COMP1005/1405 Fall 2024 Tutorial 6

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

Objectives

- Practice writing/running Python code in Visual Studio Code
- Practice applying **Loops and Function Creation** in the code

Expectations

To receive full grades for this tutorial, you must complete Problems 1-2.

Grading Scheme for Tutorials

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

- 2/2 for demonstrate problems 1-2 and your **YourStudentNumber_T6.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: demonstrate your tutorial work in person to a teaching assistant by the end of the tutorial session.
- 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 (Branching + Loop + String)

Write a Python file named **`replacement.py`** in VSCode to implement the string-related program.

1. Based on the description below, create the program with as many functions as needed. The function(s) should follow the naming requirement in **Lecture 9, Page 19**. Each function will need a function description; refer to **Lecture 10** sample code for a documentation example.

2. The program asks the user for inclusive positive integers in the range [1,10]
   - Stores the user input in the list until the user enters -1.
   - Print out all the elements of that list at the end.
   - Assume the user will given values within the specified range.

3. The program repeatedly asks the user to enter two inputs (a number num and a string data)
   - Replace all the corresponding num in the list with data.
     - For example, if num is 1 and data is “one”, it must replace all 1’s in the list with “one”.
   - Print the updated list and the number of times a number is replaced after the replacement is done.
   - The program stops when the user enters ‘q’ or “Q”.

   Hint: Use a for (counter-controlled) loop inside a while (event-controlled) loop.

Sample Output: **(Assume user inputs are 1,3,3,9,5,7,8,2,10,8,4,1,1, -1) one value per line**

```
List: [1, 3, 3, 9, 5, 7, 8, 2, 10, 8, 4, 1, 1]

Enter a number to change ['q' to quit]>> 1
with string >> one
List: ['one', 3, 3, 9, 5, 7, 8, 2, 10, 8, 4, 'one', 'one']
The number 1 was replaced 3 times.

Enter a number to change ['q' to quit]>> 8
with string >> eight
List: ['one', 3, 3, 9, 5, 7, 'eight', 2, 10, 'eight', 4, 'one', 'one']
The number 8 was replaced 2 times.

Enter a number to change ['q' to quit]>> q
```

Problem 2 (Branching + Loop + String)

Write a Python file named **`christmasTree.py`** in VSCode to use loops and functions to generate a Christmas tree made of stars (*)

1. Continuously prompt the user to input the desired height of the tree until they choose to “quit” the program.
2. Assume the user will only provide either integer values or the case-insensitive input "quit". **(Hint: Event-Control Loop)**
    a. If the input is an integer (height of the tree):
        i. Call the function checkError() to check if the user’s tree size input is valid.
        ii. Call the function generateTree() to draw the tree.
    b. If the input is "quit", exit the program.

Functions:

1. **Error Checking Function, checkError():**
    - The function will take user input as a parameter and return the error message if applied.
    - If the user input is less than 2, respond with: "Looks like we've got a baby tree here – not quite ready for Christmas cheer."
    - If the user input exceeds 10, respond with: "Whoa, that tree is too big for my cozy living room. Let's find one that fits just right."
    - If the user input is between 2 and 10, inclusively, draw the tree.
2. **Tree Generation Function, generateTree():**
    - The function will take user input as a parameter and display a tree.
    - Use loops to structure and print the tree layers with the correct number of stars.
    - Ensure the tree maintains symmetry in its triangular shape by correctly indenting each layer.
    - Represent the stars based on the specified height and display the final tree output.

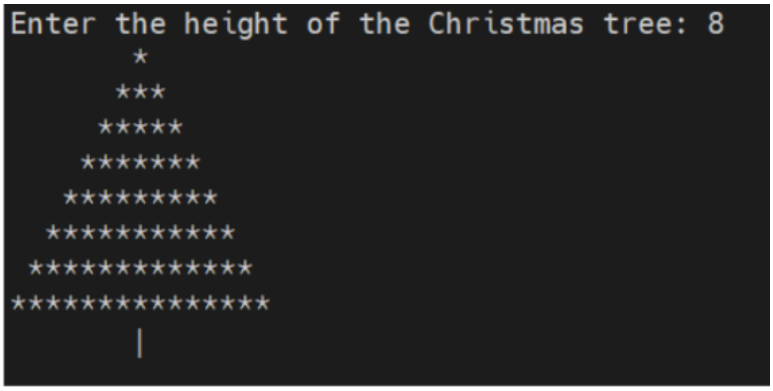
Hints:

- The formula for the number of stars in the i-th layer is 2 * i + 1. **(i starts at 0)**
- The number of spaces is related to the difference between the tree height and the current layer (height - i - 1).
- After printing the spaces and stars for a layer, use printf("\n") to move to the following line for the next layer.

(Optional Task):

Implement the Christmas tree program. The character 'O' should replace stars (*) at odd positions in this tree.

Example Output: (The height refers only to the leaves (*), excluding the trunk.)



Compress Files (zip files)

1. Create a directory/folder and copy/move all your tutorial files.
2. Enter this directory, select all files, and create a zip file as described above. Name it `YourStudentNumber_T6` (replace **YourStudentNumber** with your 9-digit student number).

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 **replacement.py, and christmasTree.py.**
3. **Extract** the .py files and execute those again to ensure they work correctly. 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 1-2:** 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.