# ASE 420 Homework 2 (Python Programming)

- 50 points in total
- Use the 'Last-First-HW#' directory as a template directory. Copy all your results in the results directory (create sub-directories if necessary) and complete (mark and grade) the grading.docx in the directory.
- Change the directory name; for example, `Cho-Samuel-HW#` is the directory name if I submit the homework (# should be replaced with the HW number). Notice that last name comes first.
- Zip the directory to make a zip file, for example, `Cho-Samuel-HW#.zip` (# should be replaced with the HW number), to submit them on Canvas.

## Learn Python Programming Using Jupyter and LearnXinYminutes.com (10 points)

- Download Python code from https://learnxinyminutes.com/docs/python/
- Use the Jupyter notebook to learn Python using the downloaded Python code.
  - Make Jupyter pages by interactively running the code on Jupyter.
  - Make comments (or documentation) if it helps your learning.
  - Export HTML files from the Jupyter pages.
- Copy all your jupyter files in the jupyter/notebook directory.
- Copy all your HTML files in the jupyter/html directory.

## Learn Python Programming by Making Applications (30 points)

- Use skeleton code, but it's OK if you want to start from scratch.
- Copy all your python programs in the code directory.

### Rock/Paper/Scissor Prototype (ver 0.1)

```
// while true
//   Show prompt
//   Read user input from console
//   If input is a valid move ("r", "p", "s")
//     Choose the AI move at random
//     Compare the player move with the AI move
//     Show the result
//   else if input is "q"
//     Quit the program
//   else
//     Invalid input
```

- We implement the Rock/Paper/Scissor program. We are given the following algorithm (business logic); write Python code to implement the logic.

  - Implement the algorithm without thinking about the structure or design of the program.
  - The focus is to make a working application as quickly as possible, so it is OK to use global variables or create a script (not an application).
- Copy your program (rps.py) in the code/rps/ver01 directory.
  - Don't forget to attach the output in the source file.

**Rock/Paper/Scissor OOP (ver 0.2)**

- We understand the business logic, and we have a prototype implementation (ver 0.1). Using the experience, we can rewrite the ver 0.1 code following the OOP principles in Python to make ver 0.2
  - Use abstraction and, if necessary, polymorphism, inheritance, and encapsulation to implement the algorithm using OOP.
  - Use the skeleton to implement your application.
- Copy your program (rps.py) in the code/rps/ver02 directory.
  - Don't forget to attach the output in the source file.

**Rock/Paper/Scissor UML Diagram**

- Use a UML class diagram to draw the design (modules and interfaces) you made in ver 0.2.
  - You can use any method to draw your design, including StarUML (https://staruml.io), Lucidchart (https://www.lucidchart.com/pages/landing/uml-diagram-software), VISIO (https://www.microsoft.com/en-us/microsoft-365/visio/flowchart-software), Creately (https://creately.com/lp/uml-diagram-tool/) or any other drawing software tools.
- Copy your UML diagram (uml1) in the code/rps/uml directory. * You can use any extension name for the uml2 file.

**Understand the Strategy Design Pattern**

- Read the strategy design pattern from the SD_design_behavioral_patterns.pdf in the reference directory.
- Think about at least two possible strategies for your AI move.
  - We used the random strategy in the original implementation, but we can use different strategies (for example, giving a different move than before).

**Refactoring Rock/Paper/Scissor using the Strategy Design Pattern (ver 0.3)**

- Refactor (redesign and rewrite) your ver 0.2 code to use the strategy pattern.
- Copy your program (rps.py) in the code/rps/ver03 directory.
  - Don't forget to attach the output in the source file.

**Update Rock/Paper/Scissor UML Diagram**

- Redraw the UML diagram from the revised design.
- Copy your UML diagram (uml2) in the code/rps/uml directory.
  - You can use any extension name for the uml2 file.

**Make Two AI Rock/Paper/Scissor Players to Play 100 times (ver 0.4)**

- Refactor your ver 0.3 code to make compete for two different strategies.
- Play the game 100 times automatically, and decide what strategy wins or loses.
- Copy your program (rps.py) in the code/rps/ver03 directory.
  - Don't forget to attach the output in the source file.
  - You don't have to copy all the results if you have too long output.

## Learn PyGame by Making Applications (10 points)

Open the pygame.zip in the reference directory.

- Learn how to make a sound with PyGame with the sound.py
- Learn how to make an animation with PyGame with the image.py
- Then make a pygame application animation.py.
  - It moves the image and plays sound when the keyboard (up, down, left, or right) is clicked.
    - The image is moved by 100 when the keyboard is clicked.
    - Use the sound.py, image.py, and tetris_ver1.py as a reference.
- Copy your code (animation.py) in the code/pygame directory.