# GGS 366: Lab 2

## General guidelines

**Please submit the .ipynb notebook containing all the codes.** Make sure the notebook is properly formatted by following the instructions below.

* Separate answers to each question using text cells and markdown headings. See example syntax here: [Markdown Guide - Colab](https://colab.research.google.com/notebooks/markdown_guide.ipynb)
* Write the essay questions in your Python notebook using text cells.
* The function body (where appropriate) must include a docstring to provide a general description of the function.
* Each step of the code needs to be commented.
* The code needs to be properly indented and readable. For more on formatting guidelines: [PEP 8 – Style Guide for Python Code | peps.python.org](https://peps.python.org/pep-0008/) and [PEP 257 – Docstring Conventions | peps.python.org](https://peps.python.org/pep-0257/)

You may use GenAI as a supporting tool. However, directly copying code from GenAI will be considered plagiarism and hinder your learning process. This can negatively impact your performance in course exams and, ultimately, your professional success. Therefore, it is in your best interest to thoroughly understand the fundamental concepts and make a genuine effort to solve the problems on your own.

## Question 1 (5 points)

You are given a string variable representing a file path:

*file\_path = “C:\Users\downloads\main\_folder\virginia\_map.jpeg”*

Using **character indexing and slicing**, extract substrings from *file\_path* to create the following variables:

1. ***file\_name*** — This should contain *"virginia\_map.jpeg"*.
2. ***folder\_name*** — This should contain *"main\_folder"*.
3. ***drive\_name*** — This should contain *"C:"*.

You **must** use string slicing with character indexes (e.g., *file\_path[start:end]*). **Do not** declare these variables directly (i.e., avoid hardcoding like *file\_name = "virginia\_map.jpeg"*).

## Question 2 (5 points)

You are given the following lists:

*day = [1, 2, 3, 4, 5, 6, 7]*

*temperature = [55, 60, 52, 54, 63, 64, 60]*

*humidity = [62, 43, 58, 56, 59, 60, 66]*

The ***day*** list represents the days of the week (1 to 7). The ***temperature*** and ***humidity*** lists contain corresponding temperature and humidity values for each day.

**Tasks:**

1. The temperature for **day 4** is incorrect. The current value is 54, but the correct temperature is 57. **Update** the *temperature* list using list indexing to reflect this correction.
2. Using **list indexing and slicing**, **create three new lists** that contain data **only for days 3 to 6**:
   * *day\_sub* → Contains days *[3, 4, 5, 6]*
   * *temperature\_sub* → Corresponding temperatures *[52, 54, 63, 64]*
   * *humidity\_sub* → Corresponding humidity values *[58, 56, 59, 60]*

You **must** **use list indexing and slicing** (e.g., *list[start:end]*). **Do not** hardcode the values directly (e.g., avoid writing *day\_sub = [3, 4, 5, 6]*).

## Question 3 (5 points)

If the *days*, *temperature*, and *humidity* data structures were written like this:

*day = (1, 2, 3, 4, 5, 6, 7)*

*temperature = (55, 60, 52, 54, 63, 64, 60)*

*humidity = (62, 43, 58, 56, 59, 60, 66)*

What do we call this data structure? Can you still update the incorrect temperature value using their index positions as you did in question 2? Justify your answer.

## Question 4 (10 points)

You can calculate the **cube of a number** by multiplying it by itself three times. For example:

Cube of 2=23=2×2×2=8

**Tasks:**

1. Write Python code to create a list named *cubes\_list* that contains the **cubes of numbers from 1 to 3**.
2. Write a Python function named *cubes\_function* that takes an **integer** as input and **returns its cube**.
3. Using the *cubes\_function*, **calculate the cubes of numbers from 4 to 10** and **append** each of these cube values to the existing *cubes\_list*.
4. **Print the final *cubes\_list*** that contains all cube values from 1 to 10.

## Question 5 (10 points)

Open Google Maps and search for the **Lincoln Memorial**. Google Maps will add a pin to the map indicating the location. Right-click on the pin and write down the latitude and longitude of the Lincoln Memorial somewhere. Remember, Google Maps displays the latitude first and the longitude second. Do the same for **the Thomas Jefferson Memorial, the Smithsonian National Museum of Natural History, and the United States Capitol**.

A screenshot of a computer

Description automatically generated

Tasks:

1. Create a Python list named *DC\_attractions* that contains **four dictionaries**, one for each location.
2. Each dictionary should have the following structure:
   1. **Key:** *location\_name* → **Value:** A string representing the name of the location.
   2. **Key:** *coord* → **Value:** A tuple containing the coordinates in the format **(longitude, latitude)**. *(Note: Make sure to switch the order from Google Maps, which shows latitude first.)*
3. Using **proper list indexing and dictionary keys** from the *DC\_attractions* list, print the following statement for **each location.**

The **[insert location name here]** is located at **[insert longitude here]** º longitude and **[insert latitude here]** º latitude.

Avoid hardcoding the print statements. **Make sure to use list indexing and dictionary keys to get the corresponding values**.