ENGR 102 - LAB ASSIGNMENT #12

Fall 2021

General course learning outcomes:

- demonstrate the use of programming techniques in the construction of computer programs, including the use of: large data structures such as lists; control structures such as conditionals and loops; user-defined functions; data from and processed to an external file.
- decomposing a complicated task into manageable pieces.
- apply programming techniques to solve problems in engineering.
- complete a team programming assignment that ties together concepts learned in the class.

Remember to document your code with comments and print labels and units when applicable. When these programs are completed, submit all files to Mimir. Remember the appropriate header information.

Deliverables (what each member of the team will be submitting to Mimir) –1 pdf file, 1 .py file:

1.) Lab12_PartsAB.pdf (Part A and Part B)
2.) Lab12_PartC.py file (Part C)

Team Activity 1

(Reminder: This assignment (100 points) will continue with Lab 13 (150 points + 50 possible challenge points))

This week your team will use bottom-up and top-down design to begin programming a 'Pokémon-ish' game, "Gotta Catch One of 'Em". The next pages lists several requirements the game must satisfy, but your team has significant freedom in the specifics of how to do so.

A. Bottom-Up: As a team, brainstorm what functionality you are likely to need in this program.

- Your goal for Part A is not to generate a complete program design, but to **identify a list of functions to create that will likely be helpful in your program** (*e.g.*, *pokeball_throw*)
- Consider functions that will call other functions only after you have designed the lower functions (*e.g., remember the trajectory_y( ) function from last week*).
- Skip parts you are uncertain about, and come back as necessary.
- For each basic task where all necessary pieces are already known, write a short function description, required input arguments (if any), and function return values (if any).

B. Top-Down:

- In a new section of your document, create and outline your team *top-down* design in a flowchart as shown earlier this semester. Make use of the functions created in part A.
- Decide what data you will need to keep that will be used in more than one "node" of the design. Determine what variables you will use in the main code; write a list of these, along with a brief description of what that variable will store. *Note: you do not need to decide on variables used locally, e.g. a loop iterator.*
- Create the sequence of steps required to go from your plan into a working program.

C. Prepare your Python Project:

- Transfer your hybrid top-down / bottom-up plan into function stubs and comments in your program:
    i. The first part should be a list of function stubs, using the *pass* command, for the functions you will write, with the first ones being from the bottom of your hierarchy.

    For *every* function, provide a name, write a docstring, and indicate what input parameters and return values are required so the functions will interact correctly. Much of this should already be completed from Part A; copy the information to code.

    ii. The second part should be comments outlining the main code sequence of general steps that will be required for the program, including where calls to the above-listed functions are required.

Program Requirements:

General Requirements:

1.) Display the rules of the game and a set of instructions of how to use your game.
2.) This is a 1 player game. More than 1 player account must be capable of being handled, but only 1 player will play at a time. (e.g. Tony Stark, Peter Parker, and Thor all have player accounts. But, only one of their accounts is loaded and 'playing' at a time.)
3.) Use at least 1 loop.
4.) Use at least one try-except block somewhere in the program.
5.) Use at least one if-elif-else statement.
6.) Use functions frequently to simplify your main code. You are required to create at least 3 functions.
    - Consider: menu building, catching Pokemon, leveling up Pokemon (optional challenge requirement), calculating CP (optional challenge requirement), displaying current Pokemon information, etc.
    - Minimize the number of functions needed by reusing functions where possible (i.e., if you find you are rewriting the same code in the program, it may be cleaner to utilize a function).

Menus: The game must provide several menus and submenus.

- A main menu should be created to allow for selection among several main functions of the game.
- Menus should be informative and intuitive
    - Game and Pokemon information must be used when populating the menu titles, options, and relevant information (e.g., displaying the current Pokemon information, and the number of candies available when viewing the level-up menu. Leveling is an optional challenge requirement.)

Selecting an Active Pokemon: A player must be able to select from all previously caught Pokemon in their game.

- A random Pokemon is assigned to each player when they start the game the first time.
- An external file should be used to load and store a list of Pokemon and their current stored candies for each player. (If including optional challenge requirement #1 and/or #2, include the level and/or CP information in the file, as well.)
- Newly 'caught' Pokemon should be added to the file.
- The external file should update if a Pokemon's stats change.

Catching: A player must 'catch' a new Pokemon through winning a type of mini-game.

- The specific game can be determined by your team. (e.g., guessing a number close enough to a random number, winning a round of hang-man, winning a game of Angry Birds as you designed last week within a certain number of guesses, etc…).
- If a player catches a new Pokemon, the player will be randomly awarded 3, 5 or 10 candies. For our game, candy is generic and the same candy works for all Pokemon.
- A CSV file is available with many Pokemons specified that a player may catch. You may modify the file as desired, but it must contain at least the same amount of information and remain a CSV file type. *If you modify the file to use with your game, you must submit the modified file with your game file.*

Everything above the blue line is required.

Everything below the blue line is optional.

CHALLENGE Requirement #1 (optional – worth 15 points):

Leveling: A player should be able to 'level up' Pokemon according to the following rules. Level information for each Pokemon should appear in any menu where information is displayed about the player's Pokemon.

- Level 40 is the maximum level. All newly caught Pokemon start at Level 1.
- One (1) candy is required to level the Pokemon by 1 level from levels 1–30, two (2) candies are required to level-up to levels 31–40.
- Note, candy is to be awarded through catching Pokemon. (See '***Catching***' section above.)

CHALLENGE Requirement #2 (optional – worth 15 points):

Combat Power (CP): Include CP stats for each Pokemon caught. See the provided Pokemon file for suggested minimum and maximum CP values for each Pokemon. But, you have freedom to determine how CP values are assigned. The CP value for each Pokemon should appear in any menu where information is being displayed about the player's Pokemon.

CHALLENGE Requirement #3 (optional – worth 20 points):

Two Player: The game must provide a method to switch between two player accounts within an 'active' session.

- Each player must be able to access their separate list of Pokemons within an 'active' session of the game.
- More than two accounts should be capable of being handled, but only two will be used at a time.

Example menu structures:

These are not required to be used verbatim.

```
-----------------------------          MAIN MENU         -----------------------------
1. View current Pokemon
2. Catch a new Pokemon
3.
```

```
-----------------------------       Current Pokemon      -----------------------------
Pokemon Name

Current CP:    _______
Current Level: _______
Candies:       _______

1 - Use Candy to Level-Up
2 - Exit to Main Menu
```

```
-----------------------------    Pokemon Selection Menu   -----------------------------
Pokemon Name

Current CP:    _______
Current Level: _______
Candies:       _______


----------------------------------------------------------------------------------------
 1. Pokemon Name #1     2. Pokemon Name #2     3. Pokemon Name #3     4. Pokemon Name #4
    CP XXX                 CP XXX                 CP XXX                 CP XXX

 5. Pokemon Name #1     6. Pokemon Name #2     7. Pokemon Name #3     8. Pokemon Name #4
    CP XXX                 CP XXX                 CP XXX                 CP XXX

 9. Pokemon Name #1    10. Pokemon Name #2    11. Pokemon Name #3    12. Pokemon Name #4
    CP XXX                 CP XXX                 CP XXX                 CP XXX
----------------------------------------------------------------------------------------
Select a new Pokemon (1-12):
```