**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?**

Answer 1:

#Define Thing class

class Thing:

pass

#Print Thing class

print(Thing)

#Create an object example from Thing class

example = Thing()

#Print example object

print(example)

The printed values are different.

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

Answer 2:

#Define Thing2 class

class Thing2:

letters = 'abc'

#Print letters from Thing2 class

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?**

Answer 3:

#Define Thing3 class

class Thing3:

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

self.letters = 'xyz'

#Create an object from Thing3 class

obj\_thing3 = Thing3()

#Print letters from the object

print(obj\_thing3.letters)

You need to make an object from the class to print 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.**

Answer 4:

#Define Element class

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

#Create an object from Element class

element\_object = Element('Hydrogen', 'H', 1)

**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.**

Answer 5:

#Create a dictionary

element\_dict = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

#Create an object hydrogen from Element using the dictionary

hydrogen = Element(\*\*element\_dict)

**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.**

Answer 6:

#Define dump method in Element class

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

def dump(self):

print(f'Name: {self.name}, Symbol: {self.symbol}, Number: {self.number}')

#Create hydrogen object from the updated Element class

hydrogen = Element('Hydrogen', 'H', 1)

#Use dump() to print its attributes

hydrogen.dump()

**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.**

Answer 7:

#Modify Element class with \_\_str\_\_ method

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

def \_\_str\_\_(self):

return f'Name: {self.name}, Symbol: {self.symbol}, Number: {self.number}'

#Create a new hydrogen object

hydrogen = Element('Hydrogen', 'H', 1)

#Print hydrogen using print()

print(hydrogen)

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

Answer 8:

#Modify Element class to make attributes private with getter properties

class Element:

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

self.\_name = name

self.\_symbol = symbol

self.\_number = number

@property

def name(self):

return self.\_name

@property

def symbol(self):

return self.\_symbol

@property

def number(self):

return self.\_number

**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.**

Answer 9:

#Define Bear, Rabbit, and Octothorpe classes

class Bear:

def eats(self):

return 'berries'

class Rabbit:

def eats(self):

return 'clover'

class Octothorpe:

def eats(self):

return 'campers'

#Create objects and print what they eat

bear = Bear()

rabbit = Rabbit()

octothorpe = Octothorpe()

print(bear.eats()) # Output: berries

print(rabbit.eats()) # Output: clover

print(octothorpe.eats()) # Output: campers

**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.**

Answer 10:

#Define Laser, Claw, and SmartPhone classes

class Laser:

def does(self):

return 'disintegrate'

class Claw:

def does(self):

return 'crush'

class SmartPhone:

def does(self):

return 'ring'

#Define Robot class with does() method

class Robot:

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

self.laser = Laser()

self.claw = Claw()

self.smartphone = SmartPhone()

def does(self):

print(f'Laser: {self.laser.does()}, Claw: {self.claw.does()}, Smartphone: {self.smartphone.does()}')

#Create Robot object and print what its components do

robot = Robot()

robot.does()