COMP1005/1405 Fall 2024 Tutorial 5

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

Objectives

- Practice writing/running Python code in the terminal
- Practice applying **Loops** in the code

Expectations

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

Grading Scheme for Tutorials

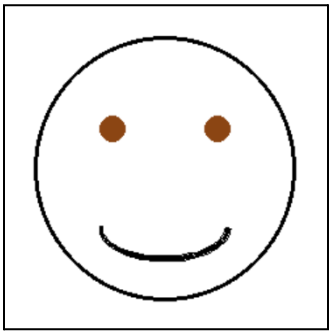
For each tutorial, you will be graded based on the following scale:

- 2/2 for demonstrate problems 1-3 and your **YourStudentNumber_T5.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 or being unable to answer the teaching assistant's questions about your tutorial. **Note: You can not email the TA your tutorial code for a grade if you missed the tutorial.**

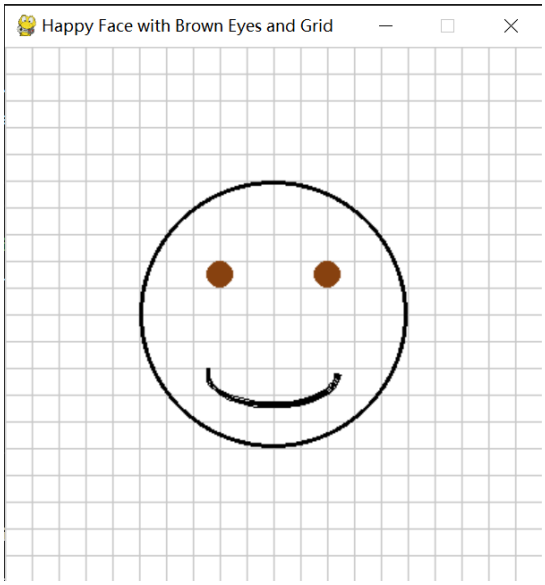
Problem 1 (Pygame)

1. **Install pygame**
   a. Before running the program, you need to install Pygame.
   b. You can install it using pip: **pip install pygame**
   c. Including the Pygame library in your code: **import pygame**
   d. Read the functions as needed by visiting the official website:
   https://www.pygame.org/docs/ref/draw.html

1. Write a program called **smile.py** that displays the smiley face. The example output is shown on the right:



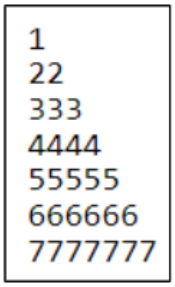
   a. Create a Pygame window of size 400 * 400 pixels.
   b. Add a grid to the drawing (**By hand or by code**):
   (Optional) Function: **pygame.draw.line()**
   (Optional) Hint: Apply the for loop with a range function with the desired grid spacing as steps to generate Vertical/Horizontal lines.



   c. **Define colors:** Black(0,0,0), White(255,255,255), Brown(139,69,19).
   d. Divide the drawing:
      i. Face: Large black circle. (**pygame.draw.circle()**)
      ii. Eyes: Smaller black circle filled in color brown.
      iii. Mouth: black arc (**pygame.draw.arc()**)
      Smile: Starting angle: 3, ending with 0
      (**Optional**: Sad: Starting angle: 0, ending with 3)
   e. **Implement the code**, display for 5 seconds, and save the image. You don't need to include the image in your tutorial zip file.

Problem 2 (Basic Loop)

2. Write a Python file named **triangle.py** in VSCode to implement a “number triangle”.
    - Prompt the user to enter an integer representing the number of rows in the "number triangle".
    - Build up a loop to construct each row of the triangle.
    - Each row should consist of a repeated sequence of digits—hint: **a string or character multiplied by an integer will repeat itself.**
    - The length of the sequence in each row should match the row number (starting from 1).
    - The digit for each row should be the same as the row number (e.g., '1' for the first row, '2' for the second row, etc.).
    - The example output is shown below:



Problem 3 (Multiplication Tables)

Write a program called **multiplication.py** that asks the user for a number between 1 to 9 (both inclusive) and displays a well-formatted multiplication table of all numbers up to the user's input.

Your solution should not hardcode the entire table in print statements, but rather, use nested loops to print each number based on the user's input.

Hints:

- Use nested for-loops. The **outer** for-loop can work with the rows (values on the horizontal lines), and the **inner** for-loop can work with the columns (values on the vertical lines).
- Use tab (“\t”) to format the columns of the table.

- Remember that the print function has an optional parameter **“end”** that can be overridden. For example: print(‘hello’, end=’\t)’.

Sample Outputs:

```
Enter a number (1-9): 5

   1      2      3      4      5
   2      4      6      8      10
   3      6      9      12     15
   4      8      12     16     20
   5      10     15     20     25


Enter a number (1-9): 9

   1      2      3      4      5      6      7      8      9
   2      4      6      8      10     12     14     16     18
   3      6      9      12     15     18     21     24     27
   4      8      12     16     20     24     28     32     36
   5      10     15     20     25     30     35     40     45
   6      12     18     24     30     36     42     48     54
   7      14     21     28     35     42     49     56     63
   8      16     24     32     40     48     56     64     72
   9      18     27     36     45     54     63     72     81
```

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_T5` (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 it from Brightspace and confirm that it is a zip file containing **smile.py**, **triangle.py** and **multiplication.py**.
3. **Extract** the Python files and execute them again to ensure they work properly. Occasionally, a problem can occur during the upload process, 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 files in VS Code to demonstrate it is working.
2. Answer the questions TA may ask.
3. Show the TA your zip file and extract the files.