Adarsh Raj, Ineuron Python OOP Assignment

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

Ans – Python's OOP is used when we incorporate objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. The concept of OOP in Python focuses on creating reusable code. This concept is also known as DRY (Don't Repeat Yourself).

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

Ans – Firstly, inheritance search looks for the attribute in the instance object itself, if not found then it moves to the class where the instance was created and from there search continues to all base class. Further the search moves to all super-classes and stops at the first place where attribute is found.

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

Ans – The basic distinguishable feature between class object and instance object is that a class is the ultimate blue print and the instance is the virtual copy of that blue print. Classes can be used to create multiple copies of instances. For ex – A blue print for house design is a "class" whereas a house built on that blue print is the "instance" of that blue print.
Classes also support operator overloading methods, which instances inherit, and treat any functions nested in the class as methods for processing instances.

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

Ans – It is special because it receives the instance object i.e., implied subject of the method call and is generally called 'safe' by convention. The calling process is automatic and the receiving process is not.

Q5. What is the purpose of the init method?

Ans – If 'init' method is present in any class then python inherently calls it everytime an instance of that class is created. So basically, it lets the class initialize the object's attributes and serves no other purpose apart from that.

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

Ans – Creating a class instance is as easy as calling the class name as if it is a function. It shows up as arguments two and beyond in the __init__ constructor method.

Q7. What is the process for creating a class?

Ans – As answered previously, a class is like an object constructor, like a blueprint to create an object. Syntax to create a class –

```
class name:
    x = Adarsh
a = name()
```

In the above code, we initiate the class by writing class and followed by the name which we would like to associate with the initiated class. Further, we insert some code depending on the requirements.

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

Ans – Super classes are defined as classes from which inheritance is derived from. They are also called as parent class. super() function in python makes the inheritance more manageable and extensible.

Q9. What is the relationship between classes and modules?

Ans – Classes are python types, which means that they can be instantiated whereas modules are static. This implies that data stored within a module exists only once. And a module doesn't need to be instantiated. In short, modules are files present inside a package, whereas a class is used to encapsulate data and functions together inside the same unit.

Q10. How do you make instances and classes?

Ans – To create an instance of a class, we call the class using its name and feed whatever arguments are required by its '__init__' method.

To create a class, we use the below syntax –

```
class name:
    x = Adarsh
a = name()
```

Q11. Where and how should be class attributes created?

Ans – Class attributes are created within the class and it doesn't require any 'self' parameter. Once created, they are shared across all the objects and instances but is owned by the class itself. Refer the below code to create class attribute –

```
class name:
    x = Adarsh
a = name()
```

In the above code, ‘x’ is an attribute of the class ‘name’.

Q12. Where and how are instance attributes created?

Ans – The unique thing about instance attributes are that they are not shared by objects. Every object has its own copy of instance attribute. We use two functions to list the attributes of an instance object – vars() and dir().

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

Ans – ‘Self’ is a parameter refers back to the current instance of the class. It is used to access variables, attributes and methods that belongs to the class.

Q14. How does a Python class handle operator overloading?

Ans – Operator overloading means that Python provides additional context to the operator beyond their predefined operational meaning. For ex – we can use ‘+’ operator to add two integers as well as two strings or even lists. This means that ‘+’ is overloaded with other classes.
Note that if a user tries two use ‘+’ operator on two user defined objects, then it’ll throw an error as python won’t know how to add to custom created objects.

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

Ans – Operator overloading allows us to provide an intuitive interface to users of our class, plus makes it possible for templates to work equally well with classes and built-in/intrinsic types. Operator overloading allows Python operators to have user-defined meanings on user-defined types (classes).

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

Ans – As described earlier, the most convenient and popular form of operator overloading can be considered as either ‘+’ (plus) or ‘ * ‘ (multiplication) operator. We can use the same operator to do multiple actions with integers, strings, list or even between classes as well.

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

Ans – Two most important concepts are ‘inheritance’ and ‘polymorphism’.

Q18. Describe three applications for exception processing.

Ans – Three applications of exception processing –

- Operating systems using exception handling to recover from crashes etc.
- Usage in arithmetic expression where division by zero
- Database connectivity using exception handling during situation of connectivity failure with the server

---

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

Ans – If the program is not written to handle the exception, then the program will be forced to terminate abruptly.

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

Ans – Different option to recover from exceptions are –

- Try and except statement – these statements are used to catch exceptions and then process them. Try clause is used to find exceptions and except clause is used to process/handle the exceptions.
- Try, except and else statement – This statement houses an extra else clause at the end and it is executed only when the try clause is unable to find any exception
- Finally keyword – This keyword is used with try and except and is always executed after them. The final block always executes after normal termination of try block or after try block terminates due to some exception.
- Manually raising exception – Python raise statement helps in enforcing any error manually. There is only one argument in this case and it is for the exception to be raised

---

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

Ans – Two methods of raising exception in script –

- Concrete exceptions
- OS exceptions

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

whether or not an exception exists.

Ans – Two methods –

- Using finally keyword
- Using throw keyword

Q23. What is the purpose of the try statement?

Ans – In short, the try statement allows you to define a block of code to be tested for errors while it is being executed. The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

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

Ans – Two most popular try statement variations are –

- try except statement: Syntax for this variation is mentioned below –

```
try:
    # Some Code
except:
    # Executed if error in the
    # try block
```

- try except else statement: Syntax for this variation is mentioned below –

```
try:
    # Some Code
except:
    # Executed if error in the
    # try block
else:
    # execute if no exception
```

Q25. What is the purpose of the raise statement?

Ans -- Python raise keyword is used to raise exceptions or errors. It raises an error and stops the control flow of the program and the syntax looks like –

```
raise [expression1[, expression2]]
```

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

Ans – Assert statement is used while debugging a python code. It helps in testing any condition within the code and if fulfilled then it returns True otherwise, it raises an assertion error.

```
name = "Adarsh"

#if condition returns False, AssertionError is raised:
assert x == "shubham", "name should be 'Adarsh'"
```

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

Ans – with/as argument is used generally in python exception handling to make the code cleaner and much more readable. It helps in replacing try-catch block with a concise block of code. Usage example can be seen below –

```
with open("example.txt", "w") as file:
    file.write("Hello World!")
```

Q28. What are *args, **kwargs?

Ans – *args and **kwargs are used to pass a variable number of arguments to a function in python. *args is used for non-keyword arguments and **kwargs is used for keyword arguments. Both these arguments are used when we are unsure on the number of arguments which needs to be passed down to the function.

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

Ans – To pass optional or keywords arguments from one function to another, arguments are collected using the * and ** in the function's parameter list. With this we can get the function's positional argument as tuple and keyword argument as dictionary and then from here these can be passed to another function using * and **.

Q30. What are Lambda Functions?

Ans – Lambda functions are small anonymous function which can take in multiple parameters but solves only one expression. Example – Below mentioned code snippet defines lambda function which adds 10 to any given number.

```
x = lambda a : a + 10
print(x(5))
```

Q31. Explain Inheritance in Python with an example?

Ans – Inheritance is when a class in python is based on another class or classes, getting the same method and attributes from them. Ex –

```
# define a superclass
class super_class:
    # attributes and method definition

# inheritance
class sub_class(super_class):
    # attributes and method of super_class
    # attributes and method of sub_class
```

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 – In python, search order for attributes and methods inside of parent's parenthesis is left-to-right so the first left superclass gets the priority. In above case, both class A and B have their own versions of method func() so when an object from class C is called then, class A is invoked.

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

Ans – To determine the type of instance and inheritance we use mainly two functions – isinstance() and issubclass() respectively.

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

Ans – Python nonlocal keyword is used to reference a variable inside nested functions, where the variable should not belong to the inner function.

Q35. What is the global keyword?

Ans – Similar to nonlocal, global keyword is used to reference a variable globally. Which means once declared inside a function, it can be used anywhere in the code.