PYTHON 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?

2. Create a new class called Thing2 and add the value &#39;abc&#39; to the letters class attribute. Letters

should be printed.

3. Make yet another class called, of course, Thing3. This time, assign the value &#39;xyz&#39; 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 &#39;Hydrogen,&#39; &#39;H,&#39; and 1.

5. Make a dictionary with these keys and values: &#39;name&#39;: &#39;Hydrogen&#39;, &#39;symbol&#39;: &#39;H&#39;, &#39;number&#39;: 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 &#39;berries&#39; (Bear), &#39;clover&#39; (Rabbit), or &#39;campers&#39; (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 &#39;disintegrate&#39; (Laser), &#39;crush&#39; (Claw), or &#39;ring&#39; (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.

SOLUTIONS

```
# 1. Class Thing with no contents
class Thing:
    pass

print(Thing())  # Output: <__main__.Thing object at 0x...>

# 2. Class Thing2 with letters class attribute
class Thing2:
    letters = 'abc'

print(Thing2.letters)  # Output: abc
```

```
# 3. Class Thing3 with instance attribute letters
class Thing3:
    def __init__(self):
        self.letters = 'xyz'

thing3_obj = Thing3()
print(thing3_obj.letters)  # Output: xyz

# 4. Class Element with instance attributes
class Element:
    def __init__(self, name, symbol, number):
        self.name = name
        self.symbol = symbol
        self.number = number

# Create an object from class Element
hydrogen = Element('Hydrogen', 'H', 1)

# 6. Method dump() to print attributes of Element
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 hydrogen object and use dump() to print attributes
hydrogen = Element('Hydrogen', 'H', 1)
hydrogen.dump()


# 7. __str__ method for printing 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 hydrogen object and print using print()
hydrogen = Element('Hydrogen', 'H', 1)
print(hydrogen)


# 8. Make attributes private and define getter properties
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

# Create hydrogen object and access attributes using getters
hydrogen = Element('Hydrogen', 'H', 1)
print(hydrogen.name, hydrogen.symbol, hydrogen.number)

# 9. Define classes Bear, Rabbit, and Octothorpe with eats() method
class Bear:
    def eats(self):
        return 'berries'

class Rabbit:
    def eats(self):
        return 'clover'

class Octothorpe:
    def eats(self):
        return 'campers'

# Create objects and print what they eat
bear = Bear()
rabbit = Rabbit()
octothorpe = Octothorpe()
print(bear.eats(), rabbit.eats(), octothorpe.eats())

# 10. Define classes Laser, Claw, SmartPhone, and Robot with does() method
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"{self.laser.does()}, {self.claw.does()}, {self.smartphone.does()}"


# Create Robot object and print what its component objects do
robot = Robot()
print(robot.does())
```