**Core Python Basic Questions & Answers**

## 1)How does the python code executes?

Answer:-

Below are the steps to execute the Python program.

Write a python program

Execute the program using Python Compiler.

Python compiler converts Python code to byte code.

By using PVM we convert byte code into machine code.

The processor reads the machine code and its result is displayed.

## 2) What is an Identifier in python?

Answer:-

The identifier is the name that consists of some letters, numbers, and special characters such as ‘\_’ underscore. It should always start with non-numeric characters. It is used to identify variables, functions, symbolic constants, classes. Python is a case-sensitive programming language.

Here are some naming conventions for Python identifiers:

1.Class names start with uppercase.

2.The identifier which starts with a single underscore is called a private identifier.

3.If the identifier starts and ends with a double underscore, then they are language-defined special names.

## 3) What are Constants in python?

Answers:-

A Constant identifier whose value does not change throughout the program execution. The way C and Java programs have constants, Python does not have constants. It is not possible to define a constant in Python whose value does not change. Constants in Python are defined at the module level and are written in capital letters to let the programmer know that it is a constant that does not have to be changed, but its value can change.

## 4) What are the different types of DataType in python?

Answers:-

Everything in Python is an object, so each value has a data type. In python, datatypes represent which type of data is stored in memory. The data type determines the type of the object in your program.

The following datatypes are available in Python:

.Built-in Data type

.User Defined Data type

## 5) What are the Built-in Datatype in Python?

Answers:-

The following built in datatypes are available in Python.

• None Type

• Numeric Types

• Sequences

• Sets

• Mappings

## 6) What are the User Defined Data types in Python?

Answers:-

The data types which are created on the basis of business requirements are called user-defined datatypes. The following data types are available in Python

. Array

. Class

. Module

## 7) What is None Type in Python?

Answers:-

None datatype is Python’s built-in data type that represents an object that does not contain any values.

## 8) What is the Numeric Type / Number in python?

Answers:-

Number datatype stores numeric values. When a value is assigned, a Number object is created. Using the del sNumber datatype stores numeric values. When a value is assigned, a Number object is created. Using the del statement, we can delete the reference to the Number object. We can delete a single object or multiple object references by using the del statement. Python supports the following different numerical types.

. int

. long

. float

. complex

## 9) What are Operators in Python?

Answers:-

The operator is a symbol that performs an operation in Python.

• Arithmetic Operators

• Comparison Operators

• Logical Operators

• Assignment Operators

• Bitwise Operators

• Membership Operators

• Identity Operators

## 10) What is Recursion in python?

Answers:-

The function which calls itself repeatedly to compute the value is called recursive function or recursion.

## 11) What is a Math Module in python?

Answers:-

It is a module that has many functions to perform mathematical operations. If we want to use the functions of this module then it has to be imported first.

from math import \*

## 12) What is Function Decorator in python?

Answers:-

The decorator function accepts a function as an input parameter and returns a function as an output parameter. The decorator accepts the result of the function, modifies it, and returns it. The function in the decorator is called as an input parameter to another function and then the inside wrapper function is called. To make any function a decorator function, @function\_name is written above that function.

## 13) How to define Nested Lambda Function in python?

Answers:-

When we write the second lambda function inside the lambda functions, it is called the nested lambda function.

add = lambda x=5:(lambda y:x\*y)

a = add()

print(a)

print('type of a:',type(a))

print(a(20))

## 14) What is map function in python?

Answers:-

These functions execute specific functions that execute for all the elements of the sequence and change the elements.

<strong>Syntax:</strong>

map(function\_name, iterable )

Function name - This is the function which performs the operation on all the elements and the modified elements are returned and they are stored in another sequence.

An iterable - iterable sequence such as list, string, or tuple

## 15) What is a Generator in python?

Answers:-

In Python, generators are functions that return a sequence of values. The yield statement in the generator is used to return the value from the function.

## 16) What is a next() Function in python?

Answer:-

This function is used to retrieve the value by element from the generator object.

## 17) What is a Comprehension in python?

Answer:-

Comprehension consists of a compact code of a single statement to perform a task.

• List Comprehension

• Set Comprehension

• Dictionary Comprehension

## 18) What is a list Comprehension in python?

Answer:-

List comprehension represents a new list is created from iterable objects that satisfy the given condition. List comprehension can have zero or more if statements. There can be more than one loop here.

lst = [i\*i for i in range(10)]

print(lst)

## 19) What is a Set Comprehension in python?

Answer:-

The set comprehension represents a new set is created from iterable objects that satisfy the given condition. A set comprehension can have zero or more if statements. There can also be more than one loop here.

s = {i for i in range(10) if i%2==0}

print(s)

## 20) What is a dictionary comprehension in python?

Answer:-

In python, Dictionary comprehension represents a new dictionary is created from iterable objects which satisfy the given condition. Dictionary comprehension can have zero or more if statements. There can also be more than one loop here.

d = {n:n\*n for n in range(5)}

print(d)

## 21) What is Pass/Call by Object Reference in python?

Aswer:-

Values are passed from object references to functions in Python. If the passed object is immutable then its modified value is not available outside the function. If the object is mutable, the modified value is available outside the function.

Immutable Objects – String, Integer and Tuple

Mutable Objects – List and Dictionary

## 22) What is a Local Variables in python function?

Answer:-

The variables which are declared inside the function are called local variables.

A local variable has scope only in the function where it is created. This means the local variable is available only inside the function and not outside the function.

def squr(num) :

x = 10

print(x)

print(x+num)

squr(5)

print(x)

## 23) What is Global Variables in the python function?

Answer:-

When a variable is declared before a function, it is called a global variable. After declaring these variables, they can be used in all the functions that are created. The scope of a global variable is in the entire program body, which is created after declaring the variable.

a = 25

def disp():

x = 50

print(a)

print(x)

disp()

print('a:',a)

## 24) What is Return Statement in function?

Answer:-

The return statement is used to return something from the function. In Python, using a return statement we can return more than one value to the caller object.

def mul(x):

return x,x\*x

print(mul(5))

## 25) What is \_\_init\_\_ in Python?

Answer:-

In Python classes, the reserved method init serves a similar purpose as constructors in object-oriented programming (OOP) terminology. When a new object is created, the init method is automatically called, initializing the object and allocating memory for it. This method can also be utilized to set initial values for variables.

class Human:

 def \_\_init\_\_(self, age):

  self.age = age

 def say(self):

print('Hello, my age is', self.age)

h = Human(22)

Output:

Hello, my age is 22

## 26) What is scope resolution?

Answer:-

In Python, a scope defines the region of code where an object remains valid and accessible. Every object in Python operates within its designated scope. Namespaces are used to uniquely identify objects within a program, and each namespace is associated with a specific scope where objects can be used without any prefix. The scope of a variable determines its accessibility and lifespan.

## 27) How does break, continue, and pass work?

Answer:-

* Python break: This statement aids in discontinuing the loop or the statement and transferring control to the subsequent statement.
* Python continue: This statement enforces the execution of the subsequent iteration when a particular condition is met, instead of terminating it.
* Python pass: This statement allows the syntactical writing of code while intending to bypass its execution. It is also recognized as a null operation, as no action is taken when the pass statement is executed.

## 28) What type of language is python? Programming or scripting?

Answer:-

Python is categorized as both a programming language and a scripting language, and this dual nature is part of what makes it so versatile and popular.

## 29) Is multiple inheritance supported in Python?

Answer:-

Yes, unlike Java, Python provides users with a range of support in terms of inheritance and its usage. Multiple inheritance refers to a scenario where a class is instantiated from more than one individual parent class. This provides a lot of functionality and advantages to users

**30)What is a module in Python with example?**

Answer:-

Module is a file containing Python definitions, functions, classes, and variables. It acts as a container for related code and can be imported into other Python programs using the import statement. Modules help in modularizing code and promote code reuse.

“**Import**” is used to get all functionalities of any particular module.  
Python provides a wide range of built-in modules that offer various functionalities like:

1. **math**: Provides mathematical functions and constants.
2. **random**: Offers functions for generating pseudo-random numbers.
3. **datetime**: Enables manipulation of dates, times, and time intervals.
4. **os**: Allows interaction with the operating system, providing functions for file operations, directory handling, etc.
5. **sys**: Provides access to system-specific parameters and functions.
6. **json**: Enables encoding and decoding of JSON data.
7. **re**: Provides regular expression matching operations.
8. **csv**: Offers functionality for reading and writing CSV files.
9. **urllib**: Allows making HTTP requests and working with URLs.
10. **sqlite3**: Provides a simple and lightweight database interface for SQLite database

**31) What is the use of Floor Division (//) in python?**

Answer:-

Floor division in Python is denoted by the double forward slash operator //. It performs division between two numbers and returns the largest integer less than or equal to the quotient

**32) What is the use Range function in python ?**

**Answer:**-

range() function in Python allows to create a sequence of numbers that can be used for iteration. It can be used to create a range of numbers with a specified start, stop, and step value.  
[ i.e. range( start, stop, steps) ]

**33) What are \*args and \*\*kwargs in python?**

**Answer:-**

**\*args (pronounced “star args”)** allows you to pass multiple number of arguments to a function without specifying their names in advance. It collects all the arguments into a tuple, which you can access within the function.

**34) What is PEP 8 ?**

Answer:-

PEP 8 is a guide that helps in writing clean, consistent, and maintainable Python code that is easy to read and understand. It stands for Python Enhancement Proposal, it specifically provides guidelines and recommendations on how to format and structure Python code to enhance readability and reliability of the code.

Some key points of PEP 8 include:

Indentation: Use 4 spaces for indentation to improve code readability.

Line Length: Limit lines to a maximum of 79 characters to ensure readability, although it can be extended up to 120 characters in certain cases.

Naming Conventions

Function and Variable Names

Imports: Import modules on separate lines and follow a specific ordering convention (standard library modules, third-party modules, local modules).

It also provides numerous other guidelines covering various aspects of coding style, including whitespace, blank lines, operator spacing, and more.

**35) What are the types of literals in Python?**

**Answer:-**

**Numeric Literal**: Numeric literals can be floating-point values, integers, or complex numbers.

**Character Literal:** A character literal consists of a single character enclosed in double quotes.

**Boolean Literal:** The boolean literals are True or False.

**Literal Collections:** There are four types of literal collections, including list literals, tuple literals, set literals, and dictionary literals.

**String Literal:** String literal is created by assigning text to a variable using single or double quotes. Multiline literals can be formed by enclosing text within triple quotes.

**36) what is pickling and Unpickling in Python ?**

**Answer:-**

**Pickling:**

Pickling is the process of converting a Python object hierarchy into a byte stream.

It allows you to save the state of an object or a collection of objects as a file or transfer it over a network.

The resulting byte stream can be stored persistently or transmitted between different systems.

Pickling is commonly used for tasks like caching, serialization, and data persistence.

**Unpickling:**

Unpickling is the process of reconstructing a Python object hierarchy from a byte stream.

It is the reverse operation of pickling and allows you to restore the state of the objects.

By unpickling, you can retrieve the original object or data structure that was pickled.

Unpickling is essential when you want to retrieve and utilize the saved data or objects.

**37) How memory is managed in python programming language ?**

**Answer:-**

In Python, memory management operates in the following manner:

Memory management in Python is handled by a private heap space. All Python objects and data structures are stored within this private heap, which remains inaccessible to programmers. The responsibility of managing this private heap lies with the Python interpreter.

The allocation of heap space for Python objects is handled by Python’s memory manager. Although programmers do not have direct access to this process, Python’s core API provides certain tools that can be utilized.

Python incorporates an internal garbage collector that is responsible for reclaiming unused memory. This ensures that memory becomes available within the heap space for future utilization.

**38) What Are The Principal Differences Between The Lambda And Def?**

Answer:-

Def can hold multiple expressions while lambda is a uni-expression function.

Def generates a function and designates a name to call it later. Lambda forms a function object and returns it.

Def can have a return statement. Lambda can’t have return statements.

Lambda supports to get used inside a list and dictionary.

**39) What is \*args and \*\*kwargs?**

Answer:-

\*args is used when the programmer is not sure about how many arguments are going to be passed to a function, or if the programmer is expecting a list or a tuple as argument to the function.

\*\*kwargs is used when a dictionary (keyword arguments) is expected as an argument to the function.

**40) How does keyword argument passing work in Python?**

Answer:-

Answer: In keyword argument passing, you can pass arguments to a function by explicitly specifying the parameter names.

**41) What is the purpose of the \* operator in function arguments?**

Answer:-

The \* operator is used to unpack iterables (like lists or tuples) when passed as arguments to a function.

**42) Can a function return multiple values?**

Answer:-

Yes, a function can return multiple values by separating them with commas or by returning a data structure like a tuple.

**43) What is a variable in Python?**

Answer:-

A variable is a named location in memory used to store data values.

**44)How do you declare a variable in Python?**

Answer:-

You can declare a variable by using a name followed by an equal sign and the value (for eg: x = 10)

**45)What are the rules for naming variables in Python?**

Answer:-

Variable names must start with a letter or an underscore (\_), followed by letters, numbers, or underscores.

**46)Can Python variable names start with a number?**

Answer:-

No, variable names in Python cannot start with a number.

**47)What is the scope of a variable in Python?**

Answer:-

The scope of a variable determines where in the code it can be accessed. Variables can have local or global scope.

**48) How do you check the data type of a variable in Python?**

Answer:-

You can use the type() function. For example: type(x)

**49) )What is the difference between == and is in Python?**

Answer:-

== checks for equality of values, while ‘is’ checks for identity (whether they refer to the same object).

**50) How do you represent a complex number in Python?**

Answers:-

Complex numbers are represented as a + bj in Python, where ‘a’ and b are real numbers, and j represents the imaginary unit.

**51) What is pass in Python?**

Answer:-

Pass means, no-operation Python statement, or in other words it is a place holder in compound statement, where there should be a blank left and nothing has to be written there.

**52) What Does The Continue Do In Python?**

Answer:-

The continue is a jump statement in Python which moves the control to execute the next iteration in a loop leaving all the remaining instructions in the block unexecuted.

The continue statement is applicable for both the “while” and “for” loops.

**53) When Should You Use The “Break” In Python?**

Answer:-

Python provides a break statement to exit from a loop. Whenever the break hits in the code, the control of the program immediately exits from the body of the loop.

The break statement in a nested loop causes the control to exit from the inner iterative block.

**54)What is Encapsulation?**

Answer:-

Encapsulation is also a part of the OOPs concept. It refers to the bundling of data with the methods that operate on that data. It also helps to restrict any direct access to some of an object’s components.

**55)What is the difference between method overloading and method overriding?**

Answer:-

There is a concept where two or more methods can have the same name. But they should have different parameters, different numbers of parameters, different types, or both. These methods are known as overloaded methods and this feature is called method overloading.

**56)What is the difference between constructors and destructor?**

Answer:-

In Python, a constructor is a special method used to initialize objects. You can think of it as blueprint instructions for creating objects.

When you create an instance of a class, the constructor method, \_\_init\_\_, is automatically called. It allows you to set up properties or execute any setup code needed for the object.

If you don’t define a constructor in your class, Python provides a default one. However, defining your own constructor gives you more control over the object initialization process.

A decorator in Python is a function that you can use to modify or extend the behavior of other functions or methods without changing their code. In the context of OOP, decorators can be used to add functionality to methods within classes, such as logging or access control.

You can think of decorators as wrappers around a function or method. When you call a decorated function, the decorator’s code runs first, allowing you to perform actions before or after the original function. It helps keep the code clean and adhere to the OOP reusability principle.

**57)What is data abstraction,How to achieve data abstraction?**

Answer:-

Data abstraction is one of the most important features of OOPs. It only allows important information to be displayed. It helps to hide the implementation details.

For example, while using a mobile, you know, how can you message or call someone but you don’t know how it actually happens.

This is data abstraction as the implementation details are hidden from the user.

Data abstraction can be achieved using two ways:

Abstract class

Abstract method

**58)How do you implement abstract classes and methods in Python?**

Answer:

In Python, you can implement abstract classes and methods using the ABC module. You would first import the module and then create a class that inherits from ABC, which stands for abstract base class.

Within this class, you can define abstract methods using the @abstractmethod decorator. These abstract methods serve as a blueprint for other classes, enforcing that the derived classes must provide a concrete implementation of these methods.

If a derived class does not implement the abstract methods, an instantiation error will occur.

**59)Differentiate between an abstract class and an interface?**

Answer:-

An interface can have only abstract methods, but an Abstract class can have abstract and non-abstract methods.

The interface should be used if just the requirement specification is known and nothing about implementation. If the implementation is known, but partially, then an abstract class should be used. If the implementation is known completely, then a concrete Class should be used.

**60)What is the use of the super() function?**

Answer:

Python’s super() function is used within a class to call a method from a parent class, often within the context of method overriding.

If you have a method in a child class with the same name as a method in the parent class, you can use super() to call the parent method within the child method. This is particularly useful when you want to extend or modify the behavior of the parent method in the child class.

By using super(), you ensure that your code follows the inheritance hierarchy. It also makes the code more maintainable, as changes in the parent class can be easily propagated to child classes.

**61)What is a static method and how is it different from a class method?**

Answer:

A static method in Python belongs to a class rather than an instance of the class. You can define it using the @staticmethod decorator. It doesn’t require reference to the class or its instance and can be called on the class itself. Unlike regular instance methods, it doesn’t take a self parameter.

A class method, on the other hand, is defined with the @classmethod decorator and takes a reference to the class itself as its first parameter – usually named “cls.” It can access and modify class-level attributes, while a static method can’t.

A class method is more versatile as subclasses can override it, whereas a static method remains unchanged.

**62)Are class and structure the same? If not, what's the difference between a class and a structure?**

Answer:-

No, class and structure are not the same. Though they appear to be similar, they have differences that make them apart. For example, the structure is saved in the stack memory, whereas the class is saved in the heap memory. Also, Data Abstraction cannot be achieved with the help of structure, but with class, Abstraction is majorly used

**63)What is meant by Garbage Collection in OOPs world?**

Answer:-

Object-oriented programming revolves around entities like objects. Each object consumes memory and there can be multiple objects of a class. So if these objects and their memories are not handled properly, then it might lead to certain memory-related errors and the system might fail.

Garbage collection refers to this mechanism of handling the memory in the program. Through garbage collection, the unwanted memory is freed up by removing the objects that are no longer needed.

**Practical**

## 1) Converting numbers from decimal to binary using while loop

Answers:-

num = int(input("Enter a number: "))

b = 0

p = 1

n = num

while n>0:

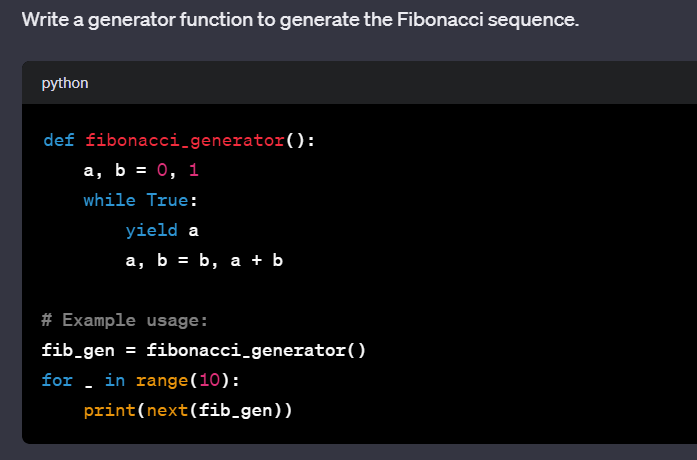
rem = n%2

b += rem \* p

p = p\*10

n = n//2

print("Binary value: ",b)

2) 

**3)python function program to display a calender?**

1. # First import the calendar module
2. **import** calendar
3. # ask of month and year
4. yy = int(input("Enter year: "))
5. mm = int(input("Enter month: "))
6. # display the calendar
7. **print**(calendar.month(yy,mm))

**4) Write a Python function that accepts a string and counts the number of upper and lower case letters.**

def string\_test(s):

d={"UPPER\_CASE":0, "LOWER\_CASE":0}

for c in s:

if c.isupper():

d["UPPER\_CASE"]+=1

elif c.islower():

d["LOWER\_CASE"]+=1

else:

pass

print ("Original String : ", s)

print ("No. of Upper case characters : ", d["UPPER\_CASE"])

print ("No. of Lower case Characters : ", d["LOWER\_CASE"])

string\_test('The quick Brown Fox')

**5) Write a Python function to create and print a list where the values are the squares of numbers between 1 and 30 (both included).**

def printValues():

l = list()

for i in range(1,21):

l.append(i\*\*2)

print(l)

printValues()

**6) Write a Python function to check whether a string is a pangram or not.**

Note : Pangrams are words or sentences containing every letter of the alphabet at least once.

For example : "The quick brown fox jumps over the lazy dog"

import string, sys

def ispangram(str1, alphabet=string.ascii\_lowercase):

alphaset = set(alphabet)

return alphaset <= set(str1.lower())

print ( ispangram('The quick brown fox jumps over the lazy dog'))

**7) Write a Python function that checks whether a passed string is a palindrome or not.**

Note: A palindrome is a word, phrase, or sequence that reads the same backward as forward, e.g., madam or nurses run.

def isPalindrome(string):

left\_pos = 0

right\_pos = len(string) - 1

while right\_pos >= left\_pos:

if not string[left\_pos] == string[right\_pos]:

return False

left\_pos += 1

right\_pos -= 1

return True

print(isPalindrome('aza'))

**8) Write a Python function that takes a number as a parameter and checks whether the number is prime or not.**

Note : A prime number (or a prime) is a natural number greater than 1 and that has no positive divisors other than 1 and itself.

def test\_prime(n):

if (n==1):

return False

elif (n==2):

return True;

else:

for x in range(2,n):

if(n % x==0):

return False

return True

print(test\_prime(9))

**8) How do you swap the values of two variables in Python without using a temporary variable? and write a program to swap values using tuple or arithmetic operations ?**

Answer:-

You can use tuple unpacking or arithmetic operations.

Example:

**(Using Tuple (a, b = b, a)**

a = 6

b = 9

b, a = a, b

print("a =", a)

print("b =", b)

Output

a = 9

b = 6

**Using Arithmetic Operations (a = a + b; b = a - b; a = a – b;)**

a = 6

b = 9

a = a + b

b = a - b

a = a - b

print("a =", a)

print("b =", b)

Output

a = 9

b = 6

**9) Filter integer values from the given list and find the sum of the integers**

**List= ["Python", "Java", "Ruby", 3, 4, 7, "PHP", 10 ,"+C++"]**

Answer:-

List= ["Python", "Java", "Ruby", 3, 4, 7, "PHP", 10 ,"+C++"]

integers = [x for x in List if isinstance(x, int)]

sum\_of\_integers = sum(integers)

print("Integers:", integers)

print("Sum of Integers:", sum\_of\_integers)

Output

Integers: [3, 4, 7, 10]

Sum of Integers: 24

**11) Write a Python program to create a class that represents a shape. Include methods to calculate its area and perimeter. Implement subclasses for different shapes like circle, triangle, and square.**

# Import the math module to access mathematical functions like pi

import math

# Define a base class called Shape to represent a generic shape with methods for calculating area and perimeter

class Shape:

# Placeholder method for calculating area (to be implemented in derived classes)

def calculate\_area(self):

pass

# Placeholder method for calculating perimeter (to be implemented in derived classes)

def calculate\_perimeter(self):

pass

# Define a derived class called Circle, which inherits from the Shape class

class Circle(Shape):

# Initialize the Circle object with a given radius

def \_\_init\_\_(self, radius):

self.radius = radius

# Calculate and return the area of the circle using the formula: π \* r^2

def calculate\_area(self):

return math.pi \* self.radius\*\*2

# Calculate and return the perimeter of the circle using the formula: 2π \* r

def calculate\_perimeter(self):

return 2 \* math.pi \* self.radius

# Define a derived class called Rectangle, which inherits from the Shape class

class Rectangle(Shape):

# Initialize the Rectangle object with given length and width

def \_\_init\_\_(self, length, width):

self.length = length

self.width = width

# Calculate and return the area of the rectangle using the formula: length \* width

def calculate\_area(self):

return self.length \* self.width

# Calculate and return the perimeter of the rectangle using the formula: 2 \* (length + width)

def calculate\_perimeter(self):

return 2 \* (self.length + self.width)

# Define a derived class called Triangle, which inherits from the Shape class

class Triangle(Shape):

# Initialize the Triangle object with a base, height, and three side lengths

def \_\_init\_\_(self, base, height, side1, side2, side3):

self.base = base

self.height = height

self.side1 = side1

self.side2 = side2

self.side3 = side3

# Calculate and return the area of the triangle using the formula: 0.5 \* base \* height

def calculate\_area(self):

return 0.5 \* self.base \* self.height

# Calculate and return the perimeter of the triangle by adding the lengths of its three sides

def calculate\_perimeter(self):

return self.side1 + self.side2 + self.side3

# Example usage

# Create a Circle object with a given radius and calculate its area and perimeter

r = 7

circle = Circle(r)

circle\_area = circle.calculate\_area()

circle\_perimeter = circle.calculate\_perimeter()

# Print the results for the Circle

print("Radius of the circle:", r)

print("Circle Area:", circle\_area)

print("Circle Perimeter:", circle\_perimeter)

# Create a Rectangle object with given length and width and calculate its area and perimeter

l = 5

w = 7

rectangle = Rectangle(l, w)

rectangle\_area = rectangle.calculate\_area()

rectangle\_perimeter = rectangle.calculate\_perimeter()

# Print the results for the Rectangle

print("\nRectangle: Length =", l, " Width =", w)

print("Rectangle Area:", rectangle\_area)

print("Rectangle Perimeter:", rectangle\_perimeter)

# Create a Triangle object with a base, height, and three side lengths, and calculate its area and perimeter

base = 5

height = 4

s1 = 4

s2 = 3

s3 = 5

# Print the results for the Triangle

print("\nTriangle: Base =", base, " Height =", height, " side1 =", s1, " side2 =", s2, " side3 =", s3)

triangle = Triangle(base, height, s1, s2, s3)

triangle\_area = triangle.calculate\_area()

triangle\_perimeter = triangle.calculate\_perimeter()

print("Triangle Area:", triangle\_area)

print("Triangle Perimeter:", triangle\_perimeter)

**12)Write a Python class named Student with two attributes student\_id, student\_name. Add a new attribute student\_class and display the entire attribute and the values of the class. Now remove the student\_name attribute and display the entire attribute with values.**

class Student:

student\_id = 'V10'

student\_name = 'Jacqueline Barnett'

print("Original attributes and their values of the Student class:")

for attr, value in Student.\_\_dict\_\_.items():

if not attr.startswith('\_'):

print(f'{attr} -> {value}')

print("\nAfter adding the student\_class, attributes and their values with the said class:")

Student.student\_class = 'V'

for attr, value in Student.\_\_dict\_\_.items():

if not attr.startswith('\_'):

print(f'{attr} -> {value}')

print("\nAfter removing the student\_name, attributes and their values from the said class:")

del Student.student\_name

#delattr(Student, 'student\_name')

for attr, value in Student.\_\_dict\_\_.items():

if not attr.startswith('\_'):

print(f'{attr} -> {value}')

**13.Write a Python class to get all possible unique subsets from a set of distinct integers.**

**Input : [4, 5, 6]**

**Output : [[], [6], [5], [5, 6], [4], [4, 6], [4, 5], [4, 5, 6]]**

class py\_solution:

def sub\_sets(self, sset):

return self.subsetsRecur([], sorted(sset))

def subsetsRecur(self, current, sset):

if sset:

return self.subsetsRecur(current, sset[1:]) + self.subsetsRecur(current + [sset[0]], sset[1:])

return [current]

print(py\_solution().sub\_sets([4,5,6]))

**14.Write a Python class to reverse a string word by word.**

**Input string : 'hello .py'**

**Expected Output : '.py hello'**

class py\_solution:

def reverse\_words(self, s):

return ' '.join(reversed(s.split()))

print(py\_solution().reverse\_words('hello .py'))

**15.Write a Python class to find a pair of elements (indices of the two numbers) from a given array whose sum equals a specific target number.**

**Note: There will be one solution for each input and do not use the same element twice.**

**Input: numbers= [10,20,10,40,50,60,70], target=50**

**Output: 3, 4**

class py\_solution:

def twoSum(self, nums, target):

lookup = {}

for i, num in enumerate(nums):

if target - num in lookup:

return (lookup[target - num], i )

lookup[num] = i

print("index1=%d, index2=%d" % py\_solution().twoSum((10,20,10,40,50,60,70),50))

**16. Write a Python class that has two methods: get\_String and print\_String , get\_String accept a string from the user and print\_String prints the string in upper case?**

class IOString():

def \_\_init\_\_(self):

self.str1 = ""

def get\_String(self):

self.str1 = input()

def print\_String(self):

print(self.str1.upper())

str1 = IOString()

str1.get\_String()

str1.print\_String()

**17.Write a Python class Employee with attributes like emp\_id, emp\_name, emp\_salary, and emp\_department and methods like calculate\_emp\_salary, emp\_assign\_department, and print\_employee\_details.**

Sample Employee Data:

"ADAMS", "E7876", 50000, "ACCOUNTING"

"JONES", "E7499", 45000, "RESEARCH"

"MARTIN", "E7900", 50000, "SALES"

"SMITH", "E7698", 55000, "OPERATIONS"

Use 'assign\_department' method to change the department of an employee.

Use 'print\_employee\_details' method to print the details of an employee.

Use 'calculate\_emp\_salary' method takes two arguments: salary and hours\_worked, which is the number of hours worked by the employee. If the number of hours worked is more than 50, the method computes overtime and adds it to the salary. Overtime is calculated as following formula:

overtime = hours\_worked - 50

Overtime amount = (overtime \* (salary / 50))

class Employee:

def \_\_init\_\_(self, name, emp\_id, salary, department):

self.name = name

self.id = emp\_id

self.salary = salary

self.department = department

def calculate\_salary(self, salary, hours\_worked):

overtime = 0

if hours\_worked > 50:

overtime = hours\_worked - 50

self.salary = self.salary + (overtime \* (self.salary / 50))

def assign\_department(self, emp\_department):

self.department = emp\_department

def print\_employee\_details(self):

print("\nName: ", self.name)

print("ID: ", self.id)

print("Salary: ", self.salary)

print("Department: ", self.department)

print("----------------------")

employee1 = Employee("ADAMS", "E7876", 50000, "ACCOUNTING")

employee2 = Employee("JONES", "E7499", 45000, "RESEARCH")

employee3 = Employee("MARTIN", "E7900", 50000, "SALES")

employee4 = Employee("SMITH", "E7698", 55000, "OPERATIONS")

print("Original Employee Details:")

employee1.print\_employee\_details()

employee2.print\_employee\_details()

employee3.print\_employee\_details()

employee4.print\_employee\_details()

# Change the departments of employee1 and employee4

employee1.assign\_department("OPERATIONS")

employee4.assign\_department("SALES")

# Now calculate the overtime of the employees who are eligible:

employee2.calculate\_salary(45000, 52)

employee4.calculate\_salary(45000, 60)

print("Updated Employee Details:")

employee1.print\_employee\_details()

employee2.print\_employee\_details()

employee3.print\_employee\_details()

employee4.print\_employee\_details()

**18)Create a list with different type of elements as input , find the string elements from that list.**

Eg:-

List inputs:-

list = [10,”ABCD”,True,300,40,”QWERTY”,”Hello”]

Output:-

[”ABCD”, ”QWERTY”, ”Hello”]

**19)Print out the reverse of a sentence without using built-in methods**

Eg:-

Input:- s=” This is python programming”

Output:- programming python is this