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) Python 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 classs

 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, python 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. Python 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)

e) Python 3.13(2024)

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)

e) Python 3.13(2024)

```
# int object demo program
a = 25 # Ref 'a' points to object 25
print(a) # Value of object 'a' i.e. 25
print(type(a)) # Type of object 'a' i.e. <class 'int'>
print(id(a)) # Address of object 'a' (may be 1000)
b = 99999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999 # Valid
```

print(b)

#999999999999999999999999999999999999999999999999999999999999999999999999999999999
99999

c = $75 # Error due to $

1) What does a = 25 do ? ---> Assigns reference 'a' to int object 25

2) a = 25

What is 25 called ? ---> Object i.e. An int class object

What is 'a' called ? ---> Reference

3) a = 25

What does object 'a' contain ? ---> Integer number 25

What does reference contain ? ---> Address (or) id of the object

4) a = 25

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

int object

---

1) What can an int object hold ? ---> An integer number such as 25

2) What is an integer ? ---> A number without decimal point

3) What is 9247 called ? ---> Positive integer

What is -682 called ? ---> Negative integer

4) What is the maximum value of integer ? ---> Infinity

What is the minimum value of integer ? ---> -Infinity

5) Why python does not support long integer ? ---> Since max value of integer is infinity

6) What is 25 called in other languages (value (or) object) ? ---> Value

What about python ? ---> Object

7) Is $75 valid ? ---> No due to special character $

8) Is int a user defined class (or) pre-defined ? ---> A pre-defined class

9) Where is int class defined ? ---> In builtins module

10) What is a module ? ---> File

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

5) Are nested comments valid ? ---> No i.e. There can not be a comment in a comment

6) ''' ---> Comment begin

Line1

Line2

Line3

''' ---> Comment end

Line4 ---> Error

Line5 ---> Error

Line6 ---> Error

''' ---> Comment begin

Line7

Line8

Line9

''' ---> Comment End

7) What about nested single line comment ? ---> Permitted

Eg: # Nested single # line comment

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 of representing a float number ? --->

Fractional number and Mantissa-Exponent number

5) What is 123.45 called ? ---> A fractional number

6) What is 9.728e3 called ? ---> A 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 i.e. float class 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

```
# 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 10.8 (may be 1000)
b = 25. # Valid and interpreted as 25.0
print(b) # 25.0
print(type(b)) # <class 'float'>
c = .689 # # Valid and 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 more than one decimal point
```

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) Is 9 + j valid ? ---> No becoz imag is missing

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

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

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

```
# complex object demo program

a = 3 + 4j  # Ref 'a' points to object 3 + 4j
print(a)  # 3+4j
print(type(a))  # Type of object 'a' i.e <class 'complex'>
print(id(a))  # Address of object 3 + 4j (may be 1000)
print(a . real)  # 3.0
print(a . imag)  # 4.0
print(type(a . real)) # <class 'float'>
print(type(a . imag))  # <class 'float'>
What is the type of real and imag ? ---> Always float

# Find outputs (Home work)

a = 6j  # Ref 'a' points to object 6j
print(a) #  6j
print(type(a))  # Type of object 'a' i.e. <class 'complex'>
print(a . real) #  0.0
print(a . imag) #  6.0
#print(5 + j6)  # Error : imag can not be after 'j'
#print(3 + 4i) # Error due to 'i'
#print(4+j)  # Error : imag is missing
print(4 + 1j)  #  4 + 1j
print(4 + 0j)  #  4 + 0j

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 built-in module

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

bool object demo program

```
a = True  # Ref 'a' points to object True
print(a) # Value of object 'a' i.e. True
print(type(a)) # Type of object 'a' i.e. <class 'bool'>
print(id(a)) # Address of object True
b = False  # Ref 'b' points to object False
print(b)  # Value of object 'b' i.e. False
print(type(b))  # Type of object 'b' i.e. <class 'bool'>
print(True + True)  # 1+1 = 2
print(True + False) # 1+0 = 1
print(False + True)  # 0+1 = 1
print(False + False) # 0+0 = 0
print(True + True + True) # 1+1+1 = 3
print(25 + 10.8 + True) # 36.8
print(True > False) # 1 > 0 is True
print(True)  # True
print(False) # False
#print(true)  # Error due to 't'
#print(false) # Error due to 'f'
```

Points to remember

---

1) Does python have main() function ? ---> No

2) int a;

float b;

Are the 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 no datatypes and also there are only objects but no 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

18) Who runs python program ? ---> PVM (python virtual machine)

19) Is PVM a software (or) hardware ? ---> Software i.e. A group of programs

```
# NoneType object demo program
a = None #  Ref 'a' points to object None
print(type(a)) # <class 'NoneType'>
print(a) #  None
print(id(a)) # Address of object None
#print(none)  # Error due to 'n'
```

1) Is NoneType a class (or) object ? ---> class

   What about None ? ---> Object

2) Where is NoneType class defined ? ---> In built-in module

3) Is None a user defined word (or) keyword ? ---> Keyword

Summary

---

1) What is 25 called ? ---> An int class object

   What is 10.8 called ? ---> A float class object

   What is 3 + 4j called ? ---> A 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

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 ? ---> The 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 contain binary number

8) What does print(binary-number) do ? ---> Prints decimal equivalent of the number

Find outputs

a = 0B10101 # Object contains decimal equivalent i.e. 16 + 4 + 1 = 21

print(a) # 21 decimal only

print(type(a)) # <class 'int'>

print(id(a)) # Address of object 21 (say 1000)

b = 0b10101 # Ref 'b' points to same object 21

print(b) # 21

print(id(b)) # Same address

c = 21 # Ref 'c' points to same object 21

print(c) # 21

print(id(c)) # Same address

d = 10101 # Decimal number

print(d) # 10101

#e = 0B1234 # Error due to 2 , 3 and 4

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

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 (may be 1000)

b = 0o6247 # Ref 'b' points to same object (int object is reusable)

print(id(b)) # Same address

print(b) # 3239

c = 3239 # Ref 'c' points to same object (int object is reusable)

print(c) # 3239

print(id(c)) # Same address

#print(0o9248) # Error due to 9 and 8 in octal number

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

Find outputs (Home work)

```
a = 0O6247
print(a)
print(type(a))
print(id(a))
b = 0o6247
print(id(b))
print(b)
c = 3239
print(c)
print(id(c))
print(0o9248)
```

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 ? ---> 10

What is the value of B ? ---> 11

What is the value of C ? ---> 12

What is the value of D ? ---> 13

What is the value of E ? ---> 14

What is the value of F ? ---> 15

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) ? ---> The decimal equivalent

6) In other words, hexa decimal number is automatically converted to decimal number and

decimal number is stored in the object

7) Finally object will never contain hexa-decimal number

8) 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 # Object contains decimal equivalent i.e. 11 * 16 ^ 3 + 14 * 16 ^ 2 + 14 * 16 ^ 1 + 15 * 16 ^ 0 = 48879

print(b) # 48879

#print(A7B9) # Error : Not starting with 0X

print('A7B9') # A7B9

#print(0XBEER) # Error due to 'R' in hexa-decimal number

#print(0XHYD) # Error due to 'H' and 'Y' in hexa-decimal number

#print(0xA7G9B) # Error due to 'G' in hexa-decimal number

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
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

Module → 2

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) In other words, it is not possible to modify characters of the string

10) Is str object indexed ? ---> Yes

11) 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

12) What is the index of 10th character from left to right ? ---> 9

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

13) 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'

14) 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'

15) What is the advantage of indexes ? ---> Random access

16) What is random access ? ---> It is possible to access 10th character of the string directly without accessing first nine characters

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

What does 'Hyd' * 3 do ? ---> Repeats 'Hyd' thrice i.e. 'HydHydHyd'

18) What does len('Hyd') do ? ---> Returns number of characters in 'Hyd' i.e. 3

19) Where is str class defined ? ---> In builtins module

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)

```
# Find outputs (Home work)
a = "Rama Rao"  # Ref 'a' points to object "Rama Rao"
print(a) #  Rama Rao
print(type(a))  # <class 'str'>
print(id(a)) # Address of object  "Rama Rao"
b = 'Hyd'  # Ref 'b' points to object  'Hyd'
print(b) # Hyd
c = '''Hyd is green city.
Hyd is hitec city.
Hyd is beautiful city.'''  # Multi line string
print(c) # Hyd is green city.  <next line>  Hyd is hitec city.  <next line>  Hyd is beautiful city.
```

1) When are triple quotes used ? ---> For multi-line string and also multi-line comment

2) When is it called multi-line string ? ---> When it is assigned to an object

3) When is it called multi-line comment ? ---> When it is not assigned to any object

4) Which operator is used for single line comment ? ---> #

```
# Index demo program (Home work)
a = 'Hyd'
print('Hyd'[0]) # Char at index 0 of object 'Hyd' i.e. 'H'
print('Hyd'[1]) # Char at index 1 of object 'Hyd' i.e. 'y'
print('Hyd'[2]) # Char at index 2 of object 'Hyd' i.e. 'd'
#print(a[3]) # Error : Index 3 does not exist in 'Hyd'
print('Hyd'[-1]) # Char at index -1 of object 'Hyd' i.e. 'd'
print('Hyd'[-2]) # Char at index -2 of object 'Hyd' i.e. 'y'
print('Hyd'[-3]) # Char at index -3 of object 'Hyd' i.e. 'H'
#print(a[-4]) # Error : Index -4 does not exist in 'Hyd'
print(a[0] == a[-3]) # 'H' == 'H' is True
#a[2] = 'c' # Error : 'd' can not be replaced with 'd' as str object is immutable
#print(25[0]) # Error : Non-sequence (such as int) is not indexed
print('25'[0]) # Char at index 0 of object '25' i.e. '2'
#print(True[1]) # Error : Non-sequence (such as bool) is not indexed
print('True'[1]) # Char at index 1 of 'True' i.e. 'r'

1) What is another name of index ? ---> Subscript
2) What does == operator do ? ---> Compares objects
3) What does = operator do ? ---> Assigns refernce to an object

# Find outputs (Home work)
a = 'Hyd'
print(a * 3) # Repeats string thrice i.e. HydHydHyd
print(a * 2) # Repeats string twice i.e. HydHyd
print(a * 1) # Repeats string once i.e. Hyd
print(a * 0) # Repeats string 0 times i.e. Empty string
print(a * -1) # Repeats string -1 times i.e. 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) # Repeats string once i.e. 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

```
# Find outputs (Home work)
a = 'Hyd'
print(a , id(a)) # Hyd <space> Address of 'Hyd'
a = a * 3 # Ref 'a' is modified to a new string object 'HydHydHyd'
print(a , id(a)) # HydHydHyd <space> Address
'
```

1) a = 'Hyd'
   a = a * 3
   What is modified ? ---> Ref 'a' is modified to the new string 'HydHydHyd'

2) Why is object not modified ? ---> Since str object is immutable

```
# len() function (Home work)
print(len('Hyd')) # 3
print(len('Rama Rao')) # 8
print(len('9247')) # 4
print(len('')) # 0 due to empty string
print(len(' ')) # 1 due to space
#print(len(689)) # Error : Argument should be a sequence but 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 a char of the string '("Hyd)' is ok

print(a) # "Hyd

print(len(a)) # 4

print(a[0]) # "

#print(""""Hyd""""") # Error due to excess closing quote

b = """""Hyd""" # Excess opening quotes are characters 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 ? ---> A new string with desired characters

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 i.e. Dayal

print(a[7 :]) # a[7:18:1] ----> string from index 7 to 17 insteps of 1 i.e. Dayal Sarma

print(a[: 6]) # a[0:6:1] ---> string from index 0 to 5 in step of 1 i.e. Sankar

print(a[:]) #a[0:18:1] ---> string from index 0 to 17 instep of 1 i.e. Sankar Dayal Sarma

print(a[: :]) # a[0 : 18 : 1] ---> string from indexes 0 to 17 in steps of 1 i.e. Sankar Dayal Sarma

print(a[1 : 10 : 2]) #a[1:10:2] ---> string from indexes 1 to 9 in steps of 2 i.e. akrDy

print(a[0 : : 2]) #a[0:18:2] ---> string from indexes 0 to 17 in steps of 2 i.e. Sna<space>aa<space>am

print(a[1 : : 2]) #a[1:18:2] ---> string from indexes 1 to 17 in steps of 2 i.e. akrDylSra

print(a[-5 : -1]) #a[-5:-1:1] ---> string from indexes -5 to -2 insteps of 1 i.e.Sarm

print(a[::-1]) # a[-1 : -19 : -1] ---> string from indexes -1 to -18 in steps of -1 i.e. Reverse string

print(a[-1:-5:-1]) #string from indexes -1 to -4 in steps of -1 i.e. amra

print(a[: : -2]) #a[-1:-19:-2]--->string from indexes -1 to -18 in steps of -2 i.e. arSlyDrka

print(a[3 : -3]) # a[3 : -3 : 1] ---> string from indexes 3 to -4 in steps of 1 i.e. kar<space>Dayal<space>Sa

print(a[2 : -5]) #a[2:-5:1] --->string from indexes 2 to -6 in steps of 1 i.e. nkar<space>Dayal<space>

print(a[-1:-5]) #a[-1:-5:1]--->string from indexes -1 to -6 in steps of 1 i.e. Empty string

print(a[3 : 3]) #a[3:3:1]--->string from indexes 3 to 2 in steps of 1 i.e. Empty string

#	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) How to obtain first two characters of the string ? ---> string[: 2]

How to obtain first three characters of the string ? ---> string[: 3]

How to obtain first 'n' characters of the string ? ---> string[: n]

2) How to obtain last two characters of the string ? ---> string[-2 :]

How to obtain last three characters of the string ? ---> string[-3 :]

How to obtain last 'n' characters of the string ? ---> string[-n :]

3) How to obtain whole string without first two characters ? ---> sting[2 :]

How to obtain whole string without first three characters ? ---> string[3 :]

How to obtain whole string without first 'n' characters ? ---> string[n :]

4) How to obtain reverse string ? ---> string[::-1]

Find outputs (Home work)

a = 'A'

#print(a[1]) # Error becoz index 1 does not exist in 'A'

print(a[1:]) # String without 1st char i.e. ''

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

Type-casting (or) Type-conversion 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()

4) Where are the above functions defined ? ---> In builtins module

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')) # Converts '0075' to 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 : complex number can not be converted to int

#print(int('25.4')) # Error : string float can not be converted to int

#print(int('Ten')) # Error : 'Ten' can not be converted to int
```

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 : complex number can not be converted to float
#print(float('Ten')) # Error : '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 : string float can not be converted to integer

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

```
# 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 : 2nd argument can not be string
#print(complex('3' , 4)) # Error : 2nd argument is not permitted as first argument is a string
#print(complex('3' , '4')) # Error : 2nd argument is not permitted as first argument is a string
#print(complex('Ten')) # Error : 'Ten' can not be converted to complex

# bool() function demo program
print(bool(0)) # False due to 0
print(bool(10)) # True : 10 is non-zero
print(bool(-25)) # True : -25 is non-zero
```

```
print(bool(0.0)) # False due to 0.0
print(bool(0.1)) # True : 0.1 is non-zero
print(bool(0 + 0j)) # False : Both real and imag are zeroes
print(bool(10 + 20j)) # True : real is non-zero
print(bool(-15j)) # True : imag is non-zero
print(bool('False')) # True : 'False' is non-empty string
print(bool('')) #False due to empty string
print(bool('Hyd')) #True : 'Hyd' is a non-empty string
print(bool(' ')) #True : ' ' is non-empty string
print(bool('True')) #True : '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

bin() function demo program

print(bin(25)) # Converts decimal number 25 to binary number i.e. 0B11001

print(bin(0O6247)) # Converts octal number to binary number i.e. 0B110 010 100 111

print(bin(0XA7B9)) # Converts hexa-decimal number to binary number 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

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

---

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

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 ? ---> With * operator
i.e. 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) What is the syntax of slice ? ---> a[begin : end : step] where 'a' is range object

19) What is obtained when range object is sliced ? ---> A new range object with sliced elements

20) Can range object have duplicate elements ? ---> No becoz step can not be 0

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

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

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 = ' , ') # 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 : Element of range object cannot be modified as it is immutable

print(a * 2) #Error : range object can not be repeated as duplicate elements are not permitted

	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

1) What is obtained when range object is sliced ? ---> A new range object with sliced elements

2) What does range-object[::-1] do ? ---> A new range object is obtained with elements in reverse order

3) How many range objects exist in the above program ? ---> Three

4) What does 1st range object contain ? ---> Elements from 10 to 45 in steps of 5

What does 2nd range object contain ? ---> Elements from 20 to 40 in steps of 5

What does 3rd range object contain ? ---> Elements from 45 downto 10 in steps of -5

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 , ... 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  <space>  ...  -1
e = range(-10)  #  range(0 ,  -10 , 1)  --->  Empty  object  due  to  0 >=  -10
print(*e)  #  Unpacks  empty  object  i.e.  Nothing
f = range(2 , 2)  #  range(2 , 2 , 1)  ---> Empty  object  due  to  2 >=  2
print(*f)  #  Unpacks  empty  object  i.e.  Nothing
#g = range(10 , 11 , 0.1) # Error :  range  object can  not  hold  float  elements
#h = range('A' , 'F') # Error : 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
3) This  rule  is  applicable  to  range  object  only  but  not  for  slice

#  Find  outputs  (Home  work)
r = range(10 ,  17 , 3)  #Object  contains  elements  from  10  to  16  in  steps  of  3
a , b , c = r  #  Unpacks  range  object  to  3  elements
print(a , b , c) #  10 <space>  13  <space>  16
r = range(3)  #Object  contains  elements  from  0  to  2  in  steps  of  1
#x , y = r  #  Error :  Excess  elements  in  range  object
#p , q  , r , s = r  #  Error :  Few  elements  in  range  object
#a , b , c = *r  #  Error  due  to  *  operator
m = r  #  Ref  'm'  points  to  object  'r'
print(id(r))  #  Address  of  range  object  'm'
print(id(m))  # Same  address
```

1) x , y , z = range-object

What does the above statement do ? ---> Unpacks range object to 3 elements

2) x = range-object

What does the above statement do ? ---> Assigns ref 'x' to same range object

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 [] operator 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

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

12) What is obatined when list is sliced ? ---> A new list with sliced (or) desired elements

13) 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

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

15) list = [10 , 20 , 15]

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

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

17) 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

18) In other words, list is growable and shrinkable

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

20) list = [10 , 20 , 15]

What does list * 2 do ? ---> Repeats list twice and produces a new list with 6 elements

i.e. [10 , 20 , 15 , 10 , 20 , 15]

21) 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 .....

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

Find outputs (Home Work)

a = [25 , 10.8 , 'Hyd' , True , 3 + 4j , None , 'Hyd' , 25] # 'a' is 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 of list 'a' 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]

1) What is obtained when list is sliced ? ---> A new list with sliced elements

2) How many lists are in the above program ? ---> Two lists i.e. Original list and sliced list i.e. a[2 : 5]

How to print list in different ways (Home work)

```
a = [25 , 10.8 , 'Hyd' , True]
print('List with print function')
print(a) # How to print list ---> [25 , 10.8 , 'Hyd' , True]
print('Iterate thru list without using indexes')
for x in a: # 'x' is each element of list 'a'
        print(x) # 25 <next line> 10.8 <next line> Hyd <next line> True <next line>
print('Iterate thru list using indexes')
for i in range(len(a)):
                print(a[i]) # prints a[i] where 'i' varies from 0 to len - 1
print('Iterate thru list in reverse order')
for i in range(1 , len(a) + 1):
                print(a[-i]) # prints a[-i] where 'i' varies from 1 to len
print('Reverse List with slice : ' , 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  empty  list
a . append(10.8)  #  Appends  10.8  to  list
a . append('Hyd')  #  Appends  'Hyd'  to  list
a . append(True)  #  Appends  True  to  list
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 :  No  '25'  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'] #  Ref 'a' points to list of 3 elements
print(a) #[25 , 10.8 , 'Hyd']
print(id(a)) # Address of list with 3 elements
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
a = a * 3 # Ref 'a' is modified to list of 9 elements
print(a) # [25 , 10.8 , 'Hyd' , 25 , 10.8 , 'Hyd' , 25 , 10.8 , 'Hyd']
print(id(a)) # Address of list with 9 elements
a = [25] # Ref 'a' is modified to list of single element
print(a , id(a)) # [25] <space> Address of list with single element
#print(a * a) # Error : operand2 should be an integer but not list

1) What does list * n do ? ---> Repeats list for 'n' times
2) How many elements are in the resultant list ? ---> n * len(list)
3) a = [10,20,30]
   a = a * 2
        What is modified (List (or) reference) ? ---> Reference but not list

4) In other words, ref 'a' points to a new list of 6 elements

# 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  i.e.  [10 , 20 , 15 , 18]

print(list(range(5)))  #  Converts   range   object to  list  i.e.  [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 denote 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]) #[-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]) #[-1:-9:-2]--->List from indexes -1 to -8 in steps of -2 i.e['Hyd', None, 'Hyd', 10.8]

print(list[-2 : : -2]) #[-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]) #[1:4:1]--->List from indexes 1 to 3 in steps of 1 i.e[10.8, (3+4j), 'Hyd']

print(list[-4 : -1])#[-4:-9:-1]--->List from indexes -4 to -8 in steps of -1 i.e[True, 'Hyd', (3+4j), 10.8, 25]

print(list[3 : -3]) # 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 []

1) How to obtain list with first two elements ? ---> list[: 2]

   How to obtain list with first three elements ? ---> list[: 3]

   How to obtain list with first 'n' elements ? ---> list[: n]

2) How to obtain list with last two elements ? ---> list[-2:]

   How to obtain list with last three elements ? ---> list[-3:]

   How to obtain list with last 'n' elements ? ---> list[-n:]

3) How to obtain list without first two elements ? ---> list[2:]

   How to obtain list without first three elements ? ---> list[3:]

   How to obtain list without first 'n' elements ? ---> list[n:]

4) How to obtain reverse list ? ---> list[::-1]

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 = ['Hyd' , True]

print('x : ' , x) # x : Hyd

print('y : ' , y) # y : True

for x in list[2:7]: # for x in [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>

for x in list:

print(x)

for x in list[2 : 7]:

print(x)

What is the difference between the two for loops ? ---> First for loop iterates thru whole list but

2nd for loop iterates thru sliced list

Find outputs (Home work)

0 1 2 3 4

a = [10 , 20 , 30 , 40 , 50]

print(a , id(a)) #[10 , 20 , 30 , 40 , 50] <space> Address of list

a[1 : 4] = [60 , 70] # Modifies elements of list from indexes 1 to 3 to 60 , 70

print(a , id(a)) # [10 , 60 , 70 , 50] <space> Same address

a[2: 4] = [100 , 200 , 300] # Modifies elements of list from indexes 2 to 3 to 100 , 200 , 300

print(a , id(a)) #[10,60,100,200,300] <space> Same address

1) a[1 : 4] = [60 , 70]

Does a[1 : 4] generate a new list (or) modifies elements of existing list ? ---> Modifies elements existing of list

2) In general , slice on left side of = modifies list elements

3) b = a[1 : 4]

What about a[1 : 4] on right side ? ---> Generates a new list

4) a[1 : 4] = [60 , 70]

How many elements are modified (single element / multiple elements) ? ---> Multiple elements

Find outputs (Home work)

a = [25]

#print(a[1]) # Error becoz index 1 does not exist in [25]

print(a[1:]) # List without first element i.e. []

Index may throw error but slice never throws error

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

Simlilarities between list and tuple

---

1) Heterogeneous

2) Duplicate elements

3) Repeat

4) Indexes

5) Slice

6) len()

Find outputs (Home work)

a = (25 , 10.8 , 'Hyd' , True , 3+4j , None , 'Hyd' , 25)

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 indexes 2 to 4 in steps of 1 i.e ('Hyd',True,3+4j)

print(*a[2 : 5]) # Unpacks tuple from indexes from 2 to 4 in steps of 1 i.e Hyd <space> True <space> 3+4j

#a[2] = 'Sec' # Error : Element of tuple can not be modified as it is immutable

#a . append('Sec') # Error : No append() method in tuple

#a . remove('Hyd')# Error : No remove() method in tuple

b = 10 , 20 , 30 # Valid : () 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'>

1) What happens when tuple is sliced ? ---> A new tuple with sliced elements

2) What happens when tuple is repeated ? ---> A new tuple with repeated elements

3) How many tuples are in the above program ? ---> 6

```
# Find outputs (Home work)
a = (25) # Integer : comma is missing
b = 25, # Tuple due to comma
c = 25 # Integer : 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) # Repeats tuple 4 times i.e. (25,25,25,25)
print(c * 4) # 25 * 4 = 100
print(d * 4) # Repeats 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]  # List due to []
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 : 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()  # Returns an empty tuple
print(b) # ()
print(len(b)) # 0
```

1) When are ( ) mandatory for tuple ? ---> Empty tuple

2) When are ( ) optional for tuple ? ---> When there is at least one element in the tuple

3) What are the two ways to denote an empty tuple ? ---> ( ) and tuple(No-args)

```
# Gift
# Find outputs (Home work)
a = (10 , 20 , 30)  # Ref 'a' points to tuple of 3 elements
print(a) # (10,20,30)
print(id(a)) # Address of tuple with 3 elements (may be 1000)
a = a * 2  # a = (10 , 20 , 30 , 10 , 20 , 30)  ---> Ref 'a' is modified to another 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) What happens to tuple of 3 elements when ref 'a' is modified to tuple of 6 elements ? →

Gets deleted by pvm as there is no reference to it

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 (like list and tuple)

4) Can set hold duplicate elements ? ---> No becoz set can hold unique elements (like range object)

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 random access is not possible as there are no indexes

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

14) How to access set elements ? ---> With for loop (sequential access)

i.e. for x in set:
    print(x)

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

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

24) Is [[10,20,15,18] , (10 , 20, 30)] valid ? ---> Yes becoz list can hold both mutable and immutable elements

Is ([10,20,15,18] , (10 , 20, 30)) valid ? ---> Yes becoz tuple can hold both mutable and immutable elements

25) In other words, nested list and nest tuple are valid but not nested set

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

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

Note:

How is set different from remaining sequences (total : 5) ? --->

1) set is unordered

2) set is not indexed

3) Set is mutable but not 100%

4) Set can not hold duplicate elements

5) Set can not hold mutable elements

```
# set object demo program (Home work)
a = {25 , 10.8 , 'Hyd' , True , 3+4j , None , 25 , 'Hyd'}
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 : set is not indexed
#print(a[1 : 4]) # Error : set can not be sliced
#a[2] = 'Sec'# Error : Element of set can not be modified as there is no index
#print(a * 2) # Error : set can not be repeated duplicate elements
#print(a * a) # Error : sets can not be multiplied
```

1) a = {25 , 10.8 , 'Hyd' , True}

   print(a)

Is set printed in same order every time program is executed ? ---> Not guranteed

2) In other words, order may change from run to run

3) a = {25 , 10.8 , 'Hyd' , True}

   print(a)

   print(a)

   print(a)

Is set printed in same order all the three times ? ---> Yes becoz it is the same set which is printed thrice

```
# 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) a = {True , 1 , 1.0}

Which element will be in set 'a' ? ---> True becoz it is first inserted into set 'a'

4) Can set have False , 0.0 and 0 ? ---> No becoz they are same

5) a = {0.0 , False , 0}

Which element will be in set 'a' ? ---> 0.0 becoz it is first inserted into set 'a'

```
# set() function demo program
a = set('Rama rAo') # Converts string to set
print(a) #{'R' , 'a' , 'm' , ' ' , 'r' , 'A' , 'o'} in any order
print(len(a)) # 7
print(set([10 , 20 , 15 , 20])) # Converts list to set i.e. {20,10,15} in any order
print(set((25 , 10.8 , 'Hyd' , 10.8))) # Converts tuple to set i.e. {25,10.8,'Hyd'} in any order
print(set(range(10 , 20 , 3))) # Converts range object to set i.e. {10,13,16,19} in any order
#print(set(25)) # Error : 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

```
# Find outputs (Home work)
a = [ ] # Empty list
b = ( ) # Empty tuple
c = { } # Empty dictionary
d = set() # Empty set
print(type(a)) # <class 'list'>
print(type(b)) # <class 'tuple'>
print(type(c)) # <class 'dict'>
print(type(d)) # <class 'set'>
print(a) # []
```

```
print(b) #  ()
print(c) #  { }
print(d) #  set()
```

1) Is {10 , 20} a set ? ---> Yes due to { }

   Is {25} a set ? ---> Yes due to { }

   Is { } a set ? ---> No and it is an empty dictionary

2) How is an empty set denoted ? ---> set() but not { }

```
# Tricky  program
# 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 :  set  already  contains  'Hyd'
a . add(1)  #  Ignored :  set  already  contains   True
print(a) # {25,10,8,'Hyd',True,None} in  any  order
print(len(a)) #   5
a . remove(25) # Removes 25 from  set
print(a)   #   {10,8,'Hyd',True,None}  (Order  is  same as  line  11)
#a . append(100) # Error :   No  append()  method  in  set
#a . add(set())  #   Error : set  can  not  be  inserted  as   it  is  mutable
a . add(()) #  Inserts  ()   any  where   in  set  'a'
#a . add([])  #   Error :  List  can  not  be  inserted  as   it  is  mutable
print(a)  #  {10.8 , 'Hyd' , True , None , ()}  in  any  order
#a . add({})  #  Error :  dict  can  not  be  inserted  as   it  is  mutable
```

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

How to print set in two differnet ways (Home work)

```
a = {25 , True , 'Hyd' , 10.8}
print('set with print function')
print(a) #  {'Hyd' , 25 , 10.8 , True} in any order
print('Iterate elements of set with for loop')
for x in a:
        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 { }

i.e. {k1 : v1 , k2 : v2 , k3 : v3 , ...}

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) How to obtain value of dictionary ? ---> a[key] where 'a' is the dictionary

5) What does dict[invalid-key] do ? ---> Throws error

6) What does dict[value] do ? ---> Throws error

7) In other words, it is possible to obtain value using key but not vice-versa

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

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

10) Can dictionary have duplicate keys ? ---> No and they should be unique

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

11) 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'}

12) Can dictionary be repeated ? --->

No becoz duplicate keys are obtainbed when dictionary is repeated which is not permitted

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

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

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

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

17) What does len(dict) do ? ---> Returns number of key : value pairs in the dictionary

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

19) What does dict[key] = x do ? ---> Modifies value of the key to 'x'

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

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

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

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

Find outputs (Home work)

```
a = {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
print(a) # {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
print(type(a)) # <class 'dict'>
print(a[10]) # Value of key 10 i.e. Ramesh
print(a[20]) # Value of key 20 i.e Kiran
print(a[15]) # Value of key 15 i.e Amar
print(a[18]) # Value of key 18 i.e sita
#print(a[19]) # Error : Key 19 does not exist in dict 'a'
```

```
#print(a[0]) # Error : Key 0 does not exist in dict 'a'
#print(a['Amar']) # Error : Key 'Amar' does not exist in dict 'a'
a[15] = 'Krishna' # Value of key 15 is modified to 'Krishna' in dict 'a'
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 : Dict can not be repeated as duplicate keys are obtained it is repeated

# Find outputs (Home work)
a = {10 : 'Hyd' , 10 : 'Sec'} # Replaces 'Hyd' with 'Sec' as key 10 is repeated
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
```

1) What happens when key is repeated in the dictionary ? ---> Value gets replaced

2) Order of keys remain same and only value gets replace

```
# 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 ? ---> Replaces 'Yes' with 'No' becoz True and 1 are same

2) What happens when 1.0 : 'May be' is encounterd ? ---> Replaces 'No' with 'May be ' becoz True and 1.0 are same

3) Value gets replaced but key remains unchanged

```
# Find outputs

#a = { [ ] : 25} # Error : key can not be a mutable object such as list

b = { ( ) : 25} # Valid : key can be immutable object such as tuple

print(b) # { ( ) : 25}

#c = { { } : 25} # Error : key can not be a mutable object such as dict

d = {'Ramesh' : [9948250500, 9848565090, 9440250404]} # Valid : key can be immutable object such as string

print(d) # {'Ramesh' : [9948250500, 9848565090, 9440250404]}

print(len(d)) # 1

#e = {set() : 10.8} # Error : key can not be a mutable object such as set

# Find outputs

a = {} # Empty dictionary

print(type(a)) # <class 'dict'>

print(len(a)) # 0

print(a) # {}

b = dict() # Functions returns an empty dictionary

print(type(b)) # <class 'dict'>

print(len(b)) # 0

print(b) # {}
```

What are the two ways to denote an empty dictionary ? ---> { } and dict(No-args)

```
# 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) .....

```
# How to print dictionary in different ways
a = {10 : 'Ramesh' , 20 : 'Kiran' , 15 : 'Amar' , 18 : 'Sita'}
print('Dictionary with print function')
print(a) # {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 in 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 for loop valid ? ---> Yes becoz keys() method is executed by default

2) for x in dictionary:

    print(x)

How is the for loop interpreted ? ---> for x in dictionary . keys()
print(x)

3) for x , y in dictionary . keys():

print(x , y)

Is the 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 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 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 . keys():

print(x)

Iteration	x
1	10
2	20
3	15
4	18

8) 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'

9) 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')

10) 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'

11)

```
for x , y in a . items():
        print(x , y , sep = ':')
```

What is the another alternative to the for loop ? --->

```
for x in a . items():
        print(x[0] , x[1])
```

What are x[0] and x[1] ? ---> x[0] is first element and x[1] is 2nd element of each tuple

12)

```
for x , y in a . items():
        print(x , y , sep = ':')
```

What is the alternative to the for loop ? --->

```
for x in a . items():
        print(*x)
```

What is the difference between x and *x ? ---> 'x' is each tuple of the list in dict_items object and *x is elements of each tuple after unpack

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) Which sequence can not hold mutable elements (Total : 4) ? ---> str , range , set and dict

13) Does python support variables ? ---> No and python supports only objects

14) Is python 100% object oriented language (OOL) ? ---> Yes becoz there are only objects but not variables and there are only classes but not datatypes

```
# Identify Error
print('Hyd')
        print('Sec')  # Error  due to space
                print('Cyb')  # Error
```

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

3) In other words, spacebar (or) tab key is not permitted before the statement

4) Finally every suite should be indented

5) Invalid:
```
stmt1
        stmt2
            stmt3
```

6)
```
{
    stmt1
    stmt2
}
```

Can suite be in braces ? ---> No

7) In other words, braces can be used for set and dictionary but not for suite

```
# Find outputs (Home work)
a = {
                    print('Hyd') ,
                    print('Sec') ,
                    print('Cyb')
   } # set of single None
print(type(a)) # <class 'set'>
print(a)  # {None}
print(len(a)) #  1
```

Outputs

```
---------
Hyd
Sec
Cyb
<class 'set'>
{None}
1
```

```
1) {
            print('Hyd'),
            print('Sec'),
            print('Cyb')
   }
```

Is it a suite ? ---> No due to { } and commas

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}

4) a = [

print('Hyd') ,

print('Sec') ,

print('Cyb')]

What is type(a) ? ---> <class 'list'>

What is the result of print(a) ? ---> [None , None , None]

5) a = (

print('Hyd') ,

print('Sec') ,

print('Cyb')

)

What is type(a) ? ---> <class 'tuple'>

What is the result of print(a) ? ---> (None , None , None)

```
# Anonymous object demo program
_ = 25 # 25 is assigned to nameless object
print(_) # 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 nameless objects can be a program ? ---> Just one (or) zero

3) What happens when another nameless object is created ? ---> Existing nameless object gets deleted

4) Can there be multiple nameless objects in a program ? ---> No

5) _ = 10

_ = 20

What happens when _ = 20 is executed ? ---> A new nameless object is created with value 20 and old nameless object with 10 is lost