Intro to Computer Science
CS-UH 1001, Spring 2022

Lecture 8 – Lab (2D lists, Nested loops)

Today's Lecture

- Python Comments
- Lab (2D lists, Nested loops)

Recap: while Loop with Initial/Exit Condition

```
students_list = []
input_more_students = True

while input_more_students == True:

    student = input("Please enter student name: ")
    students_list.append(student)
    response = input("Input more, enter 'yes': ")
    if response != 'yes':
        input_more_students = False

print("Good bye")
```

The input_more_students variable is initialized with True to ensure that the while loop will at least execute once

Change the initial condition to False to stop the loop

break in Loops

If you want to stop a loop from iterating at a certain point, you can use the break command

- Example for loop:

```
for i in range(5):
    if i > 3:
        break
    print(i)

print("Good bye")
```

- Example while loop:

```
i = 0
while True:
    if i > 3:
        break
    print(i)
    i = i + 1

print("Good bye")
```

Dice guessing (ex_7.2.py)

1. Write a program that emulates a dice roll and ask the user to guess the dice value (values between 1 and 6)
2. Inform the user if the guess was correct or wrong
3. Print the outcome of the dice as:

```
*---*
| 6 |
*---*
```

Hint: To emulate a random dice roll, you can use the random module:

```
import random   #add this to line 1 of your .py file

random_value = random.randint(FromINT, ToINT)
```

This will randomly generate an integer between the values FromINT and ToINT (both are inclusive, both must be integers).
For example, random.randint(1,2) would either return 1 or 2

Dice guessing (v2) (ex_7.2.py)

Use the dice guessing program from earlier

1. Extend it so that the program keeps asking the user to guess until the guess is correct

2. Add error checks for the input:
   - can the input be casted to an integer? (use the string method .isdigit() to check if the input is a digit)
   - is the number within the dice range?

Joke of the day

A computer scientist walks into a donut store. Orders a donut. Orders 0 donuts. Orders 99999999999 donuts. Orders a lizard. Orders -1 donuts. Orders a asdfasdf.

First real customer walks in and asks where the bathroom is. The store bursts into flames, killing everyone.

Comments

Python Comments

- Comments can serve as notes to yourself, or they can be written with the intention of other programmers being able to understand what your code is doing
- Python comments are non-executable statements and start with a #
- All characters after the # and up to the end of the line are part of the comment and will not be executed by Python

- Example:

```
# This is a comment
```

Python Comments

- Comments are helpful to document your code (and you should do that!)

- Option 1:

```
# This will print ‘Hello World’
print(“Hello World”)
```

- Option 2:

```
print(”Hello World”) # This will print ’Hello World’
```

Long Python Comments

- Python does not have a syntax for multi line comments, but there are still 2 options:

- Option 1:

```
# This is a comment
# written in more than
# one line
```

- Option 2 (Triple quotes):

```
''' This is a comment
written in more than
one line '''
```

  - Python will ignore strings that are not assigned to a variable

How NOT to Comment Code!

- x = 10 # set the value of the variable x to 10
- print(”Hello World”) # This will print ’Hello World’

Gravity (ex_8.1.py)

- Create board with dimensions 10x10. The board will be initialized with '.'s
- Print the board
  - Use import os and os.system("clear") to clear the screen before printing the board
- Randomly assign 5 positions within the board with 'X's
  - Use import random and random.randint(0, ??) to draw a random number
  - Extra task: make sure 5 unique positions are chosen
- Simulate gravity, where these 'X's fall down to the ground
  - In each iteration, all 'X's fall one row
  - Use import time and time.sleep(1) to pause the program 1 second between iterations
  - Extra task: If two 'X's are on the same column, they should stack up on top of each other