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?

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

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?

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

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.

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.

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

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.

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.

Answers

1. **Create a Class Called Thing**:
   * We’ll create a simple class called Thing with no contents.

**Python**

class Thing:

pass

# Create an object called 'example'

example = Thing()

# Print the class and the object

print(f"Class: {Thing}")

print(f"Object: {example}")

1. **Create a Class Called Thing2 with a Class Attribute**:
   * We’ll create a class called Thing2 and add the value 'abc' to the letters class attribute.

**Python**

class Thing2:

letters = 'abc'

# Print the value of the 'letters' attribute

print(f"Thing2 letters: {Thing2.letters}")

1. **Create a Class Called Thing3 with an Instance Attribute**:
   * We’ll create a class called Thing3 and assign the value 'xyz' to an instance attribute called letters.
   * We don’t need to create an object from the class to access the instance attribute.

**Python**

class Thing3:

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

self.letters = 'xyz'

# Create an object of Thing3 (not necessary for this step)

thing3\_obj = Thing3()

# Print the value of the 'letters' attribute

print(f"Thing3 letters: {thing3\_obj.letters}")

1. **Create an Element Class with Instance Attributes**:
   * We’ll create an Element class with the instance attributes name, symbol, and number.

**Python**

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

# Create an object of Element

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

1. **Create a Dictionary and an Object from Class Element**:
   * We’ll create a dictionary with the given keys and values, then create an object called hydrogen from the Element class using this dictionary.

**Python**

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

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

1. **Define a Method**dump()**in Element Class**:
   * We’ll define a method called dump() in the Element class that prints the values of its attributes.

**Python**

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)

hydrogen.dump()

1. **Call**print(hydrogen)**and Change**dump**to**\_\_str\_\_:
   * We’ll call print(hydrogen) and then change the name of the method dump to \_\_str\_\_.
   * The \_\_str\_\_ method controls what should be returned when the object is represented as a string.

**Python**

class Element:

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

self.name = name

self.symbol = symbol

self.number = number

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

return f"{self.name} ({self.symbol}) - Atomic Number {self.number}"

# Create a new hydrogen object

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

print(hydrogen)

1. **Modify Element to Make Attributes Private and Define Getters**:
   * We’ll modify the Element class to make the attributes name, symbol, and number private.
   * We’ll define getter methods for each attribute to return their values.

**Python**

class Element:

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

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

# Create a new hydrogen object

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

print(f"Name: {hydrogen.get\_name()}, Symbol: {hydrogen.get\_symbol()}, Number: {hydrogen.get\_number()}")

1. **Define Three Classes: Bear, Rabbit, and Octothorpe**:
   * We’ll define three classes: Bear, Rabbit, and Octothorpe.
   * Each class will have a method called eats() that returns the preferred food for that animal.

**Python**

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 animal eats

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

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

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

1. **Define a Robot Class with Component Objects**:
   * We’ll define a Robot class that has one instance (object) of each of the components: Laser, Claw, and SmartPhone.
   * The Robot class will have a method called does() that prints what its component objects do.

**Python**

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):

return f"Laser: {self.laser.does()}, Claw: {self.claw.does()}, SmartPhone: {self.smartphone.does()}"

# Create a Robot object

robot = Robot()

# Print what the Robot's components do

print(robot.does())