Repetition Structures
For-Loops and While-Loops
Loops

Repetition structures

Introduce nonlinearity into programs

Repeatedly execute blocks of code

Loops fall into two general categories

• Condition-controlled loops
• Count-controlled loops
While-Loops

Repeat a block of code while a condition is True

Must initialize variable and set incrementation

\[ i = 0 \]

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

For-Loops and While-Loops

```
<table>
<thead>
<tr>
<th>initialization_block</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Is condition true?</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>(yes)</td>
</tr>
<tr>
<td>body_block</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>after_block</td>
</tr>
<tr>
<td>no</td>
</tr>
</tbody>
</table>
```
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)
```
Break Statement
break

Let's 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
**Continue Statement**

`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
For-Loops and While-Loops