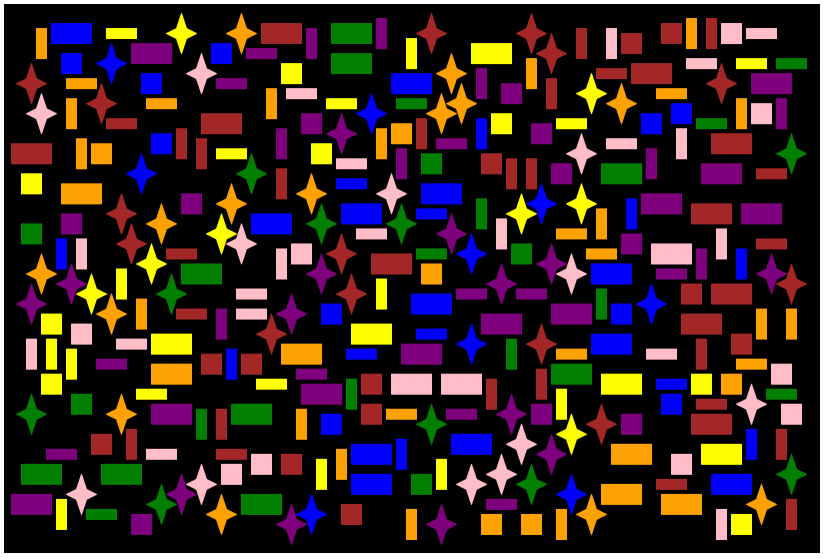
Iris - 2025



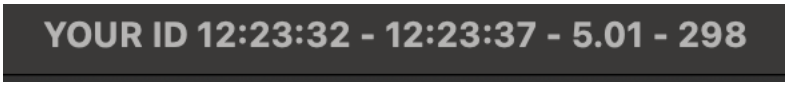
Overview

Inspired by a high school student I met in a high-school elite event organized by SSE in 2024. Her name is Iris. In this assignment, you are going to design and develop a program to display as many polygon shapes as possible on the canvas in random locations within a given period of time. Your program will load the custom shapes (name and coordinates) of all the polygons from an external file, start tracking the elapsed time and then repeatedly perform the shape-placement operation. Initially the canvas is empty. The shape-placement operation picks a random polygon shape, a random color and a random pair of x, y coordinates, trying to position the chosen polygon on the canvas at the chosen x, y coordinate, on condition that it is completely separate from all other polygons already drawn on the canvas. If it fails, your program repeatedly attempts to place the same polygon in another location by choosing another random pair of x, y coordinates. No polygon can be placed within another polygon and any polygon cannot touch or overlap any other polygons. The placement operation ceases any further attempts for another polygon once the elapsed time has expired. The duration length is defined by the user at the program's launch. Please refer to the scope for detailed requirements.

灵感来自我在 2024 年 SSE 组织的高中精英活动中遇到的一名高中生。她的名字叫 Iris。在本次作业中，您将设计和开发一个程序，在给定的 .me 时间段内，在画布上的随机位置显示尽可能多的多边形。您的程序将从外部文件加载所有多边形的自定义形状（名称和坐标），开始跟踪经过的 .me，然后重复执行形状放置操作。最初，画布是空的。形状放置操作会选择一个随机的多边形形状、一个随机的颜色和一对随机的 x、y 坐标，尝试将所选多边形放置在画布上所选的 x、y 坐标处，条件是它与画布上已绘制的所有其他多边形完全分开。如果失败，您的程序将通过选择另一对随机的 x、y 坐标反复尝试将同一个多边形放置在另一个位置。任何多边形都不能放置在另一个多边形内，任何多边形都不能接触或重叠任何其他多边形。一旦经过的 .me 到期，放置操作将停止对另一个多边形的任何进一步尝试。持续时间长度由用户在程序启动时定义。有关详细要求，请参阅范围。

SCOPE

1. Drawing Canvas - Use module “turtle” to create the drawing canvas, 75% width and 75% height of computer’s screen resolution, positioned at the center of the computer’s screen.
2. Input - prompt the user to enter the following information:
    a. Stretch Value - a value used to resize polygon in both width and height. Default is 1.
    b. Duration - total number of seconds to place random polygons on canvas. Default is 5.
    c. Random Seed - the seed value used to initialize the random module. Default is 1.
    d. Terminate - ‘y’ or ’n’. Default is ’n’
3. Termination
    a. after the duration is up, the program stops placing additional polygons on the canvas and then shows the total count of polygons successfully displayed on the canvas. In the title bar of the canvas, show your student ID, the time the operation started and ended, elapsed time (seconds) and the number of polygons displayed:



and simply print the polygon count with your student ID on the console:

    b. Change the background color of the canvas to black.
    c. if “Terminate” is ‘y’ then exit the program immediately; otherwise do nothing and keep the canvas active so that TA can verify the positions of polygons on the canvas.
4. Custom Shapes - the names and coordinates of all polygons are given in an external file called shapes.txt. Ensure the mode of the screen is set to “logo” and register the custom shapes using addshape(name, coordinates).

```
diamond: ( (–10, 0), (0,10), (10,0), (0,–10) )
rect1: ( (–15,5), (15,5), (15,–5), (–15,–5) )
rect2: ( (–5,15), (5,15), (5,–15), (–5,–15) )
rect3: ( (–20,40), (20,40), (20,–40), (–20,–40) )
hex: ( (–5,–8.66), (5,–8.66), (10, 0), (5, 8.66), (–5,8.66), (–10, 0) )
```

5. Shape-Placement Operation
    a. Track the elapsed time
    b. Pick a random color - from a predefined list of colors (see template file).

c. Pick a random polygon shape

d. Stretch the chosen polygon in both x and y directions and set the shape color accordingly using the following methods:

   i. shapesize(value, value)

   ii. color(value)

e. Try repeatedly placing the chosen polygon on the canvas at a random location:

   i. pick a random x, y coordinates

   ii. show the polygon at the chosen x, y coordinates if the following conditions are met:

      1. no further adjustment is allowed to the polygon including rotation, heading adjustment, shearing, etc.

      2. simply call goto(x, y) to move the chosen polygon

      3. the polygon must be completely separate from all other polygons already drawn on the canvas

      4. the polygon cannot be placed within another polygon

      5. the polygon cannot touch or overlap any other polygons

   iii. if the condition isn't met and there is more time to continue, repeat the above steps.

f. track of the number of polygons successfully shown on the canvas.

g. Repeat all the steps above until the total elapsed time (in seconds) exceeds the entered duration.

6. Template File - follow the provided template file to complete the assignment.

NOTE:

- Keep your entire source code in ONE SINGLE file.
- Use only Python modules as specified in the "Permitted Modules" section.
- In your design stick ONLY to functions, in other words, no class objects of your own.
  - Furthermore, the lines of code containing the sub-function(s) defined within another function will be counted as part of the parent function.
  - NOTE: Failure to adhere to the instructions outlined in the assignment handout will result in a 50% reduction in the coding style score.

Startup Options

Not applicable

Skills

In this assignment, you will be trained on the use of the followings:

- Refactoring - logic reuse or simplification based on the existing logic.
- Variable scope: global, local and function parameters.
- Coding Styles (naming convention, meaningful names, comments, doc_string, …etc)
- Problem Decomposition, Clean Code, Top-Down Design
- Functions (with parameters and return) for program structure and logic decomposition
- Standard objects (strings, numbers & lists)
- Variable Scope

Permitted Modules

Only the following Python module(s) is allowed to be used:

- turtle
- random
- time

Deliverables

Program source code (A2_School_StudentID.py), where School is SSE, SDS, SME, HSS, FE, LHS, MED and StudentID is your 9-digit student ID.

For instance, a student from SME with student ID “119010001” will name the Python file as follows:

- A2_SME_119010001.py:

Ensure that your source file is saved in standard, regular UTF-8 encoding format. On the status bar of Visual Studio Code, you can view the current encoding format as follows:

On an occasion, the encoding scheme is set to UTF-8 with BOM as follows:

The presence of the Byte Order Mark (BOM) could be due to copying from websites, older versions of editor, file conversion from other sources, default encoding settings, and so on.

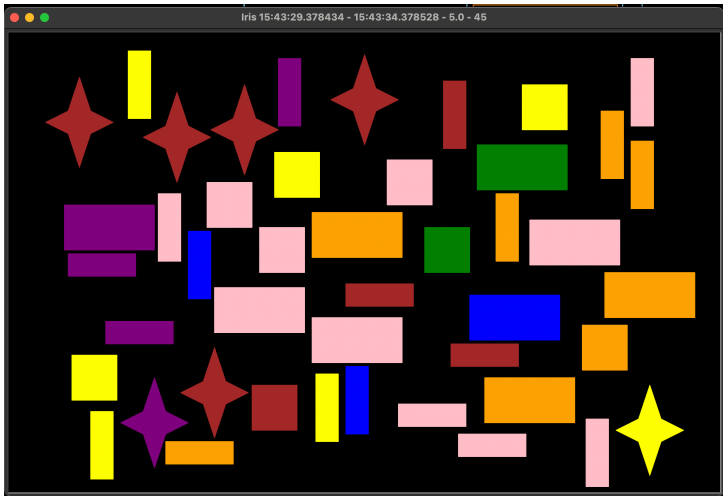
Confirm the encoding scheme is UTF-8 and the file name is correct, then submit the plain program file to the corresponding assignment folder. A deduction of 5% will be penalized if the file is incorrectly named or in wrong encoding format.

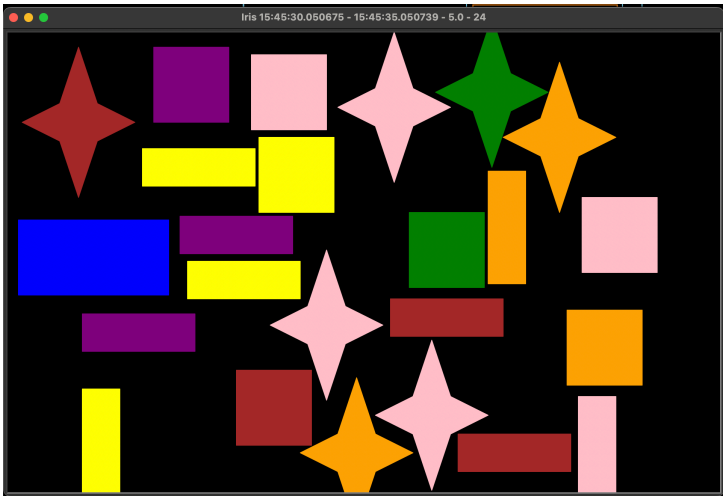
TIPS & HINTS

- Apply problem decomposition, Clean Code and Refactoring as illustrated during classes.
- Beware of variable scope as you might keep a few variables as global such as current editor content, cursor position, undo buffer, and so on.
- Refer to Python website for program styles and naming conventions (PEP 8)

SAMPLE OUTPUT

NOTE: REFER TO THE SCOPE FOR THE REQUIRED INFORMATION TO BE DISPLAYED IN THE TITLE BAR OF THE TURTLE GRAPHICS WINDOWS.





Marking Criteria

- Coding Styles – overall program structure including layout, comments, white spaces, naming convention, variables, indentation, functions with appropriate parameters and return.
- Program Correctness – whether or the program works 100% as per Scope.
- User Interaction – how informative and accurate information is exchanged between your program and the player.
- Readability counts – programs that are well structured and easy to follow using functions to break down complex problems into smaller cleaner generalized functions are preferred over a function embracing a complex logic with many nested conditions and branches! In other words, a design with a clean architecture and high readability is the predilection for the course objectives over efficiency. The logic in each function should be kept simple and short, and it should be designed to perform a single task and be generalized with parameters as needed.
- KISS approach – Keep It Simple and Straightforward.
- Balance approach – you are not required to come up with a very optimized solution. However, take a balance between readability and efficiency with good use of program constructs.

Items	Percentage	Remarks
Coding Styles	50%-60%	0% if program doesn't run
Functionality	50%-60%	Refer to Scope

Due Date

April 6th, 2025, 11:59:00PM