#CodeYork

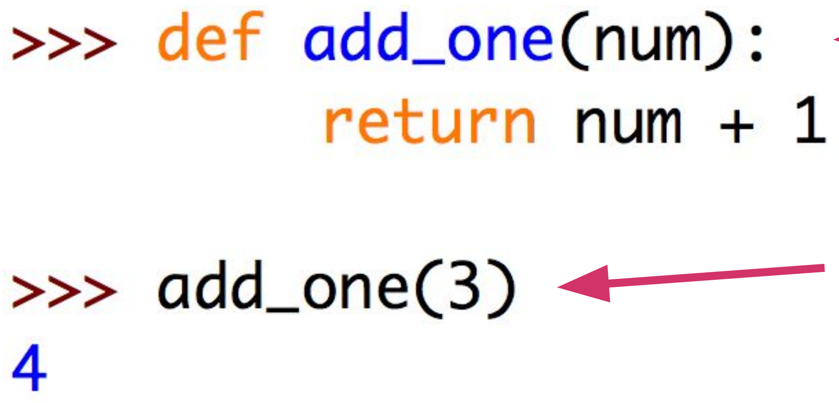
Handout 2: Functions and Control

Intro to Functions

- The ‘verbs’ of a programming language
- Can define an ‘action’ in code and ‘perform’ it at any time
    - In English, can define what ‘speak’ means and do it after, once everyone knows what it means
    - In Python, can define what ‘foo()’ means and do it after, once Python knows what it means
- Sometimes our actions need to know about the world they’re in
    - eg. ‘eating’ needs us to know who’s eating and what they’re eating
- Functions can have <u>parameters</u>, and we can pass <u>arguments</u> to the function:
    - eg. print(“Hello york!”) -> “Hello york!” is the argument

Functions in Python

- Functions are defined using the “def” keyword
- Functions may or may not “return” values



```
>>> def add_one(num):
        return num + 1

>>> add_one(3)
4
```

Function Definition

Function Call

Functions Calling Functions

- Functions can call other functions if needed

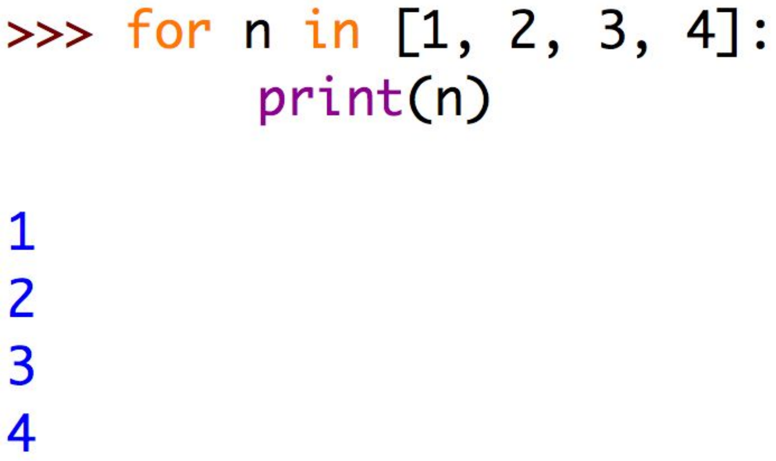
```
def add_one(num):
    return num + 1

def add_two(num):
    return add_one(add_one(num))


print(add_two(3))
```

Python’s Built-In Functions

- In the previous example, we called the “print” function
- We can call the built-in functions in exactly the same was as our own
- Some functions we’ve already seen are:
    - print
    - range
    - len

For Loops

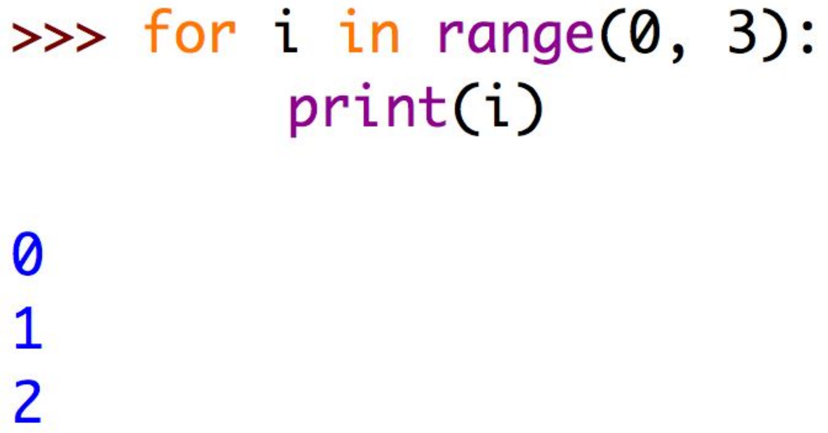
- For loops to do something for every element



```
>>> for n in [1, 2, 3, 4]:
        print(n)

1
2
3
4
```

The Range Function

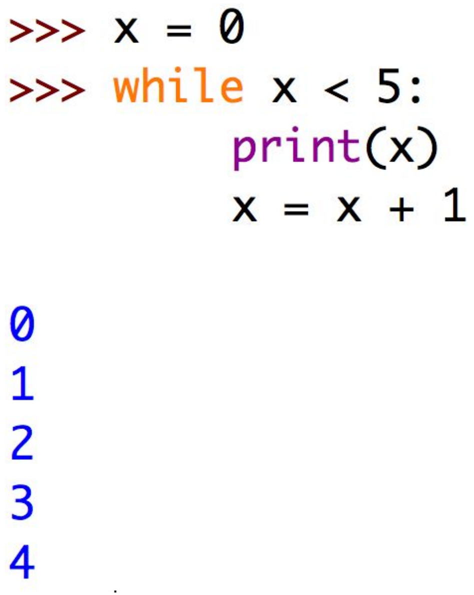
- The range function behaves a bit like a list



```
>>> for i in range(0, 3):
        print(i)

0
1
2
```

While Loops

- While loops will repeat until a condition stops being true



```
>>> x = 0
>>> while x < 5:
        print(x)
        x = x + 1

0
1
2
3
4
```