**Python OOP Assignment**

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

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

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

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

Q5. What is the purpose of the init method?

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

Q7. What is the process for creating a class?

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

Q9. What is the relationship between classes and modules?

Q10. How do you make instances and classes?

Q11. Where and how should be class attributes created?

Q12. Where and how are instance attributes created?

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

Q14. How does a Python class handle operator overloading?

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

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

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

Q18. Describe three applications for exception processing.

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

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

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

Q22. Identify two methods for specifying actions to be executed at termination time, regardless of

whether or not an exception exists.

Q23. What is the purpose of the try statement?

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

Q25. What is the purpose of the raise statement?

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

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

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

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

Q30. What are Lambda Functions?

Q31. Explain Inheritance in Python with an example?

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?

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

Q35. What is the global keyword?

**Answers-**

Ans1. The purpose of Python's OOP (Object-Oriented Programming) is to allow for the creation of objects that have their own methods and attributes, and can be used to model real-world objects and their interactions.

Ans2. An inheritance search looks for an attribute in the current class, then in its parent classes, in the order specified by the class's inheritance hierarchy.

Ans3. A class object is the blueprint for creating instances, while an instance object is a specific instance of a class, created by calling the class as a function.

Ans4. The first argument in a class's method function, often named "self", refers to the instance of the class on which the method is being called.

Ans5. The purpose of the init method is to initialize the attributes of an instance of a class when it is created.

Ans 6. To create a class instance, you need to define a class, and then call the class as a function, passing any necessary arguments to the class's init method.

Ans 7. To create a class, you need to use the class keyword, give the class a name, and define the class's methods and attributes.

Ans 8. The superclasses of a class are specified as arguments in the class definition, with the class being defined as a subclass of the superclass.

Ans 9. A module is a file containing Python definitions and statements, while a class is a blueprint for creating instances, with methods and attributes. Classes can be defined and used within a module.

Ans 10. To make an instance, you need to call the class as a function, and to make a class, you use the class keyword.

Ans 11. Class attributes can be created by defining them within the class's body, but outside of any method.

Ans 12. Instance attributes can be created by defining them within the class's init method, and setting them as instance variables using the self keyword.

Ans 13. The term "self" in a Python class refers to the instance of the class on which a method is being called.

Ans 14. Python classes handle operator overloading by defining special methods with double underscores, such as add for the + operator.

Ans 15. You should consider allowing operator overloading when it makes sense for the class and its instances to support the given operator.

Ans 16. The most popular form of operator overloading is using special methods with double underscores, such as add for the + operator.

Ans 17. The two most important concepts to grasp in order to comprehend Python OOP code are classes and objects.

Ans 18. Three applications for exception processing are handling errors, validating input, and providing alternative flow of control.

Ans 19. If you don't do something extra to treat an exception, the script will stop running.

Ans 20. The options for recovering from an exception in a script include using try/except statements, using the raise statement to throw an exception, and using the assert statement to check for errors.

Ans 21. Two methods for triggering exceptions in a script are using the raise statement to throw an exception, and using the assert statement to check for errors.

Ans 22. Two methods for specifying actions to be executed at termination time, regardless of whether or not an exception exists are the try-finally statement, and using a context manager with the "with" statement.

Ans 23. The try statement in Python is used to handle exceptions or errors that may occur during the execution of a program. It provides a way to run some code and if an exception occurs, the program can handle it and continue running.

Ans 24. The two most popular variations of the try statement are the try-except and the try-finally. The try-except block is used to catch exceptions, while the try-finally block is used to run some code regardless of whether an exception occurs or not.

Ans 25. The raise statement is used to raise an exception in Python. It allows the programmer to throw a custom exception, and provides a way to handle the exception in a specific way.

Ans 26. The assert statement is used to check if a condition is true, and if it is not, an AssertionError is raised. It is similar to using an if statement and raising an exception manually, but is more concise.

Ans 27. The with/as argument is used in the with statement, which provides a way to ensure that a particular block of code is executed within a managed context, such as a file or a lock. It is similar to the try-finally statement, but provides more context management functionality.

Ans 28. \*args and \*\*kwargs are special syntax in Python for passing variable-length arguments to a function. \*args allows for passing a variable number of non-keyword arguments to a function, while \*\*kwargs allows for passing a variable number of keyword arguments.

Ans 29. To pass optional or keyword parameters from one function to another, you can use default values for parameters in the function definition, or use the \*args and \*\*kwargs syntax to pass a variable number of arguments.

Ans 30. A lambda function is a small, anonymous function in Python. It is often used as an argument to higher-order functions, such as map, filter, or reduce.

Ans 31. Inheritance in Python is a mechanism that allows a class to inherit properties and behavior from a parent class. An example of inheritance in Python could be a class 'Car' that inherits from a class 'Vehicle', where the class 'Car' has additional properties specific to cars, such as make and model, but also inherits properties from the class 'Vehicle', such as wheels and engine.

Ans 32. In the scenario described, class C inherits from classes A and B, and if both classes A and B have their own versions of method func(), the method from class A will be invoked. This is because class C uses the method resolution order (MRO) to determine the order in which to look for methods, and in this case, class A will be searched before class B.

Ans 33. To determine the type of an instance, you can use the built-in function type(). To check if an object is an instance of a particular class, you can use the isinstance() function. To check if a class inherits from another class, you can use the issubclass() function.

Ans 34. The 'nonlocal' keyword in Python is used to indicate that a variable being assigned to is in the nearest enclosing scope that is not global. It allows for the modification of variables in an inner scope, but outside of the current function.

Ans 35. The global keyword in Python is used to indicate that a variable being assigned to is in the global scope, meaning it can be accessed from anywhere in the program. The global keyword is used to modify global variables from within a function