**Assignment 19**

**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?**

**Sure, here's the solution to your problem:**

**python**

**Copy code**

**# Define a class called Thing**

**class Thing:**

**pass**

**# Print the class**

**print(Thing)**

**# Create an instance of the class Thing**

**example = Thing()**

**# Print the instance**

**print(example)**

**Output:**

**javascript**

**Copy code**

**<class '\_\_main\_\_.Thing'>**

**<\_\_main\_\_.Thing object at 0x7fca461fd8d0>**

**As you can see, the printed values are different. The first one is the class definition, and the second one is an instance of the class.**

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

**Here's the solution to your problem:**

**python**

**Copy code**

**# Define a class called Thing2**

**class Thing2:**

**letters = 'abc'**

**# Print the class attribute letters**

**print(Thing2.letters)**

**Output:**

**Copy code**

**abc**

**As you can see, the class attribute letters is printed with the value '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?**

**Here's the solution to your problem:**

**python**

**Copy code**

**# Define a class called Thing3**

**class Thing3:**

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

**self.letters = 'xyz'**

**# Create an instance of the class Thing3**

**example = Thing3()**

**# Print the object attribute letters**

**print(example.letters)**

**Output:**

**Copy code**

**xyz**

**Yes, you need to create an object (instance) of the class Thing3 to access the object attribute letters. In this case, the attribute letters is assigned the value 'xyz' when an object of the class is created using the constructor \_\_init\_\_.**

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

**Here's the solution to your problem:**

**python**

**Copy code**

**# Define a class called Element**

**class Element:**

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

**self.name = name**

**self.symbol = symbol**

**self.number = number**

**# Create an instance of the class Element**

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

**# Print the attributes of the instance**

**print(hydrogen.name)**

**print(hydrogen.symbol)**

**print(hydrogen.number)**

**Output:**

**Copy code**

**Hydrogen**

**H**

**1**

**As you can see, we defined a class Element with the instance attributes name, symbol, and number, and created an instance of that class with the values ‘Hydrogen,’ ‘H,’ and 1. Finally, we printed the attributes of the instance.**

**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.**

**Here's the solution to your problem:**

**python**

**Copy code**

**# Define a class called Element**

**class Element:**

**def \_\_init\_\_(self, \*\*kwargs):**

**self.name = kwargs['name']**

**self.symbol = kwargs['symbol']**

**self.number = kwargs['number']**

**# Define a dictionary with the key-value pairs**

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

**# Create an instance of the class Element using the dictionary**

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

**# Print the attributes of the instance**

**print(hydrogen.name)**

**print(hydrogen.symbol)**

**print(hydrogen.number)**

**Output:**

**Copy code**

**Hydrogen**

**H**

**1**

**As you can see, we defined a class Element with the instance attributes name, symbol, and number. Then, we created a dictionary with the key-value pairs corresponding to the attributes of the class. Finally, we created an instance of the class Element using the dictionary and printed the attributes of the instance.**

**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.**

**Here's the solution to your problem:**

**python**

**Copy code**

**# Define a class called Element**

**class Element:**

**def \_\_init\_\_(self, \*\*kwargs):**

**self.name = kwargs['name']**

**self.symbol = kwargs['symbol']**

**self.number = kwargs['number']**

**# Define a method called dump**

**def dump(self):**

**print('Name:', self.name)**

**print('Symbol:', self.symbol)**

**print('Number:', self.number)**

**# Create an instance of the class Element using a dictionary**

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

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

**# Use the dump() method to print the attributes of the instance**

**hydrogen.dump()**

**Output:**

**makefile**

**Copy code**

**Name: Hydrogen**

**Symbol: H**

**Number: 1**

**As you can see, we defined a class Element with the instance attributes name, symbol, and number, and defined a method called dump that prints the values of these attributes. We then created an instance of the class Element using a dictionary and used the dump() method to print the attributes of the instance.**

**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.**

**Here's the solution to your problem:**

**python**

**Copy code**

**# Define a class called Element**

**class Element:**

**def \_\_init\_\_(self, \*\*kwargs):**

**self.name = kwargs['name']**

**self.symbol = kwargs['symbol']**

**self.number = kwargs['number']**

**# Define a method called \_\_str\_\_ (formerly dump)**

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

**return f'Name: {self.name}\nSymbol: {self.symbol}\nNumber: {self.number}'**

**# Create an instance of the class Element using a dictionary**

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

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

**# Call print(hydrogen) and see the output**

**print(hydrogen)**

**# Create a new hydrogen object and call print(hydrogen) again**

**hydrogen = Element(name='Hydrogen', symbol='H', number=1)**

**print(hydrogen)**

**Output:**

**makefile**

**Copy code**

**Name: Hydrogen**

**Symbol: H**

**Number: 1**

**Name: Hydrogen**

**Symbol: H**

**Number: 1**

**As you can see, we defined a class Element with the instance attributes name, symbol, and number. We then changed the name of the method dump to str to make it a string representation of the object. We created an instance of the class Element using a dictionary and called print(hydrogen) to print the string representation of the object. Finally, we created a new hydrogen object and called print(hydrogen) again to print the string representation of the object.**

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

**Here's the solution to your problem:**

**python**

**Copy code**

**# Define a class called Element with private attributes**

**class Element:**

**def \_\_init\_\_(self, \*\*kwargs):**

**self.\_\_name = kwargs['name']**

**self.\_\_symbol = kwargs['symbol']**

**self.\_\_number = kwargs['number']**

**# Define getter properties for the private attributes**

**@property**

**def name(self):**

**return self.\_\_name**

**@property**

**def symbol(self):**

**return self.\_\_symbol**

**@property**

**def number(self):**

**return self.\_\_number**

**# Create an instance of the class Element using a dictionary**

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

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

**# Use the getter properties to access the private attributes**

**print(hydrogen.name)**

**print(hydrogen.symbol)**

**print(hydrogen.number)**

**Output:**

**Copy code**

**Hydrogen**

**H**

**1**

**As you can see, we defined a class Element with private attributes name, symbol, and number. We then defined getter properties for these attributes using the @property decorator. This allows us to access the private attributes using hydrogen.name, hydrogen.symbol, and hydrogen.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.**

**Here's the solution to your problem:**

**python**

**Copy code**

**# Define a class called Bear**

**class Bear:**

**def eats(self):**

**return 'berries'**

**# Define a class called Rabbit**

**class Rabbit:**

**def eats(self):**

**return 'clover'**

**# Define a class called Octothorpe**

**class Octothorpe:**

**def eats(self):**

**return 'campers'**

**# Create one object from each class and print what it eats**

**bear = Bear()**

**print(bear.eats())**

**rabbit = Rabbit()**

**print(rabbit.eats())**

**octothorpe = Octothorpe()**

**print(octothorpe.eats())**

**Output:**

**Copy code**

**berries**

**clover**

**campers**

**As you can see, we defined three classes Bear, Rabbit, and Octothorpe. Each class has only one method called eats() which returns a specific value. We then created one object from each class and printed what it eats using the eats() method.**

**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.**

**Here's the solution to your problem:**

**python**

**Copy code**

**# Define a class called Laser**

**class Laser:**

**def does(self):**

**return 'disintegrate'**

**# Define a class called Claw**

**class Claw:**

**def does(self):**

**return 'crush'**

**# Define a class called SmartPhone**

**class SmartPhone:**

**def does(self):**

**return 'ring'**

**# Define a class called Robot with instances of Laser, Claw, and SmartPhone**

**class Robot:**

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

**self.laser = Laser()**

**self.claw = Claw()**

**self.smartphone = SmartPhone()**

**# Define a method called does() for the Robot that prints what its component objects do**

**def does(self):**

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

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

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

**# Create an object of class Robot and call its does() method**

**robot = Robot()**

**robot.does()**

**Output:**

**Copy code**

**The Laser disintegrate.**

**The Claw crush.**

**The SmartPhone ring.**

**As you can see, we defined three classes Laser, Claw, and SmartPhone, each with only one method called does() that returns a specific value. We then defined the class Robot with instances of Laser, Claw, and SmartPhone. The Robot class also has a does() method that prints what its component objects do. Finally, we created an object of class Robot and called its does() method to see what each of its components do.**