**Python OOPS Programming**

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

Ans.) The purpose of OOPS is the programming paradigm in which that uses class and object in the programming. They uses real world entities like inheritance, polymorphism, encapsulation, abstraction.

Q.2) 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.

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

Ans.) A class is a blueprint which you use to create objects. An object is an instance of a class. it's a concrete 'thing' that you made using a specific class. So, 'object' and 'instance' are the same thing, but the word 'instance' indicates the relationship of an object to its class.

Q.4) What makes the first argument in a class’s method function special?

Ans.) the first argument of a method is called self . This is nothing more than a convention: the name self has absolutely no special meaning to Python.

Q.5) What is the purpose of the init method?

Ans.) init method is the special method in python i.e constructor. The first argument of init method is self.

Q.6) 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. In the constructor method we pass the values of object.

Q.7) What is the process for creating a class?

Ans.) The process for creating a class with the help of class keyword.

Class Employeee just like creating the employee classs.

Q.8) How would you define the superclasses of a class?

Ans.) The class from which a class inherits is called the parent or superclass. A class which inherits from a superclass is called a subclass, also called heir class or child class. Superclasses are sometimes called ancestors as well.

Q.9) What is the relationship between classes and modules?

Ans.) module in python is simply a way to organize the code, and it contains either python classes or just functions. If you need those classes or functions in your project, you just import them.Modules are collections of methods and constants. They cannot generate instances. Classes may generate instances (objects), and have per-instance state (instance variables).

Q.10) How do you make instances and classes?

Ans.) Class can be both variables and method. To create a class we use class keyword. In the class they hold properties and methods of the object. instance it is an object that is used for class.

Q.11) Where and how should be class attributes created?

* Ans.) A class attribute is shared by all instances of the class. To define a class attribute, you place it outside of the \_\_init\_\_() method.
* Use class\_name. ...
* Use class attributes for storing class contants, track data across all instances, and setting default values for all instances of the class.

Q.12) Where and how are instance attributes created?

Ans.) An instance attribute is a Python variable belonging to only one object. It is only accessible in the scope of the object and it is defined inside the constructor function of a class. For example, \_\_init\_\_(self,..).

Q.13) 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.

Q.14) 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.

Q.15) When do you consider allowing operator overloading of your classes?

Ans.) Operator overloading is mostly useful when you're making a new class that falls into an existing "Abstract Base Class" (ABC)  indeed, many of the ABCs in standard library module collections rely on the presence of certain special methods (and special methods, one with names starting and ending with double underscores.

Q.16) 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.

Q.17) 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.

Q.18) Describe three applications for exception processing.

Ans.) Raised when the specified key is not found in the dictionary. Raised when an identifier is not found in the local or global namespace. Raised when trying to access a local variable in a function or method but no value has been assigned to it. Base class for all exceptions that occur outside the Python environment.

Q.19) What happens if you don't do something extra to treat an exception?

Ans.) An exception object is created when a Python script raises an exception. If the script explicitly doesn't handle the exception, the program will be forced to terminate abruptly.

Q.20) 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.

Q.21) Describe two methods for triggering exceptions in your script.

Ans.) 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.

Q.22) Identify two methods for specifying actions to be executed at termination time, regardless of whether or not an exception exists.

Ans.)

Q.23) 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.

Q.24) 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 .

Q.25) What is the purpose of the raise statement?

Ans.) The raise keyword is used to raise an exception. You can define what kind of error to raise, and the text to print to the user.

Q.26) 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.

Q.27) What is the purpose of the with/as argument, and what other statement is it like?

Ans.) 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.

Q.28) 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 whereas \*\*kwargs is a variable number of keyworded arguments that can be passed to a function that can perform dictionary operations.

Q.29) How can I pass optional or keyword parameters from one function to another?

Ans.) By using keyword arguments.

Q.30) 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.

Q.31) Explain Inheritance in Python with an example?

Ans.) 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.

Q.32) 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.)

Q.33) Which methods/functions do we use to determine the type of instance and inheritance?

* Ans.) Use isinstance() to check an instance's type: isinstance(obj, int) will be True only if obj.\_\_class\_\_ is int or some class derived from int .
* Use issubclass() to check class inheritance: issubclass(bool, int) is True since bool is a subclass of int .

Q.34) 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.

Q.35) What is the global keyword?

Ans.) In Python, the global keyword allows you to change a variable value outside of its current scope. It is used to make changes to a global variable from a local location. The global keyword is only required for altering the variable value and not for publishing or accessing it.