Python - Turtlympics - Sharing

Overview

Now that the mazes have been created and solved, its off to the races! You will need to show off your maze AND a turtle solving the maze. In doing so, you will describe the IPO of each function you created and discuss how you utilize an IDE to manage your project effectively.

Once mazes have been presented, you will have the opportunity to vote for your favourite mazes in regards to:

- Best Theme
- Best Aesthetics
- Most Complex

Project Criteria

Your task is to present your maze with a turtle that can successfully navigate the maze from start to finish. When you are presenting you must:

- Describe the IPO of each of the functions you created to help build your maze.
- Describe different parts of the IDE and how they help with your workflow.
- Run the program to show the maze and a turtle navigating the maze.

You will also create a short written reflection that meets the following criteria:

- No more than 300 words in length.
- Convince me that you have met the required outcomes for different steps of the project.
- Reference specific learning outcomes that you feel you exemplified.
- Include one change you would make if you had infinite resources (time and money) to your project.

Outcome Criteria

- **2 Demonstrate the ability to translate algorithms into working programs.**
  - **2.1** Describe the purpose of key parts of a programming development environment.
  - **2.3** Use the key components of a programming development environment to write and run programs.
  - **2.4** Convert algorithms into code while:
    - **2.4.1** maintaining an IPO structure
    - **2.4.2** including documentation
    - **2.4.3** using appropriate data types
    - **2.4.4** using variables and constants correctly
    - **2.4.5** supplying data using literals and input commands
    - **2.4.6** using appropriate operators
    - **2.4.7** displaying results clearly
  - **2.5** Test programs using appropriate data.
  - **2.6** Revise programs based on testing.

Mastery Rubric

	5 – Mastery of Skill	**4 – Excellence in Skill**	**3 – Proficiency in Skill**	**2 – Attempting Skill**	**1 – Starting to Attempt Skill**	**0 – Refusal to Engage with Skill**
2.1 Use a programming IDE.	Uses the IDE fluently during the presentation to manage files, run the program, and navigate code while clearly demonstrating control and efficiency.	Uses the IDE confidently during the presentation with only minor hesitation.	Uses the IDE to run and show the program, but with limited fluency.	Uses the IDE inconsistently or requires prompting to navigate or run the program.	Shows minimal ability to use the IDE during the presentation.	Does not use the IDE.
2.2 Explain why you use an IDE.	Provides a clear, detailed explanation of why an IDE is used, linking features directly to workflow efficiency and project management.	Clearly explains why an IDE is helpful, referencing multiple relevant features.	Gives a basic explanation of why an IDE is used.	Explanation is vague or incomplete.	Shows very limited understanding of why an IDE is used.	No explanation provided.
2.3 Use the features of an IDE like the code editor, terminal, and file explorer.	Effectively demonstrates and explains how multiple IDE features are used together to develop, run, and manage the project.	Uses and explains most IDE features accurately.	Uses basic IDE features correctly but with limited explanation.	Attempts to use IDE features but shows confusion or gaps.	Rarely uses IDE features correctly.	Does not use IDE features.
2.4 Convert your algorithm into code.	Presents a working program that clearly reflects the planned algorithm, maintains IPO structure, includes clear documentation, uses variables and data correctly, and produces clear output.	Program accurately reflects the algorithm with correct IPO structure and documentation, with minor issues.	Program works and reflects the algorithm, but IPO structure or documentation may be limited.	Program partially reflects the algorithm; structure or documentation is weak.	Program shows little connection to the planned algorithm.	No meaningful program demonstrated
2.5 Test your program using appropriate data.	Clearly explains and demonstrates how the program was tested, including different scenarios and outcomes.	Explains testing performed and demonstrates that it influenced results.	Mentions basic testing with limited detail.	Minimal explanation of testing; testing appears superficial.	Very limited evidence of testing.	No testing evident.
2.6 Revise programs based on testing.	Clearly explains specific revisions made based on testing and how they improved the program.	Explains revisions made after testing with clear reasoning.	Mentions revisions made, but with limited explanation.	Attempts to describe revisions, but changes are minimal or unclear.	Shows little evidence of revision.	No revisions made.

Submission

The following are *required* submissions:

- Your written reflection submitted as a word document.