
CORK
UNIVERSITY
BUSINESS
SCHOOL

IS6200/IS6143

Lab 4

Reading: Murach's Python Programming (2nd Edition)

# General guidelines

## Naming

- When creating the filenames for your programs, please use the convention specified by your instructor. Otherwise, for programs that consist of just one file, name the file *first_last_program*.py where *first_last* specifies your first and last name and *program* specifies the name of the program. For programs that have multiple files, store the files in a folder named *first_last_program*.
- When creating names for variables and functions, please use the guidelines and recommendations specified by your instructor. Otherwise, use the guidelines and recommendations specified in *Murach's Python Programming*.

## User interfaces

- You should think of the user interfaces that are shown for the projects as starting points. If you can improve on them, especially to make them more user-friendly, by all means do so.

## Specifications

- You should think of the specifications that are given for the projects as starting points. If you have the time to enhance the programs by improving on the starting specifications, by all means do so.

## Top-down development

- Always start by developing a working version of the program for a project. That way, you'll have something to show for your efforts if you run out of time. Then, you can build out that starting version of the program until it satisfies all of the specifications.
- From chapter 5 on, you should use top-down coding and testing as you develop your programs. You might also want to sketch out a hierarchy chart for each program as a guide to your top-down coding.

## Files supplied by your instructor

- Some of the projects require starting text, CSV, or binary files. These files are identified in the specifications for the projects, and your instructor should make these starting files available to you.

## Project 14-1: Rectangle Calculator

Create an object-oriented program that performs calculations on a rectangle.

### Console

```
Rectangle Calculator

Height:    10
Width:     20
Perimeter: 60
Area:      200
* * * * * * * * * * * * * * * * * * * *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
*                                       *
* * * * * * * * * * * * * * * * * * * *

Continue? (y/n): y

Height:    5
Width:     10
Perimeter: 30
Area:      50
* * * * * * * * * *
*                 *
*                 *
*                 *
* * * * * * * * * *

Continue? (y/n): n

Bye!
```

### Specifications

- Use a Rectangle class that provides attributes to store the height and width of a rectangle. This class should also provide methods that calculate the perimeter and area of the rectangle. In addition, it should provide a method that gets a string representation of the rectangle.
- When the program starts, it should prompt the user for height and width. Then, it should create a Rectangle object from the height and width and use the methods of that object to get the perimeter, area, and string representation of the object.

## Project 14-2: Card Dealer

Create an object-oriented program that creates a deck of cards, shuffles them, and deals the specified number of cards to the player.

### Console

```
Card Dealer

I have shuffled a deck of 52 cards.

How many cards would you like?: 7

Here are your cards:
Jack of Hearts
Jack of Diamonds
2 of Diamonds
6 of Spades
Jack of Spades
6 of Hearts
King of Diamonds

There are 45 cards left in the deck.

Good luck!
```

### Specifications

- Use a Card class to store the rank and suit for each card. In addition, use a method to get a string representation for each card such as "Ace of Spades", "2 of Spades", etc.
- Use a Deck class to store the 52 cards in a standard playing deck (one card for each rank and suit):

  ```
  ranks: 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King, Ace
  suits: Clubs, Diamonds, Hearts, Spades
  ```

- The Deck class should include a method that shuffles the deck, a method that counts the number of cards in the deck, and a method that deals a card from the deck, which should reduce the count of the cards in the deck by 1.
- When the program starts, it should get a new deck of cards, shuffle them, and display a message that indicates the total number of cards in the deck. To shuffle the cards, you can use the shuffle function of the random module described in chapter 6.
- The program should prompt the user for the desired number of cards. Then, it should deal the user the desired number of cards and display a message that indicates the number of cards left in the deck.