CS 1337/1337L Introduction to Object-Oriented   Fall 2017

Lab Manual

Prepared by: Vinitha Hannah Subburaj

Assistant Professor, ECSM

Textbook: MURACH'S PYTHON PROGRAMMING, 16 Ed. ISBN 9781890774974 (c) 2016, Murach.

Note:

- This manual is designed for students to plan and perform lab exercises in an organized way.
- Students will need their textbooks during every lab session.
- You need to install the source code for this book from WTClass.

General guidelines

Naming

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 case studies 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 case studies 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. That way, you'll have something to show for your efforts if you run out of time. Then, you can build out that version of the program until it satisfies all of the specifications.

Coding Style

Shebang Line

- The first line of all your Python programs should be a shebang line, which tells your computer that you want Python to execute this program.
  On Linux, the shebang line is #! /usr/bin/env python3

Layout

- Use 4 spaces per indentation level.
- Tabs or Spaces? Spaces are the preferred indentation method. Tabs should be used solely to remain consistent with code that is already indented with tabs. Python 3 disallows mixing the use of tabs and spaces for indentation.
- Limit all lines to a maximum of 79 characters.

Naming conventions

Here are some general principles when choosing names for your variables.

- Use meaningful names that convey the purpose of the variable.
- Choose names that are easy to pronounce, and avoid cryptic abbreviations. For example, use wagePerHour or hourlyWage instead of wph. Use polygon instead of p or poly or pgon.
- Be consistent.

Commenting

Programmers use comments to annotate a program and help the reader (or grader) understand how and why your program works. As a general rule, the code explains to the computer and programmer what is being done; the comments explain to the programmer why it is being done.

- Make sure that comments agree with the code. Be careful to update the comments when you update the code.
- Do not write comments that merely restate the code. Generally, comments should describe what or why you are doing something, rather than how.
- Comment any potentially confusing code, or better yet, rewrite the code so that it isn't confusing.

Index

Lab 0 – Getting Started

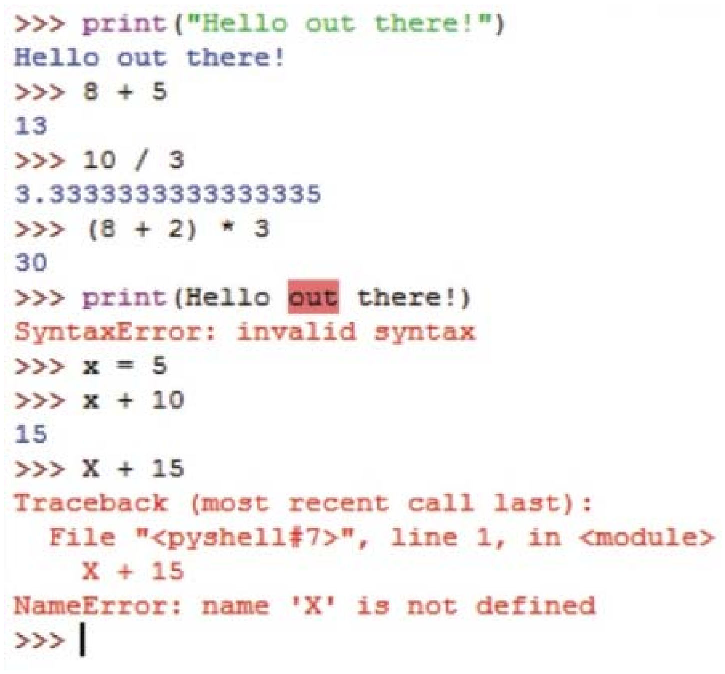
Objective

1. How to use IDLE to develop python programs
2. Installing the source code from www.murach.com

Experimenting with IDLE

Steps

1. We have different python editors installed in our lab workstations. Try out all the editors.
2. Open IDLE editor, type the following statements, save the file with a lab_0.py name extension, and try opening the file that you saved.



```
>>> print("Hello out there!")
Hello out there!
>>> 8 + 5
13
>>> 10 / 3
3.3333333333333335
>>> (8 + 2) * 3
30
>>> print(Hello out there!)
SyntaxError: invalid syntax
>>> x = 5
>>> x + 10
15
>>> X + 15
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    X + 15
NameError: name 'X' is not defined
>>> |
```

How to open, close, and restart the IDLE's interactive shell

- To start IDLE, use the features of your operating system. This opens an interactive shell.
- To close the interactive shell, click on its close button or select File→Close.
- To restart an interactive shell, select Run→ Python Shell from another IDLE window.

How to use the IDLE's interactive shell

- Enter Python code after the >>> prompt. Then, press Enter.
- If you enter valid code that produces a result, the shell displays the results.
- If you enter invalid code, the shell displays an error message.

Downloading Student resources

Download the student resources folder from your WTClass

WTClass → CS 1301 → Lessons → Lab → Student resources

Lab 1

Chapter 1

An introduction to Python programming

Objectives

1. Use IDLE to test Python expressions and statements in the interactive shell.

2. Use IDLE to open, compile, and run a Python source file.

Using your textbook,

Complete Exercise 1-1, 1-2, 1-3 from your textbook (pg: 24, 25, 26)

Take screenshots of your output screen and save them as Excercise1_1_output.jpg, Excercise1_2_output.jpg, Excercise1_3_output.jpg

Files to upload to WTClass: Excercise1_1_output.jpg, Excercise1_2_output.jpg, Excercise1_3_output.jpg

Lab 2

Chapter 2

How to write your first programs

Objectives

1. Use the IDLE shell to test numeric and string operations.

2. Code, test, and debug programs that require the skills that you've learned in this chapter. That includes the use of:

comments for documenting your code and commenting out statements

str, int, and float values and variables

arithmetic expressions

string concatenation

special characters in strings

the built-in print(), input(), str(), float(), int(), and round() functions

function chaining

2A Using your textbook,

Complete Exercises 2-1, 2-2, 2-3 (pg: 62, 63, 64). Save your work after completing the exercises using the filenames Exercise_2_1_output.py, Exercise_2_2_output.py, and Exercise_2_3_output.py

Files to upload: Exercise_2_1_output.py, Exercise_2_2_output.py, and Exercise_2_3_output.py

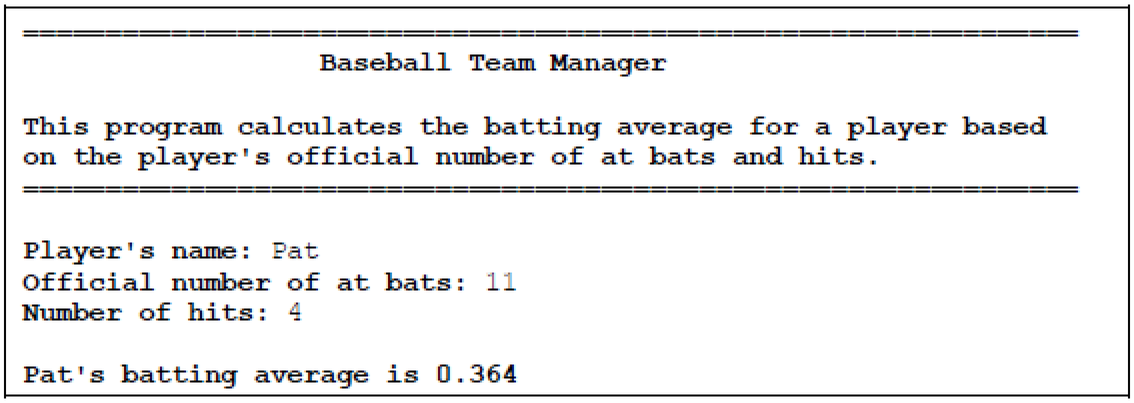
2B Case Study:

For this case study, you'll use the programming skills that you learn in Murach's Python Programming to develop a program that helps a person manage a baseball team. This program stores the data for each player on the team, and it also lets the manager set a starting lineup for each game.

Calculate a player's batting average

Create a program that displays a welcome message and calculates a player's batting average. Save your program using the file name baseball_team.py

Console



```
================================================================
                  Baseball Team Manager

This program calculates the batting average for a player based
on the player's official number of at bats and hits.
================================================================

Player's name: Pat
Official number of at bats: 11
Number of hits: 4

Pat's batting average is 0.364
```

Specifications

- The formula for calculating batting average is: average = hits / at_bats
- The program should accept integer entries.
- Assume the user will enter valid data.
- The program should round the batting averages to a maximum of three decimal places

File to upload: baseball_team.py

Lab 3

How to code control statements

Objectives

1. Code, test, and debug programs that require the skills that you've learned in this

    chapter. That includes the use of:

    if statements

    while statements

    for statements

    break and continue statements

    pass statements

2. Use pseudocode to plan your control structures and programs.

3A. Using your textbook,

Complete Exercises 3-1, 3-2, 3-3 (pg: 101, 102, 103). Save your work after completing the exercises using the filenames Exercise_3_1_output.py, Exercise_3_2_output.py, and Exercise_2_3_output.py

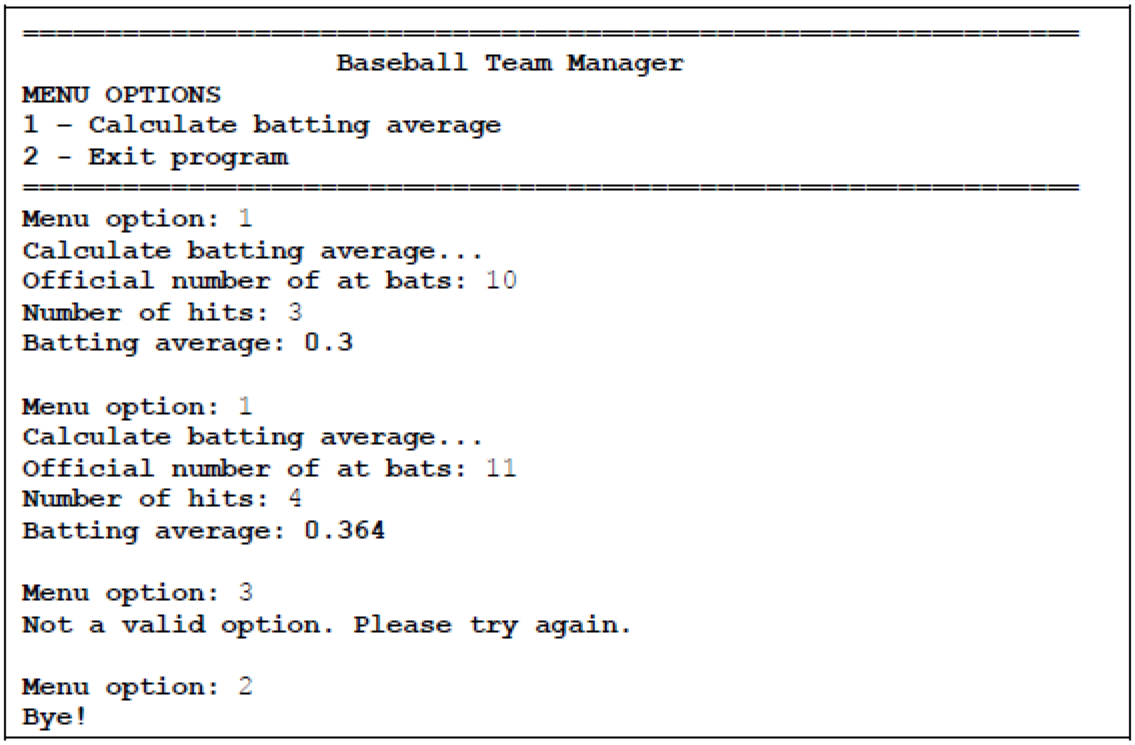
Files to upload: Exercise_2_1_output.py, Exercise_2_2_output.py, and Exercise_2_3_output.py

3B. Case Study

Add a menu

Update the program so it allows the user to select options from a menu.

Console:



```
================================================================
                   Baseball Team Manager
MENU OPTIONS
1 – Calculate batting average
2 - Exit program
================================================================
Menu option: 1
Calculate batting average...
Official number of at bats: 10
Number of hits: 3
Batting average: 0.3

Menu option: 1
Calculate batting average...
Official number of at bats: 11
Number of hits: 4
Batting average: 0.364

Menu option: 3
Not a valid option. Please try again.

Menu option: 2
Bye!
```

Specifications

- Assume the user will enter valid data.
- Display an error message if the user chooses an invalid menu option.

File to upload: baseball_team.py

Lab 4

How to define and use functions and modules

Objectives

1. Define and use functions in your programs including the use of default values, named arguments, local variables, and global variables.

2. Create, document, import, and use your own modules.

3. Import and use the random module.

4. Use a hierarchy chart or outline to plan the functions of a program.

4A. Using your textbook,

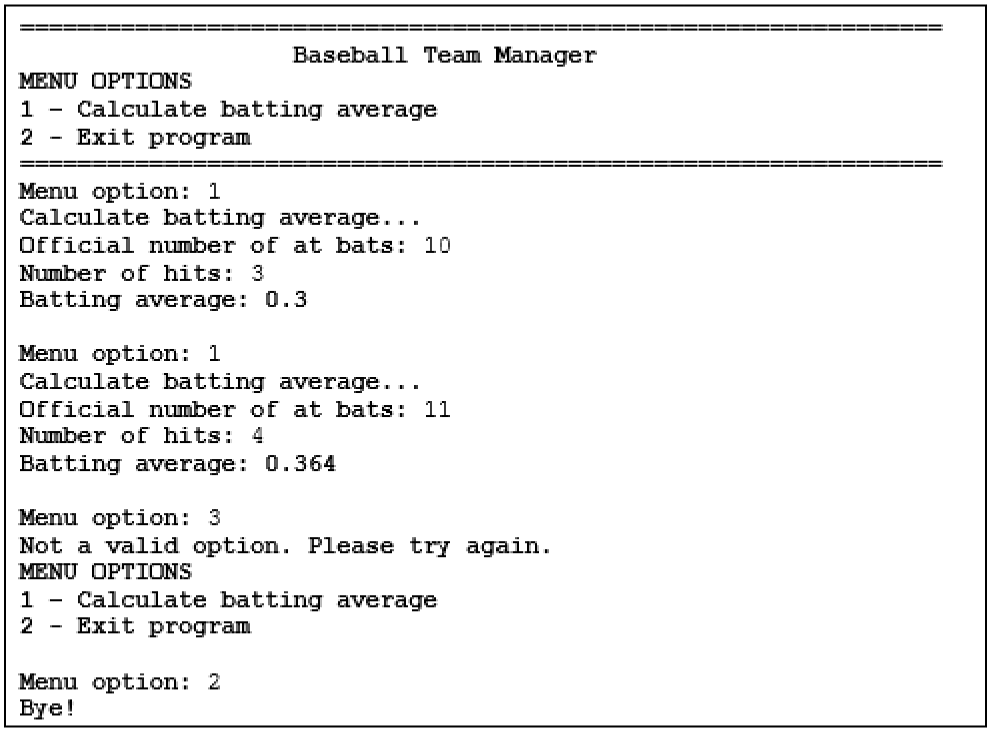
Complete Exercises 4-1 (pg: 138, 138). Save your work after completing the exercises using the filenames Exercise_4_1_output.py

Files to upload: Exercise_4_1_output.py

4B. Case Study

Update the program so it uses functions to organize the code so it's easier to reuse and maintain.

Console



```
================================================================
                   Baseball Team Manager
MENU OPTIONS
1 – Calculate batting average
2 - Exit program
================================================================
Menu option: 1
Calculate batting average...
Official number of at bats: 10
Number of hits: 3
Batting average: 0.3

Menu option: 1
Calculate batting average...
Official number of at bats: 11
Number of hits: 4
Batting average: 0.364

Menu option: 3
Not a valid option. Please try again.
MENU OPTIONS
1 – Calculate batting average
2 - Exit program

Menu option: 2
Bye!
```

Specifications

- Use a function to store the code that displays the menu.
- Use a function to store the code that calculates the batting average.
- Use a main function to store the rest of the code.
- Assume the user will enter valid data.
- If the user enters an invalid menu option, display an error message and display the menu again, so the user can clearly see the valid menu options.

File to upload: baseball_team.py