Lecture 5: For Loops

Eleanor Birrell

February 3, 2026

Review: Programming in Python

- Values
  - `47`
  - `"hello, world!\n"`
- Types
  - `str`
  - `bool`
- Variables
  - `dist_in_miles = 3.1`
  - `a_string = "hello"`
- Operations
  - `1 * 2 * 3`
  - `a_string + " world"`
- Functions
  -
```
def example(x):
    y = 2*x
    return y
```
  - `z = example(25)`
  - `print("hello, world!")`
- Conditionals
  - `if`
  - `if-else`
  - `if-elif-else`
  -
```
if x > 5:
    x = x - 5
else:
    x = x + 5
```

Review: Modules

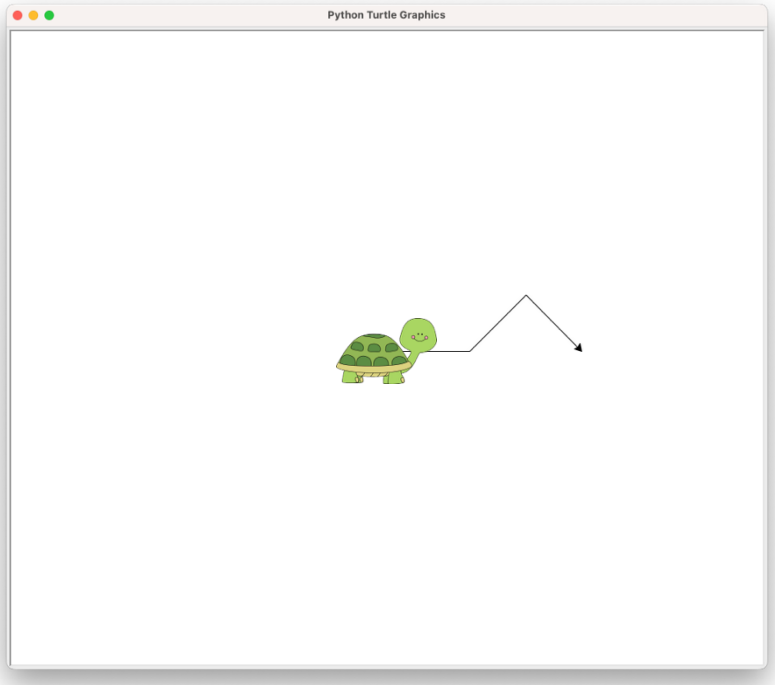
- extra pre-defined functions that you can add to Python

- Examples:
  - math: defines lots of useful math functions
  - random: generate random numbers
  - turtle: draw stuff!

- To use functions defined in a module:
  - `from <module> import <functions>`
  - `from <module> import *`
  - `import <module>`

Review: Turtle Module

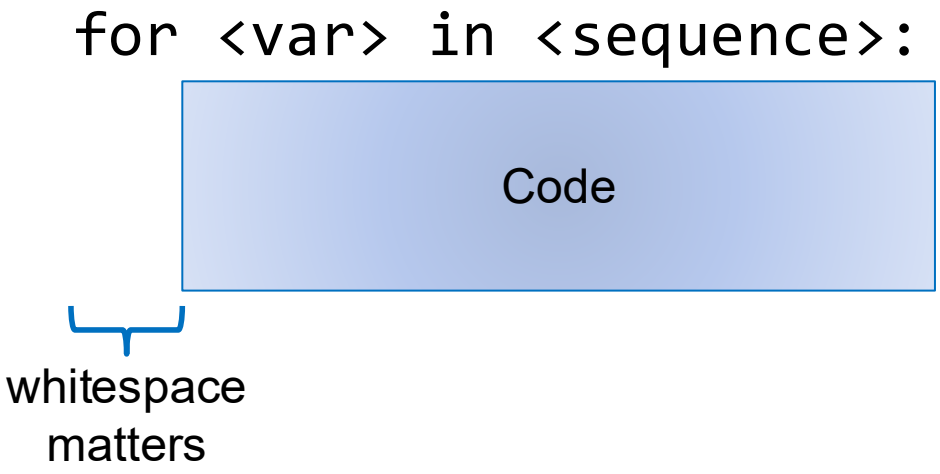
```
from turtle import *

def zig_zag():
    forward(100)
    left(45)
    forward(100)
    right(90)
    forward(100)



zig_zag()
done()
```



Turtle starts at the center of the screen (0, 0) facing right

Review: For loops

- When you want some set of statements to execute repeatedly



Review: Exact Repetition

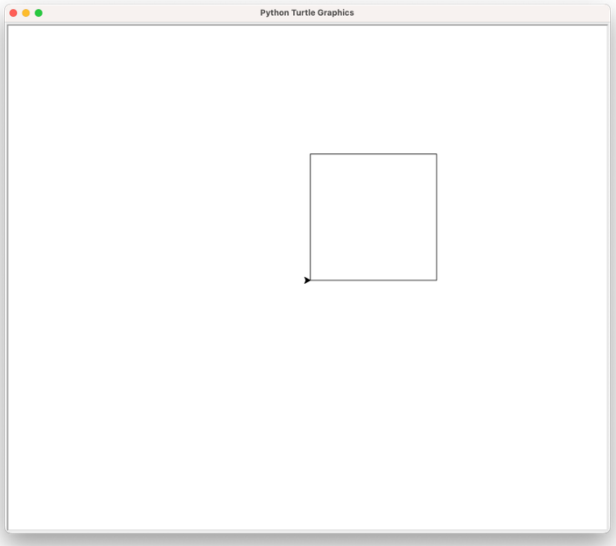
```
def say_hello():
    for x in range(5):
        print("Hello!")
```

```
def draw_square(len):
    for x in range(4):
        forward(len)
        left(90)
```

```
say_hello()
```

```
draw_square(200)
```

```
Hello!
Hello!
Hello!
Hello!
```



Review: Using the Index Variable

variable {0, 1, 2, 3, 4}

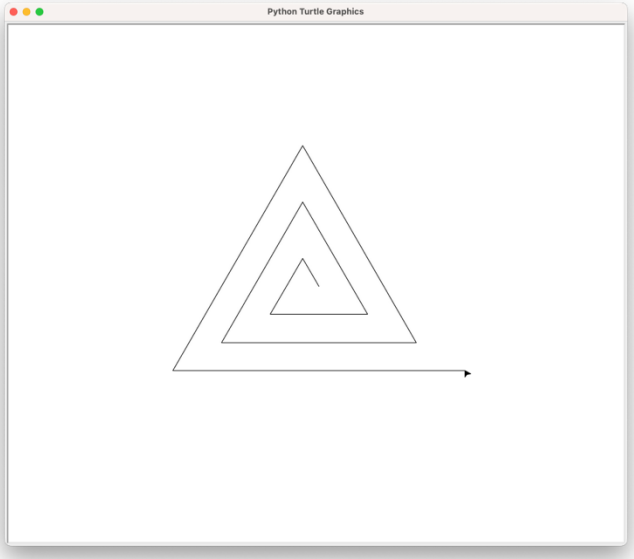
```
def count():
    for x in range(5):
        print(x)
```

```
def spiral(num):
    for len in range(num):
        forward(len*50)
        left(120)
```

```
count()
```

```
spiral(10)
```

```
0
1
2
3
4
```



Exercise 1: Using the Index Variable

Using a for loop, define a function `print_odd_squares` that takes one parameter `max` (an int) and prints the squares of the odd numbers in the range [1, max].

Hint: You should use a conditional.

```
print_odd_squares(9)
```

```
1
9
25
49
81
```

range

- `range([start,] stop [, step])`
- generates a sequence of numbers
- to see the elements, call the function `list`

```
range(5)

range(10)

range(1,10)

range(13,15)

range(1,15,2)

range(5,-5,-1)
```

Exercise 2: ranges

- `range(3)`
- `range(5, 10)`
- `range(5, 0, -1)`
- `range(0, 10, 2)`
- `range(10, 0, 2)`

Exercise 3: index variables

Modify your `print_odd_squares` function so it no longer uses a conditional.

Loop Accumulators

- You can update variables inside a loop even if they are initially defined outside the loop
- This gives you a way to iteratively accumulate a total value. Which is a very powerful programming technique!

```
def pyramid(height):
    string = ""
    for x in range(height):
        string = string + "*" * x + "\n"

    return string
```

```
full_string = pyramid(10)
print(3 * full_string)
```

Exercise 4: Loop Accumulators

Using a for loop, define a function `sum_square_odds` that takes one parameter `max` (an int) and then returns the sum of the odd values in the range [1, max]

For example, `sum_square_odds(5)` should return 9 (since 1 + 3 + 5 == 9)

Docstrings

```
def sum_square_odds(max):
    """
    adds the squares of all the odd numbers in the range [1,max]
    :param max: (int) a positive integer
    :return: (int)  the total sum
    """
```

- Function description
- Parameter description, with data type
- Return value description, with data type

Example: Putting it all together

Define a function `countdown` that takes one parameter (a positive integer) and then counts down from that value to `1` (all on one line!) and then prints "GO!" on the next line.

```
countdown(5)
```

```
5, 4, 3, 2, 1
GO!
```