'''
```

```
# complex() function demo program
print(complex(3 , 4)) # 3+4j
print(complex(0 , 4)) # 4j
print(complex(3)) # 3 + 0j
print(complex(3.8 , 4.6)) # 3.8 + 4.6j
print(complex(3.8)) # 3.8 + 0j
print(complex(3 , 4.5)) # 3 + 4.5j
print(complex(True , False)) # 1 + 0j
print(complex(True)) # 1 + 0j
print(complex(False)) # 0j
print(complex(True , 4)) # 1 + 4j
print(complex('3')) # 3 + 0j
print(complex('3.8')) # 3.8 + 0j
#print(complex(3 , '4')) # Error due 2nd arg which is string
#print(complex('3' , 4)) # Error due to 2nd arg
#print(complex('3' , '4')) # Error due to 2nd arg
#print(complex('Ten')) # Error becoz 'Ten' can not be converted to complex
# bool() function demo program
print(bool(0)) # Converts 0 to False
print(bool(10)) # True becoz 10 is non-zero number
print(bool(-25)) # True becoz -25 is non-zero number
print(bool(0.0)) # False due to 0.0
print(bool(0.1)) # True becoz 0.1 is non-zero number
print(bool(0 + 0j)) # False becoz both real and imag are zeroes
print(bool(10 + 20j)) # True becoz 10 is non-zero
print(bool(-15j)) # True becoz -15 is non-zero
print(bool('False')) # True becoz 'False' is non-empty string
print(bool('')) # False due to empty string
print(bool('Hyd')) # True becoz 'Hyd' is non-empty string
print(bool(' ')) # True becoz ' ' is non-empty string
print(bool('True')) # True becoz 'True' is non-empty string
'''
bool() function
------------------
1) What does bool(x) do ? ---> Converts object 'x' to True / False
2) Is 0 True (or) False ? ---> False
   What about non-zero ? ---> True
3) Is ''(i.e. Empty string) True (or) False ? ---> False
   What about non-empty string ? ---> True
4) When is x + yj treated as False ? ---> When both 'x' and 'y' are zeroes
    When is x + yj treated as True ? ---> When either 'x' is non-zero (or) 'y' is non-zero
'''
# str() function demo program
print(str(25)) # Converts 25 to '25'
print(str(10.8)) # Converts 10.8 to '10.8'
print(str(3 + 4j)) # Converts 3+4j to '3+4j'
print(str(True)) # Converts True to 'True'
print(str(False)) # Converts False to 'False'
print(str(None)) # Converts None to 'None'
'''
What does str(x) do ? ---> Converts object 'x' to string
'''
```

```
# oct() function demo program
print(oct(195)) # Converts decimal number to octal number i.e. 0O303
print(oct(0B10101110010)) # Converts binary number to octal number i.e. 0O2562
print(oct(0xA7B9)) # Converts hexa decimal number to octal number i.e. 0O123671
'''
oct() function
-----------------
1) What does oct(x) do ? ---> Converts object 'x' to octal number where
                           'x' can be binary / decimal / hexa-decimal number
2) Conversion of decimal number to octal number
     -----------------------------------------------------------
         Number Quotient Remainder
       195 24 3
        24 3 0
         3 0 3
    Remainders in the reverse order ---> 303
3) Conversion of binary number to octal number (2 ^ 3 = 8)
    -----------------------------------------------------------
     4 2 1 4 2 1 4 2 1 4 2 1 ---> Weights
     0 1 0 1 0 1 1 1 0 0 1 0
       2 5 6 2 ---> octal number
4) Conversion of hexa decimal number to binary number (2 ^ 4 = 16)
     --------------------------------------------------------------------
      8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 ---> Weights
      1 0 1 0 0 1 1 1 1 0 1 1 1 0 0 1 ---> Binary number
    Conversion of binary number to octal number (2 ^ 3 = 8)
    -----------------------------------------------------------
     4 2 1 4 2 1 4 2 1 4 2 1 4 2 1 4 2 1
     0 0 1 0 1 0 0 1 1 1 1 0 1 1 1 0 0 1
       1 2 3 6 7 1
'''


# hex() function demo program
print(hex(25)) # Converts decimal number to hexa decimal number i.e. 0X19
print(hex(0B10101111010111)) # Converts binary number to hexa decimal number i.e. 0X2BD7
print(hex(0O6247)) # Converts octal number to hexa decimal number i.e. 0XCA7
'''
hex() function
------------------
1) What does hex(x) do ? ---> Converts object 'x' to hexa-decimal number where
                            'x' can be binary / decimal / octal number
2) Conversion of decimal number to hexa decimal number
    --------------------------------------------------------------------
      Number Quotient Remainder
     25 1 9
      1 0 1
    Remainders in the reverse order ---> 19
3) Conversion of binary number to hexa decimal number (2 ^ 4 = 16)
    -------------------------------------------------------------------
      8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 ---> Weights
      0 0 1 0 1 0 1 1 1 1 0 1 0 1 1 1
        2 B D 7
4) Conversion of octal number to binary number (2 ^ 3 = 8)
    ---------------------------------------------------------
     4 2 1 4 2 1 4 2 1 4 2 1 ---> Weights
     1 1 0 0 1 0 1 0 0 1 1 1 ---> binary number
    Conversion of binary number to hexa decimal number (2 ^ 4 = 16)
    ------------------------------------------------------------------
     8 4 2 1 8 4 2 1 8 4 2 1 ---> Weights
     1 1 0 0 1 0 1 0 0 1 1 1
         C A 7
'''
```

float object
----
1) What can a float object hold ? ---> A float number
2) What is a float number ? ---> A number with decimal point
3) What is the maximum value of float ? ---> Infinity
   What is the minimum value of float ? ---> -Infinity
4) What are the two ways to represent a float number ? ---> Fractional number
                    and
            Mantissa-Exponent number
5) What is 123.45 called ? ---> Fractional number
6) What is 9.728e3 called ? ---> Mantissa-Exponent number
   What is 9.728 in 9.728e3 called ? ---> Mantissa becoz it is before 'e'
   What is 3 called ? ---> Exponent becoz it is after 'e'
7) What is the result of 9.728e3 ? ---> 9.728 * 10 ^ 3 = 9728.0
   What is the result of 9.728E-2 ? ---> 9.728 * 10 ^ -2 = 9.728 / 100 = 0.09728
8) Why python does not support double ? ---> Since max value of float is infinity
9) What is 10.8 called in other languages (value (or) object) ? ---> Value
   What about python ? ---> Object
10) What does x = 10.8 do ? ---> Assigns reference 'x' to float object 10.8
11) Where is float class defined ? ---> In builtins module

range object
----
1) What is a range object ? ---> A group of integer elements
2) Is range object homogeneous ? ---> Yes becoz all the elements in range object are of same type i.e. int type
3) range(x , y , z)
   What does object contain ? ---> Elements from x to y - 1 in steps of z
4) range(x , y , -z)
   What does object contain ? ---> Elements from x to y + 1 in steps of -z
5) range(x , y)
   What does object contain ? ---> Elements from x to y - 1 in steps of 1 becoz default step is 1
6) range(y)
   What does object contain ? ---> Elements from 0 to y - 1 in steps of 1 becoz default begin is 0
7) Is range() valid ? ---> No due to zero arguments
8) What does print(range-object) do ? ---> Prints range object itself but not elements of range object
                              i.e. range(x , y , z)
9) How to obtain elements of range object ? ---> print(*rangeobject)
10) print(*rangeobject)
    What does * operator do ? ---> Unpacks range object to elements
11) Can range object be modified ? ---> No becoz it is immutable
    Can new elements be appended to range object ? ---> No becoz it is immutable
    Can elements be removed from range object ? ---> No becoz it is immutable
12) In other words, range object is neither growable nor shrinkable
13) What does len(range-object) do ? ---> Returns number of elements in the range object
14) Is range object indexed ? ---> Yes becoz it is a sequence

15) What are indexes of elements from left to right ? ---> 0 , 1 , 2 , ... length - 1

What are indexes of elements from right to left ? ---> -1 , -2 , -3 , ... -length

16) What is the use of indexing ? ---> Random access

How to obtain 10th element of range object ? ---> a[9] where 'a' is range object

How to obtain 1st element of range object ? ---> a[0]

How to obtain last element of range object ? ---> a[len(a) - 1] (or) a[-1]

17) Can range object be sliced ? ---> Yes becoz it is indexed

18) Can range object have duplicate elements ? ---> No

19) In other words, range object can have only unique elements

20) Can range object be repeated with * operator ? ---> No becoz duplicate elements are obtained when range object is repated which is not permitted

```
# float object demo program (Home work)
a = 10.8 # Ref 'a' points to float object 10.8
print(a) # Value of object 'a' i.e. 10.8
print(type(a)) # Type of object 'a' i.e. <class 'float'>
print(id(a)) # Address of object 'a' (may be 1000)
b = 25. # Valid and is interpreted as 25.0
print(b) # 25.0
print(type(b)) # <class 'float'>
c = .689 # Valid and is interpreted as 0.689
print(c) # 0.689
d = 3.4E2 # 3.4 * 10 ^ 2
print(d) # 340.0
print(type(d))# <class 'float'>
e = 9.62e-2 # 9.62 * 10 ^ -2
print(e) # 0.0962
#print(9.8.2) # Error due to 2 decimal points
```

```
# Find outputs (Home work)
a = range(10 , 50 , 5) # Object contains elements from 10 to 49 in steps of 5
print(type(a)) # <class 'range'>
print(a) # range(10 , 50 , 5)
print(*a) # Unpacks object 'a' to elements i.e. 10 <space> 15 <space> 20 <space> 25 <space> 30 <space> 35 <space> 40 <space> 45
print(id(a)) # Address of range object
print(len(a)) # 8
print(*a[2 : 7] , sep = ' , ') # *a[2 : 7 : 1] ---> Elements of object 'a' from indexes 2 to 6 in steps of 1 i.e. 20 , 25 , 30 , 35 , 40
print(*a[ : : -1]) # *a[-1 : -9 : -1] ---> Elements of object 'a' from indexes -1 to -8 in steps of -1 i.e. 45 <space> 40 <space> 35 <space> 30 <space> 25 <space> 20 <space> 15 <space> 10
#a[4] = 32 # Error becoz range object can not be modified
#print(a * 2) # Error becoz range object can not be repeated
'''
                0 1 2 3 4 5 6 7
range object ---> 10 15 20 25 30 35 40 45
            -8 -7 -6 -5 -4 -3 -2 -1
'''
```

```
# Find outputs (Home work)
a = range(10 , 20) # range(10 , 20 , 1) ---> Object contains elements from 10 to 19 in steps of 1
print(*a , sep = ',') # 10,11,12,13,....19
b = range(5) # range(0 , 5 , 1) ---> Object contains elements from 0 to 4 in steps of 1
print(*b) # 0 <space> 1 <space> 2 <space> 3 <space> 4
c = range(10 , 1 , -1) # Object contains elements from 10 to 2 in steps of -1
print(*c , sep = '...') # 10 ... 9 ... 8 ... .... 2
d = range(-10 , 0) # range(-10 , 0 , 1) ---> Object contains elements from -10 to -1 in steps of 1
print(*d) # -10 <space> -9 <space> -8 ... -1
e = range(-10) # range(0 , -10 , 1) ---> Empty object becoz 0 >= -10
print(*e) # Unpacks empty object i.e. Nothing
f = range(2 , 2) # range(2 , 2 , 1) ---> Empty object becoz 2 >=
print(*f) # Unpacks empty object i.e. Nothing
#g = range(10 , 11 , 0.1) # Error becoz range object can not hold float elements
#h = range('A' , 'F') # Error becoz range object can not hold str elements
'''
1) range(x , y , +ve step)
   When is range object empty ? ---> When x >= y
2) range(x , y , -ve step)
   When is range object empty ? ---> When x <= y
'''
```

complex object

----

1) What can a complex object hold ? ---> A complex number such as 3 + 4j

2) What are the two fields of complex object ? ---> real and imag

3) What is 3 in 3 + 4j called ? ---> real

What is 4 in 3 + 4j called ? ---> imag

4) Is 5 + 6i valid ? ---> No due to 'i'

5) Is 7 + j8 valid ? ---> No becoz imag is after 'j'

6) What does a = 3 + 4j do ? ---> Assigns reference 'a' to complex object 3+4j

7) What is the value of 'j' ? ---> sqrt(-1)

8) Where is complex class defined ? ---> In builtins module

List object

----

1) What is a list ? ---> A group of elements in [ ]

2) Is [10 , 20 , 15 , 18] a list ? ---> Yes due to []

3) What is [] called ? ---> List operator

4) Can list hold different types of elements ? ---> Yes becoz it is a heterogeneous object

Eg: [25 , 10.8 , 'Hyd' , True , None , 3 + 4j]

5) Is [25 , 25] valid ? ---> Yes becoz list can hold duplicate elements

6) What does len(list) do ? ---> Returns number of elements in the list

7) Is list indexed ? ---> Yes becoz it is a sequence

8) What are the indexes of elements from left to right ? ---> 0 to length - 1

What are the indexes of elements from right to left ? ---> -1 to -length

9) What is the use of indexing ? ---> Random access

How to obtain 10th element of list ? ---> a[9] where 'a' is a list

How to obtain 1st element of list ? ---> a[0]

How to obtain last element of list ? ---> a[len(a) - 1] (or) a[-1]

10) Can list be sliced ? ---> Yes becoz it is indexed

What is the syntax of slice ? ---> list[begin : end : step]

What is obatined when list is sliced ? ---> Sub - list

11) list = [10 , 20 , 15]

Is list[1] = 18 valid ? ---> Yes becoz list can be modified as it is a mutable object

and 20 is replaced with 18

12) How to append an element to the list ? ---> With append() method of list class

13) list = [10 , 20 , 15]

What does list . append(18) do ? ---> Inserts 18 at the end of the list

14) How to remove list element ? ---> With remove() method of list class

15) list = [10 , 15 , 20 , 15 , 18]

What does list . remove(15) do ? ---> Removes first 15 from the list

What does list . remove(25) do ? ---> Throws error becoz there is no 25 in the list

16) In other words, list is growable and shrinkable

17) Can list be repeated ? ---> Yes with * operator

18) list = [10 , 20 , 15]

What does list * 2 do ? ---> Repeats list twice i.e. [10 , 20 , 15 , 10 , 20 , 15]

19) What does print(list) do ? ---> Prints list itself

i.e. [Element1 , Element2 , Element3 , ...]

What does print(*list) do ? ---> Unpacks list to elements

i.e. Element1 Element2 Element3 .....

20) What is the most frequently used sequence in python ? ---> List

```
# complex object demo program
a = 3 + 4j
print(a)
print(type(a))
print(id(a))
print(a . real)
print(a . imag)
print(type(a . real))
print(type(a . imag))
```

```
'''
What is the type of real and imag ? --->


Always float
''' # Find outputs (Home Work)
a = [25 , 10.8 , 'Hyd' , True , 3 + 4j , None , 'Hyd' , 25] # List due to []
print(a) # [25 , 10.8 , 'Hyd' , True , 3 + 4j , None , 'Hyd' , 25]
print(*a) # Unpacks list into elements i.e. 25 <space> 10.8 <space> Hyd <space> True <space> 3+4j <space> None <space> Hyd <space> 25
print(type(a))# <class 'list'>
print(id(a)) # Address of list
print(len(a)) # 8
a[2] = 'Sec' # Element at index 2 is modified to 'Sec'
print(a) # [25 , 10.8 , 'Sec' , True , 3 + 4j , None , 'Hyd' , 25]
print(a[2 : 5]) # List from indexes 2 to 4 in steps of 1 i.e. ['Sec' , True , 3+4j]
# Find outputs (Home work)
a = 6j
print(a)
print(type(a))
print(a . real)
print(a . imag)
print(5 + j6)
print(3 + 4i)
print(4+j)
print(4 + 1j)
print(4 + 0j)
# How to print list in different ways (Home work)
a = [25 , 10.8 , 'Hyd' , True]
print('List with print function')
print(a) # [25 , 10.8 , 'Hyd' , True]
print('Elements of list without using indexes')
for x in a: # How to print each element of list using for loop without using indexes
 print(x) # 25 <next line> 10.8 <next line> Hyd <next line> True <next line>
print('Elements of list using indexes')
for i in range(len(a)): # prints a[i] where 'i' varies from 0 to len - 1
  print(a[i]) # 25 <next line> 10.8 <next line> Hyd <next line> True <next line>
print('Elements of list in reverse order without slice')
for i in range(1 , len(a) + 1): # prints a[-i] where 'i' varies from 1 to len
  print(a[-i]) # True <next line> Hyd <next line> 10.8 <next line> 25 <next line>
print('Reverse List with slice')
print(a[::-1]) # a[-1 : -5 : -1] ---> List from indexes -1 to -4 in steps of -1 i.e. [True , 'Hyd' , 10.8 , 25]
'''
1) for x in a:
            print(x)
   Iteration x
  ---------------------------
        1 25
        2 10.8
        3 Hyd
        4 True
2) for i in range(len(a)):
            print(a[i])
        i a[i]
     -------------------
        0 25
        1 10.8
        2 Hyd
        3 True
 What is the difference between a[i] and 'i' ? --->
                  a[i] is each element of list and 'i' is index of each element of list
3) for i in range(1 , len(a) + 1):
                print(a[-i])
        i a[-i]
    ---------------------
        1 a[-1] ---> True
        2 a[-2] ---> Hyd
        3 a[-3] ---> 10.8
        4 a[-4] ---> 25
4) What is the result of string[::-1] ? ---> Reverse string
    What is the result of list[::-1] ? ---> Reverse list
'''
```

```
# append() and remove() methods (Home work)
a = [ ] # Empty list
print(a) # []
a . append(25) # Appends 25 to list 'a'
a . append(10.8) # Appends 10.8 to list 'a'
a . append('Hyd') # Appends 'Hyd' to list 'a'
a . append(True) # Appends True to list 'a'
print(a) # [25,10.8,'Hyd',True]
a . remove('Hyd') # Removes 'Hyd' from list 'a'
print(a) # [25,10.8,True]
#a . remove('25') # Error becoz '25' is not in list 'a'
print(a) # [25,10.8,True]
'''
1) How many lists are in the program ? ---> Single
2) The above program demonstrates that list is growable and shrinkable
'''
# Find outputs (Home work)
a = [25 , 10.8 , 'Hyd']
print(a * 3) # Repeats list thrice i.e. [25,10.8,'Hyd',25,10.8,'Hyd',25,10.8,'Hyd']
print(a * 2) # Repeats list twice i.e. [25,10.8,'Hyd',25,10.8,'Hyd']
print(a * 1) # Repeats list once i.e. [25,10.8,'Hyd']
print(a * 0) # Repeats list 0 times i.e. []
print(a * -1) # Repeats list -1 times i.e. []
#print(a * 4.0) # Error due to float operand 4.0
'''
1) What does list * n do ? ---> Repeats list for 'n' times
2) How many elements are in the resultant list ? ---> n * len(list)
'''
# list() function demo program
a = list('Hyd') # Converts string to list
print(a) # ['H' , 'y' , 'd']
print(type(a)) # <class 'list'>
print(len(a)) # 3
b = (10 , 20 , 15 , 18) # Tuple due to ()
print(list(b)) # Converts tuple to list ---> [10,20,15,18]
print(list(range(5))) # Converts range object to list ---> [0,1,2,3,4]
#print(list(25)) # Error becoz 25 is not a sequence
'''
list() function
-----------------
1) What does list(sequence) do ? ---> Converts sequence to list
2) Is list(non-sequence) valid ? ---> No becoz argument should be sequence only
3) What does list(No args) do ? ---> Returns an empty list i.e. []
4) Finally list() function does typecasting
```

```
'''
# Find outputs
a = [ ] # Empty list
print(type(a)) # <class 'list'>
print(a) # []
print(len(a)) # 0
b = list() # Returns an empty list
print(b) # []
print(len(b)) # 0
'''
What are the two ways to represent an empty list ? ---> [] and list()
'''
# Slice demo program (Home work)
# 0 1 2 3 4 5 6 7
list = [25 , 10.8 , 3 + 4j , 'Hyd' , True , None , 10.8 , 'Hyd']
# -8 -7 -6 -5 -4 -3 -2 -1
print(list[2 : 7])# list[2 : 7 : 1] ---> List from indexes 2 to 6 in steps of 1 i.e. [3 + 4j , 'Hyd' , True , None , 10.8]
print(list[ : : ]) # list[0 : 8 : 1] ---> List from indexes 0 to 7 in steps of 1 i.e. [25 , 10.8 , 3 + 4j , 'Hyd' , True , None , 10.8 , 'Hyd']
print(list[:]) # list[0 : 8 : 1] ---> List from indexes 0 to 7 in steps of 1 i.e. [25 , 10.8 , 3 + 4j , 'Hyd' , True , None , 10.8 , 'Hyd']
print(list[ : : -1]) # list[-1 : -9 : -1] ---> List from indexes -1 to -8 in steps of -1 i.e. ['Hyd' , 10.8 , None , True , 'Hyd' , 3+4j , 10.8 , 25]
print(list[ : : 2]) # list[0 : 8 : 2] ---> List from indexes 0 to 7 in steps of 2 i.e. [25 , 3+4j , True , 10.8]
print(list[1 : : 2]) # list[1 : 8 : 2] ---> List from indexes 1 to 7 in steps of 2 i.e. [10.8 , 'Hyd' , None, 'Hyd']
print(list[ : : -2]) # list[-1 : -9 : -2] ---> List from indexes -1 to -8 in steps of -2 i.e. ['Hyd' , None , 'Hyd' , 10.8]
print(list[-2 : : -2]) # list[-2 : -9 : -2] ---> List from indexes -2 to -8 in steps of -2 i.e. [10.8 , True , 3+4j , 25]
print(list[1 : 4]) # list[1 : 4 : 1] ---> List from indexes 1 to 3 in steps of 1 i.e. [10.8 , 3+4j , 'Hyd']
print(list[-4 : -1]) # list[-4 : -1 : 1] ---> List from indexes -4 to -2 in steps of 1 i.e. [True , None , 10.8]
print(list[3 : -3]) # print(list[3 : -3 : 1]) ---> List from indexes 3 to -4 in steps of 1 i.e. ['Hyd' , True]
print(list[2 : -5]) # list[2 : -5 : 1] ---> List from indexes 2 to -6 in steps of 1 i.e. [3+4j]
print(list[-1:-5]) # list[-1 : -5 : 1] ---> List from indexes -1 to -6 in steps of 1 i.e. []

# Find outputs (Home work)
# 0 1 2 3 4 5 6 7
list = [25 , 10.8 , 3+4j , 'Hyd' , True , None , 10.8 , 'Hyd']
x , y = list[3 : 5] # x , y = list[3 : 5 : 1] ---> List from indexes 3 to 4 in steps of 1 i.e. ['Hyd' , True]
print('x : ' , x) # x : Hyd
print('y : ' , y) # y : True
for x in list[2:7]: # List from indexes 2 to 6 in steps of 1 i.e. [3+4j , 'Hyd' , True , None , 10.8]
 print(x) # 3+4j <next line> Hyd <next line> True <next line> None <next line> 10.8 <next line>
'''
The above for loop iterates a part of the list due to slice
'''
# Find outputs (Home work)
# 0 1 2 3 4
```

```
a = [10 , 20 , 30 , 40 , 50]
print(a) # [10,20,30,40,50]
a[1 : 5] = [60 , 70 , 80] # Replaces elements of list 'a' from indexes 1 to 4 with 60 , 70 , 80
print(a) # [10,60,70,80]
'''
a[1 : 5] = [60 , 70 , 80] modifies 4 elements of list with 3 elements
'''
# Find outputs (Home work)
a = [25]
#print(a[1]) # Error becoz index 1 does not exist in [25]
print(a[1:]) # a[1 : 1 : 1] ---> [] becoz 1 >= 1
'''
Index may throw error but slice never throws error
'''
# Find outputs (Home work)
list = [10 , 20 , 15 , 12 , 18]
print(15 in list) # True
print(19 in list) # False
print(14 not in list) # True
print(12 not in list) # False
'''
1) x in list
   What does in operator do ? ---> Returns True when 'x' is in the list and False otherwise
2) x not in list
    What does not in operator do ? ---> Returns True when 'x' is not in the list and False otherwise
'''
'''
Write a program to remove all 15's from the list
Hint: while cond:
            statements
          statements
'''
a = [10 , 20 , 15 , 18 , 12 , 15 , 19 , 25 , 15 , 14 , 12]
while 15 in a: # Repeat until there is no 15 in the list:
  a . remove(15) # How to remove each 15 from the list
print(a) # [10 , 20 , 18 , 12 , 19 , 25 , 14 , 12]
'''
How to remove each 15 from the list ? ---> Call remove() method in a loop
'''
```

bool object

----

1) What can a bool object hold ? ---> A boolean value such as True (or) False

2) What does a = True do ? ---> Assigns reference 'a' to bool object True

3) What is the value of True ? ---> 1

   What is the value of False ? ---> 0

4) What is the result of True + False + True ? ---> 1 + 0 + 1 = 2

5) What happens when an operation is made on True and False ? --->

      The operation is internally made on 1 and 0

6) Are True and False bool class objects (or) int class objects ? --->

    int class objects when operations are made on them

      and

    bool class objects otherwise

7) Are true and false valid ? ---> No due to 't' and 'f'

8) Where is bool class defined ? ---> In builtins module

9) Are True and False user defined words (or) keywords ? ---> Keywords

```
# bool object demo program
a = True
print(a)
print(type(a))
print(id(a))
b = False
print(b)
print(type(b))
print(True + True)
print(True + False)
print(False + True)
print(False + False)
print(True + True + True)
print(25 + 10.8 + True)
print(True > False)
print(True)
print(False)
print(true)
print(false)
'''
1) When are True and False treated as 1 and 0 ? --->
When operations are made on True and False
2) When are True and False not treated as 1 and 0 ? --->
When operations are not made on True and False
'''
# Find outputs (Home work)
a = (25 , 10.8 , 'Hyd' , True , 3+4j , None , 'Hyd' , 25) # 'a' is tuple due to ()
print(a) # (25 , 10.8 , 'Hyd' , True , 3+4j , None , 'Hyd' , 25)
print(*a) # Unpacks tuple into elements i.e. 25 <space> 10.8 <space> Hyd <space> True <space> 3+4j <space>
None <space> Hyd <space> 25
print(type(a)) # <class 'tuple'>
print(len(a)) # 8
print(a[2 : 5]) # Tuple from indexex 2 to 4 in steps of 1 i.e. ('Hyd' , True , 3+4j)
print(*a[2 : 5]) # Unpacks sub-tuple ---> Hyd <space> True <space> 3+4j
#a[2] = 'Sec' # Error becoz tuple is immutable
#a . append('Sec') # Error becoz there is no append() method in tuple
#a . remove('Hyd') # Error becoz there is no remove() method in tuple
b = 10 , 20 , 30 # Valid becoz () are optional
print(b) # (10 , 20 , 30)
print(b * 2) # Repeats tuple twice i.e. (10 , 20 , 30 , 10 , 20 , 30)
c = 40 , 50 , 60, # Valid and last comma is optional
print(c) # (40 , 50 , 60)
print(type(c))# <class 'tuple'>
# Find outputs (Home work)
a = (25) # integer becoz comma is missing
b = 25, # Tuple due to comma
c = 25 # integer becoz comma is missing
d = (25,) # Tuple due to comma
print(type(a)) # <class 'int'>
print(type(b)) # <class 'tuple'>
print(type(c)) # <class 'int'>
print(type(d)) # <class 'tuple'>
print(a * 4) # 25 * 4 = 100
print(b * 4) # Repeat tuple 4 times i.e. (25,25,25,25)
print(c * 4) # 25 * 4 = 100
print(d * 4) # Repeat tuple 4 times i.e. (25,25,25,25)
'''
1) What is 25, called ? ---> Tuple due to comma
   What is (25) called ? ---> int becoz there is no comma
2) What is 10.8, called ? ---> Tuple
    What is (10.8) called ? ---> float
3) What is 3 + 4j, called ? ---> Tuple
   What is (3 + 4j) called ? ---> complex
4) What is True, called ? ---> Tuple
    What is (True) called ? ---> bool
5) What is 'Hyd', called ? ---> Tuple
     What is ('Hyd') called ? ---> str
'''
# tuple() function demo program (Home work)
a = tuple('Hyd') # Converts string to tuple
print(a) # ('H' , 'y' , 'd')
print(type(a)) # <class 'tuple'>
print(len(a)) #3
```

```
b = [10 , 20 , 15 , 18]
print(tuple(b)) # Converts list to tuple i.e. (10,20,15,18)
print(tuple(range(5))) # Converts range object to tuple i.e. (0,1,2,3,4)
#print(tuple(25)) # Error becoz 25 is not a sequence
'''
tuple() function
-------------------
1) What does tuple(sequence) do ? ---> Converts sequence to tuple
2) Is tuple(non-sequence) valid ? ---> No becoz argument should be sequence only
3) What does tuple(No args) do ? ---> Returns an empty tuple
'''
# Find outputs (Home work)
a = () # Empty tuple
print(type(a)) # <class 'tuple'>
print(a) # ()
print(len(a)) # 0
b = tuple() # Function returns an empty tupe
print(b) # ()
print(len(b)) # 0
'''
1) When are ( ) optional for tuple ? ---> When tuple has got at least one element
2) When are ( ) mandatory for tuple ? ---> Empty tuple
3) What are the two ways to represent an empty tuple ? ---> ( ) and tuple()
'''
# Gift
# Find outputs (Home work)
a = (10 , 20 , 30)
print(a) # (10 , 20 , 30)
print(id(a)) # Address of tuple with 3 elements (may be 1000)
a = a * 2 # Ref 'a' is modified to a tuple of 6 elements
print(a) # (10,20,30,10,20,30)
print(id(a)) # Address of tuple with 6 elements (may be 2000)
'''
1) a = (10 , 20 , 30)
   a = a * 2
   What is modified ? ---> Reference but not tuple
2) How many tuples are in the program ? ---> Two tuples
'''
```

Tuple Vs List

----

1) Can tuple be modified ? ---> No becoz it is an immutable object
   What about list ? ---> It can be modified becoz it is a mutable object
2) What is tuple operator ? ---> ( ) and () are optional
   What is list operator ? ---> [ ] and [] are mandatory
3) What is another name of tuple ? ---> Read-only list becoz tuple can be accessed but can not be modified
4) Is tuple growable and shrinkable ? ---> No becoz tuple is immutable
   What about list ? ---> It is growable and shrinkable
5) Is tuple size fixed (or) variable ? ---> Fixed size
   What about list ? ---> Variable size
6) Is tuple . append(x) valid ? ---> No becoz there is no append() method in tuple
   What does list . append(x) do ? ---> Inserts 'x' at the end of the list
7) Is tuple . remove(x) valid ? ---> No becoz there is no remove() method in tuple
   What does list . remove(x) do ? ---> Removes first 'x' from the list
8) How is tuple of single element denoted ? ---> (25,) and , is mandatory
   How is list of single element denoted ? ---> [25,] and , is optional

Note: List and tuple are same except the above differences set object

----

1) What is a set ? ---> A group of elements in { }

2) Is {10 , 20 , 15} a set ? ---> Yes due to { } and { } is called set operator

3) Can set hold different types of elements ? ---> Yes becoz set is heterogeneous object

*4) Can set hold duplicate elements ? ---> No becoz set can hold unique elements

Is {25 , 25 , 25} valid ? ---> Yes and it is a set of single element i.e. {25}

5) a = {25 , 10.8 , 'Hyd' , True , 3+4j , None , 'Hyd' , 25}

How many elements are in set 'a' ? ---> 6 elements but not 8

*6) Is set ordered (or) unordered ? ---> Unordered

7) What is an unordered object ? --->

Elements may not be represented in the order in which they have been inserted

8) How is {10 , 20 , 15 , 5} represented internally ? ---> Any order such as {5 , 20 , 15 , 10}

9) What is the first element in {10 , 20 , 15 , 18} ? ---> No idea becoz it is unordered

*10) Is set indexed ? ---> No becoz it is unordered

11) What does set[2] do ? ---> Throws error becoz set is not indexed

12) How to obtain 10th element of set ? ---> Not possible becoz set is unordered and not indexed

13) In other words, random access is not possible from set

14) Can set be sliced ? ---> No becoz there are no indexes

15) How to insert an element into the set ? ---> With add() method of set class

16) set = {10 , 20 , 15}

What does set . add(18) do ? ---> Inserts 18 any where in the set

What does set . add(20) do ? ---> Ignores 20 becoz set already contains 20

17) How to remove a set element ? ---> With remove() method of set class

18) set = {10 , 15 , 20 , 15 , 18}

What does set . remove(15) do ? ---> Removes 15 from the set

What does set . remove(25) do ? ---> Throws error becoz there is no 25 in the set

19) In other words, set is growable and shrinkable

20) set = {10 , 20 , 15}

Is set[1] = 18 valid ? ---> No becoz set is not indexed

21) In other words, set can not be modified becoz there are no indexes

22) Is set a mutable object (or) immutable ? ---> Mutable object but not 100% becoz modification is not permitted

23) Is {{10,20,15,18}} valid ? ---> No becoz set can not hold mutable elements such as list , set and dictionary

{(10,20,15,18)} valid ? ---> Yes becoz set can hold immutable elements and tuple is immutable

24) Can set be repeated ? ---> No becoz duplicates are obtained when set is repeated which is not permitted

25) What does len(set) do ? ---> Returns number of elements in the set

Note:

How is set different from remaining sequences (total : 4) ? ---> 1) Set can not hold duplicate elements

2) set is unordered

3) set is not indexed

4) Set can not hold mutable elements

Points to remember

----

1) Does python have main() function ? ---> No
2) int a;
   float b;
   Are the above statements valid ? ---> No becoz there are no declarations in python
3) In other words, object can be used directly without any prior declaration
4) a = 25
   What is the type(a) ? ---> int becoz 25 is an integer number
5) b = 10.8
   What is the type(b) ? ---> float becoz 10.8 is a float number
6) Therefore python is called a dynamically typed language
7) Are there values in python ? ---> No
   What is 25 called in python ? ---> An int class object
   What is 10.8 called in python ? ---> A float class object
   What is True called in python ? ---> A bool class object
8) Everything is an object in python
9) What are int , float , bool , complex called (classes (or) datatypes) ? ---> classes but not datatypes
10) Is ; mandatory at the end of statements ? ---> No and it is optional
11) Is python a 100% OOL (object oriented language) ? --->
    Yes becoz there are only classes but not datatypes
            and
    also there are only objects but not variables
12) Is python a compiler language (or) interpreter language ? ---> An interpreter language
13) What is an interpreter language ? ---> Line by line translation and execution
14) In other words, translation and execution are alternate
15) Python program is executed at the time of translation itself
16) Is python program execution fast (or) slow ? ---> Slow due to repeated translation
17) In other words, python program is translated every time program is executed

```
# set object demo program (Home work)
a = {25 , 10.8 , 'Hyd' , True , 3+4j , None , 25 , 'Hyd'} # 'a' is set due to { }
print(a) # {25 , 10.8 , 'Hyd' , True , 3+4j , None} in any order
print(type(a)) # <class 'set'>
print(len(a)) # 6
#print(a[2]) # Error becoz set is not indexed
#print(a[1 : 4]) # Error becoz set can not be sliced
#a[2] = 'Sec' # Error becoz set can not be modified as there is no index
#print(a * 2) # Error becoz set can not be repeated
#print(a * a) # Error becoz sets can not be multiplied
'''
Order may change every time program is executed
'''
# Gift
# Find outputs (Home work)
a = {1 , 'Hyd' , False , True , 0.0 , '' , 1.0 , 0}
print(a) # {1 , 'Hyd' , False , ''} in any order
print(len(a)) # 4
print(type(a)) # <class 'set'>
'''
1) Can set have duplicate elements ? ---> No
2) Can set have 1 , True and 1.0 ? ---> No becoz they are same
3) Can set have False , 0.0 and 0 ? ---> No becoz they are same
'''
# set() function demo program
a = set('Rama rAo') #Converts string to set
print(a) # {'R' , 'a' , 'm' , ' ' , 'r' , 'A' , 'o'}
print(len(a)) # 7
print(set([10 , 20 , 15 , 20])) # Converts list to set i.e. {10 , 20 , 15}
print(set((25 , 10.8 , 'Hyd' , 10.8))) # Converts tuple to set i.e. {25 , 10.8 , 'Hyd'}
print(set(range(10 , 20 , 3))) # Converts range object to set i.e. {10 , 13 , 16 , 19}
#print(set(25)) # Error becoz 25 is not a sequence
'''
set() function
----------------
1) What does set(sequence) do ? ---> Converts sequence to set
2) Is set(non-sequence) valid ? ---> No becoz argument should be sequence only
3) What does set(No args) do ? ---> Returns an empty set
'''
# Gift
# add() and remove() methods (Home work)
a = set() # Empty set
a . add(25) # Inserts 25 into empty set
a . add(10.8) # Inserts 10.8 any where in the set
a . add('Hyd') # Inserts 'Hyd' any where in the set
a . add(True) # Inserts True any where in the set
a . add(None) # Inserts None any where in the set
a . add('Hyd') # Ignored becoz set already contains 'Hyd'
a . add(1) # Ignored becoz set already contains True
print(a) # {25 , 10.8 , 'Hyd' , True , None} in any order
a . remove(25) # Removes 25 from set 'a'
print(a) # {10.8 , 'Hyd' , True , None} in same order (same as line 11)
#a . append(100) # Error becoz there is no append() method in set
'''
1) Which method is used to append an element to list ? ---> append() method
2) Which method is used to insert an element into set ? ---> add() method
3) Which method is used to remove an element from list and set ? ---> remove() method
4) a = {25 , 10.8 , 'Hyd' , True}
   print(a)
```

```
    print(a)
    print(a)
    Is set printed in the same order all the three times ? ---> Yes becoz it is the same set
5) a = {25 , 10.8 , 'Hyd' , True}
    print(a) ---> {10.8 , True , 'Hyd' , 25}
    Is set printed in the same order every time program is executed ? ---> Not guranteed
'''
# How to print set in two differnet ways (Home work)
a = {25 , True , 'Hyd' , 10.8}
print('set with print function')
print(a) # How to print set ---> {'Hyd' , 25 , 10.8 , True} in any order
print('Iterate elements of set with for loop')
for x in a: # How to iterate set with for loop
  print(x) # Hyd <next line> 25 <next line> 10.8 <next line> True <next line>
'''
1) set is iterated in the same order in which it is printed becoz it is the same set
2) a = {25 , True , 'Hyd' , 10.8}
   for i in range(len(a)):
      print(a[i])
   Is the above for loop valid ? ---> No becoz set is not indexed
'''
```

Dictionary

----

1) What is a dictionary ? ---> A group of key : value pairs in { }

2) What are the key : value pairs in college ? ---> Roll Number : Student Name

What are the key : value pairs in company ? ---> Emp Number : Emp Name

What are the key : value pairs in bank ? ---> Acct Number : Cust Name

What are the key : value pairs in India ? ---> Aadhar number : Person Name

What are the key : value pairs in Internet ? ---> Ip Address : Domain Name

3) How is dictionary different from remaining sequences ? ---> List , tuple and set are a group of elements

but dictionary is a group of key : value pairs

4) Can keys be repeated (or) duplicated ? ---> No and they should be unique

What about values ? ---> They can be repeated (or) duplicated

5) Is {10 : 'Hyd' , 10 : 'Sec'} valid ? ---> Yes and 'Hyd' is replaced with 'Sec' becoz key 10 is repeated

How many key : value pairs are in the above dictionary ? ---> 1 i.e. {10 : 'Sec'}

6) Can dictionary be repeated ? ---> No becoz duplicate keys are obtainbed when dictionary is repeated which is not permitted

7) Is {[] : []} valid ? ---> No becoz key can not be mutable object such as list

8) In other words, key should be an immutable object

9) What about value ? ---> Any python object(i.e. Immutable (or) mutable)

10) Is dictionary ordered (or) unordered ? ---> Ordered from python 3.6 (Current version : 3.13)

11) Is dictionary indexed ? ---> No due to key : value pairs

12) Can dictionary be sliced ? ---> No becoz there are no indexes

13) What does len(dict) do ? ---> Returns number of key : value pairs

14) What does dict[valid-key] do ? ---> Returns value of the key

15) In other words, it is possible to obtain value from dictionary by using key

16) What does dict[Invalid-key] do ? ---> Throws error

17) Is dict[value] valid ? ---> No and it throws error

18) In other words, it is not possible to obtain key by using value

19) Can dictionary be modified ? ---> Yes becoz it is mutable object

20) What does dict[valid-key] = value do ? ---> Modifies value of the key

What does dict[new-key] = value do ? ---> Appends new key : value pair to the dictionary

21) What does del dict[key] do ? ---> Removes key : value pair from dictionary

22) Is dictionary growable and shrinakble ? ---> Yes

23) In other words, key : value pairs can be appended and removed

```
# Find outputs
a = {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
print(a . keys()) # dict_keys([10 , 20 , 15 , 18])
print(a . values()) # dict_values(['Ramesh' , 'Kiran' , 'Amar' , 'Sita'])
print(a . items()) # dict_items([(10 , 'Ramesh') , (20 , 'Kiran') , (15 , 'Amar') , (18 , 'Sita')])
print(a) # {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
'''
```

1) What does keys() method do ? ---> Returns dict_keys object which has list of all the keys in the dictionary

2) What does values() method do ? ---> Returns dict_values object which has list of all the values in the dictionary

3) What does items() method do ? ---> Returns dict_items object which has list of tuples and

each tuple has two elements i.e. (k1 , v1) , (k2 , v2) , (k3 , v3) ....

'''

```
# NoneType object demo program
a = None # Ref 'a' points to object None
print(type(a)) # Type of object 'a' i.e. <class 'NoneType'>
print(a) # Value of object 'a' i.e. None
print(id(a)) # Address of object None
print(id(None))# Error due to 'n'
'''
1) Is NoneType a class (or) object ? ---> class
   What about None ? ---> Object
2) Where is NoneType class defined ? ---> In builtins module
3) Is None a user defined word (or) keyword ? ---> Keyword
'''
# Find outputs (Home work)
a = {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'} # Dictionary
print(a) # {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
print(type(a)) # <class 'dict'>
print(a[10]) # Value of 10 i.e. Ramesh
print(a[20]) # Value of 20 i.e. Kiran
print(a[15]) # Value of 15 i.e. Amar
print(a[18]) # Value of 18 i.e. Sita
#print(a[19]) # Error becoz 19 is not a valid key
#print(a[0]) # Error becoz 0 is not a valid key
#print(a['Amar']) # Error becoz 'Amar' is not a valid key
a[15] = 'Krishna' # Modifies value of 15 to 'Krishna'
del a[20] # Removes 20 : 'Kiran' from dict 'a'
a[25] = 'Vamsi' # Appends 25 : 'Vamsi' to dict 'a'
print(a) # {10 : 'Ramesh' , 15 : 'Krishna' , 18 : 'Sita' , 25 : 'Vamsi'}
print(len(a)) # 4
#print(a * 2) # Error becoz dict can not be repeated

# Find outputs (Home work)
a = {10 : 'Hyd' , 10 : 'Sec'} # Replaces 'Hyd' with 'Sec' becoz key 10 is duplicated
print(a) # {10 : 'Sec'}
print(len(a)) # 1
b = {'R' : 'Red' , 'G' : 'Green' , 'B' : 'Blue' , 'Y' : 'Yellow' , 'G' : 'Gray' , 'B' : 'Black'}
print(b) # {'R' : 'Red' , 'G' : 'Gray' , 'B' : 'Black' , 'Y' : 'Yellow'}
print(len(b)) # 4
'''
What happens when key is repeated in the dictionary ? ---> Value gets replaced
'''
# Gift
# Find output (Home work)
a = {True : 'Yes' , 1 : 'No' , 1.0 : 'May be'}
print(a) # {True : 'May be'}
print(len(a)) # 1
'''
1) What happens when 1 : 'No' is encounterd ? ---> 'Yes' is replaced with 'No' becoz True and 1 are same
2) What happens when 1.0 : 'May be' is encounterd ? ---> 'No' is replaced with 'May be ' becoz True and 1.0 are same
3) Value gets replaced but key remains unchanged
'''
# Find outputs
#a = { [ ] : 25} #Error becoz list is not an immutable object
b = { ( ) : 25} # Valid
print(b) # { () : 25}
#c = { { } : 25} #Error becoz dict is not an immutable object
d = {'Ramesh' : [9948250500, 9848565090, 9440250404]} # valid
print(d) # {'Ramesh' : [9948250500, 9848565090, 9440250404]}
print(len(d)) # 1
#e = {set() : 10.8} #Error becoz set is not an immutable object

# Find outputs
a = {} # Empty dictionary
print(type(a)) # <class 'dict'>
print(len(a)) # 0
print(a) # { }
b = dict() # Returns an empty dictionary
print(type(b)) # <class 'dict'>
print(len(b)) # 0
print(b) # { }

# Gift
# How to print dictionary in different ways
a = {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
print('Dictionary with print function')
print(a) # How to print dictionary ---> {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
print('Keys of dictionary')
for x in a . keys(): # 'x' is each element of the list in dict_keys object
  print(x) # 10 <next line> 20 <next line> 15 <next line> 18 <next line>
print('Values of dictionary')
for x in a . values(): # 'x' is each element of the list in dict_values object
  print(x) # Ramesh <next line> Kiran <next line> Amar <next line> Sita <next line>
print('All the tuples of dict_items object')
for x in a . items(): # 'x' is each tuple of the list in dict_items object
  print(x) # (10 , 'Ramesh') <next line> (20 , 'Kiran') <next line> (15 , 'Amar') <next line> (18 , 'Sita') <next line>
print('Elements of each tuple')
for x , y in a . items(): # 'x' and 'y' are elements of each tuple of the list of dict_items object
  print(x , y , sep = '...') # 10 ... Ramesh <next line> 20 ... Kiran <next line> 15 ... Amar <next line> 18 ... Sita <next line>
print('Keys and values of dictionary')
```

```
for x in a . keys(): # 'x' is each element of the list in dict_keys object
  print(x , a[x] , sep = ' : ') # 10 : Ramesh <next line> 20 : Kiran <next line> 15 : Amar <next line> 18 : Sita <next line>
'''
1) for x in dictionary:
       print(x)
   Is the above for loop valid ? ---> Yes becoz keys() method is executed by default
2) for x in dictionary:
       print(x)
   How is the above for loop interpreted ? ---> for x in dictionary . keys()
                                                    print(x)
3) for x , y in dictionary . keys():
   print(x , y)
   Is the above for loop valid ? ---> No becoz two variables are not permitted for keys() method
4) for x , y in dictionary . values():
   print(x , y)
    Is the above for loop valid ? ---> No becoz two variables are not permitted for values() method
5) When are two variables permitted in for loop ? ---> Only for items() method
6) for x , y in dictionary:
       print(x , y)
   Is the above for loop valid ? ---> No becoz the above for loop is interpreted as for x , y in dictionary . keys():
                                       and two variables are not permitted for keys() method
7) a = {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
   for x in a . items():
  print(x[0] , x[1] , sep = '...')
   What is 'x' in the above for loop ? ---> Each tuple of the list in dict_items object
   What are x[0] and x[1] ? ---> Elements of each tuple
8) a = {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
   for x in a . items():
  print(*x)
   What does *x do ? ---> Unpacks tuple into elements
9) a = {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
    for x in a . keys():
     print(x)
   Iteration x
   ------------------------
      1 10
      2 20
      3 15
      4 18
10) a = {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
    for x in a . values():
       print(x)
   Iteration x
   -----------------------
```

```
         1 'Ramesh'
         2 'Kiran'
         3 'Amar'
         4 'Sita'
11) a = {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
     for x in a . items():
         print(x)
    Iteration x
  ------------------------------------
           1 (10 , 'Ramesh')
           2 (20 , 'Kiran')
           3 (15 , 'Amar')
           4 (18 , 'Sita')
12) a = {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
      for x , y in a . items():
        print(x , y , sep = ':')
     Iteration x y
    ------------------------------
          1 10 'Ramesh'
          2 20 'Kiran'
          3 15 'Amar'
          4 18 'Sita'
13) a = {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
     for x in a . keys():
        print(x , a[x])
    Iteration x a[x]
    ------------------------------------
         1 10 'Ramesh'
         2 20 'Kiran'
         3 15 'Amar'
         4 18 'Sita'
'''
```

Summary
----
1) How many objects are in python ? ---> 5 + 6 = 11
2) How many objects are non-sequences and what are they ? ---> 5 i.e. int , float , complex , bool , NoneType
How many objects are sequences and what are they ? ---> 11 - 5 = 6
i.e. str , range , list , tuple , set and dict
3) What is a sequence ? ---> A group of elements
What is a non-sequence ? ---> A single element
4) Which sequences are homogeneous (Total : 2) ? ---> str and range
Which sequences are heterogeneous (Total : 4) ? ---> list , tuple , set and dict
5) Which sequences can not hold duplicates (Total : 3) ? ---> dict , set and range
6) Which sequences are not indexed (Total : 2) ? ---> set and dictionary
7) Which objects are mutable (Total : 3) ? ---> list , set and dict
Which objects are immutable (Total: 5 + 3 = 8) ? ---> int , float , complex , bool , NoneType ,
and
tuple , str , range
8) Which sequences can not be repeated (Total: 3) ? ---> set , dict and range
9) Are non-sequences indexed ? ---> No due to single element
10) Can non-sequences be repeated ? ---> No becoz * operator does multiplication but not repetition
11) What is the argument of len() function (sequence (or) non-sequence) ? ---> Sequence
12) Does python support variables ? ---> No and python supports only objects
13) Is python 100% object oriented language (OOL) ? ---> Yes becoz there are only objects but not variables
and
there are only classes but not datatypes

Summary
----
1) What is 25 called ? ---> An int class object
   What is 10.8 called ? ---> A float class object
   What is 3 + 4j called ? ---> complex class object
   What are True and False called ? ---> bool class objects
   What is None called ? ---> A NoneType class object
2) How many int objects are there ? ---> Infinite
   How many float objects are there ? ---> Infinite
   How many complex objects are there ? ---> Infinite
   How many bool objects are there ? ---> Just two i.e. True and False
   How many NoneType objects are there ? ---> Just one i.e. None

```
# Gift
# Find outputs (Home work)
a = {
  print('Hyd') ,
  print('Sec') ,
  print('Cyb')
     } # a = {None , None , None} ---> a = {None}
print(type(a)) # <class 'set'>
print(a) # {None}
print(len(a)) # 1
'''
1) {
   print('Hyd'),
   print('Sec'),
   print('Cyb')
    }
    Is it a suite ? ---> No and it is a set due to { }
2) What does print('Hyd') do ? ---> Prints Hyd and Returns None
     What does print('Sec') do ? ---> Prints Sec and Returns None
     What does print('Cyb') do ? ---> Prints Cyb and Returns None
3) Finally it is set of a single None as set can not hold duplicates
    i.e. {None , None , None} ---> {None}
'''
```

```
# Identify Error
print('Hyd')
 print('Sec') # Error due to spaces before the statement
  print('Cyb') # Error due to spaces before the statement
'''
Suite (or) Block
--------------------
1) What is a suite ? ---> A group of statements
                          Eg: stmt1
                                 stmt2
                               stmt3
                                   ....
2) What is indentation ? ---> Statements of the suite should be in the same column and
                        there should not be spacebar (or) tab key before the statement
3) In other words, every suite should be indented
4) Invalid: stmt1
                 stmt2
                       stmt3
5) {
        stmt1
        stmt2
        stmt3
    }
    Can suite be in braces ? ---> No
6) In other words, braces can be used for set and dictionary but not for suite
'''
```

Types of integers
----
1) Binary integer
2) Octal integer
3) Decimal integer
4) Hexa-Decimal integer

Binary integer
----
1) What is the prefix of binary number ? --->
0B (or) 0b
2) What are the valid digits in binary number ? --->
0 and 1
3) What is the base of binary number ? --->
2 due to two digits 0 and 1
4) Which digits are not permitted in binary number ? --->
2 to 9
5) a = 0B10101
   What does object 'a' contain ? --->
   Decimal equivalent
6) In other words, binary number is automatically converted to decimal number and
   decimal number is stored in the object
7) Object will never hold binary number
8) What does print(binary number) do ? --->
Prints decimal equivalent of the number

```
# Find outputs
a = 0B10101
print(a)
print(type(a))
print(id(a))
b = 0b10101
print(b)
print(id(b))
c = 21
print(c)
print(id(c))
d = 10101
print(d)
e = 0B1234
'''
```

1) Conversion of binary number to decimal

```
       16 8 4 2 1 ---> Weights
```

```
  1 0 1 0 1 ---> 16 + 4 + 1 = 21
2) a = 0B10101
   b = 0b10101
   c = 21
   How many objects are there ? --->
 Single object with three references and
                                                  all the three references point to the same object
'''
Octal integer
----------------
1) What is the prefix of octal number ? --->
0o (or) 0O
2) What are the valid digits in octal number ? --->
0 to 7
3) What is the base of octal number ? --->
8 due to eight digits 0 to 7
4) Which digits are not permitted in octal number ? --->
8 and 9
5) a = 0O6247
   What does object 'a' contain (octal number (or) decimal equivalent) ? --->
   Decimal equivalent
6) In other words, octal number is automatically converted to decimal number and
    decimal number is stored in the object
7) What does print(octal number) do ? --->
Prints decimal equivalent of the number

# Find outputs (Home work)
a = 0O6247 # Object contains decimal equivalent i.e. 6 * 8 ^ 3 + 2 * 8 ^ 2 + 4 * 8 ^ 1 + 7 * 8 ^ 0 = 3239
print(a) # 3239
print(type(a))# <class 'int'>
print(id(a)) # Address of object 3239
b = 0o6247 # ref 'b' points to same object 3239
print(id(b)) # Same address
print(b) # 3239
c = 3239 # ref 'b' points to same object 3239
print(c) # 3239
print(id(c)) # Same addess
#print(0o9248) # Error due to 9 and 8
'''
1) Conversion of octal number to decimal
    ------------------------------------------------
        512 64 8 1 ---> Weights
     6 2 4 7 ---> 6 * 512 + 2 * 64 + 4 * 8 + 7 * 1 = 3239
2) a = 0o6247
   b = 0O6247
```

```
    c = 3239
    How many objects are there ? ---> Single object with three references a , b and c and
                              all the three references point to the same object
'''
Hexa Decimal integer
-------------------------
1) What is the prefix of hexa-decimal number ? --->
0X (or) 0x
2) What are the valid characters in hexa-decimal number ? --->
0 to 9 , A to F and a to f
3) What is the value of A ? --->
     What is the value of B ? --->
    What is the value of C ? --->
    What is the value of D ? --->
    What is the value of E ? --->
    What is the value of F ? --->
4) What is the base of hexa-decimal number ? --->
6 + 10 = 16 due to 10 digits and 6 alphabets
5) a = 0XA7B9
    What does object 'a' contain (Hexa-decimal number (or) decimal equivalent) ? --->
    Decimal equivalent
6) In other words, hexa decimal number is automatically converted to decimal number and
     decimal number is stored in the object
7) What does print(hexa-decimal-number) do ? --->
Prints decimal equivalent of the number

# Find outputs (Home work)
a = 0XA7B9 # Object contains decimal equivalent i.e. 10 * 16 ^ 3 + 7 * 16 ^ 2 + 11 * 16 ^ 1 + 9 * 16 ^ 0 = 42937
print(a) # 42937
print(type(a)) # <class 'int'>
b = 0xBEEF # 11 * 16 ^ 3 + 14 * 16 ^ 2 + 14 * 16 ^ 1 + 15 * 16 ^ 0
print(b) # 48879
#print(A7B9) # Error becoz 0X is missing
print('A7B9') # A7B9
#print(0XBEER) # Error due to 'R'
#print(0XHYD) # Error due to 'H' and 'Y'
#print(0xA7G9B) # Error due to 'G'
'''
Conversion of hexa decimal number to decimal
----------------------------------------------------------
     4096 256 16 1 ---> Weights
      A 7 B 9 ---> 10 * 4096 + 7 * 256 + 11 * 16 + 9 * 1 = 42937
'''
Decimal integer
------------------
```

1) What is the prefix of decimal number ? --->

Nothing

2) What are the valid digits in decimal number ? --->

0 to 9

3) What is the base of decimal number ? --->

10 due to ten digits(0 to 9)

```
# Find outputs (Home work)
a = 9248 # Decimal number
print(a) # 9248
print(type(a)) # <class 'int'>
```

Summary

Property	Binary number	Octal number	Decimal number	Hexa-decimal number
Base	2	8	10	16
Prefix	0B (or) 0b	0O (or) 0o	Nothing	0X (or) 0x
Valid characters	0 to 1	0 to 7	0 to 9	0 to 9 , A to F (or) a to f

```
# Anonymous object demo program
_ = 25 # Anonymous object contains 25
print(_) # Value of nameless object i.e. 25
print(type(_)) # <class 'int'>
a , _ , c = 10 , 20 , 30 # Multiple assignment
print(a) #10
print(_) # 20
print(c) # 30
for _ in range(5):
 print(_ , 'Hello') # 0 <space> Hello <next line> 1 <space> Hello <next line> 2 <space> Hello <next line> 3 <space> Hello <next line> 4 <space> Hello <next line>
'''
```

1) What is _ called ? ---> Anonymous object (or) Nameless object

2) How many total objects are in the above program ? ---> 1 + 3 + 5 = 9

   How many objects alive are in the above program ? ---> 3 i.e. a , c and nameless object

3) How many objects are nameless in the above program ? ---> One at a time

4) In other words, old nameless object is lost every a new nameless object is created

'''