COM S 1270 - 'NIMGRAB!'

Please see the 'Course Schedule' on the Syllabus for this assignment's due date.

Assignment Objective

The goal of this assignment is to give you practice working with loops, functions, conditional logic, and the `random` module.

The activity for this assignment is for you to create your own version of the game "NIMGRAB!" in a file called `nimGrab.py`.

Instructions

The design for this game is below. Please read this design carefully, and see the example output below for details on how the game is to function. Overall, the game is open to creative interpretation, so long as the gameplay resembles that seen below.

You will need to include the rules for the game by 'abstracting' them from the 'design' shown below. The rules should be succinct, make sense to read, and adequately describe how to play the game to a user who has never played it before. Please note that simply copy/ pasting the 'design' below is insufficient - you *must* do their own original work on this aspect of the assignment.

You may program the 'AI' for the 'computer' player any way they like. The only caveat is that it cannot do anything 'catastrophic' at the end of the game. E.g., the computer cannot take all three items when there are only three items remaining. Similarly, it cannot take two when there are two items left. When there is only one item left, the computer *must* take that item.

Additionally, the `nimGrab.py` script should output a 'header,' including the name of the program ("NIMGRAB!"), your name, and the class/ section you are in as per the `Example Output` below.

DESIGN OF NIMGRAB!

Menu:

- The user is presented with a menu where they can play the game, read the rules, or quit to the terminal.
- The user decides if they want to play in 1-player mode (against a computer), or in 2-player mode (against another human).

Objective:

- The goal of the game is to take items from a row of items and to avoid being the player who takes the very last one.

Setup:

- An arbitrary number of items (typically between 20 - 25) are placed in a row.

Turns:

- Players take turns removing items from the row.
    - When it is their turn, a player must choose to take between `X` and `Y` items from the row (where `X` and `Y` are integers and `X < Y`).
        - *NOTE*: You can either choose the values of `X` and `Y` while the program is running, but before the game begins, or you can 'hard code' these values into your program. Feel free to be 'creative' with this.
- Players cannot take more items than are currently in the row.
    - Meaning - if there are 2 items in the row, a player cannot take 3 items.
- The game starts with the human player when in 1-player mode, and with whatever human decides to go first in 2-player mode.
- When playing in 1-player mode, the computer should not do anything 'catastrophic.' E.g. there are 3 items left and it takes all 3. In this scenario, it should take 2 (to force the human player to take the last one).
- Players alternate turns.

Winning:

- The player who takes the last item loses.

Game Over:

- The winner is printed to the screen and the game ends.

Restart:

- After the game ends the initial menu is displayed again.
    - Meaning - the user can choose to play the game again, read the rules, or quit to the terminal.

Other Requirements:

Additionally, you do *not* need to compensate for the crashes that occur when the user enters a value of the incorrect type. (E.g., The program asks for an integer, but the user enters a letter instead.)

It is required for you to write your name, the date you started working on your script, and a short explanation of what your code does at the top of your file. For example:

```
# Matthew Holman             6-24-2024
# Assignment #3
#
# This is the classic game of NIM - here called NIMGRAB!
```

Do *not* attempt to 'double dip' by placing your 'header' printout information at the top of your file - the header and the initial comment block are different things.

Your work should be your own, completely original effort. Meaning - you should not just copy my code

explanation above, but rather you should try to come up with one of your own.

The actual Python script itself can be programmed in any way (excluding cheating ala ChatGPT) so long as the output resembles that below. This includes the use of the initial 'header' when starting the program (i.e. the program title, who it is by, the student's class/ section, etc.). Please note that the 'header' will be strictly enforced with this assignment. Be sure to include the title, your name, and your class/ section.

Example Output

```
NIMGRAB!

By: Matthew Holman
[COM S 127 1]

Do you want to [p]lay, read the [r]ules, or [q]uit?: asdf
Do you want to [p]lay, read the [r]ules, or [q]uit?: qwer
Do you want to [p]lay, read the [r]ules, or [q]uit?: zxcv
Do you want to [p]lay, read the [r]ules, or [q]uit?: p

Do you want to play against [h]uman or [c]omputer?: asdf
Do you want to play against [h]uman or [c]omputer?: c

Items left: 20
| | | | | | | | | | | | | | | | | | | |
How many items do you want to take [1-3]?: 12
ERROR: Invalid choice. Please choose again.
How many items do you want to take [1-3]?: -2
ERROR: Invalid choice. Please choose again.
How many items do you want to take [1-3]?: 33
ERROR: Invalid choice. Please choose again.
How many items do you want to take [1-3]?: 3

Computer takes: 1

Items left: 16
| | | | | | | | | | | | | | | |
How many items do you want to take [1-3]?: 3

Computer takes: 2

Items left: 11
| | | | | | | | | | |
How many items do you want to take [1-3]?: 3

Computer takes: 2

Items left: 6
| | | | | |
How many items do you want to take [1-3]?: 3

Computer takes: 1

Items left: 2
| |
How many items do you want to take [1-3]?: 25
ERROR: Invalid choice. Please choose again.
How many items do you want to take [1-3]?: 1
```

```
Computer takes: 1

Computer took the last item. Human wins!

Do you want to [p]lay, read the [r]ules, or [q]uit?: q

Goodbye!
```

Special Notes

NOTE: This assignment will be more difficult than the previous assignments, but should not be "impossible." However:

- Completing this assignment may require you to start your work 'before the last minute.' Please plan accordingly.
- Your script **CANNOT** crash under any circumstances except for those circumstances noted below. Any portions of your code where the script crashes will receive a zero (0) for that aspect of the assignment.
    - Understand, when the word 'crash' is used in the instruction above, what this means is 'crashing under expected use cases given the level of knowledge attained in the class.'
        - You do not currently have the ability to enforce user input types (although you will later in the semester). As such, if you ask for an integer as input (e.g., 1), and the user provides a letter (e.g., 'a'), and it crashes, then this is *not* a problem *yet*.

NOTE: You are turning in your CODE - NOT the OUTPUT of your code.

- Your code will be run by the TAs, and the output of those runs, along with the code itself, is what will be evaluated.

NOTE: Assignments turned in in any other format other the specified file types will not be accepted.

- Screenshots of code **will not** be accepted.
- `.sln` files are **not** code files - they contain **no** Python code and **will not** be accepted.
- `.zip`, `.rar`, `.tar.gz`, and other compressed files **will not** be accepted unless otherwise specified.
- If your submission is not in a `.py` file, when so specified, or if the submission is not accompanied by **all** the files needed to run the submission, the submission will not be graded.
    - **THIS WILL LEAD TO YOU RECEIVING A ZERO (0) ON THE ASSIGNMENT.**
    - **You will *NOT* be allowed to re-submit your work after the final deadline in this case.**

NOTE: You have the ability (and responsibility) to 'double check' that your work/ submission is correct when you turn it in.

- If you accidentally turn in the wrong submission, you will **not** be allowed to resubmit it after the final deadline.
- Please note, you can submit your work as many times as you like *before* the final deadline.