Intro to Computer Science

PA05

A Game!

Code due by Wednesday, Feb 26th at 11:00 AM

Paperwork due the same day at the start of class

Introduction

The game Cowboy, Ninja, Bear is a simple game that two people can play to determine a winner (see this post.) Two players each, independently, select one of the three poses (Cowboy, Ninja, or Bear) and simultaneously reveal their selection as follows:

- Cowboy = both hands mimicking pistols, pointed at your opponent
- Ninja = both hands at about face level, hands flat and poised for a martial arts attack
- Bear = hands up, fingers curled like claws, ready to rip your opponent to shreds

If they select the same attack it is a tie. If they select different attacks, than one of the two is a winner, based on the following:

- Ninja beats Cowboy using lighting speed ninja kicks
- Cowboy beats Bear with his quick draw and perfect accuracy
- Bear beats Ninja with a strong swipe of his clawed paw

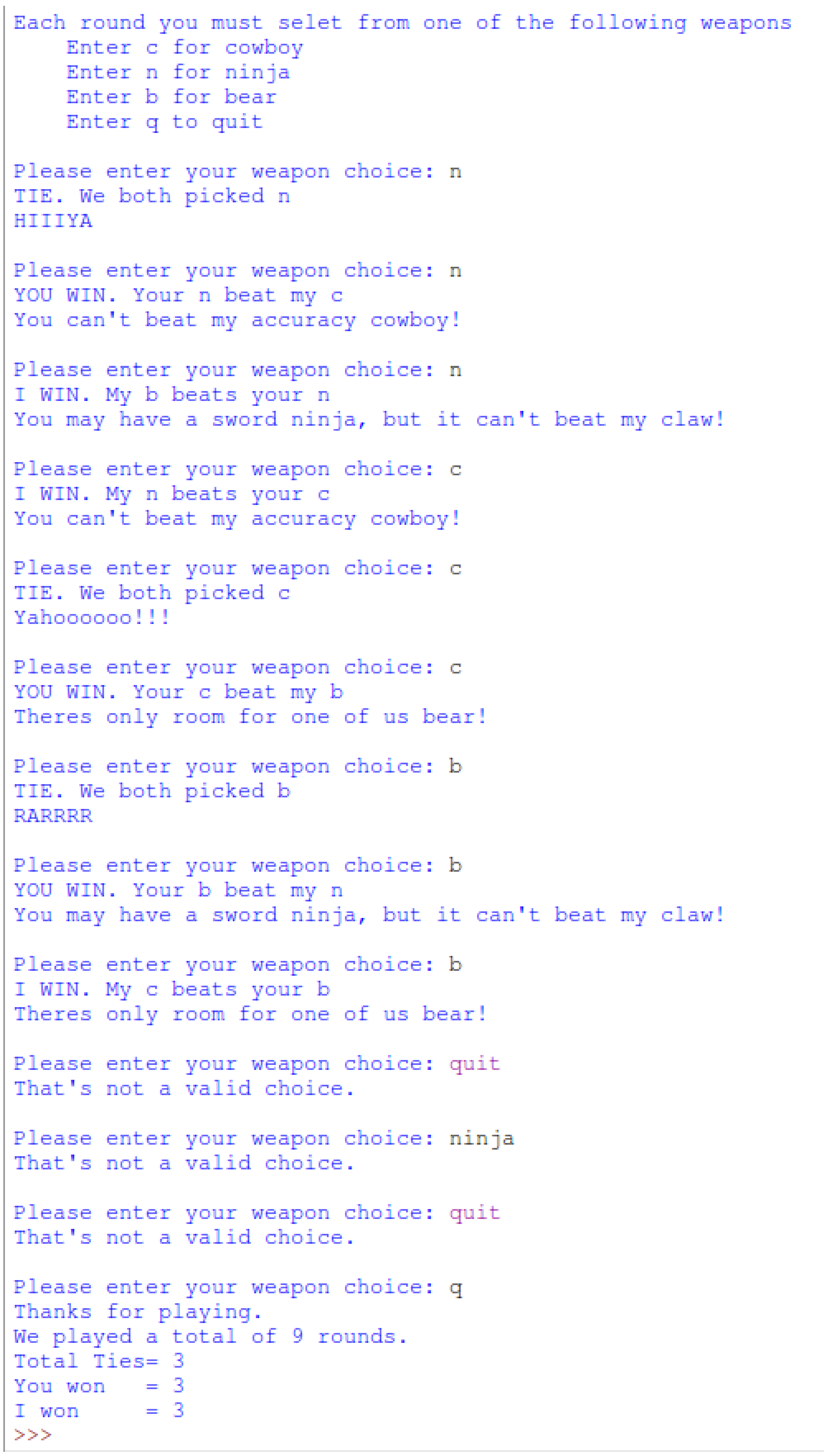
Often, it is used as a method of selection similar to flipping a coin or throwing dice to randomly select a person for some purpose. Of course, this game is not truly random since a skilled player can often recognize and exploit the non-random behavior of an opponent; for instance, if you notice that your opponent chooses Bear most frequently, you may choose Cowboy (which beats Bear) most often in an effort to win.

Original Program Specifications

Your program (saved in a file called cnb.py) will allow a human user to play several rounds of Cowboy, Ninja, Bear with the computer. Each round of the game will have the following structure:

- The program will choose an attack (Cowboy, Ninja, Bear), but its choice will not be displayed until later so the user doesn't see it.
- The program will announce the beginning of the round and ask the user for his/her attack of choice
    - The user can enter either of the three attacks or the signal to quit
    - If the user inputs something other than c, C, n, N, b, B, q or Q, the program should detect the invalid entry and ask the user to make another choice.
- If the user entered a attack :
    - The two attacks will be compared to determine the winner (or a tie) and the results will be displayed by the program
    - Depending on what attack each player entered, a sound effect of the attack will also display describing the attack executed by player. Look at sample output below for examples.
    - The computer will keep score which will be displayed later.
    - The next round will begin, and the game will continue until the user chooses to quit
- If the user enters quit than the computer:
    - Prints a goodbye message which should include a score of all of the games played

For example, one sample game might look like this:



```
Each round you must selet from one of the following weapons
    Enter c for cowboy
    Enter n for ninja
    Enter b for bear
    Enter q to quit

Please enter your weapon choice: n
TIE. We both picked n
HIIIYA

Please enter your weapon choice: n
YOU WIN. Your n beat my c
You can't beat my accuracy cowboy!

Please enter your weapon choice: n
I WIN. My b beats your n
You may have a sword ninja, but it can't beat my claw!

Please enter your weapon choice: c
I WIN. My n beats your c
You can't beat my accuracy cowboy!

Please enter your weapon choice: c
TIE. We both picked c
Yahoooooo!!!

Please enter your weapon choice: c
YOU WIN. Your c beat my b
Theres only room for one of us bear!

Please enter your weapon choice: b
TIE. We both picked b
RARRRR

Please enter your weapon choice: b
YOU WIN. Your b beat my n
You may have a sword ninja, but it can't beat my claw!

Please enter your weapon choice: b
I WIN. My c beats your b
Theres only room for one of us bear!

Please enter your weapon choice: quit
That's not a valid choice.

Please enter your weapon choice: ninja
That's not a valid choice.

Please enter your weapon choice: quit
That's not a valid choice.

Please enter your weapon choice: q
Thanks for playing.
We played a total of 9 rounds.
Total Ties= 3
You won   = 3
I won     = 3
>>>
```

For the initial version of this game you can program the computer to play randomly. To do this you should put the following code at the top of your program:

```
import random
random.seed()
```

This is the code necessary to set up a random number generator.

Each time you need to pick a random number you use:

```
picked=random.randint(1,3)
```

This will produce a value from 1 to 3 INCLUSIVE (notice this is different from the way range and slicing works).

Getting Started

1. Do all the standard startup things. Create a new file called cnb.py. Put your comments in at the top, save it.
2. Now you need to break the problem down into parts. Read the description and identify the subtasks that need to be solved. For example, one subtask would be to get proper user input. Create a design document and mark in the empty program, using comments, all the subtasks you need to solve.
3. Now address one subtask, getting user input. Do this in stages as well. Can you:
    a. Prompt for and get a choice (a string) from the user?
    b. Once you can do that, can you repeatedly prompt for a character until you see a ‘q’ or ‘Q’ for quit?
    c. Once you can do that, can you check for “legal” character responses from the user, and print an error message when an illegal response is given?
    d. Next, can you check for legal responses that are in both upper and lower case?

   Once you can do all that, move on to the next subtask.

4. Remember, save the file and run it all the time! It will make debugging the program easier.

BONUS problem for an extra two points!

Variation #1 - The computer has a memory

NOTE: If you choose to work on this variation, I strongly suggest you save your original program first so you have something to turn in if you can't get this solution in time.

The computer should select the attack most likely to beat the user, based on the user's previous choice of attacks. For instance, if the user has selected Ninja 3 times but Cowboy and Bear only 1 time each, the computer should choose Bear as the attack most likely to beat Ninja, which is the user's most frequent choice so far. To accomplish this, **your program must keep track of how often the user chooses each attack.** Note that you **do not** need to remember the order in which the attacks were used. Instead, you simply need to keep a count of how many times the user has selected each attack (Cowboy, Ninja or Bear). Your program should then use this playing history (the count of how often each attack has been selected by the user) to determine if the user currently has a preferred attack; if so, the computer should select the attack most likely to beat the user's preferred attack. **During rounds when the user does not have a single preferred attack, the computer may select any attack.** For instance, if the user has selected Cowboy and Ninja 3 times each and Bear only 1 time, or if the user has selected each of the attacks an equal number of times, then there is no single attack that has been used most frequently by the user; in this case the computer may select any of the attacks.

At the end of the game (when the user chooses 'q' or 'Q'), your program should provide additional information in the final display by including the number of times that the human user selected each attack (Cowboy, Ninja, Bear)

If you do this variation you should save your file with the name cnb_w_memory.py. This will be your signal to me to check out the possibility of bonus points.

Final Submission

To upload your homework for grading, log on to eLearning, select this class, and navigate to the "Assignment Submissions" area. Click on the "Programming Assignment 5" folder and upload the python file in its designated location.

In addition to this, you should print paper copies of your code and design document. Please submit these stapled printouts in the following order in class the day the assignment is due:

- design document
- cnb.py or cnb_w_memory.py