Project #2

Due Date

This assignment is due by 11:59 pm on **Monday, April 1st**.

- Submit it to the *Project 2* assignment on Blackboard
- *Late submissions will receive a 50% grade deduction*
- You may submit up to 3 times

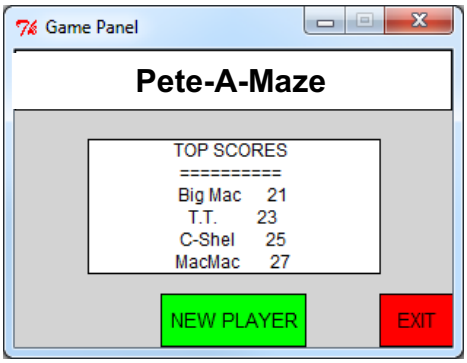
NOTE: Read the entire Project 2 document before starting this assignment

The Project Description: Pete-A-Maze

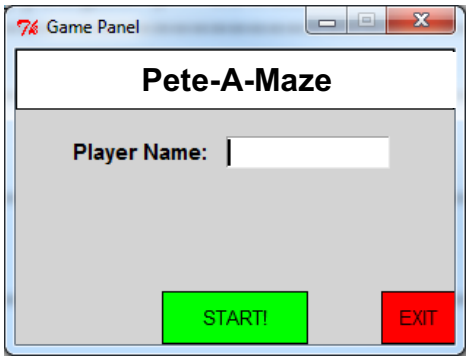
Purdue is celebrating its 150th anniversary and in honor of the occasion, you have been asked to develop a new game, the **Pete-A-Maze**. Players attempt to complete a maze in the least number of moves, crossing as few tripwires as possible. **Pete-A-Maze** tracks the players' names, moves and scores and displays the *Top Scores*.

The Game Panel: Do you want to play a game?

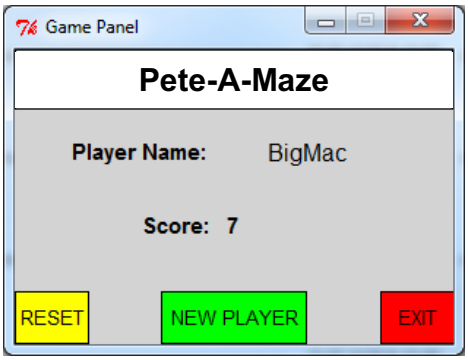
Pete-A-Maze starts by displaying the *Game Panel*, a 300x200 *Graphics* window as shown below which has controls that change based on user input. Initially, the *Game Panel* includes the *Pete-A-Maze* title, top 4 scores, and an option to either start a new game or exit the program.



Initial Game Panel



Game Panel after clicking NEW PLAYER



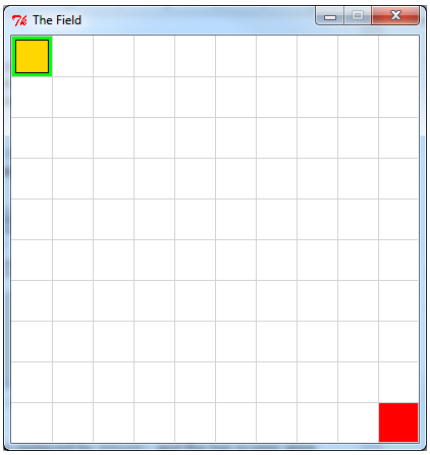
Game Panel during game play

When the user clicks on `NEW PLAYER`, the `NEW PLAYER` label is replaced by `START!` and the top scores area of the Game Panel is undrawn and replaced with an *Entry* box labeled `Player Name:`

During game play, the `START!` label is replaced by `NEW PLAYER`, the player's name and current score are shown and a new yellow control area labeled `RESET` is available.

The Field: Where `Pete` lives and plays

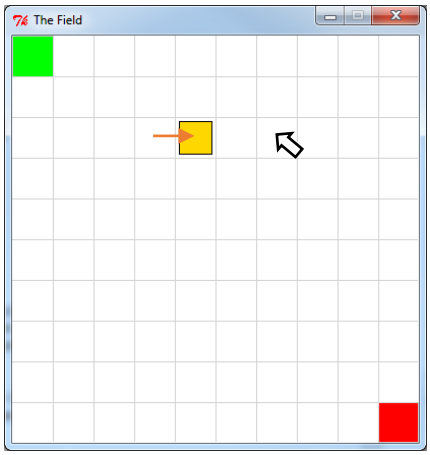
When a new game starts, the *Field* is drawn using a 400 x 400 Graphics window with a white background and a 10 x 10 grid of light-grey lines. Each grid is a 40x40 pixel square. The top-left grid is filled with green and the bottom-right corner is red.



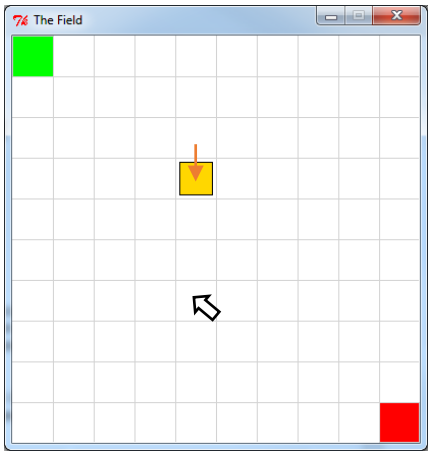
The Grid and Pete at the start

To play the game, the user controls `Pete` who is represented by a 36x36 gold square in the upper-left corner of the grid. The objective of the game is to navigate from the upper left hand corner to the lower right hand corner in as few moves as possible. The player's score is based on the number of moves it takes for `Pete` to reach the objective.

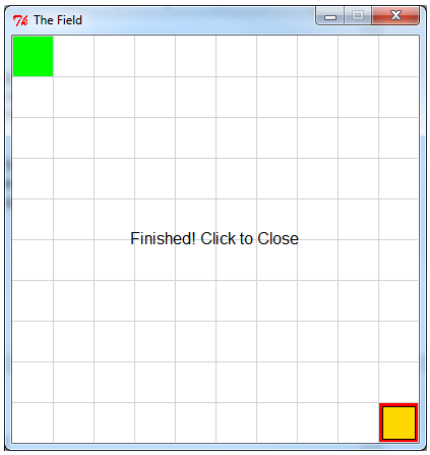
`Pete` can be moved only one grid at a time, either horizontally or vertically by clicking in any open grid in the same row or column. `Pete` *cannot* move diagonally, only up, down, right or left one grid.



*A click in his row moves Pete right **or** left*



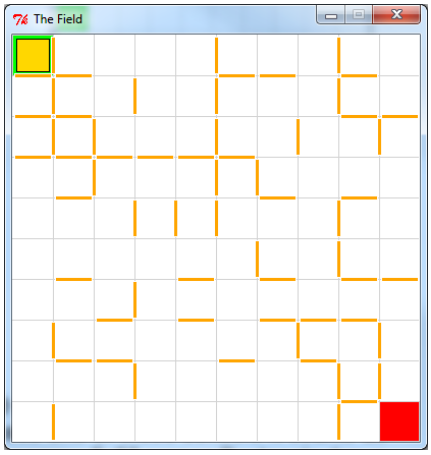
*A click in his column moves him up **or** down*

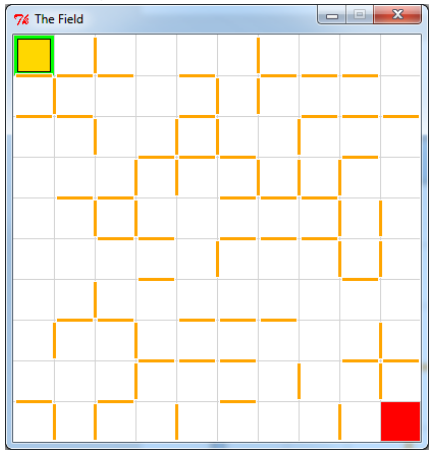


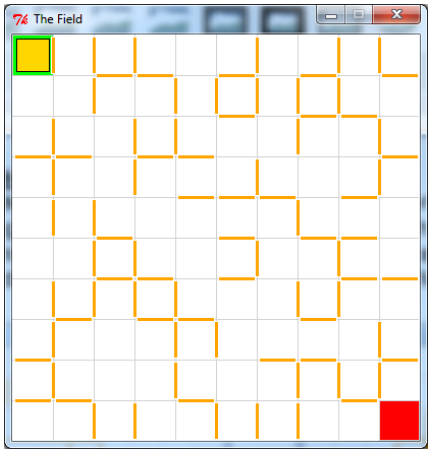
Pete reached the red square -- Game Over!

The Sensors: Prepare to Amaze!

Pete-A-Maze is challenging due to a series of *sensors* drawn on the grid. Each vertical and horizontal grid border has a 40% chance of including a sensor. As the player moves `Pete` through the grid, if he crosses one of these sensors *it counts as three (3) moves* in the player's score instead of one. The sensors are represented by a narrow, 5x36 orange *Rectangle* drawn on the grid border and each border has a 40% chance of including a sensor. Some examples of a **Pete-A-Maze** grid are shown below.







About the Project: The Tasks

In this project, you will complete several tasks:

1. Setup your `project2.py` file
2. The `panel` Graphics window
3. The `field` Graphics window
4. Animating Pete
5. Start to finish
6. Border crossings
7. Getting sensitive
8. Scoring the game
9. Top 4 scores

TODO #1: Setup your `project2.py` file

The Python program file should be setup with the following guidelines:

- File name is `project2.py`
- All library import statements should be at the top of the file
- File includes a header describing program, its purpose and function
- Header includes your name, the program name and a description of its function
- All Python code should be contained within function definitions with the exception of:
  - Library import commands
  - Header comments
  - Function definition statements themselves -- i.e. `def main():`
  - The `main()` statement which starts the program

TODO #2: The `Game Panel` Graphics window

Write a separate Python function that creates a 300 x 200 Graphics window and all the required graphics objects to create the `Game Panel` shown in the project description. You may rearrange the component groups if you choose, but they must all still be included. For this part of Project 2, focus on *creating and drawing* Graphic objects for the initial state of the `Game Panel` first, then work on the components for the additional program states.

The **Pete-A-Maze** title and `EXIT` control should be displayed at all times. If a mouse click is detected on the `EXIT` control at any time, (whether a game is active or not), the program should close all *Graphics* windows and terminate. The following table clarifies the `Game Panel` components that should be displayed. This function should **return** all of the graphics objects created.

Program State	**`Game Panel` Components Displayed**
Initial state at program start, no active game	Top 4 scores *Text* area and `NEW PLAYER` control
If `NEW PLAYER` clicked	`NEW PLAYER` control label is changed to `START!`, `Player Name:` *Text* label and *Entry* box
Game is active if `START!` clicked ***and*** `Player Name:` is not empty/blank	`RESET` and `NEW PLAYER`, player name and current, players score *Text* objects
Game is over	Top 4 scores *Text* area, `NEW PLAYER` and `RESET` controls

HINT: You can create graphics objects and then choose when to `.draw()` *or* `.undraw()` *them as needed.*

TODO #3: `The Field` Graphics window

Write a separate Python function that creates `The Field` and `Pete` as defined in the program description. *This Graphics window must match the specified design precisely*.

This function should perform the following tasks:

1. Define a 400 x 400 pixel *Graphics* window with a white background
2. Draw 10 x 10 grid pattern filling the *Graphics* window -- each grid is a 40x40 pixels with a light-grey outline
3. Draw a 40x40 green *Rectangle* with a light-grey outline in the top-left grid
4. Draw a 40x40 red *Rectangle* with a light-grey outline in the bottom-right grid
5. Draw `Pete`, a 36x36 gold *Rectangle* centered in the top-left grid
6. **Return** the `Field` and `Pete` objects

TODO #4: Animating Pete

Write a separate Python function that enables the user to move `Pete` around the `Field` grid. Use the following to help determine the operation of this function:

1. If a mouse click is detected in the same grid row as `Pete`, move one grid horizontally towards the click
2. If a mouse click is detected in the same grid column as `Pete`, move one grid vertically towards the click
3. Mouse clicks detected other than in the same grid row or column as `Pete` are ignored
4. Mouse clicks detected in the same cell as `Pete` are also ignored

HINT: Mouse clicks anywhere in the same grid row or column as `Pete` *should cause him to move.*

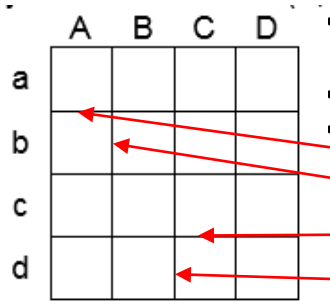
TODO #5: Start to finish

The objective of the game is to navigate `Pete` from the upper left (green) to the lower right (red) cell in the grid. Modify the function that enables `Pete` to move to meet the following requirements:

1. If `Pete` is in the same cell as the red *Rectangle* (the bottom-right grid) the game is over
2. When the game is over, draw a *Text* object containing the *String* `Finished! Click to Close` centered in the `Field` *Graphics* window
3. When the game is over, a click anywhere in the grid should close the Field window

TODO #6: Border crossings

It is important to track movements in the `Field` *Graphics* window. Specifically, your program must detect which border `Pete` crosses with each movement. You may use any technique for this portion of the Project – however one approach is described here:



An example 4x4 grid

- Think of the Graphics window as a grid with each column and row represented by letters – *in this example, the columns are* `A,B,C` and `D` *and rows are* `a,b,c` and `d`.
- Across each row (and down each column), number each border starting at 1
- For this example, the borders would be identified as follows:
  - `A1` – the first horizontal border in column `A`
  - `b1` – the first vertical border in row `b`
  - `C3` – the third horizontal border in column `C`
  - `d2` – the second vertical border in row `d`
- Keep track of the number of times Pete moves, and which border is crossed with each movement. This will be used in **TODO #8: Scoring the game**

Note: Your `Field` *Graphics window will be a 10x10 grid*

TODO #7: Getting sensitive

Each border has a 40% chance (4-in-10) of including a sensor represented by a narrow orange band as shown in the project description. Write a separate function that completes the following tasks:

1. Loop through each column and randomly decide whether to draw a 36x5 horizontal orange *Rectangle*
2. Loop through each row and randomly decide whether to draw a 5x36 vertical orange *Rectangle*
3. Define a *List* including the border locations where an orange sensor *Rectangle* was drawn
4. **Return** the *List* of sensor locations
5. Modify the function that creates the `Field` *Graphics* window to create the sensors

TODO #8: Scoring the game

Calculate the game score by counting the total number of moves necessary for `Pete` to reach the lower right (red) grid. Moving across a border where a sensor is drawn counts as three (3) moves. Modify the function that enables `Pete` to move to meet the following requirements:

1. Add one (1) to the game score when `Pete` moves
2. Add three (3) to the game score when `Pete` moves across an orange sensor
3. Display the game score in the `Game Panel` as shown in the project description
4. Update the score every time `Pete` is moved

HINT: Compare the border crossing you determined in **TODO #6** *with the List of sensor locations returned by the function in* **TODO #7***.*

TODO #9: Saving and displaying the top 4 scores

In **TODO #2**, you designed the Game Panel which displays the top 4 scores. These are read from a data file named `top_scores.txt` which can be downloaded from the Project 2 assignment on Blackboard. Modify your program to perform the following tasks:

1. Write a separate function named `scoresOut()` that updates the data file `top_scores.txt` to include the score from the most recent game
2. The `scoresOut()` function should add the most recent score, but not remove any of the existing scores in the file
3. The `scoresOut()` function should not **return** any values
4. Write a separate function named `scoresIn()` that reads the data file `top_scores.txt` and stores the values in a *List*
5. The `scoresIn()` function should **return** only the lowest four (4) scores from the *List*
6. Modify the function that creates and updates the Game Panel window to to use the List returned by `scoresIn()` to create the Text object in the top 4 scores panel
7. Modify your program so that when a game ends (TODO #5) the `scoresOut()` function is called and the player name and score from the most recent game is written to the `top_scores.txt` data file.

NOTE: The objective of the game is to make the least number of moves. Lower scores are the best scores.

Submit to Blackboard

- Upload your completed file (`project2.py`) to Blackboard by the due date.
- This assignment is due by **11:59 pm** on **Monday, April 1st**.
- *Late submissions will receive a 50% grade deduction*
- You may submit your Project 2 assignment up to 3 times

Project 2 Grading Rubric	**Points**
TODO #1: Setup your `project2.py` file	
File name is `project2.py`	1
All library import statements are at the top of the file	1
Program header included with student name, program name and short description	1
All Python code is contained within functions	2
TODO #2: The `Game Panel` *Graphics* window	
Separate function creates the Game Panel window	4
Initial state -- includes all specified components and functionality	4
`New Player` *clicked* -- includes all specified components and functionality	4
Game is active -- includes all specified components and functionality	4
Game is over -- includes all specified components and functionality	4
TODO #3: The `Field` *Graphics* window	
Separate function creates the `Field` *Graphics* window as specified	4
`Field` includes 10 x 10 grid of light-grey lines -- each grid is a 40x40 *Rectangle*	6
A 40x40 green *Rectangle* with a light-grey outline is in the top-left grid	2
A 40x40 red *Rectangle* with a light-grey outline is in the bottom-right grid	2
`Pete` is drawn as a 36x36 gold *Rectangle* centered in the top-left grid	4
The `Field` and `Pete` objects are **returned** by the function	2
TODO #4: Animating Pete	
Separate function written to move `Pete`	4
Mouse click are detected and `Pete` moves towards the click as specified	8
Mouse clicks directly on `Pete` or outside his row/column are ignored	8
TODO #5: Start to finish	
Game ends when `Pete` is in the same cell as the red	5
Text `Finished! Click to Close` displayed when game ends	5
When the game is over, a click anywhere in the grid closes the `Field` window	5
TODO #6: Border crossings	
Border locations are detected as `Pete` moves	20
TODO #7: Getting sensitive	
Separate function draws sensors as specified	15
List of sensor locations is **returned**	5
TODO #8: Scoring the game	
Score is calculated as specified	10
TODO #9: Saving and displaying the top 4 scores	
`scoresIn()` function reads values from `top_scores.txt` file	3
`scoresIn()` only returns the best 4 scores	3
`scoresOut()` function adds most recent player and score to `top_scores.txt`	2
`scoresIn()` and `scoresOut()` functions are called as specified	2
Style / Format: Python code is formatted, ***commented*** and easy to understand	10
Total Points	**150**