**Python Advance Assignment-19**

**1. How would you describe the relationship of a class to its instances? For example, is it a one-to-one or a one-to-many relationship?**

Ans.

A instance of a class is refer to an object of a class through which we refer all the class class attributes , class methods and everything which comes inside the class.

An instace of a class is an entity which refer to a class. It symbolize the class. So we can say that it one to many relationship kind of thing.

For example, we have an iphone. It is an object as the person who made that phone has refer its name as Iphone thats why we call it an Iphone. BUt at the sam etime that Ipihone is beong to some smartphone class. So we can say that it is an instance of a smartphone class.

Moreover instance and object are the same thing. Creatig an class object means creating an instance of a class, through which we refer to all those things which come under the class

**2. What kind of information is held only in an instance?**

Ans.

It store the values of the class attributes. When we give the attributes in the **init**function as an argument then the values which are stored in the object will get passed to it and attributes gets initialised and later on init function passed these values to the class

**3. What information is held in a class?**

Ans.

A class contains many information. It contains all the information about the attributes, methods function,etc. All kind of behaviours which class have in the form of methods and attributes are held inside the class.

**4. What exactly is a method, and precisely how does it differ from a standard function?**

Ans

Amethod is refer to the function only but a difference is that, a method can be perfom inside the class by using the class object. It cannot be perform outside the class.

While a standard function is a function which can be access outside any where. Working of both the method and std function is same but there is a difference in accessability.

**5. Does Python support inheritance, and, if so, what is the syntax?**

Ans.

Yes python do support inheritence in Oops concept. It is possible in python to inherit the behaviour and properties of some other class so as to avoid writing the same code again and again when it is required. We can even do the multiple inheritence. Meaning we can inherit the property of more than onw class into a single class also. Order is matter Followig is the syntax:

In [11]:

**class** A:

**pass**

**class** B(A):

**pass**

**class** C(B):

**pass**

**class** D(B,A):

**pass**

**6. To what extent does Python support encapsulation (making instance or class variables private)?**

Ans.

Python supports encapsulation. It combine all the behaviour, properties and attributes of a class inside a single class object. So that we do not have to perform separate separate operation through different means. We just need a single object to perfomr all those task. It also hide the implementation of the task. Means we do ot required to focus on the backend task. We just need to foucs on the output, result when any operation si performed.

If we talk about making instance or class variables private, then yes. It makes them both of them private so that there is not chances of changing or modifying the code or some values. So we just make those attributes and methods private which have a severe impact on the codes, so that any other person can figure it it out by just looking into it.

**7. What precisely is the difference between a class variable and an instance variable?**

Ans.

**Class variable** It is a variable which is declare below the class and outside the **init** function, where as, **instance variable** are declared inside the init function.

**Class variable** need a class name to get called as well as it can be called by the class instance name also, where as a **instance variable** can be called by an object name but not with the class name.

**Class variable\*** is Global to all the objects of a class means it is shared accross every object, while, instance variable is only for th eunique data values which are stored by the class instance.

**Class variable** could not override the instance variable but instance variable can override the class variable

Following are some examples

In [50]:

**class** a:

x**=**10

**def** \_\_init\_\_(self,var1):

self**.**var1**=**var1

**class** b(a):

print(a**.**x)

10

In [45]:

oa**=**a(20)

ob**=**b(30)

In [46]:

oa**.**x

Out[46]:

10

In [47]:

a**.**x**=**50

In [48]:

a**.**x

Out[48]:

50

In [49]:

oa**.**x

Out[49]:

50

**8. Within a class’s method definitions, when does self need to be used, if at all?**

Ans.

First of all self is nothing but an object pointer which is pointing to object itself. We need an object namespace to call the func/method which is define inside the class. So that is why there is a pointer 'self' which is refering to an class object pointer. With this **self.method** we can use the method

In [51]:

int**.**\_\_add\_\_(9,7)

Out[51]:

16

**9. What is the difference between the \_ *add* *and the radd* \_ methods?**

Ans.

**add**is simply an built method which add the two items when it is pased, whose class is having this method. Like integer and string class have this method and in case of integer this method will add up the two values and in case of string class this will concatenate the two items.

**radd**This is inside the integer class but not in string class. This also add up the two number. But the the difference is, **radd**is only called if the left object does not have an **add** method, or that method does not know how to add the two objects. If both the classes have **add**method then this will not called.

**10. When do you really need a reflection method? When do you not need it, even though you support the operation in question?**

Ans.

Reflection is a facility where you can query an object about its attributes at runtime. Meaning, we can modify the object variable, create new class atrributes during the run without initialising inside the constructor. It is then display by invoking it as an instance variable. This is an reflection facility. This is needed when we have sudden requirement of any new attributes that has to work with the class.

**11. What is the \_ *iadd* \_ method called?**

Ans.

It is an **inplace add** method which is an built in method. Like that of add , it also adds the two value but unlike \_ *add*\_ , it stores that value which was calculated.

**iadd**is actually storing value its adding into the instance variable and return the new value

**12. Is the \_ *init* \_ method inherited by subclasses? What do you do if you need to customize its behavior within a subclass?**

Ans.

**init** method is basically used for constructing the object variables meaning, initialisation of object variables happen, where the class attributes gets the values from the class instance where the values are stored. So when we want to inherit the attributes/variables of the class then we need to inherit the init method also. This implies that whatever the variables were initialise in the parent class, those will also get initialise in the subclass. We uses **super()** method in order to call the parent class.

We can customise the behaviour of the subclass by declaring some new other attributes and initialising them. However these new attributes will only get accessed by the subclass object. We could use **overridding** technique also for changing the behaviour of the init method without defining it separately inside the subclass