Python - Turtlympics - Planning

Overview

The semi-annual Turtlympics are upon us and we need some mazes for the turtles to race through. As programmers, you will be competing to create the most incredible mazes. The categories that you can compete for are:

- Best Theme
- Best Aesthetics
- Most Complex

Before creating your maze however, you must plan it following these guidelines.

Project Criteria

Your task is to create a program that creates a maze for the Turtlympics. Your program must satisfy the following criteria:

- **Define** and **use** at least 2 functions that can accept parameters.
- **Define** and **use** at least 2 functions that can return a value.
- In total you should **define** and **use** at least 4 functions when creating your maze.

In your planning for this project, you should include the following:

- A **2 sentence description about what an algorithm is and how you can use it** to draw a maze for a turtle.
- Create an **IPO chart for each function** you will be implementing.
- **Pseudocode** or a **flow chart** demonstrating the control flow of how your maze will be drawn.
- **Diagrams and/or sketches of the maze** you intend to create.

Outcome Criteria

- **1 Demonstrate introductory structured programming skills by planning and writing sequential algorithms.**
    - **1.1** Explain what an algorithm is and why it is used.
    - **1.2** Analyze simple algorithms and describe the task or tasks they perform.
    - **1.3** Determine whether a problem can be solved using an input–process–output (IPO) approach.
    - **1.4** Break a problem into input, processing, and output components.
    - **1.5** Identify which data is already available to the program and which data must be provided as input.
    - **1.6** Arrange inputs, processing steps, and outputs in the correct order so that:
        - **1.6.1** processing occurs only after all required input is available
        - **1.6.2** output occurs only after processing is complete
    - **1.7** Write algorithms in a clear, standard format.
    - **1.8** Test algorithms using appropriate data.
    - **1.9** Revise and improve algorithms based on testing results.

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**
1.1 Explain what an algorithm is and why it is used.	Clearly and precisely explains what an algorithm is using correct terminology, with a strong connection to maze drawing and turtle behavior; explanation shows depth and insight beyond examples discussed in class.	Accurately explains what an algorithm is and why it is useful, with a clear and relevant connection to drawing a maze.	Provides a basic, correct explanation of what an algorithm is and a general reason for using one.	Gives a vague or partially correct explanation; purpose of an algorithm is unclear or loosely connected to the task.	Shows very limited understanding; explanation is inaccurate or incomplete.	No explanation provided.
1.2 Analyze and describe the tasks of simple algorithms.	Thoroughly analyzes algorithms by clearly describing each task and how they work together to draw the maze.	Correctly describes the tasks performed by simple algorithms with clear reasoning.	Identifies the main tasks an algorithm performs, with limited detail.	Describes some tasks, but analysis is incomplete or unclear.	Shows minimal ability to describe what the algorithm does.	No attempt to analyze or describe tasks.
1.3 Determine whether a problem can be solved with an IPO approach.	Independently and correctly identifies that the problem fits an IPO model, with strong justification tied to the maze program.	Correctly identifies that the problem can be solved using IPO and explains why.	Correctly identifies IPO as a suitable approach, with minimal explanation.	Attempts to identify IPO use, but reasoning is weak or partially incorrect.	Shows little understanding of IPO as a problem-solving approach.	No attempt to address IPO.
1.4 Break a problem down into IPO components.	Breaks the maze problem into clear, complete, and well-justified input, processing, and output components for all functions.	Correctly identifies inputs, processes, and outputs for most parts of the problem.	Identifies basic IPO components, though some may be missing or unclear.	Partially breaks the problem into IPO components with noticeable gaps.	Shows very limited ability to separate input, process, and output.	No IPO breakdown provided.
1.5 Identify data used for inputs.	Clearly and completely identifies all required inputs, including parameters, constants, and user-provided data, with justification.	Correctly identifies all major inputs used in the program.	Identifies some inputs correctly, but misses others.	Identifies few inputs or confuses inputs with processing.	Shows minimal understanding of what counts as input data.	No inputs identified.
1.6 Order the steps of the IPO modules correctly.	All steps are logically ordered so inputs are gathered first, processing is complete before output, and dependencies are handled correctly throughout.	Steps are correctly ordered with no major logical errors.	Steps are mostly in the correct order, with minor issues.	Some ordering is correct, but several steps are misplaced.	Shows little understanding of correct sequencing.	No meaningful ordering shown.
1.7 Write algorithms in a clear standard format (pseudocode or flow charts).	Uses a clear, consistent, and professional standard format; algorithms are easy to follow and could be implemented directly.	Uses an appropriate standard format with clear structure and labels.	Uses pseudocode or a flow chart that is mostly clear, but may lack consistency.	Attempts a standard format, but clarity or structure is weak.	Format is unclear or inconsistent; difficult to follow.	No algorithm representation provided.
1.8 Test algorithms using appropriate data.	Systematically tests algorithms with varied and meaningful data; clearly documents results and observations.	Tests algorithms with appropriate data and notes outcomes.	Tests algorithms in a basic way with limited documentation.	Attempts testing, but data or results are unclear or incomplete.	Minimal or ineffective testing evident.	No testing performed.
1.9 Revise and improve algorithms based on testing results.	Thoughtfully revises algorithms based on test results, clearly explaining improvements and reasoning.	Makes clear improvements based on testing feedback.	Makes some revisions after testing, with limited explanation.	Attempts revisions, but changes are minimal or unfocused.	Very little evidence of revision or improvement.	No revisions made.

Submission

The following are *required* submissions:

- A word document with the following:
    - Description of what and algorithm is and why it is being used.
    - IPO charts for each function.
    - Pseudocode or flow chart.
    - Photos of maze sketches (upload photos to word document).