# **OOPS ASSIGNMENT**

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

Ans. Python’s OOP is basically an approach to designing a single unit program which has both ’attributes’ and ‘behavior’ of an individual object. Where attributes store the value of an object and Behavior is a method that shows the properties of a particular object.

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

Ans. The inheritance search looks for an attribute first in the instance object and then in the class from which it was created and after that, it looks in other superclasses and stops where it finds the attribute.

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

Ans. Using a python class you can create a single class object and multiple instance objects. And once you have created a class then the class object is automatically created and uh don’t need to do anything for it, We can have static variables under it and we can access them through class name. Now an instance object is created using class name with parenthesis() and once you have created an instance object then init function is automatically called if you have created an init function in class.

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

Ans. The first argument in a class’s method function is ‘Self’ and it is mandatorily present as an first argument if you have created an init function in a class. This self is basically the variable that we provide to our instance object while creating it, we are not suppose to pass all the arguments which init function had along with self, we just need other arguments except self. For ex. This is an instance object of class test; T1=test(a,b) then T1 is the self argument itself.

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

Ans. init function is basically a constructor that is automatically called by the class while creating an instance object.

For ex.

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

Self.emp\_name=name

Self.emp\_salary=salary

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

Ans. A class instance can be created simply by writing the class name with parenthesis inside which arguments will lie apart from self. For ex. Test class; Test(a,b)

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

Ans. A class is created by simply writing Class name: #and writing some code here.

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

Ans. If the class has inherited some properties from any other class then that class from which it has inherited will be called as the superclass or superclasses if more than one is there.

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

Ans. A module is a set of methods or functions that can be used anywhere as it is. A module can have zero, one, or multiple classes in it; While a class is something that has functions, attributes, and instances in it. It has one or multiple modules in it. Like math, the module has all the logic and algorithms in it so if we want to create a class of something related to math then we can use this module there without writing the complete logic and code again.

**Q10. How do you make instances and classes?**

Ans. When you create a class you simply write- class name: # some code is here

While creating instances you use the class name with the arguments and functions.

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

Ans. Class attributes are variables defined directly through the class and it’s a globally shared variable by all the objects of the class, unlike instance variables.

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

Ans. Instance attributes are the variables that are defined inside the constructor and they are tightly coupled with the particular object memory location and it can not be called explicitly anywhere. Inside a constructor init function is introduced in which we passes arguments and with the help of that later we create instance variables.

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

Ans. Self is basically a pointer to the object itself or we can say that it’s the memory address that is being passed.

**Q14. How does a Python class handle operator overloading?**

Ans. To handle the overloading in python we can use the Dunder methods or we can say mapper operators. Like ‘add’ for adding two strings/numbers.

For ex. def \_\_add\_\_(self,others):

Return self.salary+other.other

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

Ans. When we have two objects and we want to add them with + this operator then it will show you an error on directly adding two objects with ‘ + ‘. So at that time, we can use the operator overloading in our classes so that we can add even our objects as well.

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

Ans. The most popular form of operator overloading is the ‘addition’ one, as using + this symbol you can add two numbers and concatenate two strings as well.

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

Ans. The first one is Inheritance and the other one could be polymorphism. As one helps us in overloading and overriding while the other one help us in inheriting the properties of one class into the other one.

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

Ans. Three Applications for exception Processing are-

1. Dividing any number with ‘0’; ZERODIVISIONERROR
2. Printing any number from the list which is out of the range; INDEXERROR
3. Printing any key from the dictionary which does not exist; KEYERROR

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

Ans. It gives you an error and the code will stop running.

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

Ans. The options for recovering from an exception are-

1. Try: except method
2. Use of else block
3. Use of finally keyword

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

Ans. Two Methods for triggering exceptions in your script are:

1. **Try**– This method catches the exceptions raised by the program

Basically the error will be shown if any error occurs but the program will run till end here. For ex:

a=input(int(‘enter a number’))

try:

1. **Raise**– Triggers an exception manually using custom exceptions. We can use raise keyword to show the errors. It is a manual process wherein you can optionally pass values to the exception to clarify the reason why it was raised.

**Ex:** Number=input(int(‘Enter the number between 5 and 9’))

If (number < 5 or number>9):

raise ValueError(‘Value should be between 5 and 9’)

**Q22. Identify two methods for specifying actions to be executed at termination time, regardless of whether or not an exception exists.**

Ans. The methods can be the “Else block” and the “finally keyword” method. In these methods the else block will be printed irrespective of the error and similarly finally keyword block will also print itself irrespective of the exception.

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

Ans. The purpose of try statement is to execute the code if the values entered by the user are correct and if not then it passes the interpreter to except block to print the error.

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

Ans. The two most popular try statement variations are –

1. one with try/except/Else statement.
2. And the other one is with try/except/finally statement

**Q25. What is the purpose of the raise statement?**

Ans. The purpose of the raise statement is to give the error to the user if the entered values are not appropriate and stop the code there only. So that it does not waste the time and memory to execute the code for any wrong input value.

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

Ans. It is to check if the logical expression is true or false. And if the expression is true it will execute it but if it’s not then it will through you an assertive error.

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

Ans. The Purpose of the with statement is to make sure that whatever resource we are using should be closed after using it. For example, if you are opening any file with the ‘with’ statement then it will get automatically closed once the work/function is done. The with statement is a replacement for try/finally error-handling statements.

For ex:

with open(file name, “w”) as file:

File.write (‘hello world’)

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

Ans. \*args and \*\*kwargs are the arguments given in the function so that we can take or increase the number of elements in our list or tuple irrespective of the numbers. We can add or subtract the number of elements from our list at any time using the \*arg argument.

For example:

def students(\*args, \*\*kwargs):

For items in args:

print(items)

for key, values in kwargs:

print(f”{Key} is a {Value}”)

Varry= [‘harry’, ‘marry’, ‘carry’]

Kw = {‘Harry’: ’coder’, ‘marry’: ‘writer’, ‘carry’ : ‘actor’}

students(\*Varry,\*\*kw)

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

Ans. We can pass a parameter or function in a function by calling the one function in another function.

For example:

def great(brand):

Print(“Gucci is a great brand” + brand())

def profitable():

return ‘ and also a profitable one! ‘

great(profitable)

**Q30. What are Lambda Functions?**

Ans. It is used to write anonymous functions or inline functions, here don’t need def keyword and only logical expression is required.

Syntax:- lambda argument : expression

Ex:-

lambda x : x+5

add\_number= lambda x : x+ 5

y=10

print(add\_number(y))

o/p: 15

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

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

For ex:

Class parent:

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

self.name=name

def displayname(self):

print(self.name)

def isemployed(self):

print(self.name, “is un-employed”)

Class child(parent):

def isemployed(self):

print (self.name, ”is employed”)

**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. Versions of A get invoked first.

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

Ans. Python has two built-in functions that work with inheritance:

* Use [isinstance()](https://docs.python.org/3/library/functions.html#isinstance) to check an instance’s type: isinstance(obj, int) will be True only if obj.\_\_class\_\_ is [int](https://docs.python.org/3/library/functions.html#int) or some class derived from [int](https://docs.python.org/3/library/functions.html#int).
* Use [issubclass()](https://docs.python.org/3/library/functions.html#issubclass) to check class inheritance: issubclass(bool, int) is True since [bool](https://docs.python.org/3/library/functions.html#bool) is a subclass of [int](https://docs.python.org/3/library/functions.html#int). However, issubclass(float, int) is False since [float](https://docs.python.org/3/library/functions.html#float) is not a subclass of [int](https://docs.python.org/3/library/functions.html#int).

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

Ans. The nonlocal keyword is also known as the enclosed variable. And it occurs when we have nested functions; so the variable outside the nested function is a nonlocal variable and inside it will be a local variable. So to specify the nonlocal variable we use the keyword nonlocal so that our interpreter doesn’t get confused between global, local, and nonlocal variables.

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

Ans. A global keyword is used to define a global variable inside the function so that the interpreter can understand which is supposed to be used as global and which as local variable. And this global keyword will exist till the code ends i.e it’s not limited to any function or any class.