Announcements

- Reminders:
  - Program 3 due Thursday night by 11:55pm

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

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

main()
• Pretend we're computing grades for a class that has three homework assignments and three tests. The final grade 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()
```

What we want to do is:

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

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.

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

```python
# 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!

```python
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!

```python
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.

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

Capturing the return value

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

```python
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.
```

Notice average now returns the local variable, 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()
```
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()
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 `>>>`)

---

**Class Exercise**

Write a program that computes the annual household income for a family with 2 working adults.

1. Prompt the user for their and their partner’s hourly wage, as well as the tax rate.
2. Calculate the total income for each of the adults after taxes. (Assume 40 hours/week and 52 weeks/year).
3. Output the total household income.

---

**Practice**

Write a function called direction that takes two float arguments, x and y. Consider an arrow on the Cartesian plane pointing from (0,0) to (x, y). This function should return the string “NE”, “SE”, “SW”, or “NW” depending on the direction that the arrow points.

- The `def` line will be: `def direction(x, y):`
- Do not write a main() function. Test your function from the Python shell.