COMP1005/1405 Fall 2024 Tutorial 9

Do not post the tutorial or the tutorial and/or solutions on any website.

Objectives

- Practice writing/running Python code in the VSCode
- Practice coding **Dictionaries and Try-Except.**

Expectations

To receive full grades for this tutorial, you must complete Problems 1-3.

Grading Scheme for Tutorials

For each tutorial, you will be graded based on the following scale:

- 2/2 for demonstrating problems 1-3 and your **YourStudentNumber_T9.zip** file
    - Section A Tutorial Sessions: submit your work to the tutorial BrightSpace. The zip file should contain your solutions to all the required problems.
    - Section C, D, and E Tutorial Sessions: By the end of the tutorial session, demonstrate your tutorial work in person to a teaching assistant.
- 1/2 if you are missing problems or your solutions need significant improvement.
- 0/2 if you do not submit to BrightSpace or demonstrate to a teaching assistant
    - Section A Tutorial Sessions: no submission to the tutorial BrightSpace
    - Section C, D, and E Tutorial Sessions: not demonstrating your tutorial work to a teaching assistant by the end of the tutorial session

---

Problem 1 (Dictionary)

Write a Python file named **`fruit.py`** in VSCode to count the occurrences of each fruit in a list. Follow the steps to complete the function with the sample list.

1. **Define the count_fruit function:**
    1. Define a function named `count_fruit` that takes one parameter, `fruit_list`.
    2. Inside the function, create an empty dictionary called `fruit_count` that will store the counts of each fruit.
    3. Use a `for` loop to go through each fruit in `fruit_list`.
    4. Within the loop, count each type of fruit separately, using the fruit name as the key and the quantity of that fruit as the value. For each iteration:
        - Check if the name of the fruit is already a key in `fruit_count`
        - If it is, increase the count for that fruit by 1.
        - If it's not, add the fruit as a new key with an initial count of 1.
    5. Return the fruit_count dictionary.

    Hint:

    - Use `if fruit in fruit_count` to check if the `fruit` exists in the dictionary.
    - Using `dic[key] = value` will update the value if the key already exists in the dictionary; if the key doesn't exist, it will create a new (key, value) pair.

2. **Define the main function:**
    1. Please use the sample input below to test the functionality, but feel free to modify it with any input you prefer.
    2. Call the `count_fruit` function to convert the list into a dictionary and print the resulting dictionary returned by `count_fruit`.

Sample Input:

fruits = ["apple", "banana", "apple", "orange", "banana", "apple", "grape", "orange"]

Sample Output:

```
{'apple': 3, 'banana': 2, 'orange': 2, 'grape': 1}
```

Problem 2 (Try-Except)

Write a Python file named **`errorPractice1.py`** in VSCode to practice applying the try-except in your project.

1. **Define the convert_to_int function:**
    - Define a function named `convert_to_int` that takes a string `myString` and tries to convert it to an integer.
    - If the conversion fails (e.g., if `myString` is not containing a valid integer), the function should capture a **`ValueError`** using a `try-except` block and return the string `"Invalid input"`.
    - If the conversion is successful, then return the integer.
2. **Define the main function:**
    - Step 1: Prompts the user to enter an integer.
    - Step 2: Call the `convert_to_int` function to convert each input to an integer.
    - Step 3: Print the results of each conversion.
    - Repeat steps 1 - 3 three times.

Sample Output:

```
Enter a value: 10
10
Enter a value: 20
20
Enter a value: abc
Invalid Input
```

Problem 3 (Try-Except)

Write a Python file named **`errorPractice2.py`** in VSCode to divide two numbers entered by the user safely.

1. **Define the `safe_divide` Function:**
    1. Define a function named `safe_divide` that takes two parameters, `a` and `b`.
    2. Inside the function, use a `try-except` block to perform the division.
    3. If `b` is zero, the function should capture the **`ZeroDivisionError`** and return the string `"Division by zero is not allowed"`.
    4. If `b` is not zero, return the **floating point** result of dividing `a` by `b`.

2. **Create the `main` Function:**

    1. Create the `main` function to handle user input and calls the `safe_divide` function.
    2. Inside the `main`, use a `while` loop to repeatedly ask the user to input two floating point numbers (for `a` and `b`).
    3. After taking the input, attempt to convert both values to floats. If the user enters a non-numeric value for either input, raise a **`ValueError`** by catching it with a **`try-except`** block and displaying an error message (e.g., `"Invalid input, please enter numeric values."`).
    4. If both values are valid, pass them as arguments to the `safe_divide` function and print the result.
    5. After each division, ask the user if they want to perform another calculation. If the user enters `"quit"`, exit the loop; otherwise, continue asking for inputs.

    Hint: Integer values can also be converted to floating-point values.

Sample output:

```
Enter the first number (a): 10
Enter the second number (b): 2
Result: 5.0
Do you want to perform another calculation? (type 'quit' to exit): no

Enter the first number (a): 10
Enter the second number (b): 0
Result: Division by zero is not allowed
Do you want to perform another calculation? (type 'quit' to exit): quit
```

Final Step

For Section A (Submit the work before the tutorial ends):

1. **Submit** your **zip** file to our Merged Tutorial Brightspace. The due date of your submission is aligned with your tutorial session.
2. **After** you submit the file, download your submission from Brightspace and confirm that it is a zip file containing **`fruit.py`, `errorPractice1.py`, and `errorPractice2.py`.**
3. **Extract** the .py files and execute the extracted files again to ensure they work properly. Occasionally, a problem can occur during uploading, and files can become corrupted.

For Sections C, D, and E (Show the TAs your work before the tutorial ends):

1. **Problem 1-3:** Run your Python programs in VS Code to demonstrate they are working.
2. Answer the questions the TA may ask.
3. Show the TA your zip file and extract the files.