Full Stack Development Lab Instructions
ENSF381: Lab09

Python Warm up

Created by: Mahdi Jaberzadeh Ansari (He/Him)
Schulich School of Engineering University of Calgary
Calgary, Canada
mahdi.ansari1@ucalgary.ca

Week 10, March 18/20, 2024

Contents

1 Introduction

1.1 Objectives

Lab 9 is designed to extend your practical experience using Python.

1.2 Prerequisites

1. Basic understanding of computer operations and algorithms.

We do not use a Git repository for this assignment, so you must deliver all your codes and the answer sheet file in a ZIP file via D2L.

1.3 Forming groups

- In this lab session, you **MUST** work with a partner (groups of three or more are not allowed).
- The main goal of working in a group is to learn:
  - how to do teamwork
  - how to not be a single player
  - how to not be bossing
  - how to play for team
  - how to tolerate others
  - how to behave with colleagues
  - how to form a winner team
  - and ...
- Working with a partner usually gives you the opportunity to discuss some of the details of the topic and learn from each other. Also, it will give you the opportunity to practice one of the popular methods of program development called pair programming. In this method, which is normally associated with the "Agile Software Development" technique, two programmers normally work together on the same workstation (you may consider a Zoom session for this purpose). While one partner, the driver, writes the code, the other partner, acting as an observer, looks over his or her shoulder, making sure the syntax and solution logic are correct. Partners should switch roles frequently in such a way that both have equivalent opportunities to practice both roles. **Please note that you MUST switch roles for each exercise.**
- When you have to work with a partner:
  - Choose a partner that can either increase your knowledge or transfer your knowledge. (i.e., do not find a person with the same programming skill level!)
  - Please submit only one lab report with both names. Submitting two lab reports with the same content will be considered copying and plagiarism.

1.4 Before submission

- For most of the labs, you will receive a DOCX file that you need to fill out the gaps. Make sure you have this file to fill out.
- All your work should be submitted as a single file in PDF format. For instructions about how to provide your lab reports, study the posted document on the D2L called How to Hand in Your Lab assignment.

- Please note that if it is group work, only one team member must submit the solution in D2L. For ease of transferring your marks, please mention the group member's name and UCID in the description window of the submission form.
- If you have been asked to write code (HTML, CSS, JS, Python, etc.), make sure the following information appears at the top of your code:
  - File Name
  - Assignment and exercise number
  - Your names in alphabetic order
  - Submission Date:

  Here is an example in Python:

```
"""
=============================================================
Name        : lab9_exe_C.py
Assignment  : Lab 9, Exercise C
Author(s)   : Mahdi Ansari, William Arthur Philip Louis
Submission  : May 21, 2030
Description : Fetch data by Python.
=============================================================
"""
```

- Some exercises in this lab and future labs will not be marked. Please do not skip them, because these exercises are as important as the others in learning the course material.
- In courses like ENSF381, some students skip directly to the exercises that involve writing code, skipping sections such as "Read This First," or postponing the diagram-drawing until later. That's a bad idea for several reasons:
  - "Read This First" sections normally explain some technical or syntax details that may help you solve the problem or may provide you with some hints.
  - Some lab exercises may ask you to draw a diagram, and most of the students prefer to hand-draw them. In these cases, you need to scan your diagram with a scanner or an appropriate device, such as your mobile phone, and insert the scanned picture of your diagram into your PDF file (the lab report). A possible mobile app to scan your documents is Microsoft Lens, which you can install on your mobile device for free. Please make sure your diagram is clear and readable; otherwise, you may either lose marks or it could be impossible for TAs to mark it at all.
  - Also, it is better to use the Draw.io tool if you need to draw any diagram.
  - Drawing diagrams is an important part of learning how to visualize data transfer between modules and so on. If you do diagram-drawing exercises at the last minute, you won't learn the material very well. If you do the diagrams first, you may find it easier to understand the code-writing exercises, so you may be able to finish them more quickly.
- **Due Dates:**
  - You must submit your solution until 11:59 p.m. on the same day that you have the lab session.
  - Submissions until 24 hours after the due date get a maximum of 60% of the mark, and after 24 hours, they will not be evaluated and get 0.

1.5 Academic Misconduct/Plagiarism

- Ask for help, but don't copy.
  - You can get help from lab instructor(s), TAs, or even your classmates as long as you do not copy other people's work.
  - If we realize that even a small portion of your work is a copy from a classmate, both parties (the donor and the receiver of the work) will be subject to academic misconduct (plagiarism). More importantly, if you give exercise solutions to a friend, you are not doing him or her a favor, as he or she will never learn that topic and will pay off for this mistake during the exam or quiz. So, please do not put yourself and your friend in a position of academic misconduct.
  - You can use ChatGPT, but please note that it may provide similar answers for others too, or even the wrong answers. For example, it has been shown that AI can hallucinate, proposing the use of libraries that do not actually exist [1]. So, we recommend that you imagine ChatGPT as an advanced search engine, not a solution provider.
  - In order to find out who is abusing these kinds of tools, we will eventually push you toward the incorrect responses that ChatGPT might produce. In that case, you might have failed for the final mark and be reported to administration.
  - If we ask you to investigate something, don't forget to mention the source of your information. Reporting without reference can lead to a zero mark even by providing a correct answer.

1.6 Marking Scheme

- You should not submit anything for the exercises that are not marked.
- In Table 1, you can find the marking scheme for each exercise.

Table 1: Marking scheme

Exercise	Marks
A	15 marks
B	10 marks
C	25 marks
Total	50 marks

1.7 Complains

- Your grades will be posted one week following the submission date, which means they will be accessible at the subsequent lab meeting.
- Normally, the grades for individual labs are assessed by a distinct TA for each lab and section. Kindly refrain from contacting all TAs. If you have any concerns regarding your grades, please direct an email to the TA responsible for that specific lab.

[1] https://perma.cc/UQS5-3BBP

2 Exercise A (HTTP Theoretical)

2.1 Deliverable

The deliverables for this assignment are structured as follows:

1. Draw a diagram illustrating the structure of an HTTP request. Provide a detailed explanation for each component within the request, encompassing methods, headers, the endpoint, and the body.
2. Construct a diagram to depict the structure of an HTTP response. Elucidate every component of the response, including status line (version, status code, and reason phrase), headers, and body. Additionally, furnish a compendium of standard HTTP status codes, categorizing them and providing at least three examples for each category:
   - **1xx (Informational)**: Briefly describe and provide three examples.
   - **2xx (Success)**: Elaborate on the category and list three examples.
   - **3xx (Redirection)**: Explain the purpose of redirection codes and offer three instances.
   - **4xx (Client Error)**: Discuss the significance and provide three examples.
   - **5xx (Server Error)**: Detail this category and list three examples.
3. Discuss the concept of CRUD (Create, Read, Update, Delete) operations and elucidate their correlation with HTTP methods. Specifically, illustrate how each CRUD operation maps to specific HTTP methods, thereby elucidating the relationship and its significance in the context of web application development.

3 Exercise B (Setting Up VSC for Python)

3.1 Objective

This section guides you through the process of installing Python and setting up Visual Studio Code (VSC) for Python development, including the installation of essential extensions.

3.2 Prerequisites

- A computer running Windows, macOS, or Linux.
- Internet access for downloading software.

3.3 Part 1: Installing Python

3.3.1 Step 1: Download Python

1. Visit the official Python website at `https://www.python.org/`.
2. Navigate to the Downloads section. The website should automatically suggest the best version for your operating system.
3. Click on the download link for the latest version of Python.

3.3.2 Step 2: Install Python

1. Run the downloaded installer.
2. **Important**: Ensure you check the box that says **"Add Python 3.x to PATH"** before clicking on the "Install Now" button.
3. Follow the on-screen instructions to complete the installation.
4. After installation, open a terminal or command prompt and type `python --version` to verify the installation. You should see the Python version number to see if the installation was successful.
5. Make a screenshot from the terminal that shows your installed Python version, you need this screenshot later.

3.4 Part 2: Installing Visual Studio Code

Skip this section if you already have installed VSC.

3.4.1 Step 1: Download Visual Studio Code

1. Visit the official Visual Studio Code website at https://code.visualstudio.com/.
2. Click on the download link for your operating system.
3. Save the installer to your computer.

3.4.2 Step 2: Install Visual Studio Code

1. Run the downloaded installer and follow the installation instructions.
2. After installation, launch Visual Studio Code.

3.5 Part 3: Configuring Visual Studio Code for Python Development

3.5.1 Step 1: Install the Python Extension for Visual Studio Code

1. In Visual Studio Code, navigate to the Extensions view by clicking on the square icon on the sidebar or pressing `Ctrl+Shift+X`.
2. Search for **Python** in the Extensions Marketplace.
3. Find the Python extension by Microsoft (it should be the first result) and click **Install**.

3.5.2 Step 2: Install Additional Useful Extensions (Optional)

- **Pylance**: Offers fast, feature-rich language support for Python.
- **Jupyter**: Enables you to create and edit Jupyter Notebooks within VS Code.
- **autoDocstring**: Generates Python docstrings automatically.

To install these extensions, repeat the process described in Step 1, searching for each extension by name and clicking **Install**. However, we don't need them for this course, they are just some suggestions for your future.

3.5.3 Step 3: Verify the Setup

1. Create a new file with a `.py` extension in your VSC.
2. Type the following Python code: `print("Hello, world!")`
3. Right-click in the editor window and select **Run Python File in Terminal**.
4. Verify that "Hello, world!" is printed in the terminal.
5. Make a screenshot from the output and save it.

3.6 Deliverable

1. Attach your screenshot from 3.3.2.5 to the answer sheet file.
2. Attach your screenshot from 3.5.3.5 to the answer sheet file.

4 Exercise C (Interacting with JSON Data in Python)

4.1 Objective

This section is designed to enhance your understanding of working with external JSON data in Python, using arrays and strings, and improving your problem-solving skills through real-world applications.

4.2 Requirements

Create a Python application that reads product data from https://dummyjson.com/products, stores this data in a collection, and allows the user to interact with this data through the terminal. The application should enable users to:

1. Display a list of all products.
2. Search for a specific product by name and display its details.

4.3 Steps

4.3.1 Initialize a Project

1. Create a folder at your desired address (e.g., your desktop) and name it `lab9`.
2. Create a script file and name it `lab9_exe_C.py`.
3. Add the required header comment to it.

4.3.2 Installing modules

Before running Python scripts that require external modules, like `requests` module in this exercise that we are going to use for fetching data, ensure you install them using `pip install module_name` if you have not done so already. Otherwise you will see an error like this:

```
ModuleNotFoundError: No module named 'requests'
```

So, first run the following command to install the *request* module:

```
pip install requests
```

After a successful installation, you must see something like Figure 1.

```
Traceback (most recent call last):
  File "C:\Users\mahdi\OneDrive - University of Calgary\Courses\ENSF381\Labs\Lab9\lab9_exe_C.py", line 1, in <module>
    import requests
ModuleNotFoundError: No module named 'requests'

C:\Users\mahdi\OneDrive - University of Calgary\Courses\ENSF381\Labs\Lab9>pip install requests
Defaulting to user installation because normal site-packages is not writeable
Collecting requests
  Downloading requests-2.31.0-py3-none-any.whl.metadata (4.6 kB)
Collecting charset-normalizer<4,>=2 (from requests)
  Downloading charset_normalizer-3.3.2-cp312-cp312-win_amd64.whl.metadata (34 kB)
Requirement already satisfied: idna<4,>=2.5 in c:\users\mahdi\appdata\roaming\python\python312\site-packages (from reque
sts) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\mahdi\appdata\roaming\python\python312\site-packages (from
 requests) (1.26.18)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mahdi\appdata\roaming\python\python312\site-packages (from
 requests) (2024.2.2)
Downloading requests-2.31.0-py3-none-any.whl (62 kB)
   ---------------------------------------- 62.6/62.6 kB 844.1 kB/s eta 0:00:00
Downloading charset_normalizer-3.3.2-cp312-cp312-win_amd64.whl (100 kB)
   ---------------------------------------- 100.4/100.4 kB 2.9 MB/s eta 0:00:00
Installing collected packages: charset-normalizer, requests
Successfully installed charset-normalizer-3.3.2 requests-2.31.0
```

Figure 1: Installation message

4.3.3 Fetching and Parsing JSON Data

1. **Import Required Modules**: At the beginning of your script, import the `requests` module for making HTTP requests and the `json` module for parsing JSON data.

```
import requests
import json
```

2. **Fetch Data from URL**: Use the `requests` module to fetch data from `https://dummyjson.com/products`. Ensure you handle possible exceptions, such as network errors. Here is an example, feel free to extend it.

```
def fetch_product_data(url):
    try:
        response = requests.get(url)
        # Raises an error for bad responses
        response.raise_for_status()
        # The JSON structure includes a 'products' key
        return response.json()['products']
    except requests.exceptions.RequestException as e:
        print(f"Error fetching data: {e}")
        return None
```

4.3.4 Part 3: Implementing Features

1. **Display All Products**: Implement a function that takes the product data collection as an argument and prints the names of all products.

```
def list_all_products(products):
    # complete this function
```

2. **Search and Display Product Details**: Implement a function that searches for a product by name and prints its details. Ensure to handle the case where the product is not found.

```
def search_product(products, name):
    for product in products:
        # complete the for loop,
        # it must pretty print the product details with 4 indents
    print("Product not found.")
```

4.3.5 Part 4: User Interaction

1. **Main Function**: Create a main function to:

   - Fetch the product data from the URL.
   - Prompt the user to choose between listing all products or searching for a specific product.
   - Call the appropriate function based on the user's choice.

```
def main():
    products_url = 'https://dummyjson.com/products'
    products = fetch_product_data(products_url)

    if products:
        while True:
            choice = input("Choose an option:\n1. List all products\n2.
                Search for a product\n3. Exit\n> ")
```

```
            if choice == '1':
                # complete the code.
                # call suitable function(s)
            elif choice == '2':
                product_name = input("Enter the product name: ")
                # complete the code.
                # call suitable function(s)
            elif choice == '3':
                break
            else:
                print("Invalid choice. Please try again.")
    else:
        print("Failed to fetch product data.")
```

2. **Run Your Application**: Add a call to `main()` at the end of your script.

```
if __name__ == "__main__":
    main()
```

Run the script using the command `python lab9_exe_C.py` in your terminal and follow the prompts.

```
python lab9_exe_C.py
```

4.3.6 Deliverable

1. Add comments to your code explaining the logic behind each function. Submit the `.py` file along with your answer sheet file. Please do not copy and paste any code inside the answer sheet file, as we need to run and test your `.py` file.

2. Include screenshots of your terminal showing the application running and handling different user inputs.