**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:**

let's create the Thing class and then create an object example from this class and print both the class and the object:class Thing:

thing\_instance = Thing()

example\_instance = Thing()

print(Thing)

print(thing\_instance)

print(example\_instance)

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

**Answer:**

class Thing2:

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?

class Thing3:

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

self.letters = 'xyz'

thing3\_instance = Thing3()

print(thing3\_instance.letters)

we define the Thing3 class with an instance attribute letters initialized to the string 'xyz' inside the constructor (\_\_init\_\_ method) of the class. Then, we create an object thing3\_instance from the Thing3 class using Thing3

**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:**

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

# Create a class object with the values 'Hydrogen,' 'H,' and 1

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

# Printing the instance attributes of the object

print(f"Name: {hydrogen.name}")

print(f"Symbol: {hydrogen.symbol}")

print(f"Number: {hydrogen.number}")

**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:**

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

# Create a dictionary with keys and values

element\_data = {

'name': 'Hydrogen',

'symbol': 'H',

'number': 1

}

# Create an object 'hydrogen' from the Element class using the dictionary

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

# Printing the instance attributes of the object

print(f"Name: {hydrogen.name}")

print(f"Symbol: {hydrogen.symbol}")

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.

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}")

print(f"Symbol: {self.symbol}")

print(f"Number: {self.number}")

# Create an object 'hydrogen' from the Element class using the dictionary

element\_data = {

'name': 'Hydrogen',

'symbol': 'H',

'number': 1

}

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

# Use the dump() method to print the attributes of the hydrogen object

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:**

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 an object 'hydrogen' from the Element class using the dictionary

element\_data = {

'name': 'Hydrogen',

'symbol': 'H',

'number': 1

}

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

# Call print(hydrogen) before renaming the method

print(hydrogen)

# Rename the method to \_\_str\_\_

# Now, when we create a new hydrogen object and call 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:**

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 an object 'hydrogen' from the Element class using the dictionary

element\_data = {

'name': 'Hydrogen',

'symbol': 'H',

'number': 1

}

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

# Call print(hydrogen) before renaming the method

print(hydrogen)

# Rename the method to \_\_str\_\_

# Now, when we create a new hydrogen object and call print(hydrogen),

# it will automatically use the \_\_str\_\_ method to print the object's attributes.

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

element\_data\_2 = {

'name': 'Hydrogen',

'symbol': 'H',

'number': 1

}

hydrogen\_2 = Element(\*\*element\_data\_2)

# Call print(hydrogen) again, and it will use the \_\_str\_\_ method to print the object's attributes.

print(hydrogen\_2)

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:**

class Bear:

def eats(self):

return 'berries'

class Rabbit:

def eats(self):

return 'clover'

class Octothorpe:

def eats(self):

return 'campers'

# Create objects from each class

bear = Bear()

rabbit = Rabbit()

octothorpe = Octothorpe()

# Print what each object eats

print(f"Bear eats: {bear.eats()}")

print(f"Rabbit eats: {rabbit.eats()}")

print(f"Octothorpe eats: {octothorpe.eats()}")

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.