**Python OOP Assignment**

Q1. What is the purpose of Python's OOP?

Ans1. In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.

Q2. Where does an inheritance search look for an attribute?

Ans2. An inheritance search looks for an attribute first in the instance object, then in the class the instance was created from, then in all higher superclasses, progressing from left to right (by default). The search stops at the first place the attribute is found.

Q3. How do you distinguish between a class object and an instance object?

Ans3. Class objects represent the class itself, while instance objects represent individual instances of the class.

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

Ans4. The calling process is automatic while the receiving process is not (its explicit). This is the reason the first parameter of a function in class must be the object itself.

Q5. What is the purpose of the init method?

Ans5. The init method is the Python equivalent of the C++ constructor in an object-oriented approach. The init function is called every time an object is created from a class. The init method lets the class initialize the object's attributes and serves no other purpose. It is only used within classes.

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

Ans6. When you create an object, you are creating an instance of a class, therefore "instantiating" a class.

Q7. What is the process for creating a class?

Ans7. The first step in **creating** your own **class** is identifying what state you want to capture to describe objects.

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

Ans8. A superclass is the class from which many subclasses can be created. The subclasses inherit the characteristics of a superclass. The superclass is also known as the parent class or base class.

Q9. What is the relationship between classes and modules?

Ans9. Modules are collections of methods and constants. They cannot generate instances. Classes may generate instances (objects), and have per-instance state (instance variables).

Q10. How do you make instances and classes?

Ans10. 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?

Ans11. **Class attributes** are the variables defined directly in the class that are shared by all objects of the class.

Accessed using class name as well as using object with dot notation, e.g. classname.class\_attribute or object.class\_attribute

The following example demonstrates the use of class attribute count.

class Student:

count = 0

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

Student.count += 1

Q12. Where and how are instance attributes created?

Ans12. **Instance attributes** are attributes or properties attached to an instance of a class. Instance attributes are defined in the constructor.

Accessed using object dot notation e.g. object.instance\_attribute

The following demonstrates the instance attributes.

class Student:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

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

Ans13. SELF represents the instance of class. This handy keyword allows you to access variables, attributes, and methods of a defined class in Python. The self parameter doesn't have to be named “self,” as you can call it by any other name.

Q14. How does a Python class handle operator overloading?

Ans14. To perform operator overloading, Python provides some special function or magic function that is automatically invoked when it is associated with that particular operator. For example, when we use + operator, the magic method \_\_add\_\_ is automatically invoked in which the operation for + operator is defined.

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

Ans15. Consider that we have two objects which are a physical representation of a class (user-defined data type) and we have to add two objects with binary '+' operator it throws an error, because compiler don't know how to add two objects. So we define a method for an operator and that process is called operator overloading.

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

Ans16. A very popular and convenient example is the Addition (+) operator. Just think how the '+' operator operates 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?

Ans17. 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.

Q18. Describe three applications for exception processing.

Ans18. An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions.

Application 1:

# initialize the amount variable

amount = 10000

# check that You are eligible to

#  purchase Dsa Self Paced or not

if(amount > 2999)

Application2:

print("You are eligible to purchase Dsa Self Paced")

# initialize the amount variable

marks = 10000

# perform division with 0

a = marks / 0

print(a)

Application 3:

x = 5

y = "hello"

try:

    z = x + y

except TypeError:

    print("Error: cannot add an int and a str")

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

Ans19: When an exception occurred, if you don't handle it, 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?

Ans20: If you have some *suspicious* code that may raise an exception, you can defend your program by placing the suspicious code in a **try:** block. After the try: block, include an **except:** statement

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

Ans21. To avoid such a scenario, there are two methods to handle Python exceptions: Try – This method catches the exceptions raised by 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.

Ans22. Finally block always executes irrespective of an exception being thrown or not. The final keyword allows you to create a block of code that follows a try-catch block. Finally, clause is optional. It is intended to define clean-up actions which should be that executed in all conditions.

Q23. What is the purpose of the try statement?

Ans23. The try statement allows you to define a block of code to be tested for errors while it is being executed.

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

Ans24. There are two other optional segments to a try block: else and finally . Both of these optional blocks will come after the try and the except .

Q25. What is the purpose of the raise statement?

Ans25. The RAISE statement stops normal execution of a PL/SQL block or subprogram and transfers control to an exception handler.

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

Ans26. assert statement takes an expression and optional message. assert statement is used to check types, values of argument and the output of the function. assert statement is used as debugging tool as it halts the program at the point where an error occurs.

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

Ans27. In Python, the with statement replaces a try-catch block with a concise shorthand. More importantly, it ensures closing resources right after processing them. A common example of using the with statement is reading or writing to a file. A function or class that supports the with statement is known as a context manager.

Q28. What are \*args, \*\*kwargs?

Ans28. \*args and \*\*kwargs allow you to pass multiple arguments or keyword arguments to a function.

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

Ans29. When functions are defined then the parameters are written in the form “datatype keyword-name”. So python provides a mechanism to call the function using the keyword name for passing the values. This helps the programmer by relieving them not to learn the sequence or the order in which the parameters are to be passed.

Q30. What are Lambda Functions?

Ans30. 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?

Ans30. Inheritance allows you to inherit the properties of a class, i.e., base class to another, i.e., derived class.

Inheritance example:

# A Python program to demonstrate inheritance

# Base or Super class. Note object in bracket.

# (Generally, object is made ancestor of all classes)

# In Python 3.x "class Person" is

# equivalent to "class Person(object)"

class Person(object):

    # Constructor

    def \_\_init\_\_(self, name):

        self.name = name

    # To get name

    def getName(self):

        return self.name

    # To check if this person is an employee

    def isEmployee(self):

        return False

# Inherited or Subclass (Note Person in bracket)

class Employee(Person):

    # Here we return true

    def isEmployee(self):

        return True

# Driver code

emp = Person("Geek1")  # An Object of Person

print(emp.getName(), emp.isEmployee())

emp = Employee("Geek2")  # An Object of Employee

print(emp.getName(), emp.isEmployee())

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?

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

Ans33. Using isinstance() function, we can test whether 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 an object. For example, isinstance(x, int) to check if x is an instance of a class int

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

Ans34. 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.

Q35. What is the global keyword?

Ans35. 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.