**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 - 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:

letters = 'abc'

print(Thing2.letters)

abc

**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:

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

self.letters = 'xyz'

print(Thing3().letters)

xyz

**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:

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

self.symbol = symbol

element = Element("H")

# Accessing and printing the instance attribute

print(element.symbol)

H

**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, name, symbol, number):

self.name = name

self.symbol = symbol

self.number = number

element\_dict = {

'name': 'Hydrogen',

'symbol': 'H',

'number': 1

}

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

print(hydrogen.name)

print(hydrogen.symbol)

print(hydrogen.number)

**Output**

Hydrogen

H

1

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

self.name = name

self.symbol = symbol

self.number = number

def dump(self):

print("Name:", self.name)

print("Symbol:", self.symbol)

print("Number:", self.number)

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

hydrogen.dump()

Name: Hydrogen

Symbol: H

Number:1

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

Ans -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}\nSymbol: {self.symbol}\nNumber: {self.number}"

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

print(hydrogen)

Name: Hydrogen

Symbol: H

Number: 1

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

Ans- 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

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

print(hydrogen.name)

print(hydrogen.symbol)

print(hydrogen.number)

Hydrogen

H

1

**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'

# creating object and printing what they eat

bear = Bear()

rabbit = Rabbit()

octothorpe = Octothorpe()

print(bear.eats())

print(rabbit.eats())

print(octothorpe.eats())

berries

clover

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.**

Ans - class Laser:

def does(self):

return 'disintegrate'

class Claw:

def does(self):

return 'crush'

class SmartPhone:

def does(self):

return 'ring'

class Robot:

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

self.laser = Laser()

self.claw = Claw()

self.smartphone = SmartPhone()

def does(self):

print("Laser:", self.laser.does())

print("Claw:", self.claw.does())

print("SmartPhone:", self.smartphone.does())

# craeting a robot object and calling its does() method

robot = Robot()

robot.does()

Laser: disintegrate

Claw: crush

SmartPhone: ring