**Python OOP Assignment**

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

**Ans:**

* 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?**

**Ans:**

* 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?**

**Ans:**

The class = the blue print. The Object is an actual thing that is built based on the 'blue print' (like the house). An instance is a virtual copy (but not a real copy) of the object.

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

**Ans:**

* The first parameter of a function in class must be the object itself. Writing this parameter as self is merely a convention.
* It is not a keyword and has no special meaning in Python.

**Q5. What is the purpose of the init method?**

**Ans:**

* The \_\_init\_\_ method lets the class initialize the object's attributes and serves no other purpose.
* It is only used within classes.

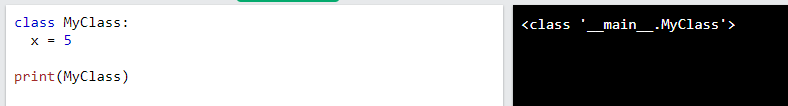
**Q6. What is the process for creating a class instance?**

**Ans:**

To create instances of a class, you call the class using class name and pass in whatever arguments its \_\_init\_\_ method accepts.

**Q7. What is the process for creating a class?**

To create a class, use the keyword class:



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

**Ans:**

The class from which the subclass is derived is called a superclass (also a base class or a parent class).

**Q9. What is the relationship between classes and modules?**

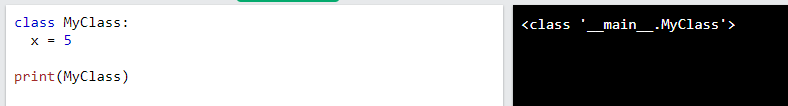
**Ans:**

* 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?**

**Ans:**

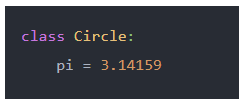
* To create instances of a class, you call the class using class name and pass in whatever arguments its \_\_init\_\_ method accepts.
* To create a class, use the keyword class:



**Q11. Where and how should be class attributes created?**

**Ans:**

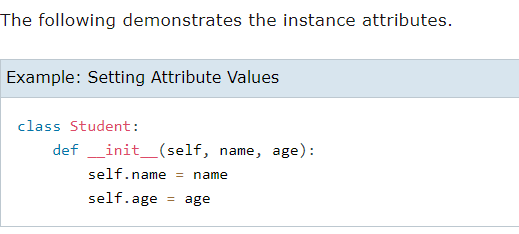
* To define a class attribute, you place it outside of the \_\_init\_\_() method. For example, the following defines pi as a class attribute:



**Q12. Where and how are instance attributes created?**

**Ans:**

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



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

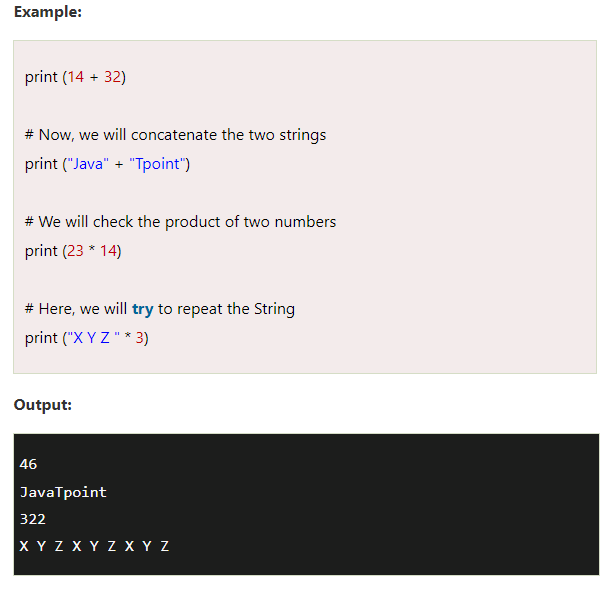
**Ans:**

* The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.
* It does not have to be named 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 behaviour for objects of different classes. This process is known as operator overloading.



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

**Ans:**

* 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?**

**Ans:**

* 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?**

**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.

**Q18. Describe three applications for exception processing.**

**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, 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:**

* You can also provide a generic except clause, which handles any exception. After the except clause(s), you can include an else-clause.
* The code in the else-block executes if the code in the try: block does not raise an exception.
* The else-block is a good place for code that does not need the try: block's protection.

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

**Ans:**

* **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.**

**Ans:**

* An exception is an error which happens at the time of execution of a program.
* However, while running a program, Python generates an exception that should be handled to avoid your program to crash.
* In Python language, exceptions trigger automatically on errors, or they can be triggered and intercepted by your code.
* The exception indicates that, although the event can occur, this type of event happens infrequently.
* When the method is not able to handle the exception, it is thrown to its caller function.
* Eventually, when an exception is thrown out of the main function, the program is terminated abruptly.

**Q23. What is the purpose of the try statement?**

**Ans:**

* The try block lets you test a block of code for errors.
* The except block lets you handle the error.
* The else block lets you execute code when there is no error.
* The finally block lets you execute code, regardless of the result of the try- and except blocks.

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

**Ans:**

* The Different Try/Except Variations. So far, we've used a try / except and even a try / except / except, but this is only two-thirds of the story.
* 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?**

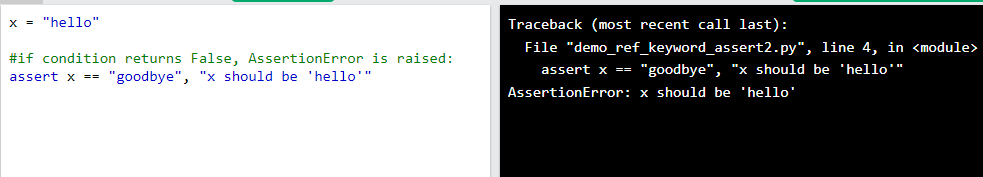
**Ans:**

* The RAISE statement stops normal execution of a PL/SQL block or subprogram and transfers control to an exception handler. RAISE statements can raise predefined exceptions, such as ZERO\_DIVIDE or NO\_DATA\_FOUND, or user-defined exceptions whose names you decide.

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

**Ans:**

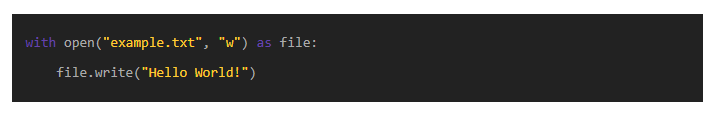
* The assert keyword is used when debugging code. The assert keyword lets you test if a condition in your code returns True, if not, the program will raise an AssertionError.
* You can write a message to be written if the code returns False, check the example below.



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

**Ans:**

The with statement is a replacement for commonly used try/finally error-handling statements. A common example of using the with statement is opening a file. To open and write to a file in Python, you can use the with statement as follows:



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

**Ans:**

* \*args specifies the number of non-keyworded arguments that can be passed and the operations that can be performed on the function in Python
* \*\*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:**

* Users can either pass their values or can pretend the function to use theirs default values which are specified.
* In this way, the user can call the function by either passing those optional parameters or just passing the required parameters.
* Without using keyword arguments. By using keyword arguments.

**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.
* Syntax:

lambda arguments : expression

**Q31. Explain Inheritance in Python with an example?**

**Ans:**

* Just like a parent-child relationship, inheritance works on derived classes relative to the base class.
* Every “Derived” class inherits from a “Base” class.
* The inheritance is represented in UML or Unified Modeling Language.
* It is a standard modeling language that includes an integrated set of diagrams to help developers specify, structure, and document software systems elements.
* Inheritance relationship defines the classes that inherit from other classes as derived, subclass, or sub-type classes.
* Base class remains to be the source from which a subclass inherits.
* For example, you have a Base class of “Animal,” and a “Lion” is a Derived class. The inheritance will be Lion is an Animal.

**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?**

**Ans:**

* 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.

**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.

**Q35. What is the global keyword?**

**Ans:**

The global keyword is used to create global variables from a no-global scope, e.g. inside a function.