Assignment 4

Due date: Sunday, April 10, 11:59 pm
Submission via Git only

Programming environment

For this assignment you must ensure your work executes correctly on the virtual machines (i.e., Senjhalla or its analogous environment for students with M1 MacBooks) you installed/configured as part of Assignment 0 as this is our "reference platform". This same environment will be used by the teaching team when grading the work submitted by the SENG 265 students. Submitted assignment that don't execute correctly on the required environment will receive a failing grade or be subject to heavy penalties.

The sample code for Part 1 of this assignment is available on Brightspace as well as on the UVic SENG server in **/seng265work/2022-spring/a4/**
Use the following **scp** command to download these files from the UVic SENG server to your computer

```
scp NETLINKID@seng265.seng.uvic.ca:/seng265work/2022-spring/a4/* .
```

Make sure that your submitted files are inside of your **a4 folder of your cloned Git repository**.
Hint: To verify whether you uploaded the files properly, simply clone the git repository to a new directory on your computer and check that the desired files are placed properly.

Individual work

This assignment is to be completed by each individual student (i.e., no group work). Naturally you will want to discuss aspects of the problem with fellow students, and such discussion is encouraged. However, sharing of code fragments is strictly forbidden. Code-similarity analysis tools are used to check submitted work for plagiarism.

Learning objectives

I. Learn object-oriented design with Python — using classes, objects, and attributes (methods and data).
II. Appreciate the concepts of abstraction and encapsulation
III. Learn how to structure Python projects into multiple files — packages and modules.
IV. Learn Python type hinting
V. Learn how to generate random numbers in Python
VI. Learn how to manipulate sequential text files using functions in Python
VII. Learn how to understand application programmer interfaces (APIs)
VIII. Learn how to generate HTML5 and SVG code programmatically
IX. Learn top-down design using function decomposition
X. Appreciate how you can configure Python programs with a configuration class.
XI. Continue to learn incremental development. The starting point for A4 is a complete Python program. You need to study and understand this program to complete the assignment.
XII. Learn how to read and understand existing code and build upon it.
XIII. Use Git to manage changes in your source code and annotate the evolution of your solution with messages provided during commits. Update your git repository after every major editing session to make sure that you don't loose your work.

Instructions

Assignment 4 consists of three (3) separate Python projects. The idea is to develop the famous SENG 265 Python Arts program incrementally. All three parts are required. The first two parts are worth 20% each and Part 3 is worth 60%. Store the three different Python projects in **three subdirectories called a41, a42, a43 of the a4 directory**.

Part 1

In Part 1 you are to develop an **object-oriented Python project** that features a **Circle** class and generates an HTML-SVG file as depicted in Figure 1 below. Viewing this file in a web browser will render the SVG drawing depicted in Figure 2 below. Start Part 1 by downloading this sample Python program (cf. Figure 1) and modify this program as follows. **Create classes for the geometric objects (i.e., Circle, Rectangle) as well as a class called ProEpilogue to generate the HTML/SVG prologue and epilogue. The class names Circle, Rectangle, and ProEpilogue are required for Part 1 for automated grading. Part 1 must generate some circles and rectangles. Please note that there are no random numbers required for Part 1.**

```
<html>
<head>
   <title>My Art</title>
</head>
<body>
   <!--Define SVG drawing box-->
   <svg width="500" height="300">
      <circle cx="50" cy="50" r="50" fill="rgb(255, 0, 0)" fill-opacity="1.0"></circle>
      <circle cx="150" cy="50" r="50" fill="rgb(255, 0, 0)" fill-opacity="1.0"></circle>
      <circle cx="250" cy="50" r="50" fill="rgb(255, 0, 0)" fill-opacity="1.0"></circle>
      <circle cx="350" cy="50" r="50" fill="rgb(255, 0, 0)" fill-opacity="1.0"></circle>
      <circle cx="450" cy="50" r="50" fill="rgb(255, 0, 0)" fill-opacity="1.0"></circle>
      <circle cx="50" cy="250" r="50" fill="rgb(0, 0, 255)" fill-opacity="1.0"></circle>
      <circle cx="150" cy="250" r="50" fill="rgb(0, 0, 255)" fill-opacity="1.0"></circle>
      <circle cx="250" cy="250" r="50" fill="rgb(0, 0, 255)" fill-opacity="1.0"></circle>
      <circle cx="350" cy="250" r="50" fill="rgb(0, 0, 255)" fill-opacity="1.0"></circle>
      <circle cx="450" cy="250" r="50" fill="rgb(0, 0, 255)" fill-opacity="1.0"></circle>
   </svg>
</body>
</html>
```

Figure 1: Generated HTML/SVG code

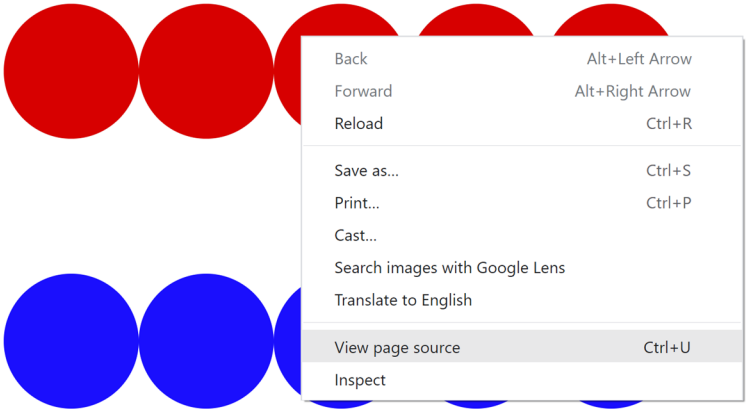


Figure 2: Sample HTML/SVG page

Part 2

Part 2 generates random numbers using a class called **GenRandom** (cf. Table 1) for generating random art using an art range configuration class called **ArtConfig** (cf. Figure 3) that can be instantiated to create different art types such as the images depicted in Figure 3 below. The output of Part 2 must be a table of random numbers as depicted in Table 1 below. Please note that not all columns are required for Part 3 but are required for Part 2. Also note that the random numbers you generate will of course be different than the numbers in Table 1. However, the random numbers should be within the ranges specified in Table 2 below. The numbers should be arranged in nicely aligned (right-justified) columns. There are no requirements with respect to white space. **The class names GenRandom and ArtConfig are required for Part 2 for automated grading.**

Table 1: Random numbers for 10 sample geometric shapes

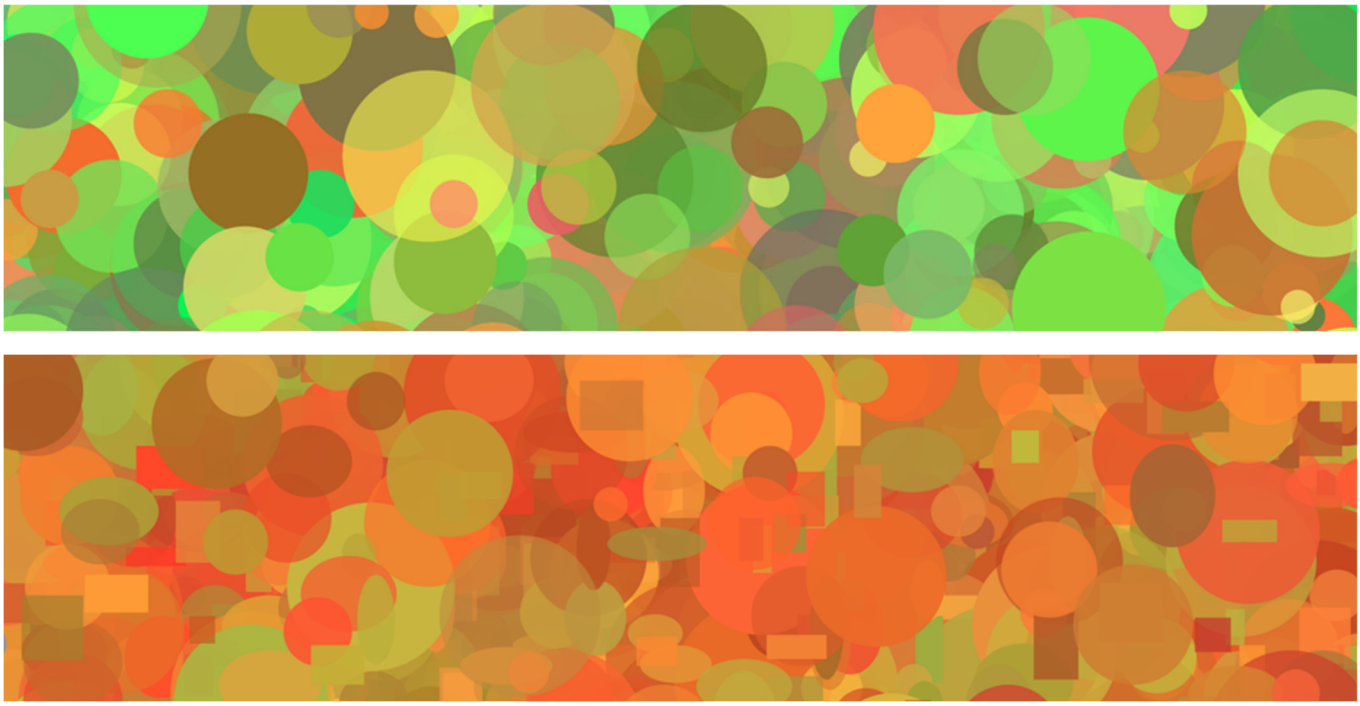
CNT	SHA	X	Y	RAD	RX	RY	W	H	R	G	B	OP
0	2	431	13	24	27	26	18	11	82	144	73	0.1
1	1	294	28	50	23	24	32	21	135	12	62	0.2
2	1	358	264	30	30	18	32	36	179	18	77	0.9
3	2	247	179	23	27	28	22	25	9	56	37	0.7
4	2	355	263	20	17	28	21	35	19	13	137	0.8
5	1	29	217	48	25	23	29	24	64	120	54	0.8
6	2	216	290	26	22	10	37	24	173	79	20	0.9
7	0	317	150	47	18	26	28	23	50	59	161	0.5
8	2	78	53	28	13	24	30	32	176	202	153	0.3
9	2	342	205	14	12	14	18	11	153	213	233	0.5

Table 2: Random numbers for 10 sample geometric shapes

VAR	Description and Range
CNT	Shape counter
SHA	Kind of shape: 0 for circle, 1 for rectangle, 3 for ellipse
X	X-coordinate of shape (e.g., circle or ellipse center, rectangle top-left corner) in viewport range
Y	Y-coordinate of shape (e.g., circle or ellipse center, rectangle top-left corner) in viewport range
RAD	Circle radius with small range 0 .. 100
RX	Ellipse radius small range 10 ..30
RY	Ellipse radius 10 ..30
W	Rectangle width small range 10 .. 100
H	Rectangle height small range 10 .. 100
R	Red color of RGB in range 0 .. 255
G	Green color of RGB in range 0 .. 255
B	Blue color of RGB in range 0 .. 255
OP	Shape opacity in range 0.0 .. 1.0

Part 3

The goal of Part 3 is to integrate the classes developed for Parts 1 and 2 into a third Python project and generate some beautiful greeting cards for your friends and family in the form of HTML-SVG pages. That is, Projects 1 and 2 are test programs for the various classes required for Part 3. Instantiate three configuration class objects to generate different art types as depicted in Figure 3 below. Image titles and captions are optional. All the class names required for Parts 1 and 2 are required for Part 3.



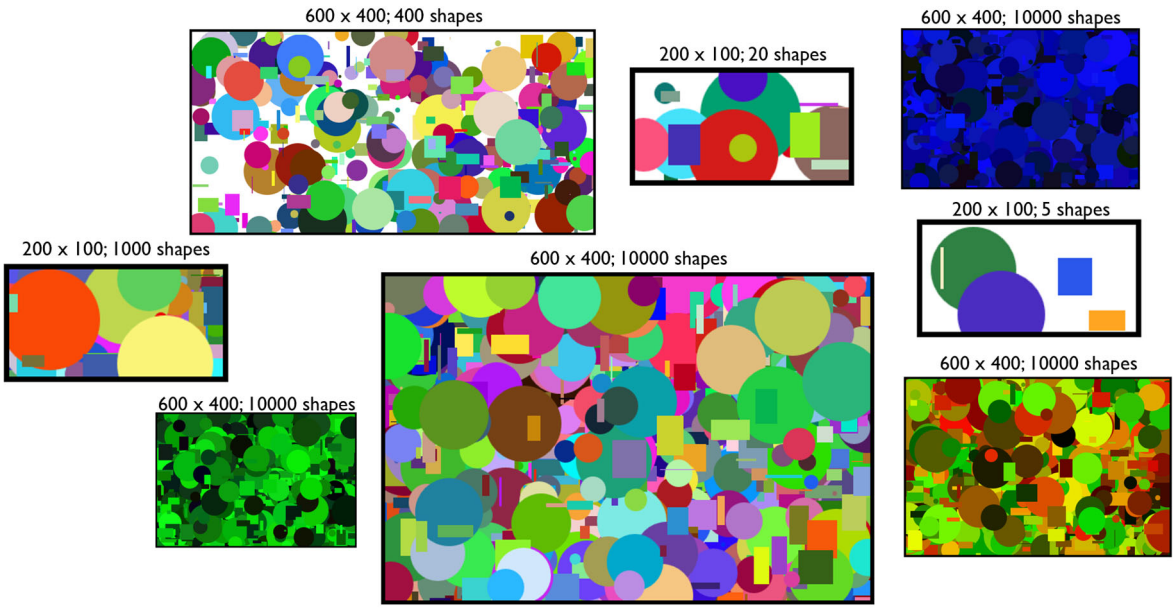


Figure 3: SENG 265 ART

Important requirements for grading

- Your Python projects must execute perfectly under the Senjhalla virtual machine.
- The most important grading requirements are effective object-oriented design and effective program decomposition.
- To facilitate automated grading, the following call names are required for the required classes: Circle, Rectangle, ProEpilogue, ArtConfig, and Genrandom.
- To facilitate automated grading, each class and each method must have a docstring containing the class or method name (e.g., ‘’’Circle class’’’ or ‘’’drawCircle method’’’) right below the class or function header.
- You must use classes and objects for geometric objects (i.e., Circle, Rectangle), art configuration (i.e., ArtConfig) as well as HTML-SVG prologue and epilogue (i.e., ProEpilogue)
- You must generate valid, multi-line, and indented HTML-SVG files for Part 1 and Part 3.
- All three projects must use Python type hints.
- You must not use program-scope or file-scope variables.

What to submit

- Submit all three parts in separate directories/folders to your **a4 folder of your Git repository** as follows.
- The three different Python projects must be stored in three subdirectories/folders called a41, a42, a43 that are in your a4 directory.
- **Hint:** To verify whether you uploaded the files properly, simply clone the git repository to a new directory on your computer and check that the desired files have been placed properly.
- Part 1: Submit your Python program (a41.py) as well as the generated HTML file (a41.html)
- Part 2: Submit your Python program (a42.py) as well as a screenshot of your random table (a42.jpg)
- Part 3: Submit your Python program (a43.py) as well as three screenshots of your art (a431.jpg, a432.jpg, and a433.jpg)

Grading assessment

- The first two parts are worth 20% each and Part 3 is worth 60%.
- Straying from the assignment requirements will result in zero marks due to automated grading.

Additional Criteria for Qualitative Assessment

- **Documentation and commenting:** the purpose of documentation and commenting is to write information so that anyone other than yourself (with knowledge of coding) can review your program and quickly understand how it works. In terms of marking, documentation is not a large mark, but it will be part of the quality assessment.
- **Proper naming conventions:** You must use proper names for functions and variables. Using random or single character variables is considered improper coding and significantly reduces code readability. Single character variables as loop variables is fine.
- **Debugging/Comment artifacts:** You must submit a clean file with no residual commented lines of code or unintended text.
- **Quality of solution:** marker will access the submission for logical and functional quality of the solution. Some examples that would result in a reduction of marks