

## Activity 1.1.5

# Run Robot, Run

## Distance Learning Support

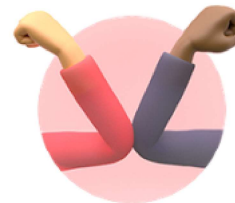
Check with your teacher about:

- ☐ What work you need to turn in and how to submit it
- ☐ Collaboration strategies
- ☐ If you are using a Chromebook, open a [blank code editor](#).



## GOALS

- Use existing algorithms in a program.
- Navigate a robot through a maze using loops and nested loops.



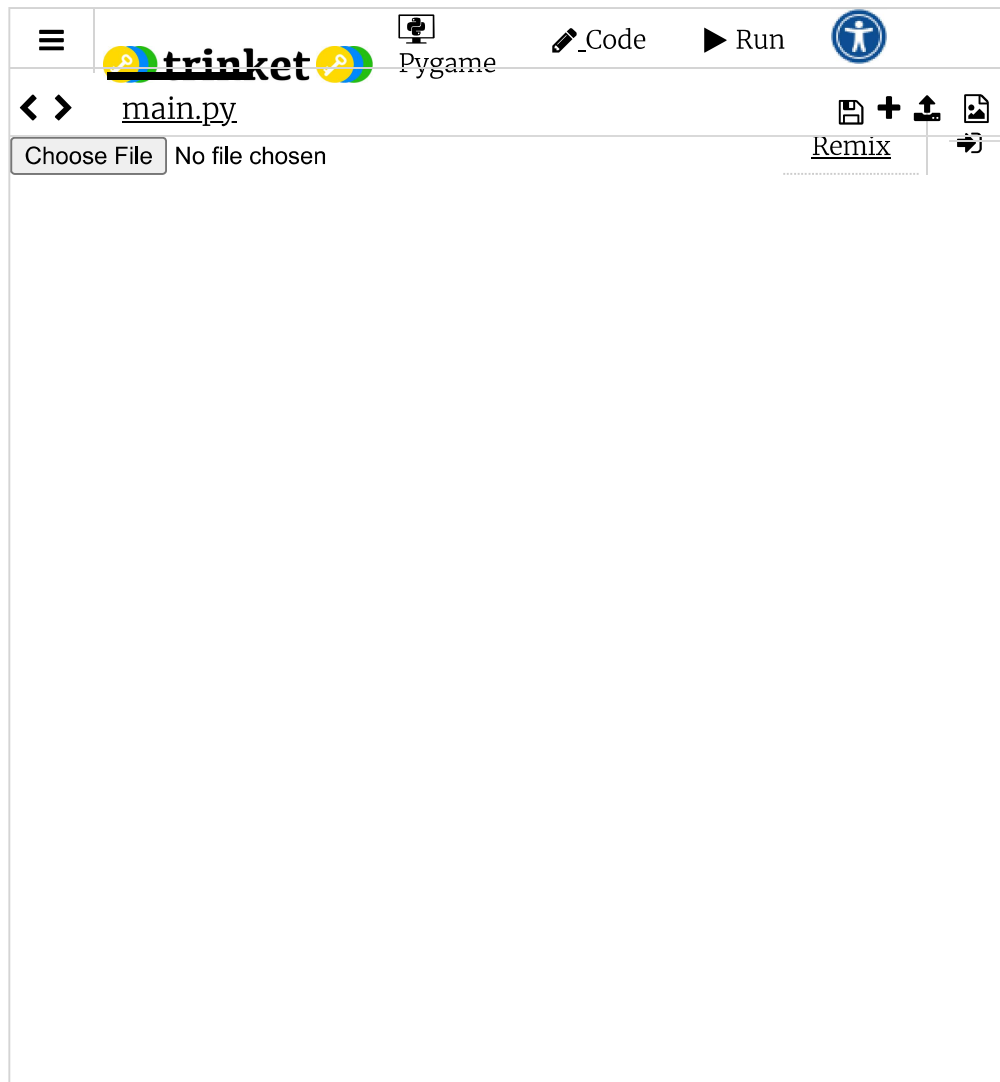


# Robot Maze

As a software developer, you will often be given existing algorithms to use in your program. These existing algorithms serve as building blocks for larger, more complex algorithms. They can reduce development time, reduce testing, and can simplify the identification of errors. They also can provide consistency when multiple developers are working on the same program.

As you gain experience, you will become familiar with common computer science algorithms, like the algorithm for determining whether a number is odd or even (using modulo). Knowing a variety of algorithms makes it easier to create variations from them and use them in new programs. In this activity, you will use existing algorithms and create some new ones to navigate a robot through a series of mazes.

Observe a robot maze program.



Near the top of the program, you will see two “robot algorithms” defined as *Python*® functions to move the robot turtle.

- The `move` function leaves a dot on the screen and moves the robot forward a certain amount.
- The `turn_left` function quickly turns the robot 90 degrees in the counter-clockwise direction.


### College Board Connection: Existing algorithms

In using these algorithms, you should not modify `move` or `turn_left`. Instead, observe the “TODO” comments for the modifications you do need to make.

- The first to-do comment tells you how you can change the maze image. The filename for the first maze, *maze1.png*, is provided for you in the code.
- In the second to-do comment, you will write iterative algorithms to move your robot. You may use only `move` and `turn_left` to move your robot.



- 1 View the images included in your program. In the code editor above, click on the image icon above and to the right of the editor window. You will see four images; three mazes and a robot. Click the editor tab to return to the code view.

When a program contains image files, you cannot copy and paste just the code into VS Code; you need to download the entire program so that you also have the necessary image files.

- 2 In the code editor above, select **Download** from the  menu at the top left.
- 3 A zip file is saved in your browser's *Downloads* folder. Extract the files so that you can access them from Visual Studio Code.
  - Extract the files to a new folder such as *1.1/115\_source*.
  - Once the files have been extracted and uncompressed, rename *main.py* to *robot\_maze1\_[studentinitials].py*.



### Download and Extraction Details

A **zip file**  is a **compressed**  file format and often contains multiple files. The compressed format saves space and makes downloads faster. Visit the [Download and Extract](#) resource for a step-by-step guide on extracting files.

4

Open your newly named *robot\_maze1.py* file in VS Code. In the Explorer panel, right-click on the folder that contains your maze program and select **Open in Terminal**. This will create a new Terminal window in the selected folder so that your program can access the image files.



**Helpful Tip:** To confirm you are in the correct directory and that the image files are in place, issue a **dir** command in the Terminal window. You should see the *.py* and image files.

5

Test that the original version of the program works. Use the command line in VS Code, **python robot\_maze1\_[studentinitials].py**.



### Helpful Tips

- In the terminal, once you have typed a few characters of the filename (such as **ro**) you can use the tab key to complete the program name.
- You can repeat previous commands with the up arrow.

A lot of the stuff that I liked from art and problem-solving was probably my favorite part of computer science. I liked figuring out a way to solve something and doing it in a way where the pieces felt like they worked together... finding the balance and then making it elegant.



Jenni Syed, Software Engineer

6

Implement the following robot navigation tasks, moving the robot from its starting position to the gray square(s), avoiding all black squares.

### Image problems?

- a. *Maze1*: Use three `for` loops to navigate through the maze.

### Hints

- b. *Maze2*: Create a copy of `robot_maze1.py` and save it as `robot_maze2.py`. Change the `bgpic` image to `maze2.png`. Use iteration to implement the two shortest routes to the gray square. One route will benefit from nested loops; remember to use different variables for the inner and outer loops. In between routes (not part of the iteration), reset the robot's location using `goto` and `startx` and `starty`.

### Hints

- c. *Maze3*: Create another copy of the maze program and save it as `robot_maze3.py`. Change the maze image to `maze3.png`. Navigate the robot to both gray squares in sequence, without starting over. Your algorithm should use

one outer loop and multiple nested inner loops. Change the robot's `penColor` when it reaches the first gray square.

### Hints



**Discussion Prompt:** Compare your solution for Maze 3 to your elbow partner's and confirm both use nested loops to produce the same result.

## CONCLUSION

- 1 You used existing algorithms to maneuver your robot through a maze. In what ways did the algorithms help or hinder your development?

Proceed to next activity