Function Review

- A function is a group of statements to which we assign a name.

- We use the def keyword to a define a function.

- When we want to execute the statements in the function, we call the function by using its name with parentheses after it.
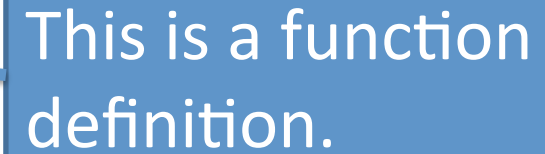
```python
def average(a, b, c):
  avg = (a + b + c)/3
  print("The average of your \
    numbers is", avg)

def main():
  x = input("Give me a number: ")
  y = input("Give me a number: ")
  z = input("Give me a number: ")
  average(x, y, z)

main()
```

```
def average(a, b, c):
  avg = (a + b + c)/3
  print("The average of your \
    numbers is", avg)

def main():
  x = input("Give me a number: ")
  y = input("Give me a number: ")
  z = input("Give me a number: ")
  average(x, y, z)

main()
```

This is a function definition.

This is a function call

```python
def average(a, b, c):
    avg = (a + b + c)/3
    print("The average of your \
        numbers is", avg)

def main():
    x = input("Give me a number: ")
    y = input("Give me a number: ")
    z = input("Give me a number: ")
    average(x, y, z)

main()
```

When main calls average, Python copies the values of x, y, and z (local variables in main) into a, b, and c (local variables in average).

- Pretend we're computing grades for a class that has three homework assignments and three tests. The final graded in the class is weighted so that 75% of the final grade is from the test average and 25% is from the homework average.

- We'd like to write a program to use our average function to take the averages of the test and homework grades, and then weight those averages appropriately to compute a final course grade.

```python
def average(a, b, c):
    avg = (a + b + c)/3
    print("The average of your numbers is", avg)

def main():
    test1 = input("Give me the first test grade: ")
    test2 = input("Give me the second test grade: ")
    test3 = input("Give me the third test grade: ")
    average(test1, test2, test3)

    hw1 = input("Give me the first HW grade: ")
    hw2 = input("Give me the second HW grade: ")
    hw3 = input("Give me the third HW grade: ")
    average(hw1, hw2, hw3)

    # some code here to weight the test average by 0.75
    # and the quiz average by 0.25 and combine them.

main()
```

```
def average(a, b, c):
    avg = (a + b + c)/3
    print("The average of you

def main():
    test1 = input("Give me th
    test2 = input("Give me th
    test3 = input("Give me th
    average(test1, test2, tes

    hw1 = input("Give me the
    hw2 = input("Give me the
    hw3 = input("Give me the
    average(hw1, hw2, hw3)

    # some code here to weight the test average by 0.75
    # and the quiz average by 0.25 and combine them.

main()
```

main can't see the "avg" variable inside of `average` because `avg` is a local variable.

Furthermore, whenever we call `average`, a new `avg` variable is created and the old one is lost. Even if we could access `avg` from main, there's no way we could have both the homework and test `avg` values at the same time.

```
def average(a, b, c):
    avg = (a + b + c)/3
```

What we want to do is:

final_grade = 0.75 * (avg from the first call to average) + 0.25 * (avg from the 2nd call)

```
    test3 = input("Give me the third test grade: ")
    average(test1, test2, test3)

    hw1 = input("Give me the first HW grade: ")
    hw2 = input("Give me the second HW grade: ")
    hw3 = input("Give me the third HW grade: ")
    average(hw1, hw2, hw3)

    # some code here to weight the test average by 0.75
    # and the quiz average by 0.25 and combine them.

main()
```

Return values to the rescue!
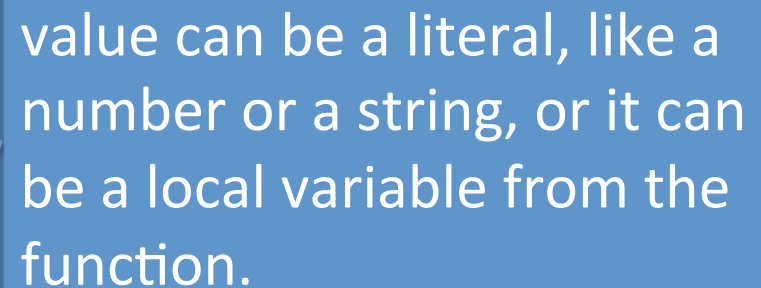
```
def function(arg1, arg2, …):
    statement
    statement
    [ more statements if desired ]
    return value
```
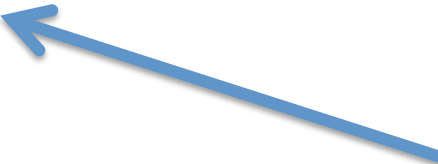
value can be a literal, like a number or a string, or it can be a local variable from the function.

Return values to the rescue!

def function(arg1, arg2, ...):

    statement

    statement

    [more statements if desired]

    return value

When Python sees a line in a function beginning with "return," the function immediately ends, and the value is sent back to the caller.

value can be a literal, like a number or a string, or it can be a local variable from the function.

Capturing the return value

- Use an assignment statement to "capture" the return value.
  - Otherwise it disappears!

```
variable = function(…)
```

When Python sees a line like this, the function is called normally. However, when the function ends and a value is "sent back" to the caller, the value is put into the variable you specify.

```python
def average(a, b, c):
    avg = (a + b + c)/3
    return avg
def main():
    test1 = input("Give me the first test grade: ")
    test2 = input("Give me the second test grade: ")
    test3 = input("Give me the third test grade: ")
    test_avg = average(test1, test2, test3)
    print("Your test average is", test_avg)
    hw1 = input("Give me the first HW grade: ")
    hw2 = input("Give me the second HW grade: ")
    hw3 = input("Give me the third HW grade: ")
    hw_avg = average(hw1, hw2, hw3)
    print("Your homework average is", hw_avg)
    final_grade = 0.75 * test_avg + 0.25 * hw_avg
    print("Your final grade is", final_grade)

main()
```

Notice `average` now returns the local variable `avg`, and the print statement is removed.

```python
def average(a, b, c):
    avg = (a + b + c)/3
    return avg
def main():
    test1 = input("Give me the first test grade: ")
    test2 = input("Give me the second test grade: ")
    test3 = input("Give me the third test grade: ")
    test_avg = average(test1, test2, test3)
    print("Your test average is", test_avg)
    hw1 = input("Give me the first HW grade: ")
    hw2 = input("Give me the second HW grade: ")
    hw3 = input("Give me the third HW grade: ")
    hw_avg = average(hw1, hw2, hw3)
    print("Your homework average is", hw_avg)
    final_grade = 0.75 * test_avg + 0.25 * hw_avg
    print("Your final grade is", final_grade)

main()
```

main calls average: values test1, test2, and test3 are copied into a, b, and c.

```
def average(a, b, c):
    avg = (a + b + c)/3
    return avg
def main():
    test1 = input("Give me the first test grade: ")
    test2 = input("Give me the second test grade: ")
    test3 = input("Give me the third test grade: ")
    test_avg = average(test1, test2, test3)
    print("Your test average is", test_avg)
    hw1 = input("Give me the first HW grade: ")
    hw2 = input("Give me the second HW grade: ")
    hw3 = input("Give me the third HW grade: ")
    hw_avg = average(hw1, hw2, hw3)
    print("Your homework average is", hw_avg)
    final_grade = 0.75 * test_avg + 0.25 * hw_avg
    print("Your final grade is", final_grade)

main()
```

average returns a copy of its local variable `avg` back to main, and the value is assigned to `test_avg`.

```python
def average(a, b, c):
    avg = (a + b + c)/3
    return avg
def main():
    test1 = input("Give me the first test grade: ")
    test2 = input("Give me the second test grade: ")
    test3 = input("Give me the third test grade: ")
    test_avg = average(test1, test2, test3)
    print("Your test average is", test_avg)
    hw1 = input("Give me the first HW grade: ")
    hw2 = input("Give me the second HW grade: ")
    hw3 = input("Give me the third HW grade: ")
    hw_avg = average(hw1, hw2, hw3)
    print("Your homework average is", hw_avg)
    final_grade = 0.75 * test_avg + 0.25 * hw_avg
    print("Your final grade is", final_grade)

main()
```

main calls average: values hw1, hw2, and hw3 are copied into a, b, and c.

```
def average(a, b, c):
    avg = (a + b + c)/3
    return avg
def main():
    test1 = input("Give me the first test grade: ")
    test2 = input("Give me the second test grade: ")
    test3 = input("Give me the third test grade: ")
    test_avg = average(test1, test2, test3)
    print("Your test average is", test_avg)
    hw1 = input("Give me the first HW grade: ")
    hw2 = input("Give me the second HW grade: ")
    hw3 = input("Give me the third HW grade: ")
    hw_avg = average(hw1, hw2, hw3)
    print("Your homework average is", hw_avg)
    final_grade = 0.75 * test_avg + 0.25 * hw_avg
    print("Your final grade is", final_grade)

main()
```

average returns a copy of its local variable `avg` back to main, and the value is assigned to `hw_avg`.

- When writing functions, you should test them to make sure they work in all kinds of situations.
  - Does average() work with negative numbers? Floating point numbers?
- You can write a program to do testing, by calling the function with varying arguments.
- Or, you can test your function using the Python Shell (the window where every line starts with >>>)