## Python OOP Assignment

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

Ans- In Python, object-oriented Programming (OOPs) is a programming model that uses classes and objects in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. The main concept of OOPs is to bind data and functions that work together as a single unit so that no other part of the code can access this data.

Following are the main Concepts of Object-Oriented Programming (OOPs)

* Class
* Objects
* Polymorphism
* Encapsulation
* Inheritance
* Data Abstraction

**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).

­­­­­­­

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

Ans- Every object has a type and object types are created using classes. Instance is an object that belongs to a class

**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

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

Ans- \_\_init\_\_ is one of the reserved methods in Python. In object oriented programming, it is known as a constructor. The \_\_init\_\_ method can be called when an object is created from the class, and access is required to initialize the attributes of the class.

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

Ans- To create an instance of a class, you call the class as if it were a function. For example, consider the following class:

class MyClass:

pass

The following statement creates an instance of the class MyClass:

x = MyClass()

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

Ans- In Python, a class can be created by using the keyword class, followed by the class name. The syntax to create a class is given below.

Syntax:

class ClassName:

**Q8. 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 or child class.

**Q9. 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.

**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.

**Q11. 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.

class Dog:

# class attribute

attr1 = "mammal"

# Instance attribute

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

self.name = name

# create object

a = Dog("Raju")

# Accessing class attribute

print(a.attr1)

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

Ans- Instance attributes are defined in the constructor. Defined directly inside a class. Defined inside a constructor using the self parameter.

class Dog:

# class attribute

attr1 = "mammal"

# Instance attribute

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

self.name = name

# create object

a = Dog("Raju")

# Accessing Instance attribute

print(a.name)

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

Ans- self represents the instance of the class. By using the “self” we can access the attributes and methods of the class in python.

**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.

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

Ans- It allows us to provide an intuitive interface to our class users, plus makes it possible for templates to work equally well with classes and built-in types.

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

Ans- The most popular form of operator overloading is adding operator ‘+’

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

Ans- Both inheritance and polymorphism are important concepts of object oriented programming

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

Ans- Exceptions: Exceptions are raised when the program is syntactically correct, but the code resulted in an error. This error does not stop the execution of the program, however, it changes the normal flow of the program.

Example:-

marks = 10000

a = marks / 0

print(a)

Output:-

ZeroDivisionError

Try-else-Finally are the type of exception handling blocks are used.

# 3 Applications:-

1. This example opens a file, writes content in the, file and comes out gracefully because there is no problem at all.

2. This example tries to open a file where you do not have write permission, so it raises an exception.

3. To perform mathematic operation and handling error we get if someone enter a wrong input or performing different operation in a systamic manner without stopping the execution process.

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

Ans- When an exception occurred and if you don't handle it, your program will stop and complain, instead of continuing and corrupting the data.

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

Ans- You can provide a except clause, which handles any exception. After the except clause, 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 protection.

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

Ans- In Python, we can handle exception using try and except code blocks. The try clause contains the code that can raise an exception, while except clause contains the code lines that handle the exception.

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

**whether or not an exception exists**.

Ans- ### 1. Finally Statement in Python

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.

#### Code Example:-

try:

raise KeyboardInterrupt

finally:

print 'welcome, world!'

Output

Welcome, world!

KeyboardInterrupt

### 2. Try block

The Try Block

Let’s talk about the try block first. It encloses the part of your code that might throw the exception. If your code throws more than one exception, you can choose if you want to:

\* Use a separate try block for each statement that could throw an exception or

\* Use one try block for multiple statements that might throw multiple exceptions.

#### Code Example:-

try:

print(x)

except:

print("An exception occurred")

print(x)

#### OUTPUT:-

Traceback (most recent call last):

File "demo\_try\_except\_error.py", line 3, in <module>

print(x)

NameError: name 'x' is not defined

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

Ans- The try block lets you test a block of code for errors.

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

Ans- 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 keyword is used to raise an exception. Raise statement is used when we want to throw exception for particular condition.

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

Ans- The assert statement is useful to ensure that given condition is True. If it is not true, it raises AssertionError. If else and raise statement is like assert statement.

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

Ans- with/as are used in conditional or pointing statements. And other statement will be if else.

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

Ans- \*args collects extra positional arguments as a tuple while \*\*kwargs collects extra keyword arguments as dictionary.

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

Ans- By using keyword arguments you can pass optional or keyword parameters from one function to another.

**Q30. What are Lambda Functions?**

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

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

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

class Person:

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

self.name = name

self.idnumber = idnumber

def display(self):

print(self.name)

print(self.idnumber)

def details(self):

print("My name is",self.name)

print("IdNumber:",self.idnumber)

class Employee(Person):

def \_\_init\_\_(self, name, idnumber,salary,post):

self.salary = salary

self.post = post

super().\_\_init\_\_(name, idnumber)

def details(self):

print("My name is",self.name)

print("IdNumber:",self.idnumber)

print("Post:",self.post)

a = Employee("Ajay",5634,1000,"Engineer")

a.display()

a.details()

**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- The order of inheritance is A - B - C in this order then

The Version of [ B ] Class would be invoked when called.

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

Ans- Using isinstance() function we can determine the type of instance and inheritance.

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

Ans- By using nonlocal keyword you can use variable of outer function in nested function(inner function).

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

Ans- If local variable and global variable has same name then function by default refers to local variable and ignores the global variable.

It means global variable is not accessible inside the function but possible to access outside of function.

In this situation if we need to access global variable inside the function we can access it using global keyword followed by variable name.