Week 7 - Monday

COMP 1800

Last time

- What did we talk about last time?
- File example
- **`while`** loops

Questions?

Assignment 5

`while` examples

Integer square root

- What if we want to find the largest integer that is less than or equal to the square root of another integer?
- Sure, we could use the **`math.sqrt()`** function and then convert to an integer
- But we could also loop through different integer values until we find the right one
- We'll need a **`while`** loop since it's not clear how many steps we'll take

Integer square root algorithm

- Start our counter at 1
- As long as the counter squared is less than or equal to the number we're trying to square root
  - Increase it by 1
- Give back an answer one less than the counter, since we overshot by 1

```
def squareRoot(n):
```

Guessing game

- We can write some Python that will guess what number we're thinking of between 1 and 100
- Each time the program guesses a number, we will answer:
  - **`H`** If the number it guessed is too high
  - **`L`** If the number it guessed is too low
  - **`F`** If it found our number

- We have to use a **`while`** loop for this problem, since we have no way of knowing how many guesses it will take

Algorithm for guessing game

- Make a variable holding the start of the range (1)
- Make a variable holding the end of the range (100)
- As long as the number hasn't been guessed:
  - Find the number in the middle of the range (by averaging the minimum and the maximum)
  - Ask the user if the number is right
  - If the number is too high, change the end of the range to the middle
  - If the number is too low, change the start of the range to the middle

List Comprehensions

List comprehensions

- What if we wanted a list with:
  - A bunch of perfect squares in it
  - A bunch of perfect squares of odd numbers
  - Any set of values that we could compute with a short loop
- We could create an empty list and add such things with a loop
- But Python has a tool called a **list comprehension** that lets you put the loop inside the list, generating the values all in one line

A list comprehension for 10 perfect squares

- Code we already know using **`append()`**:

```
values = []
for i in range(10):
  values.append(i**2)
```

- List comprehension version:

```
values = [i**2 for i in range(10)]
```

A list comprehension for perfect squares of odd numbers

- Code we already know using **`append()`**:

```
values = []
for i in range(10):
  if i % 2 == 1:
    values.append(i**2)
```

- List comprehension version:

```
values = [i**2 for i in range(10) if i % 2 == 1]
```

List comprehension syntax

- A list comprehension looks like:

```
[expression for i in iterable if condition]
```

- The **`expression`** part is any single Python expression that generates a value (and usually involves your iterating variable)
- You can use any variable, **`i`** here is just an example
- The **`iterable`** is anything a **`for`** loop can loop over, like a string, another list, or a **`range()`** function
- The **`if condition`** part is optional

Example

- Given a list with all the planets' names, write a list comprehension that puts only those names whose length is shorter than 6 into a new list

Reflections on list comprehensions

- List comprehensions are never necessary
- You can always build a list by appending
- However, list comprehensions can be faster because of internal mechanisms in Python
- They only take a line to write instead of three or four
- Python people love them, so you'll see them quite a bit in other people's code

Reading Data from the Internet

Internet data

- Have you heard of the Internet?
- It's got a lot of data on it
- All that data is sitting on computers *somewhere*
- We often use a URL to give the location of files and other resources on the Internet
- It's possible to open a file remotely if you know its URL

URL

- **URL** is an abbreviation for Uniform Resource Locator
- Format: **protocol host resource parameters**
  - **http://faculty.otterbein.edu/wittman1/comp1800/**
  - **https://www.youtube.com/watch?v=GQf25_9NOts**
- Hosts are often given as domains
  - Top-level domain: **edu**
  - Second-level domain: **otterbein**
  - Subdomain: **faculty**

JSON (JavaScript Object Notation)

- JSON is an industry standard data structure for transmitting data across network connections
- It uses dictionaries and lists to create hierarchical and structured repositories of data that can be accessed programmatically
- JSON data itself is always a string
- Example JSON data:

```
'{"artist":"Led Zeppelin", "name":"Stairway to Heaven",
"length":"7:55", "year":1971}'
```

Libraries for remotely opening files

- To open a file from the Internet with a URL called **`url`**:

```
import urllib.request
file = urllib.request.urlopen(url)
```

- Then, you can read from it like you would read another file
- To turn a JSON file with a URL called **`url`** into a Python object:

```
import json
import urllib.request
file = urllib.request.urlopen(url)
string = file.read() # reads entire file into a single string
data = json.loads(string) # turns JSON string into Python
```

Earthquake example

- We can get a lot of earthquake data stored in a JSON file:

```
import json
import urllib.request
url =
  'https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/4.
  5_month.geojson'
file = urllib.request.urlopen(url)
string = file.read() # reads file into a string
data = json.loads(string) # turns JSON string into Python
features = data['features'] # get list of features
for feature in features:
    properties = feature['properties'] # get all properties
    magnitude = properties['mag'] # get magnitude
    print (magnitude)
```

Upcoming

Next time…

- Image processing
- Read sections 6.2 and 6.3

Reminders

- Read sections 6.2 and 6.3
- Keep working on Assignment 5
  - Due next Friday before midnight