Using class & objects

COSC 101, 2018-04-18

Announcements
- Homework 8 due tomorrow @ 8pm

Outline
- Warm-up
- Classes
- Special functions
- Objects as parameters and return values

Warm-up
Assume you are given the following class definition:
class Rectangle:
    """A class representing a rectangle""

    def __init__(self, w, h):
        self.width = w
        self.height = h

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

da) What is output by the following code?
rect1 = Rectangle(5, 10)
print(rect1.area())
rect2 = Rectangle(6, 8)
print(rect2.area())
print(rect1.height, 'x', rect1.width)
print(rect2.height, 'x', rect2.width)
Output:
50
48
10 x 5
8 x 6

b) Write a perimeter function for the Rectangle class that returns a rectangle’s perimeter.
    def perimeter(self):
        return (self.width + self.height) * 2

c) Write a getDimensions function for the Rectangle class that returns a tuple with a rectangle’s width and height.
   def getDimensions(self):
       return (self.width, self.height)

d) Write a setWidth function for the Rectangle class that takes a width as a parameter and updates the rectangle’s width.
   def setWidth(self, w):
       self.width = w
Writing classes

- Write a class definition for a cylinder. The constructor should take the cylinder’s radius and height. The class should include three functions:
  - diameter (a cylinder’s diameter is double its radius)
  - volume (the volume of a cylinder is \( h \pi r^2 \))
  - setDimensions (which changes the cylinder’s radius and height to new values)

```python
import math

class Cylinder:
    def __init__(self, r, h):
        self.radius = r
        self.height = h

    def diameter(self):
        return self.radius * 2

    def volume(self):
        return self.height * math.pi * self.radius ** 2

    def setDimensions(self, r, h):
        self.radius = r
        self.height = h
```

- Write a function called cylinder_test (outside of the Cylinder class) that: creates two cylinders (with dimensions of your choosing), prints each cylinder’s volume and diameter, updates one cylinder’s dimensions, and prints each cylinder’s volume again.

```python
def cylinder_test():
    a = Cylinder(1, 2)
    b = Cylinder(3, 4)
    print(a.volume(), a.diameter())
    print(b.volume(), b.diameter())
    a.setDimensions(5, 6)
    print(a.volume(), b.volume())
```

Special functions

- What special function does every class have? --- __init__ (i.e., the constructor)
- Another special function is __str__
  - What is the purpose of this function? --- to provide a string representation of an object
  - Example for Rectangle
    ```python
    def __str__(self):
        return str(self.width) + ' x ' + str(self.height)
    ```
  - Write an __str__ function for the Cylinder class.
    ```python
    def __str__(self):
        return 'radius=' + str(self.radius) + ', height=' + str(self.height)
    ```

Objects as arguments and return values

- Object can be passed to a function as an argument --- already seen examples of this with Turtle, lists, strings, etc.
Example

def total_area(rect1, rect2):
    return rect1.area() + rect2.area()

When an object is passed as an argument to a function, the parameter name is an alias for the object

Example
def stretch(rect):
    rect.width = rect.width * 2

myrect = Rectangle(4, 6)
print(myrect)
stretch(myrect)
print(myrect)
Output:
4 x 6
8 x 6

Functions can return objects

Example
def half_rect(rect):
    new_rect = Rectangle(rect.width, rect.height / 2)

rectA = Rectangle(4, 6)
rectB = half_rect(rectA)
print(rectA)
print(rectB)
Output:
4 x 6
4 x 3

Assume you are given the following class definition:

class Pizza:
    def __init__(self, style, size):
        self.style = style
        self.size = size
        self.toppings = []

    def add_topping(self, topping):
        self.toppings.append(topping)

    def larger(self):
        p = Pizza(self.style, self.size + 4)
        for topping in self.toppings:
            p.add_topping(topping)
        return p

    def __str__(self):
        description = str(self.size) + '"
        description += str(self.style) + 'pizza'
        description += 'with ' + ','.join(self.toppings)
        return description
What is the output of each of the following programs?

a) first = Pizza('Chicago', 8)
   second = Pizza('New York', 14)
   first.add_topping('sausage')
   second.add_topping('pepperoni')
   print(first)
   print(second)
   
   Output:
   8" Chicago pizza with sausage
   14" New York pizza with pepperoni

b) mine = Pizza('Chicago', 10)
   mine.add_toppings(['black olives', 'mushrooms'])
   print(mine)
   
   Output:
   AttributeError: 'Pizza' object has no attribute 'add_toppings'

c) def add_veggies(p):
   for v in ['green pepper', 'tomato', 'onion']:
       p.add_topping(v)

   small = Pizza('Neapolitan', 8)
   large = Pizza('Neapolitan', 16)
   add_veggies(small)
   print(small)
   print(large)
   
   Output:
   8" Neapolitan pizza with green pepper, tomato, onion
   16" Neapolitan pizza with