Exercises

Resource Name: **The Quick Python Book 3rd Edition**

Chapter: **#15**

Step: **#3**

By: **Dorrin Samadian**

TRY THIS: INSTANCE VARIABLES

```
class Rectangle:
    def__init__(self):
        self.height = 6
        self.width = 4
```

TRY THIS: INSTANCE VARIABLES AND METHODS

```
class Rectangle:
    def___init__(self, height, width):
        self.height = height
        self.width = width

   def area(self):
       return self.height * self.width
```

TRY THIS: CLASS METHODS

```
class Circle:
    pi = 3.14
    circle = []

def___init__(self, radius):
    self.radius = radius
    self.__class__.circle.append(self)

def area(self):
    return Circle.pi * self.radius * self.radius

def circle_circumference(self):
    return self.radius * Circle.pi * 2

@classmethod
def final_ans(temp):
    sum = 0
   for t in temp.circle:
       sum = sum + t.circle_circumference ()
   return sum
```

TRY THIS: INHERITANCE

```
class Shape:
    def__init__(self, a, b):
        self.a = a
        self.b = b

class Rectangle(Shape):
    def__init__(self, a, b):
        super().__init__(a, b)
```

1) Squares are a subset of the rectangles, so a square can inherit from a triangle.
2) Each shape has its own area formula so putting the area function in the class Shape isn't ok.

TRY THIS: PRIVATE INSTANCE VARIABLES

```
class Rectangle():
    def___init__(self, x, y):
        self.__x = x
        self.__y = y
```

TRY THIS: PROPERTIES

```
class Rectangle():
    def___init__(self, a, b):
        self.__a = a
        self.__b = b

@property
def a(self):
    return self.__a

@a.setter
def a(self, a_updated):
    if a_updated >= 0:
        self.__a = a_updated

@property
def b(self):
    return self.__b

@b.setter
def y(self, b_updated):
     if b_updated >= 0:
         self.__b = b_updated

rectangle = Rectangle(14, 16)
print(rectangle.a, rectangle.b)
rectangle.a = 15
rectangle.b          =          17
print(rectangle.a, rectangle.b)
```

LAB 15: HTML CLASSES

```
class element:
    def___init__(self, text = None, subelement = None):
        self.subelement = subelement
        self.text = text

   def___str__(self):
       value = ‘<{}>\n’.format(self.__class__.__name__)
       if self.text:
           value += ‘{}\n’.format(self.text)
       if self.subelement:
           value += str(self.subelement)
       value += ‘</{}>\n’.format(self.__class__.__name__)
       return value

class html(element):
    def___init___(self, text = None, subelement = None):
        super().__init__(text, subelement)
    def___str__(self):
        return super().__str__()

class body(element):
    def___init___(self, text = None, subelement = None):
        return super().__init__(text, subelement)
    def___str__(self):
        return super().__str__()

class p(element):
    def___init__(self, text = None, subelement = None):
        super().__init__(text, subelement)
    def___str__(self):
        return super().__str__()

para = p(text = ‘this is some body text’)
doc_body = body(text = ‘This is the body’, subelement = para)
doc = html(subelement = doc_body)
print(doc)
```