CSCI 141
Scott Wehrwein
for loops
Goals

• Know the syntax and behavior of the `for` statement (for loop)

• Know how to loop over a list of objects using a for loop.
Hot take: for some tasks, while loops are annoying.

- Often, you want: “Do some\_thing() 10 times”

- With a while loop you need to:

  ```
  i = 0
  while i < 10:
      i += 1
      some\_thing()
  ```

- Wouldn’t it be great if we could:

  ```
  do 10 times:
      some\_thing()
  ```

We (almost) can! Using for loops.
The **for** statement: syntax

```
for var_name in sequence:
    codeblock
```

- **for keyword**
- **in keyword**
- **colon**
- **a sequence**
- **a variable name**

*an indented code block: one or more statements to be executed for each iteration of the loop*
Sequences in Python: Lists

```python
for color in ["red", "green", "blue"]:  
    print(color)
```

This is a list: an ordered collection of values. Much more on these later.

This code prints:
red
green
blue
The **for statement**: behavior

```python
for color in ["red", "green", "blue"]:  
    print(color)
```

The loop body is executed once **for each** value in the sequence (list).

This code prints:  
red  
green  
blue

In each iteration, the loop variable (color) takes on a *different* value from the sequence:
("red", then "green", then "blue")
The **for** statement: behavior

```python
for color in ["red", "green", "blue"]:  
    print(color)
```

The loop body is executed once **for each** value in the sequence (list).

This code prints:  

<table>
<thead>
<tr>
<th>red</th>
<th>green</th>
<th>blue</th>
</tr>
</thead>
</table>

In each iteration, the loop variable (color) takes on a *different* value from the sequence:

("red", then "green", then "blue")

**Notice:** the loop variable gets updated **automatically** after each iteration!
Demo

• for_demo.py

  • for loop over strings

  • for loop to square each of a list of numbers