Lab 16, Part 2: Object-Oriented Programming

Instructions:

This worksheet serves as a guide and set of instructions to complete the lab.

- You must use the starter file, found here, to get credit for the lab.
- Additionally, here is the workbook that you can read through for further context and additional (non-required) material.
- All material was sourced from the CS10 version of The Beauty and Joy of Computing course.

Submitting:

For part 2 of Lab 16, you will need to complete **all *MemePage* class methods and all *Member* class methods (6 total methods)** then submit this to Gradescope (Lab 16, Part 2: Object Oriented Programming).

- To receive full credit, you will need to complete all required methods, and the required methods must pass all tests from the autograder in Gradescope.
- For instructions on how to submit to labs to Gradescope, please see this document.

Please note, you must use the starter file, and you must NOT edit the name of any of the required functions. Failing to do either for these will result in the autograder failing.

Notes:

***This is Part 2 of Lab 16, meant for Monday (7/22).* Part 2 is due on Tuesday (7/23) @ 2359hrs**

Objectives:

Object-Oriented Programming (OOP) is a programming paradigm based on the concept of objects. Objects which can contain data (we call these attributes) and code (we call these methods). OOP is a very important level of abstraction used commonly across programming languages In today's lab you will:

- Learn how to create Class methods
- Learn how to build a constructor for a Class
- Understand the difference between Class and instance attributes
    - Along with knowing when to properly use each
- Understand the differences between instances of the same Class
- Understand and create **eq** methods

Glossary

- **Object:** An object is a collection of data with its own methods, attributes, and identity.
- **Method:** A function that an object has. For example, the Book object has the method calculate_age that calculates the age of the book.
- **Attribute:** Basically, a variable that belongs to an object. For example, a Book object's genre is an attribute.
- **Class:** A class is the blueprint of an object; it is the precise definition of an object's methods and attributes.
- **Constructor:** The constructor is the method that Python uses when it creates (instantiates) an object. Not all attributes of a class are defined immediately, the constructor lets you define an object's attributes when you actually create the object.
- **Inheritance:** A mechanism that allows a class to inherit properties and behaviors from another (parent) class. Highly promotes code reuse and establishes relationships between classes
- **Instance:** An instance is an object made from a specific class. For example, The Hunger Games could be an instance of the Book class.
- **Instantiation:** The creation of an instance. This is when the constructor actually creates the object.

Required Methods:

- MemePage Class
  - __init__(self, topic)
  - __eq__(self, other)
- Member Class
  - __init__(self, name, memepage)
  - tag_ur_friend_in_meme(self, friend, title_of_post)
  - post_in_page(self, title_of_post)
  - like_a_post_in_page(self, title_of_post)

Important Topics mentioned in the Workbook:
For better understanding of the lab we highly recommend going through these workbook pages! Topics that are important but not required for this lab will be indicated with an asterisk**. These topics are best reviewed in order and as you complete the lab.

- Constructors
- Class Attributes
- Meme Oriented Programming

Method 1.1: __init__(self, topic)

- Objective:
  - Create a constructor that initializes a MemePage object that has one input attribute: topic

- Notes:
    - Review Constructors page of workbook for guidance
    - Read doctests carefully!
        - Notice how a MemePage object has more than one attribute…
- Inputs:
    - self = N/A
    - topic = the topic of the meme page
- Output:
    - Reports: None
    - Your output should be nothing. The constructor simply initializes an object without reporting anything
- Examples:
    - Doctests available
        - python3 -m doctest <labfilename>.py to run autograder
            - Must be in correct parent file

Method 1.2: __eq__(self, other)

- Objective:
    - Create a eq method that returns a boolean representing equality of MemePage objects
- Notes:
    - When checking if one MemePage object and another MemePage object are equal, this function is called
        - View doctests for example
- Inputs:
    - self = N/A
    - other = the other MemePage object to compare with
- Output:
    - Reports: Boolean
    - Your output should be a boolean representing if the two MemePage object are equal
- Examples:
    - Doctests available
        - python3 -m doctest <labfilename>.py to run autograder
            - Must be in correct parent file

Method 2.1: __init__(self, name, memepage)

- Objective:
    - Create a constructor that initializes a Member object that has two input attributes: name and memepage

- Notes:
    - Review Constructors page of workbook for guidance
    - Read other method doctests carefully!
        - Notice how a Member object has more than two attributes…
- Inputs:
    - self = N/A
    - name = the name of the member
    - memepage = the MemePage object the member belongs to
- Output:
    - Reports: None
    - Your output should be nothing. The constructor simply initializes an object without reporting anything
- Examples:
    - Doctests available
        - python3 -m doctest <labfilename>.py to run autograder
            - Must be in correct parent file

Method 2.2: tag_ur_friend_in_meme(self, friend, title_of_post)

- Objective:
    - Write a method that tags a friend in a meme post
- Notes:
    - Notice any attribute changes?
    - Utilize f-strings!
- Inputs:
    - self = N/A
    - friend = the friend to tag in the meme
    - title_of_post = the title of the post to tag the friend in
- Output:
    - Reports: String
    - Your output should return a message indicating whether the friend was tagged or not
- Examples:
    - Doctests available
        - python3 -m doctest <labfilename>.py to run autograder
            - Must be in correct parent file

Method 2.3: post_in_page(self, title_of_post)

- Objective:
    - Write a method that posts a meme to a certain MemePage
- Notes:
    - Utilize f-strings!

    - When does one get banned in this setting?
- Inputs:
    - self = N/A
    - title_of_post = the title of the post to be posted
- Output:
    - Reports: String
    - Your output should be a message indicating the activity and number of posts on the MemePage
- Examples:
    - Doctests available
        - python3 -m doctest <labfilename>.py to run autograder
            - Must be in correct parent file

Method 2.4: like_a_post_in_page(self, title_of_post)

- Objective:
    - Write a method that likes a post in the meme page and throws an “error” if the title of the post is not found
- Notes:
    - Utilize f-strings!
    - The “error” isn’t a literal error message, just a personalized string. See doctests for example
- Inputs:
    - self = N/A
    - title_of_post = the title of the post to be liked
- Output:
    - Reports: String
    - Your output should be a message indicating the activity and number of likes on the post
- Examples:
    - Doctests available
        - python3 -m doctest <labfilename>.py to run autograder
            - Must be in correct parent file

You can always check the validity of your solutions by using the local autograder. Remember to submit on Gradescope!