Python OOPs Assignment

Q.1 What is the purpose of Python's OOP?

ANS. The concept of OOP in Python focuses on creating reusable code. This concept is also known as DRY(Don't Repeat Yourself).

Q2. Where does an inheritance search look for an attribute?
And. In Base class.

Q.3 How do you distinguish between a class object and an instance object?

Ans. In python any class can have only one class object and can have any number of instance objects.

Class objects are creates automatically when we create class and can access through class name only. So if we want to access static member variable of a class then we can access it through class object also(syntax— Class_name.static_member_variable_name).

For creating instance object:

Class Test:

A = 5 – STATIC MEMBER VARIABLE

T1 = Test()----- Instance object

Q4. What makes the first argument in a class's method function special?

Ans. The self parameter is a reference to the current instance of a class and is used to access variables that belongs to the class. It does not have to be names self, you can call it whatever you like, but it has to be the first parameter of any function in the class.

Q5. What is the purpose of the init method?

Ans. All classes have a function called __init__(), which is always executed when the class is being intiated. Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created.

Q6. What is the process for creating a class instance?

Ans. There are two method for instance creation and initialization. These are __init__() and __new__()

__init__() – It receives a fresh class instance together with initialization arguments and is responsible for intializing the class instance state.

The special method __new__() is a static method that is actually responsible for creating class instances. This __new__(cls,[,...]) method is calles prior to the __init__() initialization method.

Q7. What is the process for creating a class?

Ans. To create a class in python use the keyword class.

Example-- class Test:

CREATING A TEST CLASS

Q8. How would you define the superclasses of a class?

Ans. The class whose subclass has been made is called a superclass. Other names of superclasses are base class or parent class and other names of subclass are derived class or child class.

Q9. What is the relationship between classes and modules?

Ans.

Q10. How do you make instances and classes?

Ans. The following statement makes a class with no attributes attached-

Class Test:

    Pass

To create instances of a class, you call the class using class name and pass in whatever arguments its __init__

Method accepts.

Q11. Where and how should be class attributes created?

Ans. Class attributes are the variable defined directly in the class that are shared by all objects of the class. We can accessed using class name as well as using object with dot notation e.g. classname.class_attribute or object.class_attribute and we can change value by using classname.class_attribute = value will be reflected to all the objects.

```
class Student:
    count = 0
    def __init__(self):
        Student.count += 1

s1 = Student()
print(Student.count)
print(s1.count)
Student.count = 9
print(Student.count)
print(s1.count)
```

Q12. Where and how are instance attributes created?

Ans. Instance attributes are the attributes or properties attached to an instance of a class. Instance attributes are defined in the constructor using self parameter. Specified to object. Accessed using object dot notation e.g. object.instance_attribute and changing the value of instance attribute will not be reflected to the other objects.

```
class Student:
    def __init__(self,name,age):
        self.name = name
        self.age = age

s1 = Student("Shivam",22)
print(s1.name," -- ",s1.age)
```

Q13. What does the term "self" in a Python class mean?

Ans. The self parameter is a reference to the current instance of a class and is used to access variables that belongs to the class. It does not have to be names self, you can call it whatever you like, but it has to be the first parameter of any function in the class.

Q14. How does a Python class handle operator overloading?

Ans. The operator overloading in Python means provide extended meaning beyond their predefined operational meaning. Such as we use the “+” operator for adding two integers as well as joining two strings or merging two lists. We can achieve this as the “+” operator is overloaded by the “int” class and “str” class. The user can notice that the same inbuilt operator or function is showing different behavior for objects of different classes. This process is known as operator overloading.

```
print(1+4)

print("Shiv"+"am")
```

Q15. When do you consider allowing operator overloading of your classes?

Ans. When we want to use function in different ways for the objects of different classes then we can consider operator overloading e.g. len()

Q16. What is the most popular form of operator overloading?

Ans. A very popular form of operator overloading is the Addition(+) operator. Just like how + operator operated on two numbers and the same operator operates on two strings. It performs “Addition” on numbers whereas it performs concatenation on strings.

Q17. What are the two most important concepts to grasp in order to comprehend Python OOP code?

Ans. Both inheritance and Polymorphism are fundamental concepts of object oriented programming. These concepts help us to create code that can be extended and easily maintainable. Inheritance is a great way to eliminate repetitive code.

Q18. Describe three applications for exception processing.

Ans. 1.Database connectivity,

2.Release of resources- if any error is occurred then we can release the resources.

Q19. What happens if you don't do something extra to treat an exception?

Ans. When an exception occurred, if you don't handle it then the program terminates abruptly and the code past the line that caused the exception will not get executed.

Q20. What are your options for recovering from an exception in your script?

Ans. We can recover from an exception by only handling it (or by using exception handing)otherwise program terminates abruptly.

Q21. Describe two methods for triggering exceptions in your script.

Ans. There are two methods to handle Python exceptions:

Try – this method catches the exceptions raised b the program .

Raise – Triggers an exception manually using custom exceptions.

Q22. Identify two methods for specifying actions to be executed at termination time, regardless of whether or not an exception exists.

Ans. Try and finally statement.

Try statement – During the execution of the try statement, if no exceptions occurred then, the interpreter ignores the exception handlers for that specific try statement.

Finally statement- Finally block always executes irrespective of an exception being thrown or not.

Q23. What is the purpose of the try statement?

And. A try statement includes keyword try, followed by a colon (:) and a suite of code in which exceptions may occur. It has one or more clauses.During the execution of the try statement, if no exceptions occurred then, the interpreter ignores the exception handlers for that specific try statement.In case, if any exception occurs in a try suite, the try suite expires and program control transfers to the matching except handler following the try suite.

```
Syntax:
try:
statement(s)
```

Q24. What are the two most popular try statement variations?

Q25. What is the purpose of the raise statement?

Ans. The raise statement specifies an argument which initializes the exception object. Here, a comma follows the exception name, and argument or tuple of the argument that follows the comma.

Syntax:

```
raise [Exception [, args [, traceback]]]
```

In this syntax, the argument is optional, and at the time of execution, the exception argument value is always none.

Q26. What does the assert statement do, and what other statement is it like?

Ans. The assert keyword is used when debugging code. The assert keyword lets you test if a condition in you code returns True, if not then the program will raise an Assertion Error.

Q27. What is the purpose of the with/as argument, and what other statement is it like?

Ans. With statement is used in exception handing to make the code cleaner and much more readable. It simplifies the management of common resources like file streams.

Q28. What are *args, **kwargs?

Ans. *args specifies the number of non-keyworded arguments that can be passes and the operations that can be performed on the function in Python whereas **kwargs is a variable number of keyworded arguments that can be passed to a function that can perform dictionary operations.

Q29. How can I pass optional or keyword parameters from one function to another?

Ans. To pass optional or keyword parameters from one function to another, collect the arguments using the * and ** specifiers in the functions parameter list.

Q30. What are Lambda Functions?

Ans. A lambda function is a small anonymous function. A lambda function can take any number of arguments but can only have one expression.

Q31. Explain Inheritance in Python with an example?

Ans. Inheritance is the capability of one class to derive or inherit the properties from another class.

```
class Person:# base class
    def __init__(self, name):
```

```
        self.name = name

    def displayName(self):
        print(self.name)

    def isEmployed(self):
        print(self.name, " is Un-employed")


class Employee(Person):
    def isEmployed(self):
        print(self.name, " is Employed")


emp1 = Employee("Shivam")
print("Employee name ",emp1.name)
emp1.isEmployed()
```

Q32. Suppose class C inherits from classes A and B as class C(A,B).Classes A and B both have their own versions of method func(). If we call func() from an object of class C, which version gets invoked?

Ans. Class A version will get invoked.

Q33. Which methods/functions do we use to determine the type of instance and inheritance?

Ans. Using isinstance() function, we can test wheather an object/variable is an instance of the specified type or class such as int or list. In the case of inheritance, we can checks if the specified class is the parent class of the object.

Q34.Explain the use of the 'nonlocal' keyword in Python.

Ans. The nonlocal keyword is used to work with variables inside nested functions, where the variable should not belong to the inner function. Use the keyword nonlocal to declare that the variable is not local.

```
def myfunc():
    x = "John"
    def myfunc1():
        nonlocal x
```

```
        x = "hello"
    myfunc1()
    return x
print(myfunc())
```

o/p- will be hello if we remove nonlocal keyword the the output will be John

Q35. What is the global keyword?

Ans. A global keyword is a keyword that allows a user to modify a variable outside the current scope**.** It is used to create global variables in Python from a non-global scope, i.e. inside a function. Global keyword is used inside a function only when we want to do assignments or when we want to change a variable.