15-110: Principles of Computing, Spring 2018

Lab 10 – Thursday, April 5

Goals

This lab is intended to continue your exploration of graphics with Python. We will be using the module tkinter to create simple computer graphics on the screen. In this lab, you will create a very simple game that has a graphical component.

When you are done with this lab, you should be able to do the following:

1. Draw rectangles, circles, lines and text on a canvas.
2. Use the built-in Python function input(prompt) to read input from the standard input (keyboard). The parameter prompt is a string that is shown to the user to give instructions, and the input function returns a string with what is typed in as a response to the prompt.
3. Given an existing program with a missing component and a specification of what the resulting program should do, determine how to integrate the newly written code with the existing code.

Part 1: TA Demonstrations (Graphics in Python)

- Review the function `input_num(prompt)` below and demonstrate how to use it:

```
def input_num(prompt):
    while True:
        s = input(prompt)
        if s == "quit": return "quit"
        if s == "0": return 0
        if s == "1": return 1
        if s == "2": return 2
        print("TRY AGAIN: input 0, 1, 2 or quit")
```

- Review how to draw on the canvas:

```
c.create_rectangle(x1, y1, x2, y2, fill=color)
c.create_oval(x1, y1, x2, y2, outline=color)
c.create_line(x1, y1, x2, y2, width=size, fill=color)
c.create_text(x1, y1, text, font=style, fill=color)
```

- Review the function `play()` to observe how the given program for the game Tic-Tac-Toe is structured.

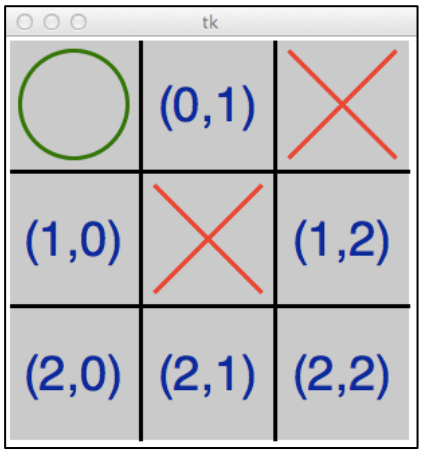
Overview of Tic-Tac-Toe

In this lab, you will develop a program that allows two players to play Tic-Tac-Toe. In Tic-tac-toe, two players alternate placing their marks ("X"'s and "O"'s, respectively) in one of the 9 positions of a 3x3 grid. The first player to put three of their marks in a vertical, horizontal, or diagonal line is the winner. If nine marks have been placed without either player getting three marks in a row, the game ends in a tie.

Download the starter code from Autolab (tic_tac_toe.py). Store in a private folder named lab10.

In order to represent the state of play in a game of Tic-tac-toe, we will use a two-dimensional 3x3 list, where each element is None (if the corresponding position is unoccupied), 0 if the position is occupied by the mark ("X") of the first player ("Player 0"), or 1 if the position is occupied by the mark ("O") of the second player ("Player 1"). For example, the Tic-tac-toe grid shown at right could be represented by the 2d Python list:

```
[[1, None, 0],
 [None, 0, None],
 [None, None, None]]
```



The function `new_grid` creates the data representation for a blank 3x3 Tic-tac-toe grid. The function `add_mark` takes parameters `grid`, `row`, `column`, `player`, and modifies `grid` to place `player`'s mark at the position specified by `row` and `column` as long as that position is unoccupied. If the position was unoccupied, then after modifying the `grid`, `add_mark` returns `True`. Otherwise (i.e., the position was already occupied), `add_mark` returns `False`.

Activities

1. Graphical Display of the Board

Complete the Python function `display_grid(grid)` that draws a game state (stored in a list of lists) to the canvas. The following algorithm may be used for `display_grid`:

I. Create a rectangle covering the entire 300 x 300 Canvas and filled with gray.
II. Draw the horizontal and vertical lines separating the positions in the grid.
III. For each grid position (rows in 0..2, columns in 0..2), do the following:
    A. If the *grid* indicates that the position should hold an "X", then draw an "X" in the square representing the specific row and column. (HINT: This requires two diagonal lines.)
    B. Otherwise if the *grid* indicates that the position should hold an "O", then draw a circle in the square representing the specific row and column.
    C. Otherwise, draw the coordinates of the row and column as text.

The following usage should result in the image above:

```
>>> display_grid([[1,None,0], [None,0,None], [None,None,None]])
```

2. Checking for a Win

Read through the Tic Tac Toe program and study how the `check_win` function works. It calls `check_win_horiz` and `check_win_vert`. Once you understand how it works, sit with another student and explain the function to each other so you are sure you both know how they work.

Now, on your own, complete the functions `check_win_diagonal1` and `check_win_diagonal2`. Test the game to see that your functions work correctly.

Once you are done, you should be able to read through the entire code and understand how everything in this game works.

3. Cleaning up the Code

As you read through the function `check_win`, you should have had the feeling that there could be a shorter way to program that function (i.e. using fewer instructions). Revise that function to have as few lines of code as you possibly can.

Submission

When you finish the lab, you should be inside the `lab10` folder, which is inside the `private/15110` directory. When you type `ls` and press the Enter key, you should see the following file: `tic_tac_toe.py`. Once you see this file, please type `cd ..` to move up one folder and press the Enter key. Then, zip your `lab10` folder by typing `zip –r lab10.zip lab10` and you should see a `lab10.zip` file in the current folder if you type `ls`. Please submit the zipped file `lab10.zip` on Autolab under 'lab 10'.