Name :- Rugvedi Jamgaonkar
Class :- M.Sc. Computer science I

Question Bank

Programming Language Python

Q.1 Name the four types of scalar objects Python has.

→ int - signed, unlimited precision integers.
float - IEEE 754 floating-point numbers.
None - a special, singular null value
bool - true / false boolean value.

Q.2 What is tuple ? How literals of type tuple are written? Five example.

→ A tuple is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. Tuples cannot be changed and use paramtheses.
A tuple is created by surrounding objects with ()'s and seperating the items with commas. An empty tuple is empty ()'s. The elements of a tuple do not have to be the same type. A tuple can be a mixture of any Python data types, including other tuples.
Example :-

```
public (double distance, Store store) GetNearestStore (double lat,
double lon)
{
var store = _storeLocator.FindNearest (lat, lon);
var distance = _storeLocator.DistanceFrom (store, lat, lon);
return (distance);
}
```

3. What is a list? How lists different from tuples?

→ Lists are used to store multiple items in a single variable. Lists are one of 4 built-in data types in python used to store collections of data, the other 3 are Tuple, set, and Dictionary, all with different qualities and usage.

Both lists and tuples are data structures in python, ~~these are~~ Lists are mutable while tuples are immutable. A list has a variable size while a tuple has a fixed size. Operations on tuples can be executed faster compared to operations on lists. The list is dynamic, whereas tuple has static characteristics.

List	Tuple
① Lists are mutable	① Tuples are immutable
② Implication of iteration is Time consuming	② Implication of iterations is comparatively faster
③ List consume more memory	③ Tuple consume less memory
④ Lists have several built-in methods.	④ Tuple does not have must builtin methods.

4. How to slice a list in python?

→ The slice() function returns a slice object. A slice object is used to specify how to slice a sequence.

slice (start, end, step)

5. Write a script to display the current date and time.

→
```
import datetime
now = datetime.datetime.now()
print("Current date and time :")
print(now.strftime("%d-%m-%Y %H:%M:%S"))
```

6. Explain with example while loop, break statements and continue statement in python.

→ While loop :-

With the while loop we can execute a set of statements as long as a condition is true.

The while loop requires relevant variables to be ready, in this example we need to define an indexing variable i, which we set to 1.

Example - print i as long as i is less than 6.

```
i = 1
while i < 6 :
    print(i)
    i += 1
```

Break Statement :-

With the break statement we can stop the loop even if the while condition is true

Example - Exit the loop when i is 3

```
i = 1
while i < 6 :
    print(i)
    if i == 3 :
        break
    i += 1
```

Continue Statement :-

With the continue statement we can stop the current iteration, and continue with the next.

Example - Continue to the next iteration if i is 3

```
i = 0
while i < 6 :
    i += 1
    if i == 3 :
        continue
    print(i)
```

7. What is a dictionary in python? Give example.

→ Dictionaries are used to store data values in key : value pairs. A dictionary is a collection which is unordered, changeable and does not allow duplicates. Dictionaries are written with curly brackets, and have keys and values.

Example - Create and print a dictionary.

```
thisdict = {
  "brand" : "Ford",
  "model" : "Mustang",
  "year" : 1964
}
print (thisdict)
```

8. Appraise with an example nested if and elif header in python.

→ Decision making statements in programming languages decides the direction of flow of program execution.

Nested if :-

A nested if is an if statement that is the target of another if statement. Nested if statements means an if statement inside another if statement.

Syntax :

```
if (condition 1) :
    # Executes when condition 1 is true
    if (condition 2) :
        # Executes when condition 2 is true
    # if Block is end here
# if Block is end here
```

if - elif - else ladder :-

A user can decide among multiple options. The if statements are executed from the top down. As soon as one of the condition

controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final else statement will be executed.

Syntax :
```
if (condition):
    statement
elif (condition):
    statement
    :
else :
    statement
```

9. Write a python program to perform linear search on a list.

→
```
def linearsearch (arr, x):
    for i in range (len(arr)) :
        if arr[i] == x :
            return i
    return -1
arr = ['H', 'E', 'L', 'L', 'O', 'W', 'O', 'R', 'L', 'D']
x = 'W'
print (" Element found at" + str (linearsearch (arr,x)))
```

Output:
Element found at 5

10. Write a program to store 'n' numbers in a list and sort the list using selection sort.

→

```
A = ['t','u','t','o','r','i','a','l']
for i in range (len(A)):
    min = i
    for j in range (i+1, len(A)):
        if A[min] > A[j]:
            min = j
    #swap
A[i], A[min] = A[min], A[i]
#main
for i in range (len(A)):
    print (A[i])
```

Output: a i l o r t t u

11. Tabulate the different modes for opening a file and explain the same.

→ File opening modes in C

A file has to be opened before the beginning of reading and writing operations. Opening a file creates a link between the operating system and the file function.

Syntax for opening a file:

```
FILE *fp;
fp = fopen ("filename with extension", "mode");
```

Opening of file in detail:

FILE: structure defined in stdio.h header file. FILE structure provides us the necessary information about a FILE.

Python has several functions for creating, reading, updating and deleting files.

There are four different methods (modes) for opening a file.

"r" - Read - Default value. Opens a file for reading, error if the file does not exist.
"a" - Append - Opens a file for appending, creates the file if it does not exist
"w" - Write - Opens a file for writing, creates the file if it does not exist.
"x" - Create - creates the specified file, returns an error if the file exists
In addition you can specify if the file should be handled as binary or text mode
"t" - Text - Default value. Text mode
"b" - Binary - Binary mode (e.g. images)
Syntax
To open a file for reading it is enough to specify the name of the file:

```
f = open("demofile.txt")
```

The code above is same as:

```
f = open("demofile.txt", "rt")
```

Because "r" for read, and "t" for text are the default values, you do not need to specify them.

12. Appraise the use of try block and except block in python.

→ Python Try Except

try() is used in Error and Exception Handling. There are two kinds of errors:

- Syntax error: Also known as Parsing errors. Arise when the python parser is unable to understand a line of code.
- Exception: Errors which are detected during execution.

eg. - ZeroDivisionError

List of Exception Errors:

IOError : if file can't be opened

KeyboardInterrupt : When an unrequired key is pressed by the user.

ValueError : When built-in function receives a wrong argument

EOFError : if End-of-file is hit without reading any data.

ImportError : if it is unable to find the module.

Basic syntax:

```
try:
    //code
except:
    // code
```

13. Explain with an example exception with arguments in python.

→ An exception can have an argument, which is a value that gives additional information about the problem. The contents of the argument vary from exception to exception.

Example:

```
try:
b = float (56+78/0)
except Exception, Argument :
print 'This is the Argument \n', Argument
```

Output:

```
This is the Argument
integer division or module by zero
```

14. State the reason to divide programs into functions.

→ Python allows us to divide a large program into the basic building blocks known as a function. The function contains the set of programming statements enclosed by {}. A function can be called multiple times to provide reusability and modularity to the python program.

15. Relate strings and lists.

→ Strings and lists share many similarities. ~~as we have seen throughout this lesson~~. However there are some differences between them.

Strings can only consist of characters, while lists can contain any data type. We cannot easily make a list into a string, but we can make a string into a list of characters, simply by using the list() function.

Strings are immutable, meaning that we cannot update them while lists are mutable, that can be modified.

16. Write a function that can take a value and return the first key mapping to that value in a dictionary.

→ Using next() + iter() : This task can be performed using these functions. In this, we take the first next key using next() and iter function is used to get the iterable conversion of dictionary items.

Function :

```
test_dict = {'Gfg' : 1, 'is' : 2, 'best' : 3}
print ("The original dictionary is : " + str(test_dict))
res = next (iter(test_dict))
print ("The first key of dictionary is : " + str (res))
```

17. What is a module? Give Example.

→ A module allows you to logically organize your python code. Grouping related code into a module makes the code easier to understand and use. A module is a python object with arbitrarily named attributes that you can bind and reference.

A module is a file consisting of Python code. A module can define functions, classes and variables. A module can also

include runnable code.

Example :- The python code for a module named aname normally resides in a file named aname.py. Here's an example of a simple module, support.py

```
def print_func(par):
    print "Hello :", par
    return
```

```
import support
support.print_func("Zara")
```

Output :- Hello : Zara

18. Python strings are immutable, Justify with an example

→ In Python, strings are made immutable so that programmers cannot alter the contents of the object (even by mistake). This avoids unnecessary bugs. Some other immutable objects are integer, float, tuple, and bool. This means a string value cannot be updated.

19. Write a python code to perform binary search. Trace it with an example of your choice.

⇒

```
def binary_search(item_list, item):
    first = 0
    last = len(item_list) - 1
    found = False
    While (first <= last and not found):
        mid = (first + last) // 2
        if item_list[mid] == item:
            found = True
        else:
            if item < item_list[mid]:
                last = mid - 1
            else:
                first = mid + 1
    return found
```

```
print (binary_search ([1,2,3,5,8], 6))
print (binary_search ([1,2,3,5,8], 5))
```

Output:

False

True

20. Discuss the different options to traverse a list.

→ List is equivalent to arrays in other languages, with the extra benefit of being dynamic in size. It is used to store multiple data at the same time. Lists in Python are ordered and have a definite count. Different options to traverse a list :- ①Using for loop ,, ②For loop and range () :- To use the traditional for loop which iterates from number x to number y, ③using while loop. ④Using list comprehension ⑤Using enumerate():- to convert the list into an iterable list of tuples, you can use enumerate () function, ⑥ using numpy :- for every large n-dimensional lists it is sometimes better to use an external library such as numpy.

21. Demonstrate the working of +, * and slice operator in python.

→ Arithmetic operators : are used to perform mathematical operations like addition, subtraction, multiplication and division.

Operator	Description	Syntax
+	Addition: adds two operands	x+y
*	Multiplication: multiplies two operands	x*y

Example :-

```
a = 9
b = 4
add = a+b
mul = a*b
print (add)
print (mul)
```

Slice () function returns a slice object Syntax (start, end, step) It is used to specify how to slice a sequence.

Example :-

```
a = ("a", "b", "c", "d")
x = slice (2)
print (a[x])
```

Output :- ('a', 'b')

22. Compare and constrast tuples and lists in python.

→ List and Tuple in Python are the class of data structure. The list is dynamic, whereas tuple has static characteristics. Lists are just like the arrays, declared in other languages. Lists need not be homogeneous always which makes it a most powerful tool in python.

Syntax:

```
list_data = ['an', 'example', 'of', 'a', 'list']
```

Tuple is also a sequence data type that can contain elements of different data types, but these are immutable in nature. A tuple is a collection of Python objects seperated by commas.

Syntax:

```
tuple_data = ('this', 'is', 'an', 'example', 'of', 'tuple')
```

List	Tuple
1. Lists are mutable	1. Tuples are immutable.
2. Implication of iterations is Time-consuming.	2. Implication of iterations is comparatively Faster
3. The list is better for performing operations, such as insertion and deletion.	3. Tuple data type is appropriate for accessing the elements.
4. Lists consume more memory	4. Tuple consume less memory as compared to list.
5. Lists have several built-in methods.	5. Tuple does not have must built-in methods.
6. The unexpected changes and errors are more likely to occur.	6. In tuple, it is hard to take place.

23. Write a script in Python to sort n numbers using selection sort.

→ (repeated question 10)

24. Explain the commands used to read and write into a file with eg.

→ Reading Files in Python :-

To read a file in Python, we must open the file in reading "r" mode. We can use the read (size) method to read in the size number of data. If the size parameter is not specified, it reads and returns up to the end of file.

```
f = open("test.txt", 'r', encoding = 'utf-8')
f.read(4)
'This'
f.read(4)        f.read()
'is'             'my first file\nThis newfile\n'
for line in f:
    print(line, end = '')
This is my first line.
This new file
```

25. Describe how exceptions are handled in python with necessary examples.

→ In Python, exceptions can be handled using a try statement. The critical operation which can raise an exception is placed inside the try clause. The code that handles the exceptions is written in the except clause. We can thus choose what operations to perform once we have caught the exception. Exceptions are raised when the program is syntactically correct but the code resulted in an error. This error does not stop the execution of the program, however, it changes the normal flow of the program.

26. Comment with an example on the use of local and global variable with the same identifier name.

→ A global variable is a variable that is accessible globally. A local variable is one that is only accessible to the current scope, such as temporary variables used in a single function defn. It is also possible to use a global and local variable with the same name simultaneously. Built-in function globals() returns a dictionary object of all global variables and their respective values. Using the name of the variable as key, its value can be accessed and modified.

27. How to create a list in python? Illustrate the use of negative indexing of list with example.

→ Lists in Python can be created by just placing the sequence inside the square brackets []. List doesn't need a built-in function for creation of list.

List may contain mutable elements.

Negative Indexing :- Use negative indexes to start the slice from the end of the string

Example : Get the characters from position 5 to position 1, starting the count from the end of the string :

```
b = "Hello, World!"
print(b[-5:-2])
```

Output :- orl

28. Mention the different types of iterative structure allowed in python. Explain the use of continue and break statements with example.

→ There are two main iterative structures in python: While loops and for loops.

A for loop is used for iterating over a sequence. The for loop does not require an indexing variable to set beforehand. The while loop requires relevant variables to be ready, With the while loop we can execute a set of statements as long as a condition is true.

The continue statement :- it returns the control to the beginning of the while loop. The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop. The continue statement can be used in both while and for loops.

The break statement :- it terminates the current loop and resumes execution at the next statement. The most common use for break is when some external condition is triggered requiring a hasty exit from a loop. The break statement can be used in both while and for loops.

29. Mention the list of keywords available in Python. Compare it with variable name.

→ Python has a set of keywords that are reserved words that cannot be used as variable names, function names, or any other identifiers:

and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, lambda, not, or, return, True, try, while

30. Analyse string slicing. Illustrate how it is done in python with example

→ Python slicing is about obtaining a substring from the given string by slicing it respectively from start to end. Python slicing can be done in two ways.

- slice() constructor
- Extending Indexing

Slice() constructor

The slice() constructor creates a slice object representing the set of indices specified by range (start, stop, step).

Syntax :

slice (stop)

slice (start, stop, step)

Parameters :

Start : Starting index where the slicing of object starts.

stop : Ending index where the slicing of object stops.

step : It is an optional argument that determines the increment between each index for slicing.

Return Type : Returns a sliced object containing elements in the given range only.

Example :

```
String = 'ASTRING'
s1 = slice(3)
s2 = slice(1,5,2)
s3 = slice(-1,-12,-2)
print("String slicing")
print(String[s1])
print(String[s2])
print(String[s3])
```

Output :

```
String slicing
AST
SR
GITA
```

31. Write a python code to search a string in the given list.

→ Example :-

```
test_string = "There are 2 apples for 4 persons"
test_list = ['apples', 'oranges']
print("The original string :" + test_string)
print("The original list :" + str(test_list))
res = [ele for ele in test_list if (ele in test_string)]
print("Does string contain any list element :" + str(bool(res)))
```

Output :-

The original string : There are 2 apples for 4 persons
The original list : ['apples', 'oranges']
Does string contain any list element : True.

32. Demonstrate with code the various operations that can be performed on tuples.

→ Operations that can be performed on tuple in Python :-

① Changing the elements of a tuple.

```
my_data = (1, [9,8,7], "World")
print(my_data)
my_data[1][2] = 99
print(my_data)
```

Output :-

(1, [9,8,7], 'World')
(1, [9,8,99], 'World')

② Delete operation on tuple

```
my_data = (1,2,3,4,5,6)
print(my_data)
del my_data
```

Output :-

(1, 2, 3, 4, 5, 6)

③ Slicing operation in tuples

```
my_data = (11,22,33,44,55,66,77,88,99)
print (my_data)
print (my_data[2:5])
```

Output:

(11, 22, 33, 44, 55, 66, 77, 88, 99)
(33, 44, 55)

33. What is user defined function? How can we pass parameters in user defined function?

→ A function is a set of statements that take inputs, do some specific computation and produce output. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can call the function.

Syntax:

```
def function_name():
    statements
    .
    .
```

Example:

```
def fun():
    print ("Inside function")
fun()
```

Output:

Inside function.

34. Illustrate the use of range() in python along with an eg.

→ The range() function is used to generate a sequence of numbers over time. At its simplest, it accepts an integer and returns a range object.

Syntax:

range(start, stop, step)

Example:

```
x = range(3,6)
for n in x:
    print(n)
```

Output:

3
4
5

35. Write a short note on class in object-oriented programming.

→ In object oriented programming, a class is an extensible program-code-template for creating objects, providing initial values for state and implementations of behaviour. A class that create classes is called a metaclass. In many languages, the class name is used as the name for the class, the name for the default constructor of the class and as the type of objects generated by instantiating the class; these distinct concepts are easily conflated.

36. Discuss anonymous function.

→ In Python, an anonymous function is a function that is defined without a name. While normal functions are defined using the def keyword in Python, anonymous functions are defined using the lambda keyword. Hence, anonymous functions are also called lambda functions.

37. What is List? Explain with example.

→ (repeated)

38. Explain map() function with example.

→ map() function returns a map object of the results after applying the given function to each item of a given iterable.

Syntax: map(fun, iter)

Parameters:

fun - It is a function to which map passes each element of given iterable.

iter - It is a iterable which is to be mapped.

Example:

```
def addition (n):
    return n+n

numbers = (1, 2, 3, 4)
result = map(addition, numbers)
print(list(result))
```

Output:

[2, 4, 6, 8]

39. How to create Dictionary in Python?

→ Dictionaries are used to store data values in key:value pairs. A dictionary is a collection which is unordered changeable and does not allow duplicates. Dictionaries are written with curly brackets, and have keys and values.

Example:

```
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
}
print (thisdict)
```

Output:

{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}

40. Explain random module and its function.

→ The python random value module functions depend on a pseudo-random number generator function random(), which generates the float number between 0.0 and 1.0. There are different types of functions used in a random module which is given below:

random.random()

This function generates a random float number bet^n 0.0 and 1.0

random.randint()

This function returns a random integer between the specified integers.

42. What is the role of indentation in Python? Provide examples to support your answer.

→ Indentation refers to the spaces at the beginning of a code line. where In order programming languages the indentation in code is for ~~red~~ readability only. the indentation in Python is very important. Python uses indentation to indicate a block of code.

Example:

① if 5 > 2:
    print ("five is greater than two!")

② if 5 > 2:
    print ("Five is greater than two!")
    print (" Five is greater than two!")

Q.43 What is the use of del statement?

→ The del keyword is used to delete objects. In Python everything is an object, so the del keyword can also be used to delete variables, lists or part of a list etc.

Example

```
x = "hello"
del x
print(x)
```

Q.44 Write two methods of dictionary datatype.

→ Dictionaries are an essential data structure in Python and any other modern programming language such as javascript and swift.

i) clear() - removes all the elements from the dictionary

e.g. car = { "brand": "Ford",
"model": "Mustang",
"Year": 1964
}

car.clear() print (car)

ii) copy() - returns a copy of the dictionary

```
eg - car = { "brand" : "ford",
             "model" : "Mustang",
             "year" : 1964
           }

x = car.copy()
print(x)
```

Q.45 How to create a shallow copy of an object in Python Explain with example.

→ A shallow copy means constructing a new collection object and then populating it with references to the child objects found in the original. In case of shallow copy, a reference of object is copied in other object. It means that any changes made to a copy of object do reflect in the original object. In python this is implemented using "copy()" function.

Q.46 Example :-

```
# initializing list
li1 = [1,2,3,[5,4],6]
# using copy to shallow copy
li2 = copy.copy(li1)
# original elements of list
print("The original elements before shallow copying")
for i in range(0, len(li1)):
    print(li1[i], end=" ")
print("\r")
# adding an element to new list
li2[3][0]=7
# checking if change is reflected
```

```
print("the original element after shallow copying")
for i in range (0, len(li1)):
    print (li1[i], end = " ")
```

Output:

the original element before shallow copying
1 2 3 [5,4] 6
the original element after shallow copying
1 2 3 [7,4] 6

Q.46 What is the difference between deep and shallow copy.

	Shallow copy	Deep Copy
i)	Shallow copy stores the reference of object to the original memory address	i) Deep copy stores copies of the object's value.
ii)	Shallow copy reflects changes made to the new/copied object in the original object	ii) Deep copy dosen't reflect changes made to the new copied object in the original object.
iii)	Shallow copy stores the copy of the original object and points the references to the objects.	iii) Deep copy stores the copy of the original object and recursively copies the objects aswell.
iv)	Shallow copy is faster	iv) Deep copy is comparatively slower.

Q.47 Python supports lazy evaluation, justify with example.

→ The range method in Python follows the concept of lazy evaluation. It saves the execution time for larger ranges and we never require all the values at a time, so it saves memory consumption as well.

Example:

```
r = range (10)
print (r)
range (0, 10)
print (r[3])
```

Output -

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
3
```

48. How to create a module and use it in a python program explain with an example.

→ i) create a module - to create a module just save the code you want in a file with the extension .py

ii) use a module - now we can use the module we just created by using import statement

iii) Naming a module - you can name module whatever you like.

Example

```
import numpy as np
x = int(input("Enter the value for x:"))
y = int(input("Enter the value for y:"))

vec1 = np.array([x,y])
vec1 = np.array([-x,-y])
def add(a,b):
    return a+b
def mult(a,b):
    return a*b
def dot(a,b):
    return a[0]*b[0]+a[1]*b[1]
```

```
answer1 = add(x,y)
answer2 = mult(x,y)
answer3 = dot(vec1,vec2)
print answer1
print answer2
print answer3
```

Output :-

```
Enter the value for x : 2
Enter the value for y : 3
5
6
13
```

Q50 Explain the concept of namespaces with an example.

→ A namespace is a group of related element that such each have a unique name or identifier. Namespaces are used in many areas of computing, such as domain names, File paths, and xml documents. Below are examples.

1. Domain names - The namespace syntax for domain names is specified by the Domain name System or DNS. It includes the top level domain [eg. "techterms.com"] and subdomain such as "www." in the URL

Example

```
# var1 is an the global namespace
var1 = 5
def some_func():
    # var2 is in local namespace
    var2 = 6
    def some_inner_func():
        # var3 is nested local namespace
        var3 = 7
```

```
count = 5
def some_method():
    global count
    count = count + 1
    print(count)
some_method()
```

Output : 6

51. Write about the concept of scope of a variable in a function.

→ A variable is only available from inside the region it is created. This is called scope.

Local Scope :- A variable created inside a function belongs to the local scope of that function, and can only be used inside that function.

Example

A variable created inside a function ~~belongs to the local scope of that function, and can only be used inside that function~~ is available inside that function:

```
def myfunc():
    x = 300
    print(x)
myfunc()
```

Output : 300

52. Write about different types of arguments in a function.

→ Types of Arguments in Python function:-

1. Default arguments :-

① default arguments are values that are provided while defining functions.

② The assignment operator = is used to assign a default value to the argument

③ Default arguments become optional during the function calls.

2. Keyword Arguments:

① Functions can also be called using keyword arguments of the form kwarg=value.

② During function call, values passed through arguments need not be in the order of parameters in the function definition.

③ This can be achieved by keyword arguments.

3. Positional arguments

① During function call, values passed through arguments should be in the order of parameters in the order of parameters in the function definition. This is called positional arguments.

4. arbitrary positional arguments

① For arbitrary positional argument, an asterisk (*) is placed before a parameter in function definition which can hold non-keyword variable length arguments.

② These arguments will be wrapped up in a tuple.

③ Before the variable number of arguments, zero or more normal arguments may occur.

5. arbitrary keyword arguments:

① For arbitrary positional argument, a double asterisk (**) is placed before a parameter in function which can hold keyword variable length arguments.

53. List the features and explain about different Object Oriented features supported by Python. Explain the concept of method overriding with an example.

→ Major OOP concept of Python includes class, object, method, inheritence, polymorphism, data abstraction and encapsulation.

Inheritence - Inheritence is single inheritence, multilevel inheritence, multiple inheritence, hierarchical inheritence. It is mechanism in which one class aquires the property of another class. For eg. child inherits the traits of parents.

Polymorphism - It defines methods in the child class that have the same name as the method in the parent class. The child class have same name as method in parent class.

Encapsulation - is the process of wrapping up variable & methods into single entity

Method of overriding - It is ability of any object-oriented programming language that allows a subclass or child class to provide a specific implementation of method that is already provided by one. Super classes or parent classes

eg.
```
class parent():
    def __init__(self)
        self.value = "Inside Parent"
    def show(self):
        print(self.value)
class child(parent):
    def __init__(self):
    self.value = "inside child"
    def show(self):
        print(self.value)
```

54. How to declare a constructor method in python? Explain

→ Creating a constructor in Python:
This method is called when the class is instantiated. It accepts the self-keyword as a first argument which allows accessing the attributes or method of class. We can pass any number of argument at the time of creating the class object, depending upon the __init__() definition.

example:-

```
class Employee:
    def __init__(self, name, id):
        self.id = id
        self.name = name
    def display(self):
        print("ID: %d \n Name: %s" %(self.id, self.name))
emp1 = Employee("John", 101)
emp2 = Employee("David", 102)
```

55. Explain the feature of inheritence in Python with an example.

→ Inheritence allows us to define a class that inherits all the methods and properties from another class. Parent class is the class being inherited from also called base class child class is the class that inherits from another class also called derived class.

Example -

```
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname
    def printname(self):
        print(self.firstname, self.lastname)
```

56. Differentiate between an error & exception

	Error	Exception
Type	classified as an unchecked type.	Classified as checked & unchecked
Package	It belongs to java.lang. error	It belongs to java.lang. Exception.
occur	It can't be occur at compile time	It can occur at run as well as Compile time.

57. How to create a user defined exception?

→ Programmers may name their own exceptions by creating a new exception class. Exceptions need to be derived from the Exception class, either directly or indirectly. Although not mandatory, most of the exceptions are named as names that end in "Error" similar to naming of the standard exceptions in python.

For example:

```
class MyError (Exception):
    def __init__ (self, value):
        self.value = value
    def __str__ (self):
        return (repr(self.value))

try:
    raise (MyError (3*2))
except MyError as error:
    print ('A New Exception occured:',
    error.value)
```

Output:

('A New Exception occured:', 6)

~~58~~ ~~How to create a user defined exception?~~

58. How to handle an exception using try except block? Explain with the help of a program.

→ The try block lets you test a block of code for errors. The except block lets you handle the error. These exceptions can be handled using the try statement.

The try block will generate an exception, because x is not defined:

```
try:
    print(x)
except:
    print("An exception occured")
```

59. Explain about methods in Lists of python with appropriate examples.

→ Python has a set of built-in methods that you can use on lists/arrays.

append() - adds an element at the end of the list

clear() - removes all the elements from the list

copy() - returns a copy of the list

count() - returns the no. of elements with the specified value

extend() - add the elements of a list to the end of the current list

index() - returns the index of the first element with the specified value

insert() - adds an element at the specified position.

pop() - removes the element at the specified position.

remove() - removes the first item with the specified value.

reverse() - reverses the order of the list

sort() - sorts the list.

60. write a python program to describe different ways of deleting an element from the given list.

→ There are many ways of clearing the list through methods of different constructs offered by Python language. Let's try to understand each of the method one by one.

```
list1 = [1,2,3]
list2 = [5,6,7]
print ("List1 before deleting is :" + str (list1))
del list1 [:]
print ("List1 after clearing using del :" + str (list1))
print ("List2 before deleting is :" + str (list2))
del list2 [:]
print ("List 2 after clearing using del :" + str (list2))
```

Output :

List 1 before deleting is : [1, 2, 3]
List 1 after clearing using del : []
List 2 before deleting is : [5, 6, 7]
List 2 after clearing using del : []

61. What are the different operations that can be performed on a list? Explain with examples.

→ ~~appendeeee~~

```
myList = [1,2,3, 'EduCBA', 'makes learning fun!']
```

① append :- myList.append (4)
print (myList)

Output :- [1, 2, 3, 'EduCBA', 'makes learning fun!', 4]

② extend :- myList.extend ([4,5,6])
print (myList)

Output :- [1, 2, 3, 'EduCBA', 'makes learning fun!', 4, 5, 6]

(3) insert() :- mylist.insert(3,4) (4,5) (5,6)
print(mylist)
Output :- [1,2,3,4,5,6, 'EduCBA', 'makes learning fun!']

(4) remove() :- mylist.remove('makes learning fun') print(mylist)
output :- [1,2,3, 'EduCBA']

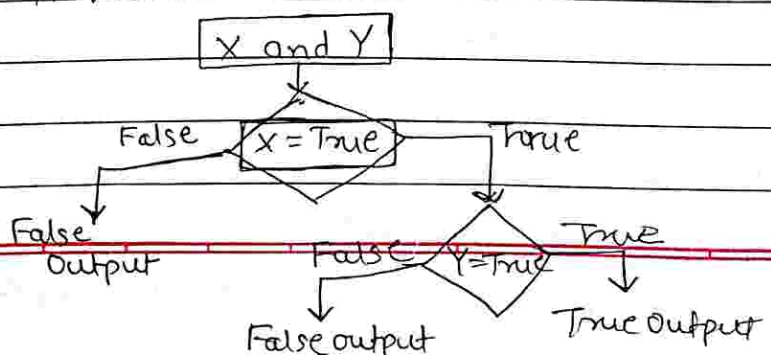
(5) pop() :- mylist.pop(4)
mylist.pop(3)
print(mylist)
Output :- [1,2,3]

(6) Slice() :- print(mylist[:4])
print(mylist[2:])
Output :-
[1,2,3, 'EduCBA']
[3, 'EduCBA', 'makes learning fun!']

62. Explain about different logical operators in python with eg.
Explain about different Relational operators in Python with eg.

→ Logical operators :-
In Python, logical operators are used on conditional statements (either True or False). They perform Logical AND, Logical OR, and Logical NOT operations.

- Logical AND operator
Logical operator returns True if bothe the operands are True else it returns False.



Example :

```
a = 10
b = 10
c = -10
if a > 0 and b > 0 :
    print ("The numbers are greater than 0")
if a > 0 and b > 0 and c > 0 :
    print ("The numbers are greater than 0")
else :
    print ("Atleast one number is not greater than 0")
```

Output :-

The numbers are greater than 0
Atleast one number is not greater than 0.

- Logical or operator

Logical or operator returns True if either of the operands is True.

Example :

```
a = 10
b = -10
c = 0
if a > 0 or b > 0 :
    print ("Either of the no is greater than 0")
else :
    print ("No number is greater than 0")
if b > 0 or c > 0 :
    print ("Either of the number is greater than 0")
else :
    print ("No number is greater than 0")
```

Output : Either of the number is greater than 0
No number is greater than 0.

- Logical not operator :
Logical not operator work with the single boolean value. If the boolean value is True it returns False and vice-versa.
Example :

```
a=10
if not a:
    print ("Boolean value of a is True")
if not (a % 3 ==0 or a %5 ==0):
    print (" 10 is not divisible by either 3 or 5")
else :
    print (" 10 is divisible by either 3 or 5")
```

Output :- 10 is divisible by either 3 or 5

63. → Explain about membership operators in python.
Membership operators are operators used to validate the membership of a value. It test for membership in a sequence, such as strings, lists or tuples.
1. 'in' operator : The 'in' operator is used to check if a value exists in a sequence or not. Evaluates to true if it finds a variable in the specified sequence and false otherwise.
2. 'Not in' operator : Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.

Python's membership operators test for membership in a sequence, such as strings, lists or tuples.

64. Explain about Identity Operators in Python.

→ In Python Identity Operators are used to determine whether a value is of a certain class or type. They are usually used to determine the type of data a certain variable contains. There are different identity operators such as -

1. 'is operator' :- Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.

```
x = 5
if (type(x) is int):
    print("true")
else:
    print("false")
```

Output:

true

2. 'is not' operator :- Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.

```
x = 5.2
if (type(x) is not int):
    print("true")
else:
    print("false")
```

Output:

true.

66. List different conditional statements in python.

→ (repeated)

67. What are different Loops available in Python?

→ Python programming language provides following types of loops to handle looping requirements.

While loop:

Repeats a statement or group of statements while a given condition is True. It tests the condition before executing the loop body.

for loop:

Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

Nested loops:

You can use one or more loop inside any another while, for or do... while loop.

68. What are different loop control statements available in python?

→ Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Python supports the following control statements.

- Continue statement:

It returns the control to the beginning of the loop.

Continue statement causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

- Break statement:

Terminates the loop statement and transfers execution to the statement immediately following the loop.

69. Give a note on i) quotes (Single, double and triple)
ii) multiline statements
iii) indentation

→ A string as a sequence of characters not intended to have numeric value. In Python, such sequence of characters is included inside single or double quotes. As far as language syntax is concerned, there is no difference in single or double quoted string. Both representations can be used interchangeably. However, if either single or double quote is a part of the string itself, then the string must be placed in double or single quotes respectively.

Example:

```
Str 2 = "Hello 'Python' "
```

Python's triple quotes comes to the rescue by allowing strings to span multiple lines, including verbatim newlines, TABS, and any other special characters.

The syntax for triple quotes consists of three consecutive single or double quotes.

```
para_str = """this is a long string"""
print para_str
```

70. What is lambda function?

→ In Python, an anonymous functions means that a function is without a name. As we already know that the def keyword is used to define a normal function in Python. Similarly, the lambda keyword is used to define an anonymous function in Python.

Syntax: lambda arguments : expression

This fun can have any number of arguments but only one expression, which is evaluated and returned.

71. What are the characteristics of a lambda function? Give an example.

→
- This function can have any number of arguments but only one expression, which is evaluated and returned.
- One is free to use lambda functions wherever function objects are required.
- You need to keep in your knowledge that lambda functions are syntactically restricted to a single expression.
- It has various uses in particular fields of programming besides other types of expressions in functions.

72. Write a recursive Python function that recursively computes sum of elements in a list of lists.

→
```
list1 = [11, 5, 17, 18, 23]
def sumOfList (list, size) :
    if (size==0) :
        return 0
    else :
        return list [size-1] + sumOfList (list, size-1)
total = sumOfList (list1, len(list1))
print ("Sum of all elements in given list :", total)
```

Output:

Sum of all elements in given list : 74

73. What are different types of inheritence supported by Python?

→ Types of Inheritence depends upon the number of child and parent classes involved. There are four types of inheritence in Python:

① Single Inheritence : enables a derived class to inherit properties from a single parent class, thus enabling code reusability and the addition of new features to existing code.

② Multiple Inheritence : When a class can be derived from more than one base class this type of inheritence is called multiple inheritence.

③ Multilevel Inheritence : features of the base class and the derived class are further inherited into the new derived class.

④ Hierarchical Inheritence : When more than one derived classes are created from a single base this type of inheritence is called hierarchical inheritence.

⑤ ~~Hybrid~~

74. What is the difference between else block and finally block in exception handling? Explain with an example.

→ Else code comes into execution only when there is no exception raised in the try block. The code inside this block is the same as normal code. If an exception is raised then this block will not run and may stop the program. Finally code executes at the last when all other blocks have completed execution, it will work even if there was no exception or an uncaught exception or there is a return statement in any of the other above blocks, it will run in every case.

75. Explain the List Accessing Methods and List comprehension.

→ Python has a set of built-in methods that you can use on lists/arrays.

append() - adds an element at the end of the list.
clear() - removes all the elements from the list
copy() - returns a copy of the list
count() - returns the number of elements with the specified value.
extend() - Add the elements of a list, to the end of the current list
index() - returns the index of the first element with the specified value.
insert() - Adds an element at the specified position.
pop() - removes the element at the specified position.
remove() - removes the first item with the specified value.
reverse() - reverse the order of the list
sort() - sorts the list

76. Describe about variable length arguments with example.

→ You may need to process a function for more arguments than you specified while defining the function. These arguments are called variable-length arguments and are not named in the function definition, unlike required and default arguments.

Example :-

```
def printinfo (arg1, *vartuple):
"This prints a variable passed arguments"
print "Output is :"
print arg1
for var in vartuple:
print var
return:
printinfo (10)
printinfo (70, 60, 50)
```

Output is :
10
Output is :
70
60
50

77. What are the two ways of importing a module? Which one is more beneficial?

→ import module_name

When import is used, it searches for the module initially in the local scope by calling __import__() function. The value returned by the function are then reflected in the output of the initial code.

```
import math
print(math.pi)
```

Output :- 3.141592653589793

import module_name.member_name

In the above code module math is imported, and its variables can be accessed by considering it to be a class and pi as its object. The value of pi is returned by __import__(). pi as whole can be imported into our initial code, rather than importing the whole module.

```
from math import pi
print(pi)
```

Output :- 3.141592653589793

from module_name import *

In the above code module math is not imported, rather just pi has been imported as a variable. All the functions and constants can be imported using *

```
from math import *
print(pi)
print(factorial(6))
```

Output : 3.141592653589793
720

78. Describe the features of python.

→ ① Easy to code :- Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in python language.

② Free and Open source :-
Python language is freely available at the official website and you can download it from that given. Source code is also available to the public.

③ Object-Oriented Language :-
One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

④ High-level language :-
It is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

⑤ Extensible feature :-
It is a Extensible language. We can write some python code into C or C++ language and also we can compile that code in C/C++ language.

⑥ Python is Portable language :-
Python language is also a portable language.

79. ~~Explain the~~ difference between Class variables & instance variables.

→

Instance variables	Class variables.
① Declared in class but outside a method, constructor or any block.	① Declared with the static keyword in a class, but outside a method constructor or a block.

② created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.

③ It can be accessed directly by calling the variable name inside the class:

ObjectReference.VariableName

② created when the program starts and destroyed when the program stops.

③ It can be accessed by calling with the class name

ClassName.VariableName

81. What are the different function prototypes?

→ Function Prototypes :-

i. Function without arguments and without return type. :-
In this type no argument is passed through the function call and no output is return to main function.

ii. Function with arguments and without return type :-
Arguments are passed through the function call but output is not return to the main function.

iii. Function without arguments and with return type :-
In this type no argument is passed through the function call but output is return to the main function.

iv. Function with arguments and with return type. :-
In this type arguments are passed through the function call and output is return to the main function.

82. Explain various string pattern matching functions in Python.

→ Regex is used in Python to match a string of three numbers, a hyphen, three more numbers, another hyphen and four nos.

```
import re
phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
mo = phoneNumRegex.search('My no is 415-555-4242.')
print('Phone No found:' + mo.group())
```

Output:- Phone no found : 415-555-4242.

83. Can a Python Function return multiple values? If yes, how it works?

→ Python functions can return multiple values. These values can be stored in variables directly. A function is not restricted to return a variable, if it can return zero, one, two or more values. This is the default property of python to return multiple values / variables which is not available in many other programming languages like c++ or Java. For returning multiple values from a function, we can return tuple, list or dictionary object as per our requirement.

84. Write a python program that interchanges the first and last character of a given string.

→
```
def swap(str):
    # storing the first character
    start = str[0]
    # storing the last character
    cmd = str[-1]
    swapped_str = cmd + str[1:-1] + start
swap("Python")
```
Output: nythoP

85. Give a comparison between lists, tuples, dictionaries and sets.

→ ① A list is a sequence of elements in a specific order. You can access elements with a numerical index, e.g. the list[3]. The time taken for several operations such as testing if the list contains an element is O(n). i.e. proportional to the length of the list.

② A tuple is basically an immutable list, meaning you

cannot add, remove or replace any elements.

③ A set has no order, but has the advantage over a list that testing if the set contains an element is much faster, almost regardless of the size of the set. It also has some handy operations such as union and intersection.

④ A dictionary is a mapping from keys to values where the keys can be all sorts of different objects, in contrast to lists where the keys can only be members. So you can have the dict = {'abc':3, 'def':8} and then the dict['abc'] is 3. The keys of a dict are much like a set. They have no order and you can test for their existence quickly.

⑤ The elements of a set and the keys of a dict must be hashable. Numbers, strings, tuples and many other things are hashable. Lists, sets and dicts are not hashable

86. What type of parameter passing is used in Python? Justify your answer with sample programs.

→ Python passes arguments by assignment i.e. when you call a python function, each function argument becomes a variable to which the passed value is assigned. In Python every variable name is a reference. When pass a variable to a function a new reference to the object is created. Parameter passing in Python is same as reference passing in Java.

```
# Here x is a new reference to same list lst
def fun(x):
    x[0] = 20
# note that lst is modified after function call
lst = [10, 11, 12, 13, 14, 15]
```

But when we pass a reference and change the received reference to something else, the connection bet^n passed and received parameter is broken. For example, consider the below program.

```
def fun(x):
    # after below line link of x with previous object gets
    # broken. A new object is assigned to x.
    x = [20, 30, 40]
# Note that lst is not modified after function call
lst = [10, 11, 12, 13, 14, 15]
    func(lst);
print(lst)
```

Output: [10, 11, 12, 13, 14, 15]

87. Write a Python function that prints all factors of a given number.

→

```
def print_factors(x):
    print("The factors of", x, "are:")
    for i in range(1, x+1):
        if x%i == 0:
            print(i)
num = int(input("Enter a number"))
print factors(num)
```

Output: The factor of 36 are:

1
2
3
4
6
9
12
18
36

88. Explain creating classes in Python with examples.

→ A class is a user defined blue print or prototype from which objects are created. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instances can have attributes attached to it for maintaining its state.

Class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. Classes are created by keyword class.

Class definition syntax:

```
class ClassName:
    #statement-1
    :
    # statement-N
```

Defining a class -

```
class Dog:
    pass
```

In the above example, class keyword indicates that you are creating a class followed by the name of the class

89. Write a function is reverse (string 1, string 2) to return true if string 1 is reverse of string 2 : is reverse ('pots', 'stop')
True

```
def is_reverse (word 1, word 2):
    if len (word 1) != len (word 2):
        return false
    else:
        i=0
        j = len (word 2) - 1
```

```
while i>=0:
if word1[i] != word2[i]:
        return False
else:
        i= i+1
        j= j-1
return True
b= is_reverse('pots','stop')
print(b)
b = is_reverse('potss','stop')
print(b)
```

Output : True
False

90. Write a Python code to check whether the given number is a strong number. Strong Numbers are numbers whose sum of factorial of digits is equal to the original number. (Example : $145 = 1! + 4! + 5!$)

→

```
def strongno(n):
    n!=n
    s = 0
    while n!=0:
        d = n % 10
        i = 1
        f = 1
        while i<=d:
            f = f * i
            i = i+1
        s = s+f
        n = n //10
```

```
if s==n1:
    print ("{0} is a strong no". format (n1))
else:
    print ("{0} is not a strong no". format (n1))
```

Output:

strongno(145) — 145 is a strong no.

strongno(125) — 125 is not a strong NO.

91. List and explain any five exceptions in Python.

→ The table below shows built in exceptions that are usually raised in Python.

Exception	Description
Arithmetic Error	Raised when an error occurs in numeric calculations.
Assertion Error	Raised when an assert statement fails.
Attribute Error	Raised when attribute reference or assignment fails.
Exception	Base class for all exceptions.
EOF Error	Raised when the input() method hits an end of life condition (EOF)
Import Error	Raised when an imported module does not exists
Indentation Error	Raised when indentation is not Correct.

92. Name and explain magic methods of Python that are used in the initialization and deletion of class objects with the help of a code snippet.

→ Magic methods are not ment to be invoked directly by us,

but the invocation happens internally from the class on a certain action. For example, when you add 2 numbers using the + operator, internally, the --add--() method will be called.

--new--() method

In Python the --new--() magic method is simplicitly called before the --init--() method. The --new--() method returns a new object, which is then initialized by --init--()

Example: --new--()

```
class employee:
        def --new--(cls):
    print("--new-- magic method is called")
    inst = object.--new--(cls)
            return(inst)
            def --init--(self):
    print("--init-- magic method is called")
    self.name = 'sam'
```

The above example will create the following output when you create an instance of the employee class.

```
>>> employee()
--new-- magic method is called
--init-- magic method is called
```

Thus the --new--() method is called before the --init--() method. The --del--() method is known as a destructor method in Python. It is called when all references to the object have been deleted i.e. when an object is garbage collected.

93. What is the significant difference between list and dictionary?

→ ① Lists are just like the arrays, declared in other languages. Lists need not be homogeneous always which makes it a most powerful tool in Python. A single list may contain Datatypes like Integers, Strings as well as Objects. Lists are mutable, and hence, they can be afford even after their creation.

② Dictionary on the other hand is an unordered collection of the data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimized. Each key value pair in a Dictionary is seperated by a colon, whereas each key is seperated by a 'comma'.

94. Write a Python code to get the following dictionary as output:
{1:1, 3:9, 5:25, 7:49, 9:81}

→ odd_squares = {x:x*x for x in range(11) if x%2==1}
print(odd_squares)

Output:
{1:1, 3:9, 5:25, 7:49, 9:81}

95. What is the output of the code given below:

```
>>> squares = {1:1, 2:4, 3:9, 4:16, 5:25}
>>> print (squares[5])
>>> print (squares[6])
```

→ Output: 25
Traceback (most recent call last):
File "<string>", line 3 in <module>
KeyError: 6

Since there is no key 6 present in the dictionary it will show error on line 3.

96. Explain the role of the Regular Expressions in the following snippets:

→ >>> p=re.compile ('\d+')

- \d+ will match a group on [0-9], group of one or greater size.

>>> p.findall ('12 drummers, drumming, 11 pipers piping, 10 lords a-leaping')

- findall () searches for the Regular Expression and return a list upon finding ['12', '11', '10']

>>> p = re.compile ('ca*t')

- ca*t will match 'ct' (0 'a' characters), 'cat' (1 'a'), 'caaat' (3 'a' characters).

97. Write a program to sort a dictionary in ascending and descending order of values.

→ import operator.

```
d = {1:2, 3:4, 4:3, 2:1, 0:0}
print ('original dictionary :', d)
sorted_d = sorted (d.items (), key = operator.itemgetter(1))
print ('Dictionary in ascending order by value :', sorted_d)
sorted_d = dict (sorted (d.items (), key = operator.itemgetter(1),
                reverse = True))
print ('Dictionary in descending order by value :', sorted_d)
```

Sample output:

Original dictionary : {1:2, 3:4, 4:3, 2:1, 0:0}

Dictionary in ascending order by value : [(0,0), (2,1), (1,2), (4,3), (3,4)]

Dictionary in descending order by value : {3:4, 4:3, 1:2, 2:1, 0:0}

98. Write a program to take a character from the user and search that character in the file. If the character is present then print total count of that character in the file or else display the message "No such character found".

→
```
f = open ("FileH_testing.txt", "r")
user_ip = input("Enter a character to search it in the file:")
flag = 0
count = 0
for j in f.read():
    if j == user_ip:
        count += 1
        flag = 1
if flag == 1:
    print ("The given character", user_ip, "occured", count, "times!")
else:
    Print(" character", user_ip, "not found")
```

99. How can string be traversed with a loop? Give suitable example. Explain the following string functions with example.
1) startswith() 2) rstrip()

→

A string can be traversed as a substring by using the Python slice operator ([]). It cuts off a substring from the original string and this allows to iterate over it partially. To use this method provide the starting and ending indices along with a step value and then traverse the string.

1) startswith() method returns a boolean. It returns True if the string starts with the specified prefix. It returns False otherwise.

2) rstrip() - method removes any trailing character. space is the default trailing character to remove.

100. Explain the continue statement with example.

→ It returns the control to the beginning of the while loop. The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop. The continue statement can be used in both while and for loops.

Example:

```
for letter in 'Python':
    if letter == 'h':
        continue
    print 'Current Letter', letter
```

Output:

Current letter: P
Current letter: y
Current letter: t
Current letter: o
Current letter: n

101. Write a program in Python to create a function using lambda to find square of a number.

→

```
num = int (input ("Enter a number"))
print ("In Square of", num, "is")
square_num = (lambda x: x * x) (num)
print (square_num)
```

102. Give Python statement for the following: Create a string mystring with value "Python world" Print last character of above string

→

→ mystring = "Python world"
print (mystring[-1])
Output : d

103. Give the output of following statements:
A0 = dict (zip (('a', 'b', 'c', 'd', 'e'), (1,2,3,4,5)))
A1 = range (10)
A2 = sorted ([i for i in A1 if i in A0])
~~A3 = [i for i in A1 if i~~
A3 = sorted ([A0[s] for s in A0])
A4 = [i for i in A1 if i in A3]
A5 = {i : i*i for i in A1}
A6 = [[i, i*i] for i in A1]

→ A0 = {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4}
A1 = range (0,10)
A2 = []
A3 = [1,2,3,4,5]
A4 = [1,2,3,4,5]
A5 = {0:0, 1:1, 2:4, 3:9, 4:16, 5:25, 6:36, 7:49, 8:64, 9:81}
A6 = [[0,0], [1,1], [2,4], [3,9], [4,16], [5,25], [6,36], [7,49], [8,64], [9,81]]

104. Predict the output of following.

```
def f(x, l=[]):
    for i in range(x)
    l.append (i*i)
    print (l)
f (2)
f (3, [3,2,1])
f (3)
```

Output: [0, 1]
[3,2,1,0,1,4]
[0,1,0,1,4]

105. What does this stuff mean: *args, **kwargs? And why would we use it?

→ Use *args when we aren't sure how many arguments are going to be passed to a function, or if we want to pass a stored list or tuple of arguments to a function. **kwargs is used when we dont know how many keyword arguments will be passed to a function, or it can be used to pass the values of a dictionary as keyword arguments. The identifiers args and kwargs are a convention, you could also use *bob and **billy but that would not be wise.

106. What is global and local variables in Python?

→ Global variable - we can access anywhere in the program.
Local variable - we can access inside a function

107. Create a tuple which contains the following:
- a string
- a sub list of values belonging to a single type.
- a sub Tuple with values belonging to a single data type.

→ tuple_1 = ("abcdef", [1,2,3], (1.1, 2.3, 4.6))

108,109. Explain the concept of Tuple Packaging and unpacking

→ In Python there is a very powerful tuple assignment feature that assigns right hand side of values into left hand side. In other way it is called unpacking of a tuple of values into a variable. In packing, we put values into a new tuple while in unpacking we extract those values into a single variable. In unpacking of tuple, number of variables on left side should be equal to number of values in given tuple.

WORLD STAR™
Date :
Page :

10. How split and join methods works in Python?

→ Python split method splits the given string based on a specific seperator (which can be a character, symbol or even empty space) It returns a list of substrings which are seperated based on the given seperator value.

Syntax : String.split (seperator / specific value, max splits)

The join method helps combine or concatenate strings but not like the + operator. When we use the join method, it combines every element of the input iterable with the calling string and returns a string.

Syntax : string.join (iterable)

11. Create a module in Python. How will you share the global variables across the modules.

→ To create a module just save the code you want in a file with file extension .py :

Save the below example code as mymodule.py

```
def greeting (name):
    print ("Hello", + name)
```

Then import the module and all greeting function Impo

```
import mymodule
mymodule.greeting ("John")
```

The best way to share global variables across modules across a single program is to create a config module. Just import the config module in all modules of your application; the module then becomes available as a global name.

- config.py

```
x=0
```

- mod.py

```
import config.py
```

config.x = 1
main.py:
import config
import mod
print (config.x)

Output : 1

112. What is docstring in Python?

→ Python docstrings are string literals that appear right aft. the definition of a function, method, class or module. They are used to document our code. We can access these docstrings using the --doc-- attribute.

113. What is set in Python?

→ Sets are used to store multiple items in a single variable. Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, tuple and Dictionary. A set is a collection which is both unordered and unindexed.

Example: thisset = {"apple", "banana", "cherry"}

Sets are unorderd.

114. What is numpy arrays and its benefits?

→ Numpy is used to work with arrays. The array object in Numpy is called ndarray. A numpy array is a grid of values, all of the same type and is indexed by a tuple of nonnegative integers.

Benefits : 1) Contiguous allocation in Memory
2) Vectorized operations.
3) Boolean selection
4) Sliceability

115. What is difference between numpy arrays and traditional arrays?

→ Python does not have built in support for Array but Python lists can be used instead.

Numpy arrays is a typed array, the array in memory stores a homogeneous densely packed members. Python lists is a hetergeneous list, the list in memory stores references to objects rather than the number themselves. This means that Python list requires dereferencing a pointer every time the code needs to access the number. while numpy array can be processed directly by numpy vector operations, which makes these vector operations much faster than anything you can code with list. The drawback of numpy array is that if you need to access single items in the array, numpy will to box/unbox the number into a Python numeric object, which can make it slow in certain situations and that it can't hold heteregeneous data.

116. How can you calculate computational time for a python program?

→ We have a method called time() in the time module in Python, which can be used to get the current time. See the following steps:

- Store the starting time before the first line of the program executes
- Store the ending time after the last line of the program executes.
- The difference between ending time and starting time will be the running time of the program.

Example :

```
import time
start = time.time()
for i in range(10):
    print(i)
time.sleep(1)
end = time.time()
print("Runtime : {end-start}")
```

117. How can you generate random variables in numpy?

→ An array of random variables can be generated using the randint(). Numpy function. This function takes three arguments the lower and of the range, the upper, end of the range, and the number of integer values to generate or the size of the array. Random integers will be drawn from a uniform distribution including the lower value and excluding the upper value, e.g. in the interval [lower, upper).

118. How can you generate random variables with distributions in numpy?

→ Generating random numbers with Numpy distributions

- Import Numpy

```
import numpy as np
```

- Generate A Random Number from the normal distribution

```
np.random.normal()
0.5661104974399703
```

- Generate 4 random numbers from the normal distribution.

```
np.random.normal(size=4)
-array([0.0019323, 0.51932356, 0.8765684, 0.33684494])
```

119. What is Pandas? What is use of it?

→ Pandas is a software library written for the Python Programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

120. How to import csv files using pandas?

→ ① Capture the file path
Firstly capture the full path where your csv file is stored.
For eg c:\users\sam\Desktop\1.csv
② Apply python code
import Pandas as pd
df = pd.read_csv ('path where the csv file is stored')
print (df)
③ Run the code.

124. What is series?

→ Series is a one-dimensional labeled array capable of holding data of any type. The axis labels are collectively called index.
A Panda series can be created using the following constructor
pandas.series (data, index, dtype, copy)

122. What is Dataframe?

→ Dataframe is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table or a dict of series objects. It is generally the most commonly used Pandas object.