Escape the Maze: Pseudocode to code

CSC160, Prof. Phil Lombardo

(*A special thanks to Dr. Hank Feild for inspiring this assignment. It has been adapted from his original work with simple drawings to suit the needs of our class.*)

In this lab you'll get practice designing algorithms, writing pseudocode, and then translating the pseudocode it to real code.

Make your own individual copy of this lab. There are parts of this lab that you should do by yourself, and others you should do with your lab partner. Each of you should submit your completed personal copy.

You and your partner will each individually need one piece of graph paper for this lab.

# Team members

|  |  |
| --- | --- |
| **Your name:** |  |
| **Partner's name:** |  |

# Learning Objectives

By the end of this part, you should feel comfortable:

* designing an algorithm
* writing pseudo code
* translating simple pseudo code to Python code
* running a Python program on the command line
* using an application programming interface (API) documentation website

# Instructions

1. After breaking you into pairs, I have assigned each of you a ***secret maze number.****Don't let your partner know your maze number!*

Sit in such a way that your partner cannot see your computer screen! Put your assigned maze number in the box below.

|  |
| --- |
|  |

2. In VisualStudio Code, open the **lab2** directory for this lab. You can open a folder by going to "File" -> "Open Folder...". If you did this correctly, you should see five different maze images. (You can also double check by opening a terminal and printing the working directory with pwd.)

Now create a new text file named **lab2.txt**; text files are easy to create, just go to "File" -> "New Text File", then use Ctrl+S or Cmd+S to save it, giving it the name **lab2.txt**.

3. Design your algorithm.

In VS Code, click on the secret maze that you have been assigned. For example, if I was assigned maze 1, I would look at the file maze1.png. Imagine your partner is located at the red dot, but the lights are off so he or she can’t see. Even worse, the walls are electrified! Your task is to send your partner a text file to help them navigate the maze.

In the text file **lab2.txt**, write a pseudocode algorithm describing how your partner can escape the maze without touching any of the walls. Use the grid to aid in your instructions.

Remember, this is pseudocode; your guiding principle is that a reasonable person should be able to follow your pseudocode and escape the maze safely. Your pseudocode will most likely involve a lot of actions (move your pen here, draw a line between these two points, etc.).

Copy your pseudocode from your text file and paste it below.

|  |
| --- |
|  |

4. **Partner work.**

When both you and your partner are done with the previous steps, exchange your pseudocode (via Discord, email, or just swap laptops). Do not tell each other your maze numbers yet!

Follow your partner's pseudocode to create a path on a fresh sheet of graph paper. *Your* task is not to think about the maze at all, but to follow their pseudocode exactly. Hand them back the completed drawing of the path they described with their pseudocode.

When you receive the picture *your partner* created using *your* pseudocode, take a picture of it and paste it below.

|  |
| --- |
|  |

4. Does the picture produced by your partner using your pseudocode resemble the correct path out of your maze?

If not, why not? Was it a problem with your pseudocode or did your partner miss a step, do something out of order, etc.? If the cause is a bug in your pseudocode, fix the pseudocode in your **lab2.txt** file. In the box below, describe the issue and paste in the updated pseudocode if you did in fact update it.

If the picture your partner made closely resembles your original drawing, then say so in the box below. Otherwise, share what modifications you made.

|  |
| --- |
|  |

5. Create a new text file in VS Code name **lab2.py**. The ".py" extension will tell VS Code that this is a Python program and it will use colors and text decorations to highlight parts of the program accordingly.

In that file, paste the following code.

|  |
| --- |
| import turtle  import time  # Start a turtle object and a screen object  pat = turtle.Turtle()  screen = pat.getscreen()  # set Screen size in pixels  screen.setup(1050, 732)  screen.setworldcoordinates(-525,-358,525,358)  # modify the cursor look (just for fun :) )  pat.color('#a16bff')  pat.shapesize(2,2,1)  # set the screen background (we’ll change this soon)  screen.bgpic("maze0.png")  # move turtle to the start  pat.penup()  pat.goto(-512,185)  time.sleep(1)  pat.speed(3)  pat.pendown()  # TODO: Your escape algorithm goes below!  screen.exitonclick()  screen.mainloop() |

Save your program and then run it as follows. Open an integrated terminal in VS Code by going to "View" --> "Terminal". Type the following to run the program:

python3 lab2.py

What happens when you do that? Describe in words below.

|  |
| --- |
|  |

6. Exit the window that popped up by clicking anywhere inside it, or by clicking the "X" in the upper corner).

In the program file on the line below where it says "# TODO", add the following lines of code:

pat.forward(44)

pat.left(90)

pat.forward(88)

Make sure you save your program, then run it the same way you did before. Now what happens? Paste a screenshot below.

|  |
| --- |
|  |

7. Remove the Python statements you added in the previous step.

The statements from above use the Turtle library. You will now translate your pseudocode from earlier to Python using the Turtle library.

**Collaborate with your partner** on the remaining questions. Each of you will have your own Python program for your pseudocode, but you should write each of them together—this is something called paired programming and it has been shown to be effective in learning to program as well as producing higher quality code.

Before moving on, please make sure you understand the following: if you try implementing all of your pseudocode in one go, you will likely run into problems that will take quite a while to figure out and fix. Instead, **implement your pseudocode incrementally**: write a couple lines of code, then save and run to make sure it's doing what you want. If there's an issue, it's likely with the most recent lines you wrote. You're also less likely to repeat errors later on.

**Knowledge check:** How many lines of code should you implement before running when using incremental implementation?

|  |
| --- |
|  |

8. Before you start programming, take a look at the the [Turtle application programming interface (API) documentation website here](https://docs.python.org/3/library/turtle.html). The top of the Turtle API has some background and examples. The most useful bit is [the list of functions](https://docs.python.org/3/library/turtle.html#turtle-methods) (technically, these are a special kind of function called a *method*, but we'll talk about that more later this semester). This listing links to the documentation for each of the methods.

Find the documentation for each of the following functions and indicate what they do and what parameters (the values that are specified in the parentheses) they take. If it doesn't take a parameter, say "none". If it can be called with and without parameters, describe the one with parameters. I've filled in the first one for you:

|  |  |  |
| --- | --- | --- |
| **Function/method** | **Purpose** | **Parameters** |
| goto | Moves the turtle to the given spot on the canvas | x -- the x-coordinate to move to  y -- the y-coordinate to move to |
| forward |  |  |
| right |  |  |
| left |  |  |

9. *Write your code.*

Using the Turtle API for reference, implement your partner’s pseudocode as Python code, inserting it in in your **lab2.py** file just below the "# TODO: Your escape algorithm goes below" comment.

Due to the rectangular nature of the mazes, you will likely only need to use the three functions (methods) in the table above. However, if you want to get fancy, feel free to try some other functions as well. Search and browse the documentation to find out how to use other Turtle functions. In the code I provided, "pat" is the name of the Turtle instance, so use that instead of "turtle", which is what is used in the documentation.

Note that you may need to do things a bit differently than your pseudocode, and that's okay, but try sticking to your pseudocode as much as possible.

|  |
| --- |
| ***Coding Tip:*** An astute observer will notice that the width and height are different for the graph paper image. I recommend setting variables WIDTH and HEIGHT for these dimensions, so you can adjust them later if needed. Now, moving two widths in the graph paper can be accomplished by pat.forward(2\*WIDTH). I recommend setting the width around 48, and the height around 40. |

You are finished implementing once your program produces an image that closely resembles your original drawing. Paste your final code below:

|  |
| --- |
|  |

10. Take a screenshot of the drawing produced by your Python program and paste it below.

|  |
| --- |
|  |

11. Testing!

We are now ready to see if we can escape the maze. At this point, share your maze number with your partner. We will make one quick change before running the program you wrote above.

Above the code you wrote, locate the line of code that reads: screen.bgpic("maze0.png"). Replace the image name in quotes with the image for the maze of your partner. For example, if my partner had maze 3, that line would read screen.bgpic("maze3.png").

Now run your code and see how your algorithm does! Did you escape the maze?

If so, great. If not, it’s time to reflect:

* Can you fix the code with small adjustments? (*Hint:* think about adjusting the starting position, as well as the height and width variables.)
* Does the code require a change in the algorithm? If so, where?

Based on your reflection above, modify parts of your code so that the algorithm is successful. In the box below, describe the modifications that you made. (Do not past the code again; just describe your changes.)

|  |
| --- |
|  |

When you’re *both* successful, call me over to show me that you successfully escaped your mazes!

# Reflection

Please answer the following questions individually, though you may discuss your answers with your group.

1. What confused you most about this activity?

|  |
| --- |
|  |

2. What questions do you have either about the activity directly or about related topics that came to mind while you did the activity?

|  |
| --- |
|  |

3. Do you feel like you achieved all of the learning objectives listed at the top of each part that you completed? If you feel uncertain about any of them, please list them here.

|  |
| --- |
|  |