**Assignment-15**

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?

A) The printed values are indeed different, but they both represent instances of the Thing class. The specific memory addresses (usually displayed as 0x...) will be different because each instance is a separate object. It's like having two identical-looking houses, but they are located at different addresses.

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

A) class Thing2:

letters = 'abc'

print(Thing2.letters)

abc

letters class attribute is now part of the Thing2 class, and it printed successfully!

1. 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?

A) class Thing3:

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

self.letters = 'xyz'

# Create an object (instance) of Thing3

example = Thing3()

# Print the letters attribute

print(example.letters)

xyz

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

A) class Element:

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

self.name = name

self.symbol = symbol

self.number = number

print(hydrogen.name)

print(hydrogen.symbol)

print(hydrogen.number)

# Create an object with values 'Hydrogen,' 'H,' and 1

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

Hydrogen

H

1

1. 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.

A) element\_data = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

# Create an object using the dictionary

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

print(hydrogen.name)

print(hydrogen.symbol)

print(hydrogen.number)

Hydrogen

H

1

1. 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.

A) 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 the hydrogen object

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

# Use the dump method to print its attributes

hydrogen.dump()

Name: Hydrogen, Symbol: H, Number: 1

1. 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.

A) 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)

# Call print(hydrogen)

print(hydrogen)

Name: Hydrogen, Symbol: H, Number: 1

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

A) 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

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

return f"Name: {self.\_\_name}, Symbol: {self.\_\_symbol}, Number: {self.\_\_number}"

# Create a new hydrogen object

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

# Access attributes using the getter properties

print(hydrogen.name)

print(hydrogen.symbol)

print(hydrogen.number)

# Call print(hydrogen)

print(hydrogen)

1. 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.

A) class Bear:

def eats(self):

return 'berries'

class Rabbit:

def eats(self):

return 'clover'

class Octothorpe:

def eats(self):

return 'campers'

bear = Bear()

rabbit = Rabbit()

octothorpe = Octothorpe()

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

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

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

Bear eats: berries

Rabbit eats: clover

Octothorpe eats: campers

1. 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.

A) class Laser:

def does(self):

return 'disintegrate'

class Claw:

def does(self):

return 'crush'

class SmartPhone:

def does(self):

return 'ring'

Now, let's define the Robot class with one instance of each of these and a does() method for the Robot:

class Robot:

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

self.laser = Laser()

self.claw = Claw()

self.smartphone = SmartPhone()

def does(self):

print(f"Laser does: {self.laser.does()}")

print(f"Claw does: {self.claw.does()}")

print(f"SmartPhone does: {self.smartphone.does()}")

robot = Robot()

robot.does()