Week 3

Strings and Loops

String accumulators

String formatting

Booleans

Conditionals
Strings and loops

We can use loops to generate patterns of strings

# Print greeting 5 times
greeting = "hello world"
for i in range(5):
    print(greeting)
Print a given word N times

$ python3 stringLoop.py
Enter a string: "hello"
Enter a number of times: 3
hello
hello
hello

```
def main():
    # Get input
    word = input("Enter a string: ")
    num = int(input("Enter a number of times: "))

    for i in range(num):
        print(word)

main()
```
More strings and loops

Strings are a sequence of characters!

What is the output of the following program?

```python
greeting = "hello world"
for i in range(len(greeting)):
    print(greeting[i])
```
Print every character in a given word on its own line

Part 1
1. Ask the user for a word
2. Get the length of the word
3. Use the range function to generate indices from 0 to len(word) - 1
4. In a loop, we just need to print word[i]

```python
def main():
    word = input("Enter a string: ")

    lengthOfWord = len(word)
    for i in range(lengthOfWord):
        print(word[i])

main()
```
Strings can be accumulators

Idea: We can use loops to generate strings

```
accum = ...

for i in range(...):
    # do something
```
Strings can be accumulators

Idea: We can use loops to generate strings

```python
accum = "" # usually initialize with the empty string
for i in range(...):
    # do something
```
Strings can be accumulators

Idea: We can use loops to generate strings

accum = "" # usually initialize with the empty string

for i in range(...):
    # Use concatenation to add to the string
    accum += ...

Exercise - Substrings

1. Ask the user for a word
2. Output increasing substrings of the input

Step 1: Sketch the algorithm on paper
   What are we accumulating?
   What should the start value be?
   How should you update each frame?
   How many times should you loop?

Step 2: Implement the algorithm with code

$ python3 substrings.py
Enter a word: Crackle
C
Cr
Cra
Crac
Crack
Crackl
Crackle
Print the first i characters on its own line

**Algorithm**

1. Define a string accumulator
2. In a loop,
   2a. Add the next letter to the accumulator
   2b. Print out the accumulator

```python
def main():
    word = input("Enter a string: ")

    accum = "" # Set accum to the empty string
    lengthOfWord = len(word)
    for i in range(lengthOfWord):
        accum = accum + word[i]
        print(accum)

main()
```
How does the accumulator change each iteration?

word = “hello”
accum = “”

<table>
<thead>
<tr>
<th>Iteration</th>
<th>i</th>
<th>word[i]</th>
<th>accum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>“h”</td>
<td>accum = “” + “h” = “h”</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>“e”</td>
<td>accum = “h” + “e” = “he”</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>“l”</td>
<td>accum = “he” + “l” = “hel”</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>“l”</td>
<td>accum = “hel” + “l” = “hell”</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>“o”</td>
<td>accum = “hell” + “o” = “hello”</td>
</tr>
</tbody>
</table>
Exercise - Square of text

1. The user will input the size N that the square should be
2. Output N lines. Each line repeats “*” N times

Hint: Use the * operator to repeat a character based on size

Step 1: Write out the algorithm on paper

   How to generate a single line?

   How to repeat that line N times?

Step 2: Implement the algorithm with code

$ python3 square.py
Enter an integer: 1
*

$ python3 square.py
Enter an integer: 4
****
****
****
****
Exercise - Double letters

1. Ask the user for a word
2. Output the word with each letter doubled

Step 1: Sketch the algorithm on paper

   What are we accumulating?
   What should the start value be?
   How should you update each frame?
   How many times should you loop?

Step 2: Implement the algorithm with code

$ python3 doubleletter.py
Enter a word: banana
bbaannaannaa

$ python3 doubleletter.py
Enter a word: lol
llooll