CS100: CPADS

Iteration (Loops)

David Babcock / Don Hake
Department of Physical Sciences
York College of Pennsylvania

Iteration

- **What is iteration?**
  - Repeatedly executing a sequence of statements
    - May need to do the same thing over-and-over again
  - Also referred to as looping
- **Use fixed iteration when the number of iterations is known at the time of programming**
  - The iterations are tracked by a loop counter
- **Use conditional iteration when the number of iterations is based on a logical condition that is updated inside the loop**
  (will be covered another day)

Structure of a `for-loop`

- **Most programming languages (including Python) use the keyword '`for`' to indicate the beginning of a 'loop'**

```
[code before the loop]                               # executes once

for loop_counter in range(num_loops):
    [indented Python statements]                      # executes num_loops times
    [include the work that you want to repeat]        # executes num_loops times

[the code after the loop is not indented]             # executes once
```

Structure of a `for-loop`

- **Most programming languages (including Python) use the keyword '`for`' to indicate the beginning of a '`for-loop`'**

```
[code before the loop]                          # executes once

for loop_counter in range(num_loops):
    [indented Python statements]                 # executes num_loops times
    [include the work that you want to repeat]  # executes num_loops times

[the code after the loop is not indented]        # executes once
```

Each `for-loop` starts with the Python keyword `for`

Structure of a `for-loop`

- **Most programming languages (including Python) use the keyword '`for`' to indicate the beginning of a '`for-loop`'**

```
[code before the loop]                             # executes once

for loop_counter in range(num_loops):
    [indented Python statements]                    # executes num_loops times
    [include the work that you want to repeat]      # executes num_loops times

[the code after the loop is not indented]           # executes once
```

Each `for-loop` contains a `loop_counter` variable that *starts at 0* and increases after each pass through the loop

Note that the `loop_counter` variable can be named anything you like, just like any other variable. Additionally, the loop_counter is AUTOMATICALLY assigned 0 when the loop first runs.

Structure of a `for-loop`

- **Most programming languages (including Python) use the keyword '`for`' to indicate the beginning of a '`for-loop`'**

```
[code before the loop]                               # executes once

for loop_counter in range(num_loops):
    [indented Python statements]                      # executes num_loops times
    [include the work that you want to repeat]        # executes num_loops times

[the code after the loop is not indented]             # executes once
```

The `in range` keywords are part of the `for-loop` declaration and should be used in conjunction with the `for` keyword

Structure of a **`for-loop`**

- **Most programming languages (including Python) use the keyword ‘`for`’ to indicate the beginning of a ‘`for-loop`’**

```
[code before the loop]                              # executes once

for loop_counter in range(num_loops):
    [indented Python statements]                     # executes num_loops times
    [include the work that you want to repeat]       # executes num_loops times

[the code after the loop is not indented]            # executes once
```

Specifies the number of times the loop will execute. Can be a literal value (e.g. **`5`**, **`22`**, etc.) or a variable that is assigned before the **`for-loop`** starts.

Note, the **`loop_counter`** variable will start at 0 and increase on each pass through the loop until it equals **`num_loops-1`**.

How Many Numbers Do you See?

1 2 3 4

What Is the Largest Number You See?

If `num_loops=5`,
the `loop_counter` will range from `0` to `4`

$\uparrow$ `num_loops-1`

Structure of a **`for-loop`**

- **Most programming languages (including Python) use the keyword '`for`' to indicate the beginning of a '`for-loop`'**

```
[code before the loop]                                # executes once

for loop_counter in range(num_loops):
    [indented Python statements]                       # executes num_loops times
    [include the work that you want to repeat]         # executes num_loops times

[the code after the loop is not indented]              # executes once
```

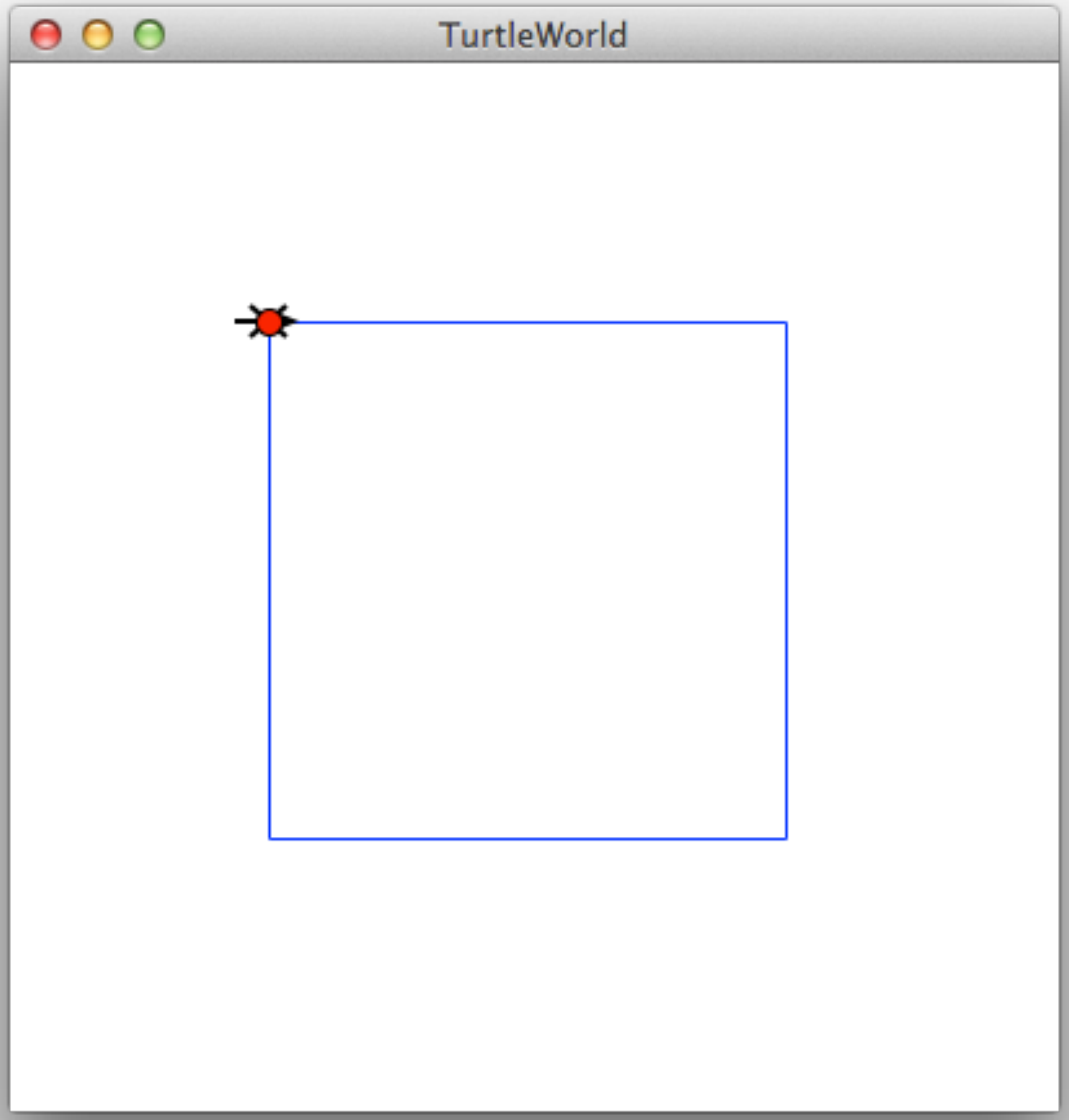
The loop-body of the **`for-loop`** (i.e. the repeated code) contains one or more indented Python statements

A Simple Example

- **Draw a square where each side is of length `size`**

```
# Draw a square
fd(t, size)
rt(t, 90)
fd(t, size)
rt(t, 90)
fd(t, size)
rt(t, 90)
fd(t, size)
rt(t, 90)
```

Note that the code repeats itself.
So, why write it over and over again?

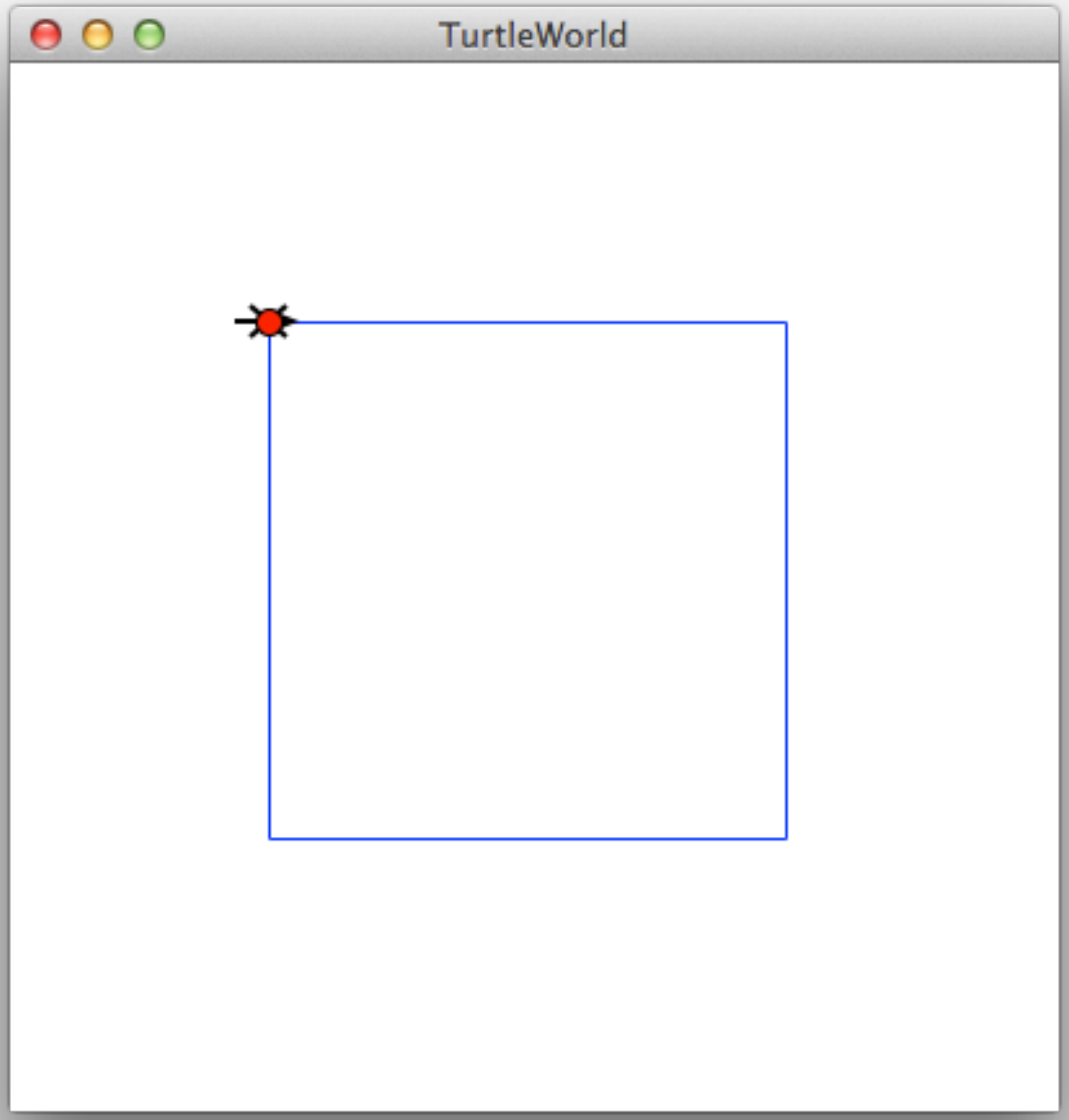


A Simple Example

- **Draw a square where each side is of length `size`**
  - Simplify with a **`for-loop`**

```python
# Draw a square
for i in range(4):
  fd(t, size)
  rt(t, 90)
```

This code will execute 4 times



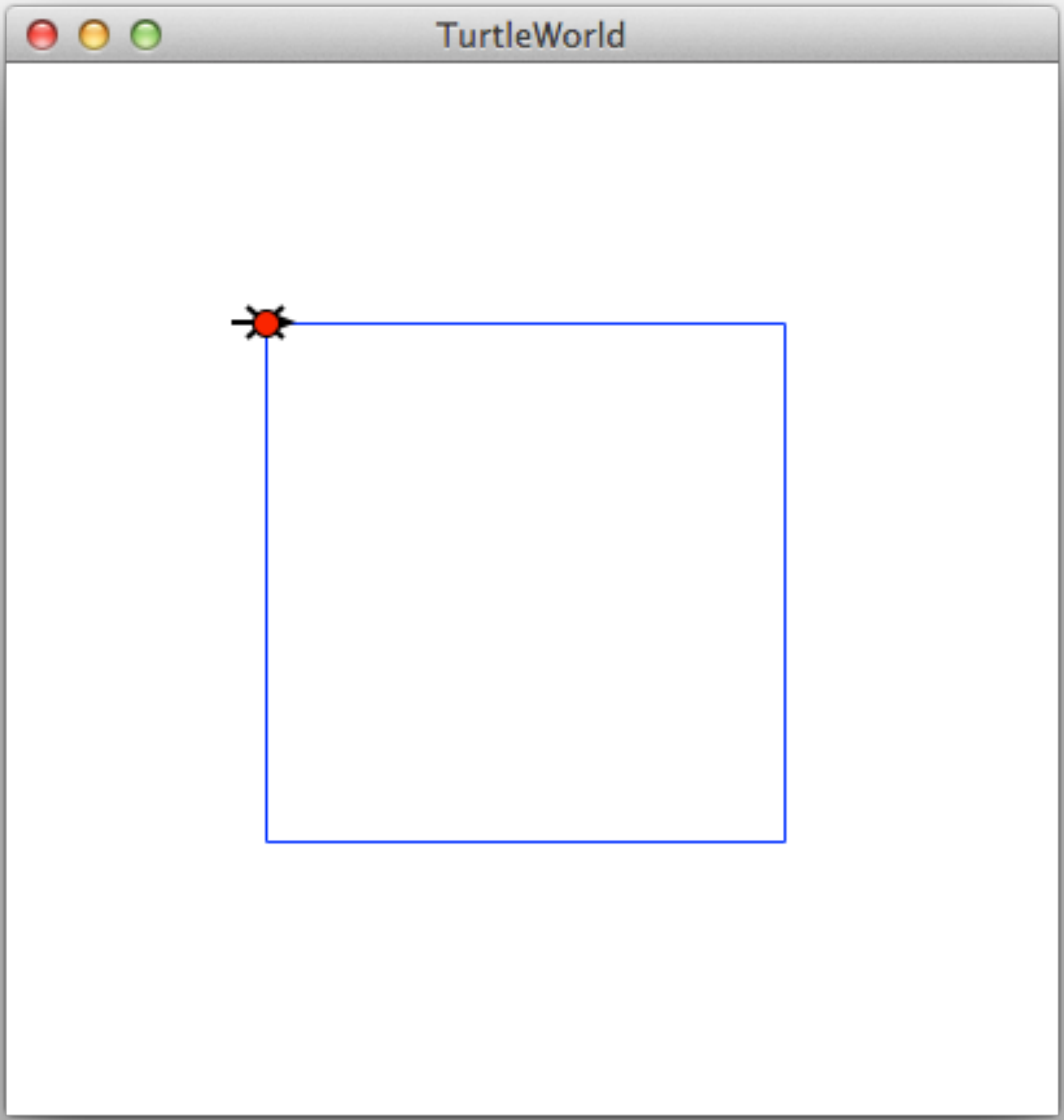
A Simple Example

- **Draw a square where each side is of length `size`**
    - Simplify with a `for-loop`

```python
# Draw a square
num = 4

for i in range(num):
    fd(t, size)
    rt(t, 90)
```

Remember, the range can also be specified using a variable



A More Interesting Example

- **It is possible to use the `loop_counter` variable inside the loop**

```python
# Draw a shape
size = 100
num = 4

for j in range(num):
  fd(t, size*j)
  rt(t, j*30)
```

j=0	j=1	j=2	j=3
fd(t, 0)	fd(t, 100)	fd(t, 200)	fd(t, 300)
rt(t, 0)	rt(t, 30)	rt(t, 60)	rt(t, 90)