**Points to Remember in Python:**

**1. Swapping of two numbers using Tuples**:

X , Y = 15 , 20

print(X , Y) # 15 , 20

X , Y = Y , X

print(X , Y) # 20 , 15

**2. Use of Ternary Operator for Conditional Assignment**

**Syntax**: [true\_statement] if [Condition] else [false\_statement]

**Ex**: Program to find smallest of 3 numbers

def small(a,b,c):

return a if a <= b and a <= c else (b if b <= a and b <= c else c)

print (small(10,20,30)) # 10

print (small(25,20,30)) # 20

print (small(35,40,30)) # 30

**3. To know absolute location of modules imported in code**:

import threading

import socket

print (threading) # <module 'threading' from 'C:\\Program Files (x86)\\Python37-32\\lib\\threading.py'>

print (socket) # <module 'socket' from 'C:\\Program Files (x86)[\\Python37-32\\lib\\socket.py](file:///\\Python37-32\\lib\\socket.py)'>

**4. Use of Interactive “\_” Operator**

The **“\_”** (an underscore) references to the output of the last executed expression.

>>> 2 + 1

3

>>> \_

3

>>> print \_

3

**5. Debugging scripts**:

We can set breakpoints in our Python Script using the <pdb> module.

We can specify <pdb.set\_trace()> anywhere in the script and set a breakpoint there.

import pdb

pdb.set\_trace()

**6. Inspect an Object in Python**:

We can inspect an object in Python by calling the dir() method as follows

test = [1,3,5,7]

print(dir(test)) # ['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_',

'\_\_delitem\_\_', '\_\_delslice\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_',

'\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_getslice\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_iadd\_\_',

'\_\_imul\_\_', '\_\_init\_\_', '\_\_iter\_\_', '\_\_le\_\_', '\_\_len\_\_', '\_\_lt\_\_', '\_\_mul\_\_', '\_\_ne\_\_',

'\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_reversed\_\_', '\_\_rmul\_\_',

'\_\_setattr\_\_', '\_\_setitem\_\_', '\_\_setslice\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_',

'append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']

**7. Return Multiple Values from functions**:

**Ex**:

def X():

return 1,2,3,4

a, b, c, d = X()

print(a, b, c, d) # 1, 2, 3, 4

8. Python is Case Sensitive language and identifiers can be of any length in Python.

9. As Python has no concept of Private Variables, leading underscores are used to indicate

variables that must not be accessed from outside the class.

10. In Python, all **Keywords are in lower case** while True, False and None are capitalized.

11. In Python, Power Operator is X\*\*Y (i.e., 2\*\*4 = 16)

12. Order of Precedence in Python: (**PEMDAS**)

(i) Parentheses (iv) Division

(ii) Exponential (v) Addition

(iii) Multiplication (vi) Subtraction

Example:

The output of the Expression 3\*1\*\*4 = 3

Reason: First the expression will solve 1\*\*4 because exponential has higher precedence than multiplication.

So 1\*\*4=1 and 3\*1=3.So Final Answer is 3.

13. Operators with same precedence are Evaluated in Left to Right Manner and Exponential operator follows the Right to Left manner.

**Ex**: >>> 2\*\*3\*\*1

8

14. **Same Precedence** level for:

(i) Addition and Subtraction

(ii) Multiplication and Division

15. When given a function that does not throw any value then Python Shell throws a None Type object back.

16. **id() in Python**:

Python id() is a Built-in function returns the “**identity**” of the object.

The **identity** of the Object is an **integer** and Which is Guaranteed to be **Unique and Constant** for the object during its life-time.

**int** is the return type of the function id.

**Syntax** : id(object)

17. When we execute following Python code snippet:

**Apple = mango** # Output : **NameError** : name ‘mango’ is not defined

18. Python codes have to be **indented** properly.

19. **List** data type can store any values within it.

Ex: L = [ ‘Hello’, 2, 4.6, 2 ]

20. In order to store Values in terms of **Key and Value**, We use **Dictionary** data type.

21. print(‘hello’, ‘how’, ‘are’, ‘you’) # **Output** : hello how are you

print(‘hello’, ‘how’, ‘are’, ‘you’ + ‘-‘ \* 4) # **Output**: hello how are you----

print(‘hello-‘ + ‘how-are-you’) # **Output** : hello-how-are-you

print(‘hello’ + ‘-‘ + ‘how’ + ‘-‘ + ‘are’ + ‘you’) # **Output** : hello-how-areyou

22. **trunc() function**:

**Truncate** function in python behaves as a **ceiling** function for Negative numbers and **floor** function for Positive Numbers.

**Ex:**

# **Python program to show output of floor(), ceil()**

# **truncate() for a positive number.**

import math

print math.**floor**(3.5) # floor # **Output** : 3.0

print math.**trunc**(3.5) # work as floor # **Output** : 3

print math.**ceil**(3.5) # ceil # **Output** : 4.0

# **Python program to show output of floor(), ceil()**

# **truncate() for a negative number.**

import math

print math.**floor**(-3.5) # floor # **Output** : -4.0

print math.**trunc**(-3.5) # work as ceil # **Output** : -4

print math.**ceil**(-3.5) # ceil # **Output** : -3.0

23. ~X Equivalent to –(X+1)

24. Numbers starting with 0 are **Octal Numbers** but 9 isn’t allowed in octal numbers.

25. **cmp(x,y)**

returns **1** if x > y

returns **0** if x == y

returns **-1** if x < y

26. Python **rounds off** numbers away from 0 when the number to be rounded off is exactly halfway through.

**Ex**:

round(0.5) = 1

round(-0.5) = -1

27. Bitwise XOR gives

0 if both bits are 1

1 if either of bits is 1

28. If we want to multiply a given number ‘a’ by 4, we can use the expression : a<<2

29. **Code**:

**if** (9 < 0) **and** (0 < -9):

**print**("hello")

**elif** (9 > 0) **or** False:

**print**("good")

**else**:

**print**("bad")

**Output** : good

30. Python Language can be used for Web development, Mobile App development, AI, Machine Learning, Big data and Internet of Things applications.

31. **Assert** is used for **Debugging Purposes** i.e., to check the correctness of code. If a statement evaluated to true, nothing happens, but when it is false, **AssertionError** is raised. One can also **print a message with the Error, separated by a comma**.

**Syntax**: assert condition, error\_message(optional)

**Ex**: # Python 3 code to demonstrate

# working of assert

# initializing number

a = 4

b = 0

# using assert to check for 0

print ("The value of a / b is : ")

**assert b != 0, "Divide by 0 error"**

print (a / b)

**Output**:

The Value of a / b is :

**Runtime Error**:

Assert b != 0, “Divide by 0 error”

AssertionError : Divide by 0 error

32. **Comments** are helpful to understand the program.

# ---- Single Line Comment Symbol

' ' ' Multiple lines of data in comment ' ' ' ---- Multiline Comment Symbol (We can use double quotes in place of single quotes)

33. Tuples are Immutable. So we can not add or delete elements. There are Operations in Tuple.

34. Strings, List and Tuple are called Sequences in Python.

35. range(1,11) == [1,2,3,4,5,6,7,8,9,10]

36. **Absolute Value** is nothing but the distance of a value from Zero(0). We can find absolute value using **abs()** function.

**Ex**: >>> abs(-6)

6

>>> abs(7)

7

37. **<stringName>.capitalize()** function will capitalize only first letter of String .

**<stringName>.title()** function will capitalize first letter of every word in the String.

38. **swapcase()** function will swap the lower case letters to upper case and upper case letters to lower case in a String.

Ex:

>>> str1 = "Hello Python SWAP"

>>> str1.swapcase()

'hELLO pYTHON swap'

39. **Strip()** function is used to remove left and right side characters of String.

Ex:

>>> str1 = 'aaaaaaahelloaaaaahelloaaaa'

>>> str1.strip('a')

'helloaaaaahello'

**lstrip()** function is used to remove only **left side** characters of string.

Ex:

>>> str1 = 'aaaaaaahelloaaaaahelloaaaa'

>>> str1.lstrip('a')

'helloaaaaahelloaaaa'

**rstrip()** function is used to remove only **right side** characters of string.

Ex:

>>> str1 = 'aaaaaaahelloaaaaahelloaaaa'

>>> str1.rstrip('a')

'aaaaaaahelloaaaaahello'

40. **Concatenate 2 dictionaries**:

We can not concatenate 2 dictionaries with **‘+’** Operator.

We have to **update()** Method to concatenate 2 dictionaries.

**Syntax of update() Method**:

**Dictionary1.update(dictionary2)**

Ex:

>>> dict1 = {1:"apple",2:"mango",3:"Orange"}

>>> dict2 = {8723:"Carrot"}

>>> dict1.update(dict2) # Updating dict1 by adding dict2 to dict1

>>> dict1

{1: 'apple', 2: 'mango', 3: 'Orange', 8723: 'Carrot'}

>>> dict2

{8723: 'Carrot'}

41. In Python, below quick operands can be used for different operations in sets:

**|** for Union Operation

**&** for Intersection Operation

* for Difference operation

**^** for Symmetric Difference Operation

**Ex:**

# sets are define

A = {0, 2, 4, 6, 8};

B = {1, 2, 3, 4, 5};

# union – combines of elements in A and B

print("Union :", A | B)

# intersection – return **common** elements in A and B

print("Intersection :", A & B)

# difference – return elements belongs to A but not B for A – B

# difference – return elements belongs to B but not A for B - A

print("Difference :", A - B)

# symmetric difference – return the values which are **not** **common** in A and B

print("Symmetric difference :", A ^ B)

**Output:**

Union : {0, 1, 2, 3, 4, 5, 6, 8}

Intersection : {2, 4}

Difference : {0, 8, 6}

Symmetric difference : {0, 1, 3, 5, 6, 8}

42. **clear()** method is used to delete all elements in a set. No Arguments can be provided in this method.

**Ex**:

>>> A = {1,2,3,4,5,6}

>>> A.**clear**(5)

Traceback (most recent call last):

TypeError: clear() takes no arguments (1 given)

>>> A.clear()

>>> A

set()

43. To **rename** or **remove** a file using Python we have to **import** **os** Module.

**Syntax**:

**import** **os**

**os.rename**(oldname,newname)

**os.remove**(filename)

44. We can list the sub-directories present in a particular directory using **os.listdir**(“C:\\”).

45. We can change the current working directory using the **os.chdir()** method.

46. We can remove **empty** directory using **os.rmdir()** method where as we can remove **non-empty** directory **shutil.rmtree()** method (**rmtree()** method is inside **shutil** module).

47. Using **raise** statement, programmer can force the Exception to occur.

**Ex 1**:

>>> raise ValueError(4)

Traceback (most recent call last):

raise ValueError(4)

ValueError: 4

**Ex 2**:

>>> raise NameError

Traceback (most recent call last):

raise NameError

NameError

48. **Class Variable**: The variable which is belong to all objects in the Class is called class variable.

class Student:

clg = 'svce' # Class Variable

49. By using **@Staticmethod** we can avoid **self** parameter in methods in class

**Ex**:

**class** person:

**@Staticmethod**

**def** display(): # creating display method without self parameter

print("Hello!!!!")

p=person()

p.display()

**Output**:

Hello!!!!

50. In **Multi-level Inheritance**, there will be one Base Class and more than one Derived Classes.

**Syntax**:

Baseclass 🡪 Derivedclass1 🡪 Derivedclass2

51. In **Multiple Inheritance**, there will be more than one Base class and one Derived Class.

**Syntax**:

Baseclass1 Baseclass2

\ /

Derivedclass

52. Assigning a Value to a Variable is called as **Binding**.

53. **Formatted Strings**:

name = 'Jhonny'

age = 56

**Syntax 1 ( Recommended)**: print(f'Hello {name}. You are {age} years old') # Using **f** before String

**Syntax 2**: print('Hello {}. You are {} years old'**.format**('Jhonny','56'))

**Syntax 3**: print('Hello {0}. You are {1} years old'.format(name,age)) # index of name = 0, age = 1

**Syntax 4**: print('Hello {new\_name}. You are {new\_age} years old'.format(new\_name='Maxwell',new\_age= 99))

54. String = 'I am PYTHON'

print(String[1:100]) # OUTPUT : am PYTHON

print(String[-3:]) # OUTPUT : HON

print(String[::-1]) # OUTPUT : NOHTYP ma I

55. Strings in Python are IMMUTABLE i.e., Strings cannot be modified. The only way to change value of a string is to assign entirely a new value to the existing string

Example:

string = '01234567'

string[1]=8 # OUTPUT : **TypeError**: **'str' object does not support item assignment**

string = 'ABCDEF'

print(string) # OUTPUT : ABCDEF

56. A Key has to be **UNIQUE** in Dictionaries.

57. **Short Circuiting in Python**:

Short Circuiting means Stoppage of execution of Boolean expression if the truth value of expression has been determined already. The evaluation of expression takes from left to right.

The Boolean Operators that support short circuiting in Python are **and**, **or** and **not**.

In-Built functions **all()** and **any()** in python also support short circuiting.

1. Using **all()**, **stops execution when false occurs**, tells the compiler that if even one is false, all cannot be true, hence stop execution further.
2. Using **any()**, **stops execution when true occurs,** tells the compiler that if one is true, expression is true, hence stop execution further.

58. Iterables can be iterated one by one by checking each item in the collection.

Iterables can be a string, list, dictionary, tuple, set

‘int’ object is not iterable.

59. **Function Arguments**:

Using **\*args** as a function parameter enables you to pass an arbitrary number of arguments to that function. The arguments are then accessible as the **tuple args** in the body of the function.

**\*\*kwargs** (keyword arguments) allows you to handle named arguments that you have not defined in advance and **return a dictionary** in which the keys are the argument names, and the values are the argument values.

Example:

**def** function(named\_arg,\*args,\*\*kwargs):  
 print(named\_arg)  
 print(args)  
 print(kwargs)

function(1,2,3,4,5,a=6,b=7,c=8)

OUTPUT:

1

(2, 3, 4, 5)

{'a': 6, 'b': 7, 'c': 8}

60. **Major 3rd-Party Libraries in Python**:

1. **Django**: The most frequently used web framework written in Python.

2. **CherryPy** and **Flask**: Popular web frameworks

3. **Beautifulsoup**: For scraping data from websites, this library is very useful, and leads to better results than building our own scraper with regular expressions.

4. **matplotlib**: This module allows us to create graphs based on data in Python.

5. **NumPy**: This module allows for the use of multi-dimensional arrays and contains functions to perform mathematical operations such as matrix transformation on the arrays.

6. **SciPy**: This library numerous extensions to the functionality of NumPy.

7. Python can also be used for **game development**.

* For 3D games, the library **Panda3D** can be used. For 2D games, we can use **pygame**.