###### Q.no 1.Discuss interaction with Python program with example ?

**QUESTION BANK**

1. The Python program that we have installed will by default act as an interpreter.
2. An interpreter takes text commands and runs them as we enter text.
3. After Python opens, it will show some contextual information like this : Python 3.5.0 (default, dec 20 2019, 11 : 28 : 25)

[GCC 5.2.0] on Linux

Type “help”, “copyright”, “credits” or “license” for more information

>>>

1. The prompt >>> defines that we are now in an interactive Python interpreter session, also called the Python shell.
2. Now enter some code for Python to run such as print(“Hello World!”) and press the Enter key.
3. The interpreter’s response should appear on the next line like this :

>>> print (“Hello World!”) Hello World!

1. After showing the results, Python will bring you back to the interactive prompt, where we could enter another command or code.

Python program communicates its result to user using print statement

###### QuesNo:2 What is Python ? How Python is interpreted ? What are the tools that help to find bugs or perform static analysis ? What are Python decorators ?

**.**

**Python :** Python is a high-level, interpreted, interactive and object-oriented scripting language. It is a highly readable language. Unlike other programming languages, Python provides an interactive mode similar to that of a calculator.

###### Interpretation of Python :

###### An interpreter is a kind of program that executes other programs.

1. When we write Python programs, it converts source code written by the developer into intermediate language which is again translated into the machine language that is executed.
2. The python code we write is compiled into python bytecode, which creates file with extension .pyc .
3. The bytecode compilation happened internally and almost completely hidden from developer.
4. Compilation is simply a translation step, and byte code is a lower-level, and platform-independent, representation of source code.
5. Each of the source statements is translated into a group of bytecode instructions. This bytecode translation is performed to speed execution. Bytecode can be run much quicker than the original source code statements.
6. The .pyc file, created in compilation step, is then executed by appropriate virtual machines.
7. The Virtual Machine iterates through bytecode instructions, one by one, to carry out their operations.
8. The Virtual Machine is the runtime engine of Python and it is always present as part of the Python system, and is the component that actually runs the Python scripts.

It is the last step of Python interpreter

###### Following tools are the static analysis tools that help to find bugs in python :

1. **Pychecker :** Pychecker is an open source tool for static analysis that detects the bugs from source code and warns about the style and complexity of the bug.

###### Pylint :

* 1. Pylint is highly configurable and it acts like special programs to control warnings and errors, it is an extensive configuration file
  2. It is an open source tool for static code analysis and it looks for programming errors and is used for coding standard.
  3. It also integrates with Python IDEs such as Pycharm, Spyder, Eclipse, and Jupy

###### Python decorators :

1. Decorators are very powerful and useful tool in Python since it allows programmers to modify the behavior of function or class Decorators allow us to wrap another function in order to extend the behavior of wrapped function, without permanently modifying it.
2. In decorators, functions are taken as the argument into another function and then called inside the wrapper function.

###### Syntax :

@gfg\_decorator

def hello\_decorator(): print(“Gfg”)

gfg\_decorator is a callable function, will add some code on the top of some another callable function, hello\_decorator function and return the wrapper function.

###### Ques No 03: What do you mean by comments in Python ?

**Comments :**

1. Python allows us to add comments in the code.
2. Comments are used by the programmer to explain the piece of code to be understood by other programmer in a simple language. Every programming language makes use of some character for commenting.
3. Python uses the hash character (#) for comments. Putting # before a text ensures that the text will not be parsed by the interpreter.
4. Comments do not affect the programming part and the Python interpreter does not display any error message for comments.

**For example :** Commenting using hash mark (#)

>>> 8 + 9 # addition

17 # Output

>>>

In this example, ‘addition’ is written with a hash mark. Hence the interpreter understands it as a comment and does not display any more message.

**Ques No 04:Explain identifiers and keywords with example?**.

1. A Python identifier is the name given to a variable, function, class, module or other object.
2. An identifier can begin with an alphabet (A – Z or a – z), or an underscore (\_) and can include any number of letters, digits, or underscores and spaces are not allowed.
3. Python will not accept @, $ and % as identifiers.
4. Python is a case-sensitive language. Thus, Hello and hello both are different identifiers. In python, a class name will always start with a capital letter.

What do you mean by data types ? Explain numeric and string data type with example

**Data types :**

1. The data stored in the memory can be of many types. For example, a person’s name is stored as an alphabetic value and his address is stored as an alphanumeric value.
2. Python has six basic data types which are as follows :
   1. Numeric
   2. String
   3. List
   4. Tuple
   5. Dictionary
   6. Boolean

###### Numeric :

1. Numeric data can be broadly divided into integers and real numbers (*i.e*., fractional numbers). Integers can be positive or negative.
2. The real numbers or fractional numbers are called, floating point numbers in programming languages. Such floating point numbers contain a decimal and a fractional part.

###### For example :

>>> num1 = 2 # integer number

>>>num2 = 2.5 # real number (float)

>>>num1

2 # Output

>>>num2

2.5 # Output

>>>

###### Ques No 05: Discuss list and tuple data types in detail ?

**List :**

1. A list can contain the same type of items.
2. Alternatively, a list can also contain different types of items.
3. A list is an ordered and indexable sequence.
4. To declare a list in Python, we need to separate the items using commas and enclose them within square brackets ([ ]).
5. Operations such as concatenation, repetition and sub-list are done on list using plus (+), asterisk (\*) and slicing (:) operator.

###### For example :

>>>first = [1, “two”, 3.0, “four” ] # 1st list

>>>second = [“five”, 6] # 2nd list

>>>first # display 1st list

[1, ‘two’, 3.0, ‘four’] # Output

###### Tuple :

1. A tuple is also used to store sequence of items.
2. Like a list, a tuple consists of items separated by commas.
3. Tuples are enclosed within parentheses rather than within square brackets.

###### For example :

>>>third = (7, “eight”, 9, 10.0)

>>>third

(7, ‘eight’, 9, 10.0)

**Ques No 6: : What is expression?**

1. An expression is a combination of symbols that evaluates to a value.
2. An expression is a combination of variables, operators, values sub-expressions and a reserve keyword.
3. Whenever we type an expression in the command line, the interpreter evaluates it and produces the result.
4. Expressions that evaluate to a numeric type are called arithmetic expressions.
5. A sub-expression is any expression that is part of a larger expression. Sub-expressions are denoted by the use of parentheses.

**For example :** 4 + (3 \* *k*)

An expression can also consist of a single literal or variable. Thus, 4, 3, and *k* are each expression.

This expression has two sub-expressions, 4 and (3 \* *k*). Sub-expression (3 \* *k*) itself has two sub-expressions, 3 and *k*.

**Ques No 7:What are conditional statements in Python ?**

1. Conditional statements help in making a decision based on certain conditions.
2. These conditions are specified by a set of conditional statements having Boolean expressions which are evaluated to true or false.
3. Conditional statements are also known as decision-making statements.
4. Python supports conditional execution using if-else statements.
5. In Python, we use different types of conditional statements:
   1. If statement
   2. If-else statement
   3. Nested-if statement
   4. Elif statement

###### Ques No 8.Explain if statement with the help of an example ?

**.**

1. An if statement consists of a Boolean expression followed by one or more statements.
2. With an if clause, a condition is provided; if the condition is true then the block of statement written in the if clause will be executed, otherwise not.

###### Syntax :

If (Boolean expression) : Block of code #Set of statements to execute if

the condition is true

###### For example :

num = 5

if (num > 10) :

print (“Number is greater than 10”) else :

print (“Number is less than 10”)

print (“This statement will always be executed”)

###### Output :

###### Number is less than 10

###### Ques No 09:Write a program to find whether a number is even or odd number ?

number = int (input (“Enter an integer number :”))

>>> if (number % 2) == 0 : print “Number is even”

else :

print “Number is odd” Enter an integer number : 6

###### Number is even

###### Ques No 10:Write a Python program to check if the input year is a leap year or not ?

>>> year = int (input (“Enter year :”))

>>> if (year % 4) == 0 :

if (year % 100) == 0 :

if (year % 400) == 0 :

print (year, ‘is leap year’) else :

print (year, ‘is not leap year’)

else :

print (year, ‘is leap year’)

else :

print (year, ‘is leap year’)

###### Output :

Enter a year : 2016 2016 is leap year **Output :**

Enter a year : 1985 1985 is not leap year

###### Ques No 11.What is nested loop ? Explain ?

**.**

1. Loop defined within another loop is known as nested loops.
2. Nested loops are the loops that are nested inside an existing loop, that is, nested loops are the body of another loop.

###### Syntax :

for condition1 :

for condition2 :

Body of for loop

###### For example :

for i in range(1,9,2) : for j in range(i) :

print( i, end = ‘ ’) print()

###### Output :

1

3 3 3

5 5 5 5 5

7 7 7 7 7 7

###### Question No12: Define function and write its advantages ?

1. Functions are self-contained programs that perform some particular tasks.
2. Once a function is created by the programmer for a specific task, this function can be called anytime to perform that task.
3. Each function is given a name, using which we call it. A function may or may not return a value.
4. There are many built-in functions provided by Python such as dir (), len ( ), abs ( ), etc.
5. Users can also build their own functions, which are called user-defined functions.

###### Advantages of using functions :

1. They reduce duplication of code in a program.
2. They break the large complex problems into small parts.
3. They help in improving the clarity of code (*i*.*e*., make the code easy to understand).
4. A piece of code can be reused as many times as we want with the help of functions.

**Ques No 13:How to define and call function in Python** ? **Explain different parts of a function ?**

Function is defined by “def” keyword following by function name and parentheses.

**Syntax of function definition :** def function\_name ( ) :

**Syntax of functional call :** function\_name ( )

###### For example :

2 arguments x and y (formal args)

Function name

Keyword

if (x > y) : return x

else

return y

def

max

(x, y) :

‘‘‘return maximum among x and y’’’

|  |  |
| --- | --- |
|  | |
|  | Body e fion |
|  |

Call to the function. Actual args are 8 d 6.

Doumenton comment (docstring)

1. **Keyword :** The keyword ‘def’ is used to define a function header.
2. **Function name :** We define the function name for identification or to uniquely identify the function. In the given example, the function name is max. Function naming follows the same rules of writing identifiers in Python.
3. A colon (:) to mark the end of function header.
4. **Arguments :** Arguments are the values passed to the functions between parentheses. In the given example, two arguments are used, *x* and *y*. These are called formal arguments.
5. **Body of the function :** The body processes the arguments to do something useful. In the given example, body of the function is intended

w.r.t. the def keyword.

1. **Documentation comment (docstring) :** A documentation string (docstring) to describe what the function does. In the given example, “return maximum among *x* and *y*” is the docstring.
2. An optional return statement to return a value from the function.
3. **Function call :** To execute a function, we have to call it. In the given example, *a* = max (8, 6) is calling function with 8 and 6 as arguments.

###### Ques No 14: What do you mean by the term strings ?

1. Strings are created by enclosing various characters within quotes. Python does not distinguish between single quotes and double quotes.
2. Strings are of literal or scalar type. The Python interpreter treats them as a single value.
3. Strings, in Python, can be used as a single data type, or, alternatively, can be accessed in parts. This makes strings really useful and easier to handle in Python.

###### For example :

>>> var1 = ‘Hello Python!’

>>> var2 = “Welcome to Python Programming!”

>>> print var1

Hello Python! # Output

>>> print var2

Welcome to Python Programming! # Output

Define tuples. How are tuples created in Python

1. Tuples are the sequence or series values of different types separated by commas (,).
2. Values in tuples can also be accessed by their index values, which are integers starting from 0.

###### For example :

The names of the months in a year can be defined in a tuple :

>>> months = (‘January’, ‘February’, ‘March’, ‘April’, ‘May’, ‘June’, ‘July’, ‘August’, ‘September’, ‘October’, ‘November’, ‘December’)

###### Creating tuples in Python :

1. To create a tuple, all the items or elements are placed inside parentheses separated by commas and assigned to a variable.
2. Tuples can have any number of different data items (that is, integer, float, string, list, etc.).

###### For examples :

1. **A tuple with integer data items :**

>>> tuple = (4, 2, 9, 1)

>>> print tuple

(4, 2, 9, 1) # Output

###### A tuple with items of different data types :

>>>tuple\_mix = (2, 30, “Python”, 5.8, “Program”)

>>>print tuple\_mix

(2, 30, ‘Python’, 5.8, ‘Program’) # Output

###### Nested tuple :

>>>nested\_tuple = (“Python”, [1, 4, 2], [“john”, 3.9])

>>> print nested\_tuple

(‘Python’, [1, 4, 2], [‘john’, 3.9]) # Output

###### Tuple can also be created without parenthesis :

>>>tuple = 4.9, 6, ‘house’

>>>print tuple

(4.9, 6, ‘house’) # Output.

**Ques No 15: Explain Built-in list operator in list ?**

1. **Concatenation :** The concatenation operator is used to concatenate two lists. This is done by the + operator in Python.

###### For example :

>>>list1 = [10, 20, 30, 40]

>>>list2 = [50, 60, 70]

>>>list3 = list1 + list2

>>>print list3

[10, 20, 30, 40, 50, 60, 70] # Output

1. **Repetition :** The repetition operator repeats the list for a given number of times. Repetition is performed by the \* operator.

###### For example :

>>>list1 = [1, 2, 3]

>>>list 1 \* 4

[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3] # Output

###### In operator :

* 1. The in operator tells the user whether the given string exists in the list or not.
  2. It gives a Boolean output *i.e*., true or false.
  3. If the given input exists in the string, it given true as output, otherwise, false.

###### For example :

>>>list = [‘Hello’, ‘Python’, ‘Program’]

>>> ‘Hello’ in list True # Output

>>>‘World’ in list False # Output

###### Ques No 16:What are mutable sequences ? Discuss with example ?

**.**

1. Python represents all its data as objects. Mutability of object is determined by its type.
2. Some of these objects like lists and dictionaries are mutable, meaning we can change their content without changing their identity.
3. Other objects like integers, floats, strings and tuples are immutable, meaning we cannot change their contents.

###### Dictionaries are mutable in Python :

Dictionaries in Python are mutable.

* 1. The values in a dictionary can be changed, added or deleted.
  2. If the key is present in the dictionary, then the associated value with that key is updated or changed; otherwise a new key : value pair is added.

###### For example :

>>> dict1 = {‘name’ : ‘Akash’, ‘age’ : 27}

>>> dict1[‘age’] = 30 # updating a value

>>> print dict

{‘age’ : 30, ‘name’: ‘Akash’} # Output

>>> dict1[‘address’] = ‘Alaska’ # adding a key : value

>>>print dict1

{‘age’: 30, ‘name’: ‘Akash’, ‘address’: ‘Alaska’} # Output

In the given example, we tried to reassign the value ‘30’ to the key ‘age’, Python interpreter first searches the key in the dictionary and then update it. Hence, the value of ‘age’ is updated to 30. However, in the next statement, it does not find the key ‘address’; hence, the key: value ‘address’ : ‘Alaska’ is added to the dictionary.

###### Strings are immutable :

* 1. String are immutable which means that we cannot change any element of a string.
  2. If we want to change an element of a string, we have to create a new string.

###### For example :

>>> var = ‘hello Python’

>>> var [0] = ‘p’

###### Output :

Type error : ‘str’ object does not support item assignment

Here, we try to change the 0th index of the string to a character *p*, but the python interpreter generates an error.

Now, the solution to this problem is to generator a new string rather than change the old string.

###### For example :

>>> var = ‘hello Python’

>>> new\_var = ‘p’ + var[1 :]

>>> print new\_var

pello Python # Output

In the given example, we cut the slice from the original string and concatenate it with the character we want to insert in the string. It does not have any effect on the original string.

###### Lists are mutable :

* 1. Lists are mutable means that the value of any element inside the list can be changed at any point of time.
  2. The elements of the list are accessible with their index value.
  3. The index always starts with 0 and ends with *n* – 1, if the list contains *n* elements.
  4. The syntax for accessing the elements of a list is the same as in the case of a string. We use square brackets around the variable and index number.

###### For example :

>>> list = [10, 20, 30, 40]

>>> list [1]

20 # Output

In the given example, we access the 2nd element of the list that has 1 as index number and the interpreter prints 20.

Now, if we want to change a value in the list given in the example :

###### For example :

>>> list [3] = 50

>>> print list

[10, 20, 30, 50] # Output

Note that the value of the 4th element is changed to 50

.

**Ques No 17:What do you mean by sets ? Explain the operations performed on sets ?**

In Python we also have one data type which is an unordered collection of data known as set.

1. A set does not contain any duplicate values or elements.

###### Operations performed on sets are :

* 1. **Union :** Union operation performed on two sets returns all the elements from both the sets. It is performed by using and operator.
  2. **Intersection :** Intersection operation performed on two sets returns all the element which are common or in both the sets. It is performed by using ‘|’ operator.
  3. **Difference :** Difference operation performed on two sets set1 and set2 returns the elements which are present on set1 but not in set2. It is performed by using ‘–’ operator.
  4. **Symmetric difference :** Symmetric difference operation performed on two sets returns the element which are present in either set1 or set2 but not in both. It is performed by using ^ operato

###### Ques No 18: Explain higher order respect to lambda expression. Write a Python code to count occurrences of an element in a list ?

1. Reduce(), filter(), map() are higher order functions used in Python.
2. Lambda definition does not include a “return” statement, it always contains an expression which is returned.
3. We can also put a lambda definition anywhere a function is expected, and we do not have to assign it to a variable at all.
4. Lambda functions can be used along with built-in higher order functions like filter(), map() and reduce().

###### Use of lambda with filter() :

1. The filter() function in Python takes in a function and a list as arguments.
2. This function helps to filter out all the elements of a sequence “sequence”, for which the function returns true.

**For example :** Python program that returns the odd numbers from an input list :

# Python code to illustrate filter() with lambda li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

final\_list = list(filter(lambdax: (x%2!=0), li)) print(final\_list)

###### Output :

[5, 7, 97, 77, 23, 73, 61]

###### Use of lambda() with reduce() :

1. The reduce() function in Python takes in a function and a list as argument.
2. The function is called with a lambda function and a list and a new reduced result is returned. This performs a repetitive operation over the pairs of the list.
3. This is a part of functools module.

###### For example :

# Python code to illustrate reduce() with lambda() to get sum of a list from functools import reduce

li = [5, 8, 10, 20, 50, 100]

sum = reduce((lambda x, y : x + y), li) print (sum)

###### Output :

193

###### Question No 19: What is Sieve of Eratosthenes ?

1. Sieve of Eratosthenes is a simple and ingenious ancient algorithm for finding all prime numbers up to any given limit.
2. It does so by iteratively marking as composite (*i*.*e*., not prime) the multiples of each prime, starting with the first prime number, 2.
3. The multiples of a given prime are generated as a sequence of numbers starting from that prime, with constant difference between them that is equal to that prime.
4. Following is the algorithm to find all the prime numbers less than or equal to a given integer *n* by Eratosthenes’ method :
   1. Create a list of consecutive integers from 2 to *n* : (2, 3, 4, …, *n*).
   2. Initially, let *p* equal 2, the first prime number.
   3. Starting from *p*2, count up in increments of *p* and mark each of these numbers greater than or equal to *p*2 itself in the list. These numbers will be *p*(*p*+1), *p*(*p*+2), *p*(*p*+3), etc.
   4. Find the first number greater than *p* in the list that is not marked. If there was no such number, stop. Otherwise, let *p* now equal this number (which is the next prime), and repeat from step c.

What are files ? How are they useful

1. A file in a computer is a location for storing some related data.
2. It has a specific name.
3. The files are used to store data permanently on to a non-volatile memory (such as hard disks).

As we know, the Random Access Memory (RAM) is a volatile memory type because the data in it is lost when we turn off the computer. Hence, we use files for storing of useful information or data for future reference

Describe the opening a file function in Python

1. Python has a built-in open () function to open files from the directory.
2. Two arguments that are mainly needed by the open () function are :
   1. **File name :** It contains a string type value containing the name of the file which we want to access.
   2. **Access\_mode :** The value of access\_mode specifies the mode in which we want to open the file, *i.e*., read, write, append etc.

###### Syntax :

file\_object = open(file\_name [, access\_mode])

###### For example :

>>>f = open (“test.txt”) #Opening file current directory

>>>f = open (“C:/Python27/README.txt”)

#Specifying full path #Output

>>>f

<open file ‘C:/Python27/README.txt’, mode ‘r’ at 0x02BC5128>

#Output

###### Ques No 20 :Explain the closing a file method in Python?

**.**

1. When the operations that are to be performed on an opened file are finished, we have to close the file in order to release the resources.
2. Python comes with a garbage collector responsible for cleaning up the unreferenced objects from the memory, we must not rely on it to close a file.
3. Proper closing of a file frees up the resources held with the file.
4. The closing of file is done with a built-in function close ().
5. **Syntax :** fileObject. close () **For example :**

# open a file

>>> f = open (“test. txt”, “wb”) # perform file operations

>>> f. close() # close the file

Discuss writing to a file operation

1. After opening a file, we have to perform some operations on the file. Here we will perform the write operation.
2. In order to write into a file, we have to open it with *w* mode or *a* mode, on any writing-enabling mode.
3. We should be careful when using the *w* mode because in this mode overwriting persists in case the file already exists.

###### For example :

# open the file with w mode

>>> f = open (“C :/Python27/test.txt”, “w”) # perform write operation

>>>f. write (‘writing to the file line 1/n’)

>>>f. write (‘writing to the file line 2/n’)

>>>f. write (‘writing to the file line 3/n’)

>>>f. write (‘writing to the file line 4’) # clos the file after writing

>>> f.close ()

The given example creates a file named test.txt if it does not exist, and overwrites into it if it exists. If we open the file, we will find the following content in it.

###### Output :

Writing to the file line 1

Writing to the file line 2 Writing to the file line 3 Writing to the file line 4

###### Ques No 21: Explain reading from a file operation with example?

1. In order to read from a file, we must open the file in the reading mode (*r* mode).
2. We can use read (size) method to read the data specified by size.
3. If no size is provided, it will end up reading to the end of the file.
4. The read() method enables us to read the strings from an opened file.

###### Syntax :

file object. read ([size])

###### For example :

# open the file

>>> f = open (“C :/Python27/test.txt”, “r”)

>>>f . read (7) # read from starting 7 bytes of data ‘writing’ # Output

>>>f. read (6) # read next 6 bytes of data ‘to the’ # Output

What do you mean by exceptions

While writing a program, we often end up making some errors. There are many types of error that can occur in a program

1. The error caused by writing an improper syntax is termed syntax error or parsing error; these are also called compile time errors.
2. Errors can also occur at runtime and these runtime errors are known as exceptions.
3. There are various types of runtime error in Python.
4. For example, when a file we try to open does not exist, we get a FileNotFoundError. When a division by zero happens, we get a ZeroDivisionError. When the module we are trying to import does not exist, we get an ImportError.
5. Python creates an exception object for every occurrence of these run- time errors.
6. The user must write a piece of code that can handle the error.
7. If it is not capable of handling the error, the program prints a trace back to that error along with the details of why the error has occurred.

###### For example :

Compile time error (syntax error)

>>> a = 3

>>> if (a < 4) # semicolon is not included SyntaxError : invalid syntax # Output **ZeroDivisionError :**

>>>5/0

###### Output :

Traceback (most recent call last) :

File “<pyshell#71>”, line 1, in <module> 5/0

ZeroDivisionError : Integer division or modulo by zero

###### Ques No 22: Give example of try….except?

**.**

>>>try:

... file = open(“C:/Python27/test.txt”,“w”)

... file write(“hello python”)

... exceptIOError :

... print “Error: cannot find file or read data

... else :

... print “content written successfully”

>>> file. close ( )

1. In the given example, we are trying to open a file test.txt with write access mode, and want to write to that file. We have added try and except blocks.
2. If the required file is not found or we do not have the permission to write to the file, an exception is raised.
3. The exception is handled by the except block and the following statement printed :

Error : cannot find file or read data

1. On the other hand, if the data is written to the file then the else block will be executed and it will print the following.

###### Output :

Content written successfully

Define the term modules

1. A module is a file containing Python definitions and statements. A module can define functions, classes and variables.
2. It allows us to logically organize our Python code.
3. The file name is the module name with the suffix .py appended.
4. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use.
5. Definitions from a module can be imported into other modules or into the main module.

###### For example :

Here is an example of a simple module named as support.py def print\_func( par ):

print “Hello : ”, par

return

What do you mean by objects

1. An object is an instance of a class that has some attributes and behaviour.
2. The object behaves according to the class of which it is an object.
3. Objects can be used to access the attributes of the class.
4. The syntax of creating an object in Python is similar to that for calling a function.

###### Syntax :

obj\_name = class\_name ( )

###### For example :

s1 = Student ()

In the given example, Python will create an object s1 of the class student.

###### Ques No 23 :Give an example of class ?

**.**

>>>class Student :

... ‘student details’

... def fill\_details(self, name, branch, year):

... self.name = name

... self.branch = branch

... self.year = year

... print(“A Student detail object is created”)

... def print details (self) :

... print(‘Name: ’, self.name)

... print(‘Branch: ’,self.branch)

... print(‘Year: ’,self.year)

In the given example, we have created a class Student that contains two methods: fill\_details and print\_details. The first method fill\_details takes four arguments: self, name, branch and year. The second method print\_details takes exactly one argument: self.

What is \_\_str\_\_ method in class

1. str\_\_method is the “informal” or nicely printable string representation of an object. This is for the end user.
2. It is called by str(object) and the built-in functions such as format() and print().
3. The return value of this method is a string object.

###### For example :

Class Account :

def \_\_str\_\_(self) :

return ‘Account of { } with starting amount : { }’ format (self.owner, self.amount)

Now we can query the object in various ways and always get a nice string representation :

>>> str(acc)

‘Account of bob with starting amount : 10’

###### Ques No 24 :Define the term inheritance ?

**Que 4.10.**

1. A class ‘*A*’ that can use the characteristics of another class ‘*B*’ is said to be a derived class, *i.e*., a class inherited from ‘*B*’. The process is called inheritance.
2. In OOP, it means that reusability of code.
3. It is the capability of a class to derive the properties of another class that has already been created.

**For example :** Vehicle is a class that is further divided into two subclasses, automobiles (driven by motors) and pulled vehicles (driven by men). Therefore, vehicle is the base class and automobiles and pulled vehicles are its subclasses. These subclasses inherit some of the properties of the base class vehicle.

**Answer**

###### Ques No 25:Give syntax of inheritance and explain with the help of

**example ?**

###### Syntax :

Class sub\_classname(Parent\_classname): ‘Optional Docstring’

Class\_suite

###### For example :

#Define a parent class Person

>>>class Person(object) :

‘returns a Person object with given name’ def get\_name (self ,name) :

self.name = name def get\_details (self) :

‘returns a string containing name of person’ return self.name

#Define a subclass Student

>>>class Student (Person) :

‘return a Student object, takes 2 arguments’ def fill\_details (self, name, branch) :

Person.get\_name(self,name) self.branch = branch

def get\_details(self):

‘returns student details’ print(“Name:”, self.name) print(“Branch: ”, self.branch)

#Define a subclass Teacher

>>>class Teacher(Person) :

‘returns a Teacher object, takes 1 arguments’ def fill\_details(self, name, branch) :

Person.get\_name(self,name) def get\_details(self) : print(“Name:”, self.name)

#Define one object for each class

>>>person 1 = Person ()

>>>student 1 = Student ()

>>>teacher 1 = Teacher () #Fill details in the objects

>>> person1.get\_name(‘John’)

>>> student1.fill\_details(‘Jinnie’, ‘CSE’)

>>> teacher1.fill\_details(‘Jack’)

#Print the details using parent class function

>>>print(personl.get\_details()) John # Output

>>>print(studentl.get\_details()) Name: Jinnie # Output Branch: CSE # Output

>>>print(teacherl.get\_details())

**Ques No 26: What do you mean by multiple inheritance ? Explain in**

**.**

**detail ?**

In multiple inheritance, a subclass is derived from more than one base class

The subclass inherits the properties of all the base classes

###### Syntax :

# Define your first parent class class A

...................... class\_suite ..................

# Define your second parent class class B

.......................class-suite......................

# Define the subclass inheriting both A and B class C(A, B)

.......................class-suite......................

###### For example :

>>> class A : # Defining class A def x(self):

print(“method of A”)

>>> class B : # Defining Class B def x(self):

print(“method of B”)

>>> class C(A, B) : # Defining class C pass

>>> y = c ()

>>> B.x(y)

###### Ques No 27: What do you mean by iterator ?

**.**

1. An iterator is an object that contains a countable number of values.
2. An iterator is an object that can be iterated upon, meaning that we can traverse through all the values.
3. Python iterator, implicitly implemented in constructs like for-loops, comprehensions, and python generators.
4. Python lists, tuples, dictionary and sets are all examples of in-built iterators.
5. These types are iterators because they implement following methods :
   1. **\_\_iter\_\_ :** This method is called on initialization of an iterator. This should return an object that has a next() method.
   2. **next() (or \_\_next\_\_) :** The iterator next method should return the next value for the iterable. When an iterator is used with a ‘for in’ loop, the for loop implicitly calls next() on the iterator object. This method should raise a StopIteration to signal the end of the iteration.

###### For example :

# An iterable user defined type class Test:

# Constructor

def \_\_init\_\_(self, limit): self.limit = limit

# Called when iteration is initialized def \_\_iter\_\_(self):

self.x = 10 return self

# To move to next element. def next(self):

# Store current value of x x = self.x

# Stop iteration if limit is reached if x > self.limit:

raise StopIteration

# Else increment and return old value self.x = x + 1;

return x

# Prints numbers from 10 to 15 for i in Test(15):

print(i)

# Prints nothing for i in Test(5):

print(i) **Output :** 10

11

12

13

14

###### Ques No 28: Define recursion. Also, give example ?

**.**

###### In Python, recursion occurs when a function is defined by itself.

1. When a function calls itself, directly or indirectly, then it is called a recursive function and this phenomenon is known as recursion.
2. Recursion is the property how we write a function. A function which performs the same task can be written either in a recursive form or in an iterative form.
3. Recursion is the process of repeating something self-similar way.

###### For example :

def fact (n) :

if n == 0:

return 1 else :

return n \* fact(n – 1) print(fact(0)) print(fact(5))

###### Output :

1

120

###### Ques No 29: Explain Fibonacci series using Python ?

**.**

1. Fibonacci series is a series of numbers formed by the addition of the preceding two numbers in the series.
2. It is simply the series of numbers which starts from 0 and 1 and then continued by the addition of the preceding two numbers.
3. Example of Fibonacci series: 0, 1, 1, 2, 3, 5.

###### Python code for recursive Fibonacci :

def FibRecursion(n) : if n <= 1 :

return n else :

return(FibRecursion(n – 1) + FibRecursion(n – 2)) nterms = int(input(“Enter the term : ”)) # take input from the user if nterms < = 0: # check if the number is valid

print (“Please enter a positive integer”) else :

print (“Fibonacci sequence :”) for i in range (nterms) :

print(FibRecursion(i)) **Output :** Enter the term : 5 Fibonacci sequence :

0 1 1 2 3

1. In the given Python program, we use recursion to generate the Fibonacci sequence.
2. The function FibRecursion is called recursively until we get the output.
3. In the function, we first check if the number *n* is zero or one. If yes, it returns the value of *n*. If not, we recursively call FibRecursion with the values *n* – 1 and *n* – 2.

###### Ques No 30 :What is simple (linear) search ? Explain with the help of

**.**

**example?**

1. Linear search is a method for finding a particular value in a list.
2. Linear search is good to use when we need to find the first occurrence of an item in an unsorted collection.
3. Linear (Simple) search is one of the simplest searching algorithms, and the easiest to understand.
4. It starts searching the value from the beginning of the list and continues till the end of the list until the value is found.

###### Code :

def seach(arr, n, x) : i = 0

for i in range(i, n) : if (arr[i] == x) :

return – i return – 1

###### For example :

arr = [3, 10, 30, 45]

x = 10

n = len(arr)

print(x, “is present at index”, search(arr, n, x))

**Output :** 10 is present at index 1

###### Ques No 31 : Discuss binary search in Python ?

**.**

1. Binary search follows a divide and conquer approach. It is faster than linear search but requires that the array be sorted before the algorithm is executed.
2. Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of item is returned.
3. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item.
4. Otherwise, the item is searched for in the sub-array to the right of the middle item.
5. This process continues on the sub-array as well until the size of the sub-array reduces to zero.

###### Code :

def binarysearch(arr, 1, r, x) : while 1 <= r :

mid = 1 + (r – 1)/2;

# Check if x is present at mid if arr[mid] == x:

return mid

# If x is greater, ignore left half elif arr[mid] < x :

l = mid + 1

# If x is smaller, ignore right half else :

r = mid – 1

# If we reach here, then the element was not present return – 1

# Test array

arr = [2, 3, 4, 10, 40]

x = 10

# Function call

result = binarySearch(arr, 0, len(arr) – 1, x) if result ! = – 1:

print “Element is present at index % d” % result else :

print “Element is not present in array”

###### Output :

Element is present at index 3

What do you mean by selection sort ? Discuss in detail

1. The selection sort algorithm sorts an array by repeatedly finding the smallest element (considering ascending order) from unsorted list and swapping it with the first element of the list.
2. The algorithm maintains two sub-arrays in a given array:
   1. The sub-array which is already sorted.
   2. Remaining sub-array which is unsorted.
3. In every iteration of selection sort, the smallest element from the unsorted sub-array is picked and moved to the sorted sub-array.

###### Code :

def slectionSort(nlist) :

for fillslot in range(len(nlist) – 1, 0, – 1) : maxpos = 0

for location in range(1, fillslot + 1) : if nlist[location]>nlist[maxpos] :

maxpos = location temp = nlist[fillslot] nlist[fillslot] = nlist[maxpos] nlist[maxpos] = temp

nlist = [14, 46, 43, 27, 57, 41, 45, 21, 70]

selectionSort(nlist) print(nlist) **Output :**

[14, 21, 27, 41, 43, 45, 46, 57, 70]

###### Time complexity :

* 1. **Best case :** O(*n*2)
  2. **Worst case :** O(*n*2)
  3. **Average case :** O(*n*2)

###### Ques No 32: Explain merge list ?

**.**

1. Merging is defined as the process of creating a sorted list/array of data items from two other sorted array/list of data items.
2. Merge list means to merge two sorted list into one list.

###### Code for merging two lists and sort it :

a=[ ]

c=[ ]

n1=int(input(“Enter number of elements:”)) for i in range(1, n1+1):

b=int(input(“Enter element:”)) a.append(b)

n2=int(input(“Enter number of elements:”)) for i in range(1, n2+1):

d=int(input(“Enter element:”)) c.append(d)

new=a+c new.sort( )

print(“Sorted list is:”, new)

**Ques No 33. Explain merge sort with the help of example ?**

**.**

1. Merge sort is a divide and conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves.
2. The merge() function is used for merging two halves.
3. The merge(arr, *l*, *m*, *r*) is key process that assumes that arr[*l*..*m*] and arr[*m* + 1 ..*r*] are sorted and merges the two sorted sub-arrays into one.

###### Code :

def mergeSort(arr) if len(arr) >1:

mid = len(arr)//2 #Finding the mid of the array L = arr[:mid] # Dividing the array elements

R = arr[mid:] # into 2 halves mergeSort(L) # Sorting the first half mergeSort(R) # Sorting the second half i = j = k = 0

# Copy data to temp arrays L[] and R[] while i < len(L) and j < len(R):

if L[i] < R[j]:

arr[k] = L[i] i + = 1

else :

arr[k] = R[j]

j + = 1 k + = 1

# Checking if any element was left while i < len(L):

arr[k] = L[i] i + = 1

k + = 1

while j < len(R): arr[k] = R[j] j + = 1

k + = 1

# Code to print the list def printList(arr):

for i in range(len(arr)): print(arr[i],end=“ ”)

print()

# driver code to test the above code if \_\_name\_\_ == ‘\_\_main\_\_ ’ :

arr = [12, 11, 13, 5, 6, 7 ]

print (“Given array is”, end = “\n”) printList(arr)

mergeSort(arr)

print(“Sorted array is: ”, end = “\n”) printList(arr)

###### Output :

Given array is 12, 11, 13, 5, 6, 7

Sorted array is 5, 6, 7, 11, 12, 13

1. **Time complexity :** Recurrence relation of merge sort is given by

*T*(*n*) = 2*T*(*n*/2) + *Cn*

= 2(2*T*(*n*/4) + *Cn*/2) + *Cn* = 22*T*(*n*/4) + 2*Cn*

= 22(2*T*(*n*/8) + *Cn*/4) + *Cn* = 23*T*(*n*/8) + 3*Cn*

= ... // keep going for *k* steps

= 2*kT*(*n*/2*k*) + *k*\**Cn*

Assume *n* = 2*k* for some *k*. *k* = log2 *n*

Then, *T*(*n*) = *n*\**T*(1) + *Cn*\*log2*n*

* 1. Time complexity of Merge sort is O(*n* log *n*) in all three cases (worst, average and best) as merge sort always divides the array into two halves and take linear time to merge two halves.

What are the features of Python

**Features of Python :**

* + 1. The code written in Python is automatically compiled to byte code

and executed.

* + 1. Python can be used as a scripting language, as a language for implementing web applications, etc.
    2. Python supports many features such as nested code blocks, functions, classes, modules and packages.
    3. Python makes use of an object oriented programming approach.

It has many built-in data types such as strings, lists, tuples, dictionaries, etc

###### Ques NO 34.What is range () function ?

The range () function is a built-in function in Python that helps us to iterate over a sequence of numbers. It produces an iterator that follows arithmetic progression.

###### Ques No 35.Give an example of range () function ?

>>> range (8)

[0, 1, 2, 3, 4, 5, 6, 7]

range (8) provides a sequence of number 0-7. That is to say range

(*n*) generates a sequence of number that starts with 0 and end with (*n* – 1).

###### Ques No 36.What are control statements ?

**.** A control statement is a statement that determines the control flow of a set of instructions. There are three fundamental forms of control that programming languages provide: sequential control, selection control, and iterative control.

###### Ques No 37.What is short-circuit evaluation ?

**Ans.** In short-circuit (lazy) evaluation, the second operand of Boolean operators AND and OR is not evaluated if the value of the Boolean expression can be determined from the first operand alone.

###### Ques No 38.Define the terms : header, suite and clause?

A header in Python starts with a keyword and ends with a colon. The group of statements following a header is called a suite. A header and its associated suite are together referred to as a clause.

###### Ques No 39. What do you mean by iterative control ?

An iterative control statement is a control statement providing the repeated execution of a set of instructions. An iterative control structure is a set of instructions and the iterative control statement(s) controlling their execution.

###### Ques No 40 :What do you mean by definite loop ?

A definite loop is a program loop in which the number of times the loop will iterate can be determined before the loop is executed.

###### Ques No41 :What do you mean by indefinite loop ?

An indefinite loop is a program loop in which the number of times that the loop will iterate cannot be determined before the loop is executed.

###### Ques No 42.Is indentation optional in Python ?

**.** No indentation in Python is compulsory and is part of its syntax. Indentation is a way of defining the scope and extent of the block of codes. Indentation provides better readability to the code.

###### Ques No 43.What is raw\_input ( ) function ?

Raw\_input ( ) takes the input from the user but it does not interpret the input and also it returns the input of the user without doing any changes.

###### Ques No 44.Differentiate fruitful functions and void functions.

**.** The main difference between void and fruitful function in python is :

* + 1. Void does not return any value
    2. Fruitful function returns some value

###### Question No 45. What are escape characters ?

The backslash character (/) is used to escape characters. It converts difficult-to-type characters into a string. For example, we need the escaping character concept when we want to print a string with double quotes or single quotes. When single or double quotes are used with the string, Python normally neglects them and prints only the string.

###### Question No 46.What do you mean by tuple assignment ?

**.** Tuple assignment allows the assignment of values to a tuple of variables on the left side of assignment from the tuple of values on the right side of the assignment.

###### Ques No 47:What do you understand by traversing a list ?

Traversing of the list refers to accessing all the elements or items of the list. Traversing can be done using any conditional statement of Python, but it is preferable to use for loop.

###### Ques No 48:What are the different methods used in deleting elements from dictionary ?

**Methods used in deleting elements from dictionary are :**

* + - * 1. **pop( ) :** pop() method removes that item from the dictionary for which the key is provided. It also returns the value of the item.
        2. **popitem( ) :** popitem() method is used to remove or delete and return an arbitrary item from the dictionary.
        3. **clear( ) :** clear() method removes all the items or elements from a dictionary at the same time.

###### Ques No 49:What are the two properties of key in the dictionary ?

**Properties of key :**

* + - * 1. One key in a dictionary cannot have two values, *i.e.*, duplicate keys are not allowed in the dictionary; they must be unique.
        2. Keys are immutable, *i.e*., we can use string, integers or tuples for dictionary keys, but cannot use something like [‘key’].

###### Ques No 50:Why we use functions ?

* + - * 1. Break up complex problem into small sub-programs.
        2. Solve each of the sub-problems separately as a function, and combine them together in another function.
        3. Hide the details and shows the functionality.

###### Ques No 51:What are mathematical functions ? How are they used in Python ?

Python provides us a math module containing most of the familiar and important mathematical functions. A module is a file that contain some predefine Python codes. A module can define functions, classes and variables. It is a collection of related functions

grouped together.

Before using a module in Python, we have to import it For example, to import the math module, we use :

>>> import math

What are user-defined functions

###### Ques No52.What are directories ?

**.** If there is a large number of file, then related files are placed in different directories. Directory can be said to be a collection of files and sub directories. The module os in Python enables us to use various methods to work with directories.

###### Ques No 53.What are the basic methods performed on directories ?

Following are the four basic methods that are performed on directories :

1. mkdir () method (Creating a directory)
2. chdir() method (Changing the current directory)
3. getcwd () method (Displaying the current directory)
4. rmdir () method (Deleting the directory).

###### Ques N0 54.What are the advantages of recursion ?

* + 1. It requires few variables.
    2. The programs are easy to implement if the problem has a recursive definition.

###### Ques No 55.Give some disadvantages of recursion?

Debugging is difficult.

It is not easy to write the program in a recursive form.

It can be inefficient as it requires more time and space.

###### Ques No 56.What are the applications of Tower of Hanoi problem ?

The Tower of Hanoi is frequently used in psychological research on problem solving.

There also exists a variant of this task called Tower of London for neuropsychological diagnosis and treatment of executive functions.

The Tower of Hanoi is also used as a Backup rotation scheme when performing computer data Backups where multiple tapes/media are involved.

The Tower of Hanoi is also used as a test by neuropsychologists trying to evaluate frontal lobe deficits.

###### Ques No 57.What are the advantages and drawbacks of simple search ?

**Advantages :**

It is a very simple search and easy to program.

###### In the best-case scenario, the item we are searching for may be at the start of the list in which case we get it on the very first try Drawbacks :

Its drawback is that if our list is large, it may take time to go through the list.

In the worst-case scenario, the item we are searching for may not be in the list, or it may be at the opposite end of the list.

###### Ques No 58.Write algorithm of simple search ?

if start > end:

return False if a[start]==key:

return True

return search(a, start + 1, end, key)

###### Ques NO 59:Give the algorithm for binary search?

**.** if start > end :

return False

mid = (start + end) //2 if a [mid] = = key:

return True if (a[mid] > key):

return binsearch(a, start, mid – 1, key) else :

return binsearch(a, mid + 1, end, key)

###### Ques No 60:Define floor division with example?

Floor division returns the quotient in which the digits after the decimal point are removed. But if one of the operands (dividend and divisor) is negative, then the result is floored, i.e., rounded away from zero (means, towards the negative of infinity). It is denoted by “//”.

###### For example :

5.0 // 2

2.

###### Ques No 61.Discuss interaction with Python program with example?

**.**

1. The Python program that we have installed will by default act as an interpreter.
2. An interpreter takes text commands and runs them as we enter text.
3. After Python opens, it will show some contextual information like this : Python 3.5.0 (default, dec 20 2019, 11 : 28 : 25)

[GCC 5.2.0] on Linux

Type “help”, “copyright”, “credits” or “license” for more information

>>>

1. The prompt >>> defines that we are now in an interactive Python interpreter session, also called the Python shell.
2. Now enter some code for Python to run such as print(“Hello World!”) and press the Enter key.
3. The interpreter’s response should appear on the next line like this :

>>> print (“Hello World!”) Hello World!

1. After showing the results, Python will bring you back to the interactive prompt, where we could enter another command or code.

Python program communicates its result to user using print statement

**Ques No 62.What do you mean by type conversion ?**

The process of converting one data type into another data type is known as type conversion.

1. There are mainly two types of type conversion methods in Python :

###### Implicit type conversion :

* + 1. When the data type conversion takes place during compilation or during the run time, then it called an implicit data type conversion.

###### Python handles the implicit data type conversion, so we do not have to explicitly convert the data type into another data type For example :

a = 5

b = 5.5

sum = a + b print (sum)

print (type (sum)) # type() is used to display the datatype of a variable

###### Output :

10.5

<class ‘float’>

1. In the given example, we have taken two variables of integer and float data types and added them.
2. Further, we have declared another variable named ‘sum’ and stored the result of the addition in it.
3. When we checked the data type of the sum variable, we can see that the data type of the sum variable has been automatically converted into the float data type by the Python compiler. This is called implicit type conversion.

###### Explicit type conversion:

i Explicit type conversion is also known as type casting.

1. Explicit type conversion takes place when the programmer clearly and explicitly defines the variables in the program.

###### For example :

# adding string and integer data types using explicit type conversion

a = 100

b = “200”

result1 = a + b b = int(b) result2 = a + b print (result2) **Output :**

Traceback (most recent call last):

File “”, line 1, in

TypeError : unsupported operand type (s) for +: ‘int’ and ‘str’ 300

1. In the given example, the variable *a* is of the number data type and variable *b* is of the string data type.
2. When we try to add these two integers and store the value in a variable named result1, a TypeError occurs. So, in order to perform this operation, we have to use explicit type casting.

We have converted the variable *b* into integer type and then added variable *a* and *b*. The sum is stored in the variable named result2, and when printed it displays 300 as output

###### Question No 63. Discuss float representation in Python?

1. Floating point representations vary from machine to machine.
2. The float type in Python represents the floating-point number.
3. Float is used to represent real numbers and is written with a decimal point dividing the integer and fractional parts.
4. For example: 97.98, 32.3 + e18, – 32.54e100 all are floating point numbers.
5. Python float values are represented as 64-bit double-precision values.
6. The maximum value any floating-point number can be is approx 1.8 × 10308.
7. Any number greater than this will be indicated by the string inf in Python.
8. Floating-point numbers are represented in computer hardware as base 2 (binary) fractions.
9. For example, the decimal fraction 0.125 has value 1/10 + 2/100 + 5/1000, and in the same way the binary fraction 0.001 has value 0/2 + 0/4 + 1/8.

**For example :** # Python code to demonstrate float values. Print(1.7e308)

# greater than 1.8 \* 10^308

# will print ‘inf’ print(1.82e308) **Output :** 1.7e+308

inf

###### Ques No 64. Define the term operator?

**.**

1. An operator is a symbol that represents an operation that may be performed on one or more operands.
2. Operators are constructs used to modify the values of operands.
3. Operators that take one operand are called unary operators.
4. Operators that take two operands are called binary operators.
5. Based on functionality operators are categories into following seven types :
   1. Arithmetic operators.
   2. Assignment operators.
   3. Bitwise operators.
   4. Comparison operators.
   5. Identity operators.
   6. Logical operators.

Membership operators

**Ques No 65. How memory is managed in Python? Explain PEP 8. Write a Python program to print even length words in a string Memory management ?**

1. Memory management in Python involves a private heap containing all Python objects and data structures.
2. The management of this private heap is ensured internally by the Python memory manager.
3. The Python memory manager has different components which deal with various dynamic storage management aspects, like sharing, segmentation, preallocation or caching.

At the lowest level, a raw memory allocator ensures that there is enough room in the private heap for storing all Python-related data by interacting with the memory manager of the operating system

1. On top of the raw memory allocator, several object-specific allocators operate on the same heap and implement distinct memory management policies adapted to the peculiarities of every object type.
2. For example, integer objects are managed differently within the heap than strings, tuples or dictionaries because integers imply different storage requirements and speed/space tradeoffs.
3. Python memory manager thus delegates some of the work to the object-specific allocators, but ensures that the latter operate within the bounds of the private heap.

###### PEP 8 :

1. A PEP is a design document providing information to the Python community, or describing a new feature for Python or its processes or environment.
2. The PEP should provide a concise technical specification of the feature.
3. PEP is actually an acronym that stands for Python Enhancement Proposal.
4. PEP 8 is Python’s style guide. It is a set of rules for how to format the Python code to maximize its readability.
5. A PEP is a design document providing information to the Python community, or describing a new feature for Python or its processes or environment.

###### Ques No 66. Program to print even length words in a string ?

def printWords(s) :

# split the string s = s.split(‘ ’)

# iterate in words of string for word in s:

# if length is even if len(word)%2==0:

print(word) # Driver Code

s = “i am muskan” printWords(s) **Output :**

am muskan

**Ques No 67.Write a python program to print the numbers for a user provided range?**

# Pythons program to print the prime numbers for a user provided range

# input range is provided from the user

>>> low = int(input(“Enter lower range : ”))

>>> up = int (input(“Enter upper range : ”))

>>> for n in range (low, up + 1) : if n > 1 :

for i in range (2, n) : if (n % i) == 0 : break

else :

print (n)

###### Output :

Enter lower range : 100 Enter upper range : 103

107

109

113

127

131

137

139

149

151

157

163

167

173

**Question No 68:Discuss the concatenation and repeat operation in Python with example**

**Concatenation ?**

1. Concatenation means joining two operands by linking them end-to-end.
2. In list concatenation, + operator concatenate two lists with each other and produce a third list.
3. For example, to concatenate two lists :

>>> x = [1, 2, 3]

>>> y = [4, 5, 6]

>>> z = x + y # concatenate two lists

>>> print z

[1, 2, 3, 4, 5, 6]

###### Repeat / replicate :

1. Lists can be replicated or repeated or repeatedly concatenated with the asterisk (\*) operator.

2, For example,

>>> aList = [1, 2, 3]

>>> print aList\*3 # list aList repeats three times [1, 2, 3, 1, 2, 3, 1, 2, 3]

Here, the list aList multiplied by 3 and printed three times

###### Ques No 69.What is len( ) function ?

**.**

1. len ( ) is a built-in function in Python. When used with a string, len returns the length or the number of character in the string.
2. Blank symbol and special characters are considered in the length of the string.

###### For example :

>>> var = “Hello Python!”

>>> len(var)

13 # Output

Here, we took a string ‘Hello Python!’ and used the len function with it. The len function returned the value 13 because not only characters, but also the blank space and exclamation mark in our string will also be counted as elements

###### Ques No 70.Define tuples. How are tuples created in PythonTuples are the sequence or series values of different types separated by commas (,)?

Values in tuples can also be accessed by their index values, which are integers starting from 0.

###### For example :

The names of the months in a year can be defined in a tuple :

>>> months = (‘January’, ‘February’, ‘March’, ‘April’, ‘May’, ‘June’, ‘July’, ‘August’, ‘September’, ‘October’, ‘November’, ‘December’)

###### Creating tuples in Python :

To create a tuple, all the items or elements are placed inside parentheses separated by commas and assigned to a variable.

Tuples can have any number of different data items (that is, integer, float, string, list, etc.).

###### For examples :

**A tuple with integer data items :**

>>> tuple = (4, 2, 9, 1)

>>> print tuple

(4, 2, 9, 1) # Output

###### A tuple with items of different data types :

>>>tuple\_mix = (2, 30, “Python”, 5.8, “Program”)

>>>print tuple\_mix

(2, 30, ‘Python’, 5.8, ‘Program’) # Output

###### Nested tuple :

>>>nested\_tuple = (“Python”, [1, 4, 2], [“john”, 3.9])

>>> print nested\_tuple

(‘Python’, [1, 4, 2], [‘john’, 3.9]) # Output

###### Tuple can also be created without parenthesis :

>>>tuple = 4.9, 6, ‘house’

>>>print tuple

(4.9, 6, ‘house’) # Output

###### Ques No 71: Define the term modules?

**.**

1. A module is a file containing Python definitions and statements. A module can define functions, classes and variables.
2. It allows us to logically organize our Python code.
3. The file name is the module name with the suffix .py appended.
4. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use.
5. Definitions from a module can be imported into other modules or into the main module.

###### For example :

Here is an example of a simple module named as support.py def print\_func( par ):

print “Hello : ”, par

return

###### Ques No 72. Define class?

1. A class can be defined as a blue print or a previously defined structure from which objects are made.
2. Classes are defined by the user; the class provides the basic structure for an object.
3. It consists of data members and method members that are used by the instances of the class.
4. In Python, a class is defined by a keyword Class.
5. Syntax : class class\_name;

**For example :** Fruit is a class, and apple, mango and banana are its objects. Attribute of these objects are color, taste, etc.

###### Ques No 73.Define operator associativity with its type?

* + 1. Associativity decides the order in which the operators with same precedence are executed.
    2. There are two types of associativity :
       1. **Left to right :** In left to right associativity, the operator of same precedence are executed from the left side first.
       2. **Right to left :** In right to left associativity, the operator of same precedence are executed from the right side first.

###### Question No. 74. What are escape characters ?

**Ans.** The backslash character (/) is used to escape characters. It converts difficult-to-type characters into a string. For example, we need the escaping character concept when we want to print a string with double quotes or single quotes. When single or double quotes are used with the string, Python normally neglects them and prints only the string

###### Question No 75. Write a program to print the calendar for the month of March 1991.

**Ans.** >>> import calendar

>>> c = calender.month(1991, 3)

>>> print c March 1991

|  |  |  |  |
| --- | --- | --- | --- |
| Mo Tu We | Th | Fr Sa | Su |
|  |  | 1 2 | 3 |
| 4 5 6 | 7 | 8 9 | 10 |
| 11 12 13 | 14 | 15 16 | 17 |
| 18 19 20 | 21 | 22 23 | 24 |
| 25 26 27 | 28 | 29 30 | 31 |

###### Ques No 76.Write a function which accepts two numbers and returns their sum.

**Ans.** >>> def sum (arg1, arg2) :

sum = arg1 + arg2

return sum

# Now calling the function here

>>> a = 4

>>> b = 3

>>> total = sum(a, b) # calling the sum function

>>> print(total)

7 # Output

###### Ques No 77.What are the types of arguments used for calling a function ?

Four types of arguments used for calling a function :

1. Required argument
2. Keyword argument
3. Default argument
4. Variable length argument

###### Question No 78.Give one-one example for zip, max and min methods.

>>> tuple1 = (‘a’, ‘b’, ‘c’)

>>> tuple1 = (1, 2, 3)

>>> max (tuple 2)

|  |  |  |
| --- | --- | --- |
| 3 | # | Output |
| >>>min(tuple 1) |  |  |
| ‘a’ | # | Output |
| >>>zip(tuple 1, tuple 2) |  |  |
| {(‘a’, 1), (‘b’, 2), (‘c’, 3)} | # | Output |

###### Ques No 79. What are the various file positions methods

###### In Python, the tell() method tells us about the current position of the pointer. The current position tells us where reading will starts from at present.

We can also change the position of the pointer with the help of the seek() method. We pass the number of bytes to be moved by the pointer as arguments to the seek() method.

###### Ques No 80.What are directories ?

**.** If there is a large number of file, then related files are placed in different directories. Directory can be said to be a collection of files and sub directories. The module os in Python enables us to use various methods to work with directories.

###### Ques No 81.What are the basic methods performed on directories ?

Following are the four basic methods that are performed on directories :

1. mkdir () method (Creating a directory)
2. chdir() method (Changing the current directory)
3. getcwd () method (Displaying the current directory)
4. rmdir () method (Deleting the directory).

###### Ques No 82.What are user-defined exceptions ? Give one example.

**.** Python allows users to define their own exceptions by creating a new class. Exception needs to be derived, directly or indirectly from exception class.

###### For example :

>>> class error(Exception) pass

###### Ques No 83. Write some built-in exception in Python.

1. AssertionError
2. FloatingPointError
3. SystemError
4. RunTimeError
5. ZeroDivisionError

**Ques No 84.Explain all the conditional statement in Python using small code example**

**Different types of conditional statement are :**

###### If statement :

* 1. An if statement consists of a Boolean expression followed by one or more statements.
  2. With an if clause, a condition is provided; if the condition is true then the block of statement written in the if clause will be executed, otherwise not.

###### Syntax :

If (Boolean expression) : Block of code #Set of statements to execute if

the condition is true

###### For example :

var = 100

if ( var == 100 ) : print “value of expression is 100” print “Good bye !”

###### Output :

value of expression is 100 Good bye!

###### If else statement :

* 1. An if statement can be followed by an optional else statement, which executes when the Boolean expression is False.
  2. The else condition is used when we have to judge one statement on the basis of other.

###### Syntax :

If (Boolean expression): Block of code #Set of statements to

execute if

condition is true

else : Block of code #Set of statements to execute if condition

is false

###### For example :

num = 5

if (num > 10) :

print (“Number is greater than 10”) else :

print (“Number is less than 10”)

print (“This statement will always be executed”)

###### Output :

Number is less than 10.

###### Nested-if statement :

Nested-if statements are nested inside other if statements. That is, a nested-if statement is the body of another if statement

* 1. We use nested if statements when we need to check secondary conditions only if the fist condition executes as true.

###### Syntax :

if test expression 1 :

# executes when condition 1 is true body of if statement

if test expression 2 :

# executes when condition 2 is true Body of nested-if

else :

body of nested-if :

else :

body of if-else statement

###### For example :

a = 20

if (a == 20) :

# First if statement if (a < 25) :

print (“a is smaller than 25”) else :

print (“a is greater than 25”) else :

print (“a is not equal to 20”)

###### Output :

a is smaller than 25

###### Elif statement :

* 1. Elif stands for else if in Python.
  2. We use elif statements when we need to check multiple conditions only if the given if condition executes as false.

###### For example :

a = 50

if (a == 29) :

print (“value of variable a is 20”) elif (a == 30) :

print (“value of variable a is 30”) elif (a == 40) :

print (“value of variable a is 40”) else :

print (“value of variable a is greater than 40”)

###### Output :

value of variable a is greater than 40

###### Ques No 85.What is the difference between Python Arrays and lists ?

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Arrays** | **Lists** |
| 1. | Arrays can only store homogeneous data (data of the same type). | Lists can store heterogeneous and arbitrary data. |
| 2. | Arrays use less memory to store data. | Lists require more memory to store data. |
| 3. | The length of an array is  pre-fixed while creating it, so more elements cannot be added. | The length of a list is not fixed, so more elements can be added. |

###### Ques No 86. What is the difference between list and tuples in Python ?



|  |  |  |
| --- | --- | --- |
| **S. No.** | **List** | **Tuples** |
| 1. | Lists are mutable, *i.e*., they can be edited. | Tuples are immutable (they are lists that cannot be edited). |
| 2. | Lists are usually slower than tuples. | Tuples are faster than lists. |
| **3.** | **Syntax :**  list\_ 1 = [10, ‘Quantum’, 20] | **Syntax :**  tup\_ 1 = (10, ‘Quantum’, 20) |

###### Ques NO 87.Explain the file object attributes in detail.

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| file.closed | It will return true if the file is closed ; it will otherwise return false. |
| file.mode | It will return the access mode with which the file is opened. |
| file.name | It will return name of the file |
| file.softspace | It will return false if space explicitly required with print; otherwise it will return true. |

###### Ques No 88.What is the difference between list and tuples in Python ?

|  |  |  |
| --- | --- | --- |
| **S. No.** | **List** | **Tuples** |
| 1. | Lists are mutable, *i.e*., they can be edited. | Tuples are immutable (they are lists that cannot be edited). |
| 2. | Lists are usually slower than tuples. | Tuples are faster than lists. |
| **3.** | **Syntax :**  list\_ 1 = [10, ‘Quantum’, 20] | **Syntax :**  tup\_ 1 = (10, ‘Quantum’, 20) |

###### Ques No 89.Explain various arithmetic methods in detail.

|  |  |
| --- | --- |
| **Method :** | **Description** |
| \_\_add\_\_(self, other) | To get called on add operation using + operator |
| \_\_sub\_\_(self, other) | To get called on subtraction operation using – operator. |
| \_\_mut\_\_(self, other) | To get called on multiplication operation using \* operator. |
| \_\_floordiv\_\_(self, other) | To get called on floor division operation using // operator. |
| \_\_div\_\_(self, other) | To get called on division operation using / operator. |

###### Ques No 90. Explain Sieve of Eratosthenes with example.

1. Let us take an example when *n* = 50. So, we need to print all print numbers smaller than or equal to 50.
2. We create a list of all numbers from 2 to 50.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |

1. According to the algorithm we will mark all the numbers which are divisible by 2 and are greater than or equal to the square of it.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 | 3 | 4\* | 5 | 6\* | 7 | 8\* | 9 | 10\* |
| 11 | 12\* | 13 | 14\* | 15 | 16\* | 17 | 18\* | 19 | 20\* |
| 21 | 22\* | 23 | 24\* | 25 | 26\* | 27 | 28\* | 29 | 30\* |
| 31 | 32\* | 33 | 34\* | 35 | 36\* | 37 | 38\* | 39 | 40\* |
| 41 | 42\* | 43 | 44\* | 45 | 46\* | 47 | 48\* | 49 | 50\* |

1. Now we move to our next unmarked number 3 and mark all the numbers which are multiples of 3 and are greater than or equal to the square of it.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 | 3 | 4\* | 5 | 6\* | 7 | 8\* | 9\* | 10\* |
| 11 | 12\* | 13 | 14\* | 15\* | 16\* | 17 | 18\* | 19 | 20\* |
| 21\* | 22\* | 23 | 24\* | 25 | 26\* | 27\* | 28\* | 29 | 30\* |
| 31 | 32\* | 33\* | 34\* | 35 | 36\* | 37 | 38\* | 39\* | 40\* |
| 41 | 42\* | 43 | 44\* | 45\* | 46\* | 47 | 48\* | 49 | 50\* |

1. We move to our next unmarked number 5 and mark all multiples of 5 and are greater than or equal to the square of it.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 | 3 | 4\* | 5 | 6\* | 7 | 8\* | 9\* | 10\* |
| 11 | 12\* | 13 | 14\* | 15\* | 16\* | 17\* | 18\* | 19 | 20\* |
| 21\* | 22\* | 23 | 24\* | 25\* | 26\* | 27\* | 28\* | 29 | 30\* |
| 31 | 32\* | 33\* | 34\* | 35\* | 36\* | 37\* | 38\* | 39\* | 40\* |
| 41 | 42\* | 43 | 44\* | 45\* | 46\* | 47\* | 48\* | 49 | 50\* |

We continue this process and our final table will :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 | 3 | 4\* | 5 | 6\* | 7 | 8\* | 9\* | 10\* |
| 11 | 12\* | 13 | 14\* | 15\* | 16\* | 17 | 18\* | 19 | 20\* |
| 21\* | 22\* | 23 | 24\* | 25\* | 26\* | 27\* | 28\* | 29 | 30\* |
| 31 | 32\* | 33\* | 34\* | 35\* | 36\* | 37 | 38\* | 39\* | 40\* |
| 41 | 42\* | 43 | 44\* | 45\* | 46\* | 47 | 48\* | 49\* | 50\* |

So the prime numbers are the unmarked ones : 2, 3, 5, 7, 11, 13, 17, 19, 23, 29,

31, 37, 41, 43, 47

Write a function in find the HCF of some given numbers

>>>def hcf (a, b) : if a > b :

small = b else :

small = a

for i in range (1, small + 1) :

if (a % i == 0) and (b % i == 0)) : hcf = i

return hcf

>>> hcf (20, 40)

20 #Output

>>>hcf (529, 456)

1 #Output

###### Ques No91.Write a function to find the ASCII value of the character ?

>>>def ascii\_val\_of (a) :

print (“The ASCII value of ‘“ + a + ”’ is”, ord (a))

>>>ascii\_val\_of (‘A’)

(“The ASCII value of ‘A’ is”, 65) #Output

>>>ascii\_val\_of (‘ ’) #Finding ASCII value of space (“The ASCII value of ‘ ’ is”, 32)

**Ques No 92 Explain Lambda expression ?**

1. Lambda expressions is used to create the anonymous function.
2. The anonymous functions are the functions created using a lambda keyword.
3. They are not defined by using def keyword. For this reason, they are called anonymous functions.
4. We can pass any number of arguments to a lambda form functions, but still they return only one value in the form of expression.
5. An anonymous function cannot directly call print command as the lambda needs an expression.
6. It cannot access the parameters that are not defined in its own namespace.
7. An anonymous function is a single line statement function.

###### Syntax :

lambda [arg1 [,arg2, , argn]] : expression

The syntax of the lambda function is a single statement

###### For example :

# function definition here

>>>mult = lambda val1, val2 : val1\*val2 ; # function call here

>>> print “value :”, mult(20,40) Value : 800 # Output

In the given example, the lambda function is defined with two arguments val1 and val2. The expression val1\*va12 does the multiplication of the two values. Now, in function call, we can directly call the mult function with two valid values as arguments and produce the output as shown in given example.

###### Ques No 93.Write some built in dictionary methods used in Python with example.

**.**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Function** | **Description** |
| 1. | all (dict) | It is a Boolean type function, which returns true if all keys of dictionary are true (or the dictionary is empty). |
| 2. | any (dict) | It is also a Boolean type function, which returns true if any key of the dictionary is true. It returns false if the dictionary is empty. |
| 3. | len (dict) | It returns the number of items (length) in the dictionary. |
| 4. | cmp (dict1, dict2) | It compares the items of two dictionaries. |
| 5. | sorted (dict) | It returns the sorted list of keys. |
| 6. | dict.clear ( ) | It deletes all the items in a dictionary at once. |
| 7. | dict. copy ( ) | It returns a copy of the dictionary. |
| 8. | dict.get (key, default = None) | For key key, returns value or default if key not in dictionary. |
| 9. | dict.items ( ) | It returns a list of entire key : value pair of dictionary. |
| 10. | dict.keys ( ) | It returns the list of all the keys in dictionary. |
| 11. | dict.update (dict2) | It adds the items from dict2 to dict. |
| 12. | dict.values ( ) | It returns all the values in the dictionary. |