**FSDS MAY BATCH 2022(Python Assignment -2)**

**Submitted by: Shubham Tiwari**

Q1. What is the relationship between classes and modules?

Ans: A module in python programming language is basically a way to organize the code, and it contains either python classes or just functions,it can basically have zero, one or multiple classes whereas, class is an abstraction that stores data (characteristics) and method (behaviour) definitions to represent a specific type of objects. It is a way to organize our codes **logically.It can have one or more module files.**

Q2. How do you make instances and classes?

Ans: To create instances of a class, we basically call the class using class name and pass in whatever arguments its *\_\_*init*\_\_* method accepts and for creating classes we just write class keyword followed by class name such as

Class **Students**

Q3: Where and how should be class attributes created?

Ans:Class attributes are those which are derived from the class itself and are same for every object and are used outside the \_\_init\_\_function..For example:

class Railway:

Passenger = "indianrailways.org"

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

self.name = name

self.ticket = ticket

Passenger 1 = Passenger ("Shubham", "C++")

Passenger 2 = Passenger ("Ravi", "Python")

print(Passenger 1.name) # Shubham

print(Passenger 2.name) # Ravi

Here in this example the Passenger variable acts as a **class attribute** while name and ticket are instance attributes.

Q4: Where and how are instance attributes created?

Ans:They are basically defined under constructor using “self” parameter ,the most important feature of it that changing the value of the instant attribute will not affect other objects i.e it is somewhat dynamic in nature .The example which I already mentioned in the Q3,Let us break that to understand instant attribute:

We created two objects from the Railway class i.e Passenger1 and Passenger2. Each of these objects, by default, will have all the variables created in the class. But each object is able to have its own name and course variable because they were created in the \_\_init\_\_() function ,but when we print it both gave us the same value because the passenger variable is a class attribute and name and ticket are instant attribute,in other words we can say that instant attributes can have different values in each object.

Q5: What does the term self in a Python class mean?

Ans: By using self parameter we can easily means access the variables which are assigned to that particular class and it is not necessary at all to call it always “self” we can name it whatever we like, but it has to be the first parameter of any function in the class.

Example:

class Colours():

# init method or constructor

def \_\_init\_\_(self, red, white):

self.red = red

self.white = white

def show(self):

print("red is", self.red)

print("white is", self.white )

Camel = Colours("Camel", "brown")

Crayons = Colours(“Crayons", "yellow")

Camel.show()

Crayons.show()

**Output**

red is Camel

white is Brown

red is Crayons

white is yellow

Q6:How does a Python class handle operator overloading?

Ans:In Python, operator overloading is must with the user defined data type since the compiler does not know which variable of user defined data type we need to add,multiply or compare. Overloading is achieved by overriding the method which is specifically for that operator, in the user-defined class.

For example \_add\_(self,x), is for overloading (+)operator and\_eq\_(self,x) is for overloading (==).

Q7: 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](http://docs.python.org/library/collections.html#module-collections) rely on the presence of certain special methods and special methods, one with names starting and ending with double underscores.

Q8: What is the most popular form of operator overloading?

Ans:There are basically two types which are most popular i.e. addition(+)operator overloading and equal(==)operator overloading.

Q9: What are the two most important concepts to grasp in order to comprehend Python OOP code?

Ans:Although there are 4 basic methods to comprehend Python OOP code but two most important concepts are **inheritance** and **polymorphism.**

Inheritance means inherit characteristics or features from other classes For example : A child is inherited from his father.

Polymorphism means “many”i.e it can perform different operations in different condtions.