**PYTHON:**

--------------------------00**-Python Object and Data Structure Basics**------------------------------

1. Basic Arithmetic- python store orignal division: 4/3 = 1.333333333

To get integer use 4//3 =1 (FLOOR DIVISION)

2. POWERS 2\*\*3 square--> 4\*\*2

squareroot--> 16\*\*0.5

3. #variable--> use DYNAMIC TYPING i.e. can be reassigned to a different datatype.

4. REVERSE a String--> s[ : :-1]

5. l1 = x.split() --> will split words in string x(based on whitespace)

x.split(e) --> will split words in string x(based on every e encountered in x)

5.2. #now to get back x, use join, to join the list elements as a string

" ".join(l1) -->joint with spaces b/w them

6. #string- IMMUTABLE

7. string INTERPOLATION: x='sam'

print(f'hello {x} ') --> f-string

print("hello {}". format(x))--> .format()

print('hello' + x = 'there') --> old style way

8. LISTS: merge two lists- concatinate 2 list --> l1=['hi'] l2=['sam']

li1+l2 --> ['hi' , 'sam']

9. l1.append() l1.pop() sorted(l1) l1.reverse() len(l1) clear(l1) remove() extend() copy() count()

10. DICTIONARIES: d1.keys() d1.values() d1.items()

11. TUPLE- immutable ()

SETS- unique {}

12. return the index--> t1.index('one')

return the no of times a value appears--> t1.count('one')

13. #since immutability, tuples can't grow. Once a tuple is made we can not add to it.

hence append() will not work

14. list1 = [1,1,2,2,3,1]--> Cast as set to get unique values

set(list1)-->{1, 2, 3}

15. #We can use None as a placeholder for an object that we don't want to reassign yet:

b = None

16. NESTED DICTIONARY --> d = {'k1':[1,2,{'k2':['this is tricky',{'tough':[1,2,['hello']]}]}]}

d['k1'][2]['k2'][1]['tough'][2][0]

17. sorting-NO --> dictionaries are mappings not a sequence.

**-------------------------------02-PYTHON CONTROL FLOW STATEMENTS---------------------------**

18. type(input) --> always string.

19. str even/odd --> len(str)%2==0

20. def check\_even(num):

return num%2 == 0

#return true if condition satisfy else false(no need for if else)

21. any() is a core library function in Python.

it returns true if at least one value in the sequence is true.

It returns as soon as it finds the first true value.

22. from random import shuffle/randint(generators)

23. shuffle does not return anything,shuffles existing list, call list to see result.

#same with all other generators, like range

24. enumerate --> it returns INDEX with respective LETTER in string as TUPLE

for a,b in enumerate(word):

print(b)

#tuple unpacking + enumeratation

25. print(\*range(1,5),sep="") --> 1234

26. zip(l1,l2) --> zip two lis together, --> [(1,'a'),(2,'b')]

27. max() min()

28. TUPLE UNPACKING--> for a,b in t1: print(a)

29. if-elif-else

30. while is followed with else.

31. PASS keyword. does not do anything. let the fn complie w/o any code-body

32. [0]\*3 same as [0,0,0]

33. LIST COMPREHENSION: l1=[i\*\*2 for i in range(0,11) if i%3==0] --> [0, 9, 36, 81]

34. even num --> for i in range(0,10,2):

print(i)

#instead of if i%2==0 use skip size as 2

33. SETS:

33.1 set union() --> a.union(b)

A.union(\*other\_sets) OR a|b

Note: \* is not part of the syntax. It is used to indicate that the method can take 0 or more arguments.

33.2 for intersection

33.3 – for difference

33.4 ^ for symmetric difference

34. raw\_input and input --> raw\_input always accepts strings(python old version)

35. n=int(input())

l1=[]

for i in range(n):

ele=int(input())

l1.append(ele)

#method 2 : l1 = input().split(" ") --> takes input as a string elements(or num) stored as words

l1=[int(i) for i in l1] --> converting elements to integer values

36. abs(n1,n2) return subtraction of higher num - lower

37. The eval() evaluats an expression. The expression can be a Python statement, or a code object.

38. Multiple variable assignment -->

x = y = z = 0

39. python does not support increment/decrement operators

40. count occourence of substr in string:

s='hxxxxh' #without overlapping

s.count('xx')

s='hxxxxh' #with overlapping

count=0

for i in range(len(s)):

if 'xx'==s[i:i+2]:

count=count+1

print(count)

41. if no\_of\_cats == no\_of\_dogs:

return True

else:

return False

can be written as --> return no\_of\_cats == no\_of\_dogs

#it will either evaluate to true or false. so instead of if else, directly write condition with return statement

42. if flag==True: ---> same as ---> f flag:

#in this case, if flag is true, condition will be true. i.e. so directly write if flag as the condition.

43. in for loop for i in iterable--> you can access i+1 i-2 elements

44. str='catdogcat'

for i in range(len(str)-3):

#range(len(str)-3) because you want to check for occourence of cat. use str[i:i+3] so last two elements will not have i+1 and i+2

#it will throw error

45. a,b=2,3 --> will assign a=2 and b=3

**------------------------------------METHODS AND FUNCTION--------------------------------------------**

46. LEGB Rule:Local -->Enclosing function locals -->Global (module)-->Built-in (Python)

47. def myfunc(\*args): ---> will accept any no oof arguments

return sum(args)\*.05

myfunc(40,60,20)

48. def makes\_twenty(n1,n2):

return (n1+n2)==20 or n1==20 or n2==20

#jab sirf true ya false(bool value) return krna ho --> return with if condition

49. l1=['hello','world']--> t1[0][0] =h

50. (a==11 or b==11 or c==11) --> 11 in [a,b,c]

51. str.replace() --> USE CASE-can remove space in str

52. to calculate sum of list, initialize sum=0

but to calculate product of list, initialize mul=1

53. fn name--> name\_of\_function (SNAKE CASING)

class name--> NameOfClass (CAMEL CASING)

**--------------------------Object Oriented Programming------------------------**

54. flow: class keyword--class level attribute--init method for user defined attibutes

55. #constructor body automatically executes when object is created. so if any print st is written inside will execute too.

56. to delete the object use --> del objectname

57. PyPi- repository

Use pip install to import module from library

Python has library for all use case s

58.



59. \_\_str\_\_ when you print the object --> print(objname); this fn will execute

\_\_len\_\_ when you print the length of object --> len(objname); this fn will execute

\_\_del\_\_ when you delete obj --> del objname; this fn will execute

60. MODULES AND PACKAGES:

Create a program, use use its fn into another program🡪

From progname import fn-name

For package:

Create \_\_init\_\_.py empty file in the folder

And its ready to be used as package.

From folder name import file name

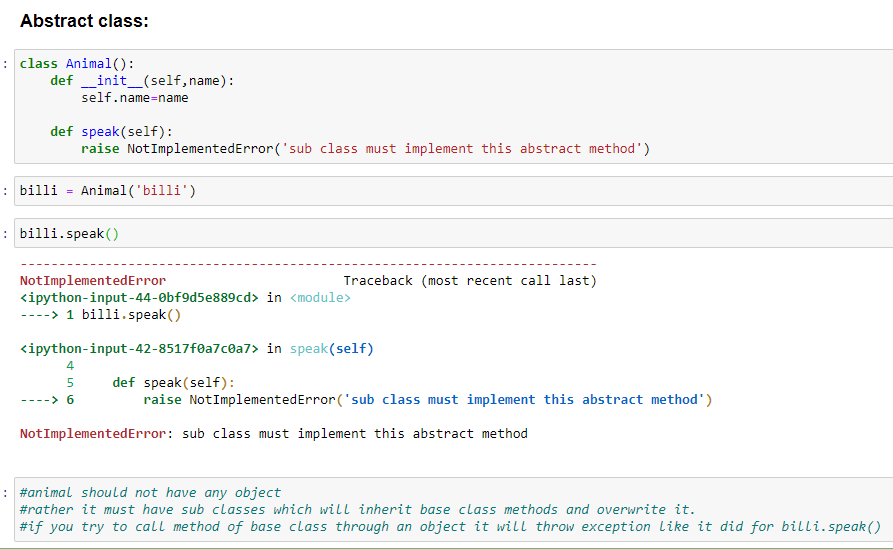
Filename.fnname()

61.

62. POLYMORPHISM



****



**1. Write a C program to find length of a string.**

**2. Write a C program to copy one string to another string.**

**3. Write a C program to concatenate two strings.**

**4. Write a C program to compare two strings.**

**5. Write a C program to convert lowercase string to uppercase.**

**6. Write a C program to convert uppercase string to lowercase.**

**7. Write a C program to toggle case of each character of a string.**

**8. Write a C program to find total number of alphabets, digits or special character in a string.**

**9. Write a C program to count total number of vowels and consonants in a string.**

**10. Write a C program to count total number of words in a string.**

**11. Write a C program to find reverse of a string.**

**12. Write a C program to check whether a string is palindrome or not.**

**13. Write a C program to reverse order of words in a given string.**

**14. Write a C program to find first occurrence of a character in a given string.**

**15. Write a C program to find last occurrence of a character in a given string.**

**16. Write a C program to search all occurrences of a character in given string.**

**17. Write a C program to count occurrences of a character in given string.**

**18. Write a C program to find highest frequency character in a string.**

**19. Write a C program to find lowest frequency character in a string.**

**20. Write a C program to count frequency of each character in a string.**

**21. Write a C program to remove first occurrence of a character from string.**

**22. Write a C program to remove last occurrence of a character from string.**

**23. Write a C program to remove all occurrences of a character from string.**

**24. Write a C program to remove all repeated characters from a given string.**

**25. Write a C program to replace first occurrence of a character with another in a string.**

**26. Write a C program to replace last occurrence of a character with another in a string.**

**27. Write a C program to replace all occurrences of a character with another in a string.**

**28. Write a C program to find first occurrence of a word in a given string.**

**29. Write a C program to find last occurrence of a word in a given string.**

**30. Write a C program to search all occurrences of a word in given string.**

**31. Write a C program to count occurrences of a word in a given string.**

**32. Write a C program to remove first occurrence of a word from string.**

**33. Write a C program to remove last occurrence of a word in given string.**

**34. Write a C program to remove all occurrence of a word in given string.**

**35. Write a C program to trim leading white space characters from given string.**

**36. Write a C program to trim trailing white space characters from given string.**

**37. Write a C program to trim both leading and trailing white space characters from given string.**

**38. Write a C program to remove all extra blank spaces from given string.**

