ASSESSMENT 3

Python OOP Assignment

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

Ans - It allows us to develop applications using an Object-Oriented approach. In Python, we can easily create and use classes and objects. An object-oriented paradigm is to design the program using classes and objects. The object is related to real-word entities such as book, house, pencil, etc.

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

1)	Object is an instance of a class.	Class is a blueprint or template from which objects are created.
2)	Object is a real world entity such as a pen, laptop, mobile, bed, keyboard, mouse, chair etc.	Class is a group of similar objects.
3)	Object is a physical entity.	Class is a logical entity.
4)	Object is created through new keyword mainly e.g.	Class is declared using class keyword e.g.

	Student s1=new Student();	class Student{}
5)	Object is created many times as per requirement.	Class is declared once.
6)	Object allocates memory when it is created.	Class doesn't allocated memory when it is created.
7)	There are many ways to create object in java such as new keyword, newInstance() method, clone() method, factory method and deserialization.	There is only one way to define class in java using class keyword

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

Ans - The calling process is automatic while the receiving process is not (its explicit). This is the reason the first parameter of a function in class must be the object itself

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?

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?

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

```
1. class ClassName:
2.     #statement_suite
```

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, also called heir class or child class. Superclasses are sometimes called ancestors as well. There exists a hierarchical relationship between classes.

Q9. What is the relationship between classes and modules?

Ans - class is used to define a blueprint for a given object, whereas a module is used to reuse a given piece of code inside another program.

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.
- 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.

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.

Defined inside a constructor using the self parameter.

Shared Specific to object.

Accessed using object dot notation e.g. object.instance_attribute

Changing the value of the instance attribute will not be reflected on other objects.

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.

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 - Ensures that objects of a class behave consistently with built-in types and other user-defined types. Makes it simpler to write code, especially for complex data types. Allows for code reuse by implementing one operator method and using it for other operators.

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

Ans - The most frequent instance is the adding up operator '+', where it can be used for the usual addition and also for combining two different strings. As mentioned on top, the plus symbol's practice in dissimilar forms is the largest classic example of the operator level overloading process

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.

Ans - Exception handling is managed by the followings

1. try
2. catch
3. finally
4. throw

Python Try Statement

A try statement includes keyword try, followed by a colon (:) and a suite of code in which exceptions may occur. It has one or more clauses.
During the execution of the try statement, if no exceptions occurred then, the interpreter ignores the exception handlers for that specific try statement.
In case, if any exception occurs in a try suite, the try suite expires and program control transfers to the matching except handler following the try suite.
Syntax:

try:

statement(s)

The catch Statement

Catch blocks take one argument at a time, which is the type of exception that it is likely to catch. These arguments may range from a specific type of exception which can be varied to a catch-all category of exceptions.
Rules for catch block:

- You can define a catch block by using the keyword catch
- Catch Exception parameter is always enclosed in parentheses
- It always represents the type of exception that catch block handles.
- An exception handling code is written between two {} curly braces.
- You can place multiple catch block within a single try block.
- You can use a catch block only after the try block.
- All the catch block should be ordered from subclass to superclass exception.

EX - try
}
catch (ArrayIndexOutOfBoundsException e) {
System.err.printin("Caught first " + e.getMessage()); } catch (IOException e) {
System.err.printin("Caught second " + e.getMessage());
}

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.

```
try:
   raise KeyboardInterrupt
```

```
finally:
   print 'welcome, world!'
```
Output
Welcome, world!
KeyboardInterrupt

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 - To handle the exception, we have put the code, result = numerator/denominator inside the try block. Now when an exception occurs, the rest of the code inside the try block is skipped. The except block catches the exception and statements inside the except block are executed.

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 - Exception handling allows you to separate error-handling code from normal code.

- An exception is a Python object which represents an error.
- As with code comments, exceptions helps you to remind yourself of what the program expects.
- It clarifies the code and enhances readability.
- Allows you to stimulate consequences as the error-handling takes place at one place and in one manner.
- An exception is a convenient method for handling error messages.
- In Python, you can raise an exception in the program by using the raise exception method.
- Raising an exception helps you to break the current code execution and returns the exception back to expection until it is handled.
- Processing exceptions for components which can’t handle them directly.

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.

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

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.

Q25. 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.

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

Ans - Python's assert statement allows you to write sanity checks in your code. These checks are known as assertions, and you can use them to test if certain assumptions remain true while you're developing your code. If any of your assertions turn false, then you have a bug in your code.

Q27. 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. A function or class that supports the with statement is known as a context manager.

Q28. What are *args, **kwargs?

The special syntax **args* in function definitions in Python is used to pass a variable number of arguments to a function. It is used to pass a non-keyworded, variable-length argument list.

EX ; def myFun(*argv):

```
    for arg in argv:
        print(arg)
```

myFun('Hello', 'Welcome', 'to', 'Ineuron')

OUT PUT - Hello

Welcome

to

Ineuron

The special syntax ***kwargs* in function definitions in Python is used to pass a keyworded, variable-length argument list. We use the name *kwargs* with the double star. The reason is that the double star allows us to pass through keyword arguments (and any number of them).

Ex - def myFun(**kwargs):

```
    for key, value in kwargs.items():

        print("%s == %s" % (key, value))
```

Driver code

myFun(first='Raj', mid='Gowda', last='CS')

Out put - first == Raj

mid == Gowda

last == CS

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

Ans - Gather the arguments using the * and ** specifiers in the function's parameter list. It gives us positional arguments as a tuple and the keyword arguments as a dictionary. Then we can pass these arguments while calling another function by using

and **

```
 def funl (a, *tup, ** keyword.Arg):
```

keyword Argj'width'j = '23.3c'

fun2 (a, *tup, **keywordArg)

Q30. What are Lambda Functions?

Ans - lambda is a keyword in Python for defining the anonymous function. argument(s) is a placeholder, that is a variable that will be used to hold the value you want to pass into the function expression

Ex:

str1 = 'Rajegowda'

```
# lambda returns a function object
rev_upper = lambda string: string.upper()[::-1]
print(rev_upper(str1))
```

Output - adwogejar

Q31. 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.

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 - multiple inheritance

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

- 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 .

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 - Global keyword is used to modify the global variable outside its current scope and meaning. It is used to make changes in the global variable in a local context. The keyword 'Global' is also used to create or declare a global variable inside a function