1. Make a class called Thing with no contents and print it. Then, create an object called example from this class and also print it. Are the printed values the same or different?

Ans => yes both are same

*class* Thing():

    pass

print(Thing())

example = Thing()

print(example)

2. Create a new class called Thing2 and add the value 'abc' to the letters class attribute. Letters should be printed.

Ans =>

*class* Thing2():

*def* \_\_init\_\_(*self*):

*self*.letters = 'abc'

print(Thing2().letters)

3. Make yet another class called, of course, Thing3. This time, assign the value 'xyz' to an instance (object) attribute called letters. Print letters. Do you need to make an object from the class to do this?

Ans =>

*class* Thing3():

    pass

obj = Thing3()

obj.letters = 'xyz'

print(obj.letters)

4. Create an Element class with the instance attributes name, symbol, and number. Create a class object with the values 'Hydrogen,' 'H,' and 1.

Ans =>

*class* Element():

    pass

obj = Element()

obj.name = 'Hydrogen'

obj.symbol = 'H'

obj.number = '1'

print(obj.name)

5. Make a dictionary with these keys and values: 'name': 'Hydrogen', 'symbol': 'H', 'number': 1. Then, create an object called hydrogen from class Element using this dictionary.

Ans =>

*class* Element():

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

        for k, v in *dictionary*.items():

            setattr(*self*, k, v)

dictionary = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

hydrogen = Element(dictionary)

print(hydrogen.name)

6. For the Element class, define a method called dump() that prints the values of the object’s attributes (name, symbol, and number). Create the hydrogen object from this new definition and use dump() to print its attributes.

Ans =>

*class* Element():

*def* dump(*self*, *dictionary*):

        for k, v in *dictionary*.items():

            print(v)

dictionary = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

hydrogen = Element()

print(hydrogen.dump(dictionary))

7. Call print(hydrogen). In the definition of Element, change the name of method dump to \_\_str\_\_, create a new hydrogen object, and call print(hydrogen) again.

Ans =>

*class* Element():

*def* \_\_str\_\_(*self*):

        pass

hydrogen = Element()

print(hydrogen)

8. Modify Element to make the attributes name, symbol, and number private. Define a getter property for each to return its value.

Ans=>

dictionary = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

*class* Element:

*def* \_\_init\_\_(*self*, *name*='Hydrogen',*symbol*='H',*number*=1):

*self*.name = *name*

*self*.symbol = *symbol*

*self*.number = *number*

*def* get\_name(*self*):

        return *self*.name

*def* get\_symbol(*self*):

        return *self*.symbol

*def* get\_number(*self*):

        return *self*.number

hydrogen = Element()

print(hydrogen.get\_name())

9. Define three classes: Bear, Rabbit, and Octothorpe. For each, define only one method: eats(). This should return 'berries' (Bear), 'clover' (Rabbit), or 'campers' (Octothorpe). Create one object from each and print what it eats.

Ans =>

*class* Bear:

*def* eats(*self*):

         return 'berries'

*class* Rabbit:

*def* eats(*self*):

         return 'clover'

*class* Octothorpe:

*def* eats(*self*):

         return 'campers'

bearObj = Bear()

rabbitObj = Rabbit()

octothorpeObj = Octothorpe()

print(bearObj.eats()+' (Bear)')

print(rabbitObj.eats()+' (Rabbit)')

print(octothorpeObj.eats()+' (Octothorpe)')

10. Define these classes: Laser, Claw, and SmartPhone. Each has only one method: does(). This returns 'disintegrate' (Laser), 'crush' (Claw), or 'ring' (SmartPhone). Then, define the class Robot that has one instance (object) of each of these. Define a does() method for the Robot that prints what its component objects do.

Ans=>

class Laser:

def does(self):

return 'disintegrate'

class Claw:

def does(self):

return 'crush'

class SmartPhone:

def does(self):

return 'ring'

obj1 = Laser()

obj2 = Claw()

obj3 = SmartPhone()

print(obj1.does()+' (Laser)')

print(obj2.does()+' (Claw)')

print(obj3.does()+' (SmartPhone)')

class Robot():

def \_\_init\_\_(self,obj1,obj2,obj3):

self.obj1 = obj1

self.obj2 = obj2

self.obj3 = obj3

def does(self):

return self.obj1

obj4 = Robot(obj1,obj2,obj3)

print(obj4.does())