Repetition Structures
Loops

Repeatedly execute blocks of code
Introduce nonlinearity into programs
Loops fall into two general categories
• Condition-controlled loops
• Count-controlled loops
**Condition-Controlled**

“While” Loops

Repeat a block of code while a condition is True

Must initialize variable and set incrementation

\[
i = 0
\]

\[
\text{while } i < 10: \\
\quad \text{print}(i) \\
\quad i = i + 1
\]
Loops

↓

initialization_block

↓

Is condition true?

(yes) ↓

body_block

(no) ➔

after_block

↓
Count-Controlled
“For” Loops

Repeat a block of code a specified number of times

Assumptions are built into for-loops so it is not necessary to initialize variables and set incrementation

```python
for i in range(n):
    print(i)
```
**Count-Controlled String Iteration**

For loops can also be used to access characters in a string.

This is referred to as “iterating over a string” and can be used to access characters in text or lines in a file.

```python
for c in "example":
    print(c)
```

In this example above, for each loop `c` becomes the next character in the string. Therefore the loop will execute 7 times.
**Loop Keywords**

**break**

- Lets you jump out of a loop from within the loop body
- Allows you to skip over unnecessary statements
- Should only be used when it makes your code simpler
Loop Keywords

continue

Related to break statement

Lets you jump to the next iteration of a currently-executing loop

Good for when you want to continue with the loop without doing anything
Introduction to Computer Programming
CSCI-UA 2

Repetition Structures
Loops