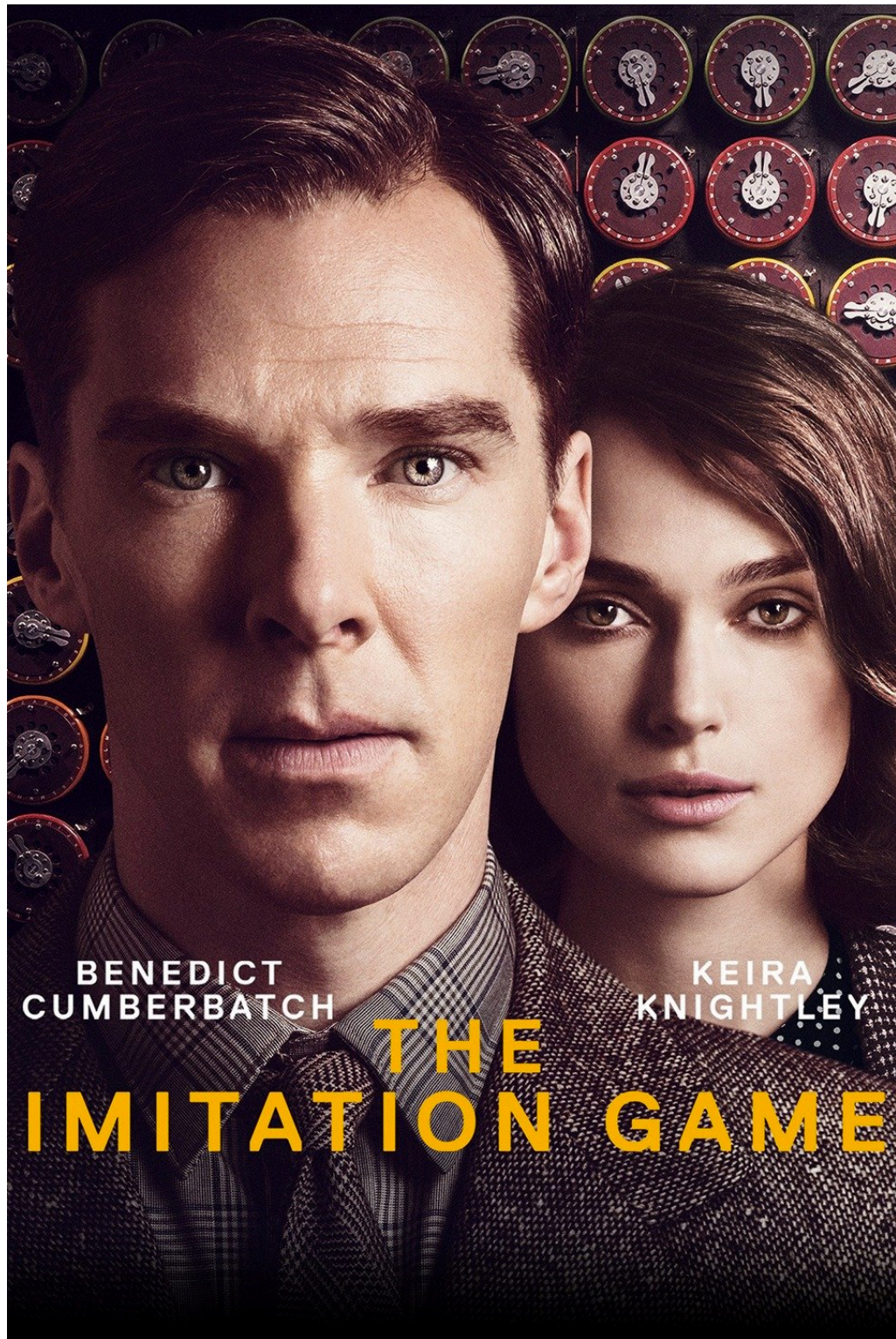Alan Turing was an English mathematician and computer scientist. He is well-known for formalizing the concept of what an algorithm is, and what computers are capable of doing (and what they're not capable of doing)! He's also known for the Turing test, an attempt to define what it means for a machine to be intelligent.
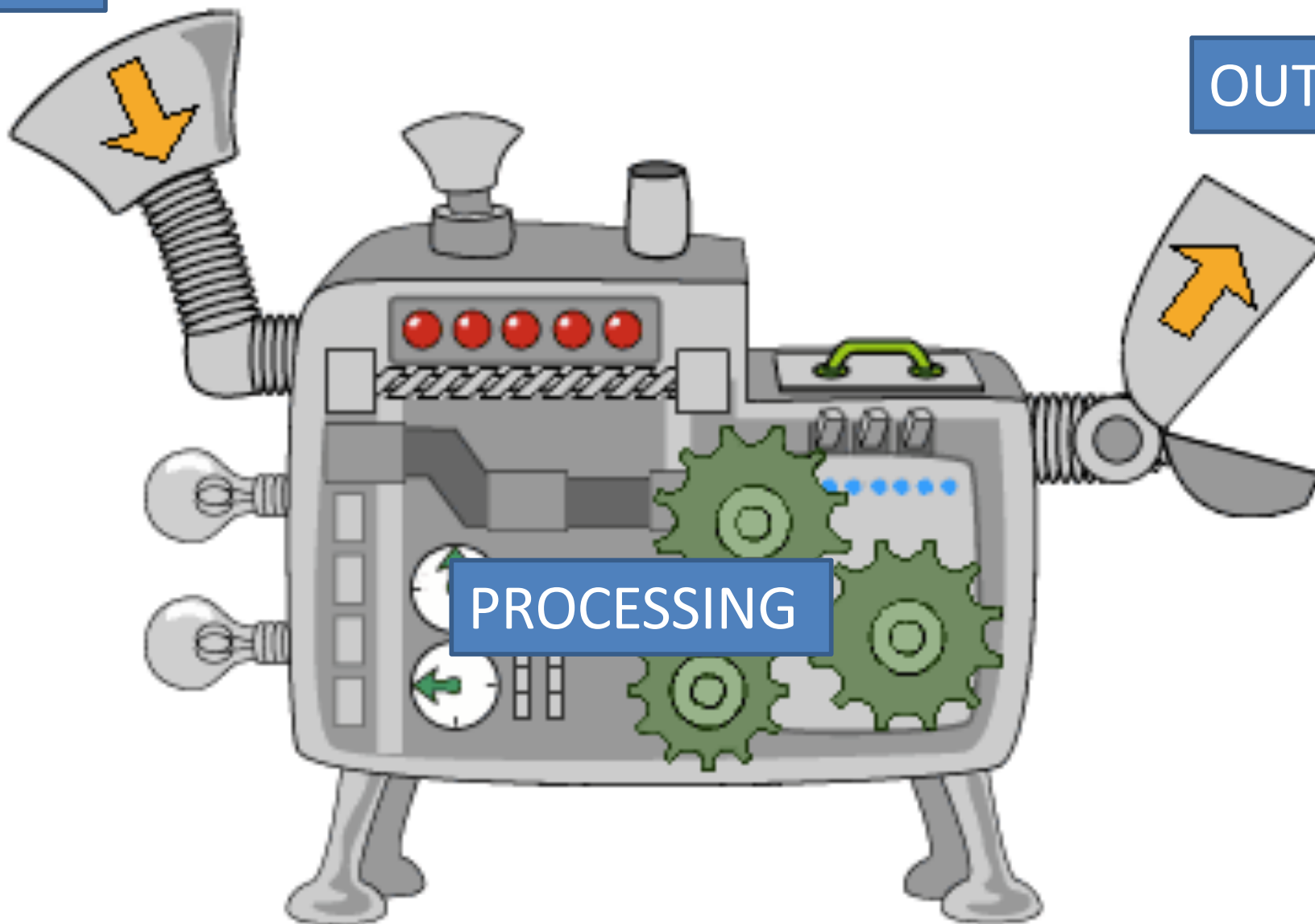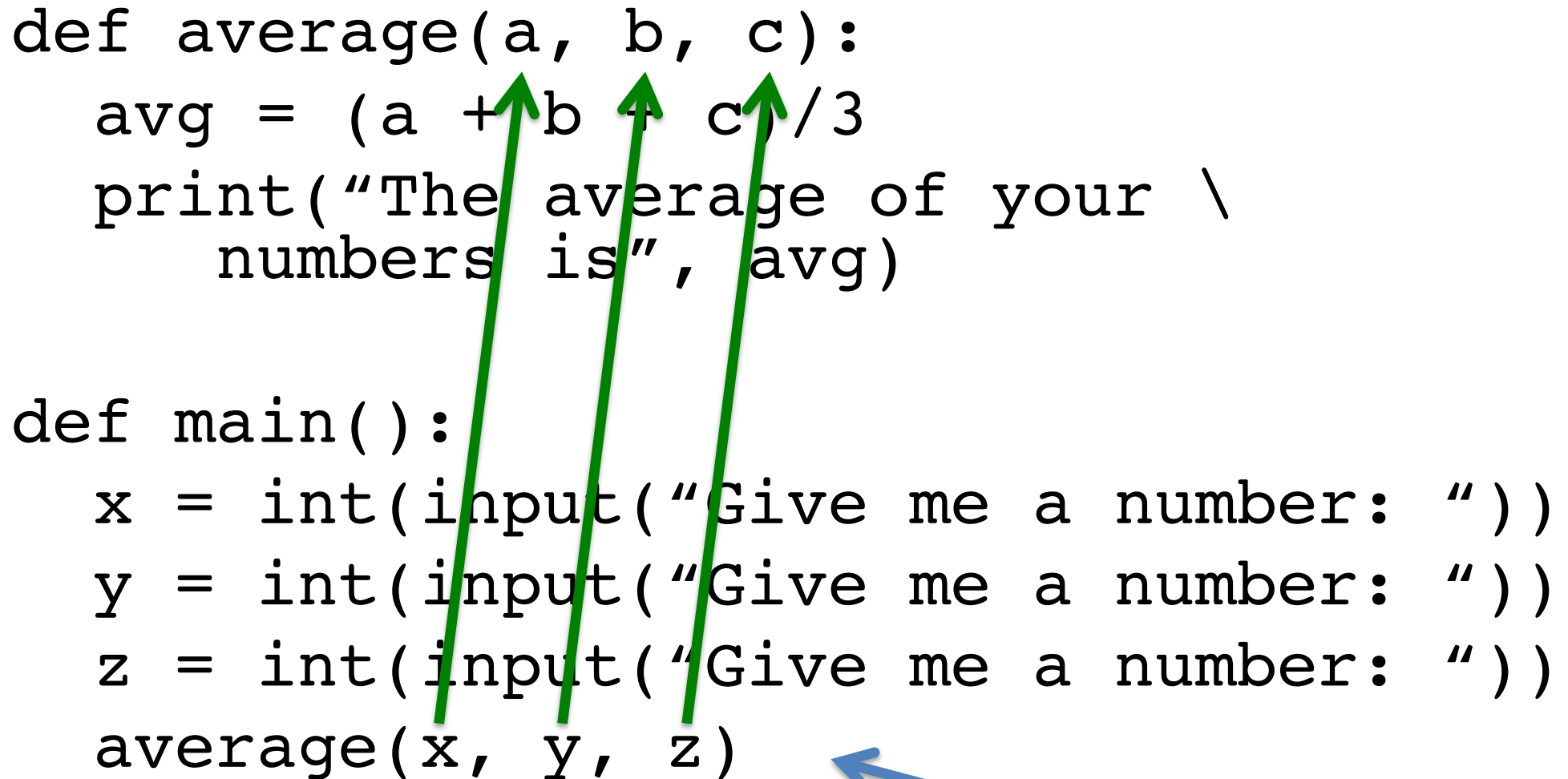
INPUT

OUTPUT

PROCESSING

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

```python
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()
```

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 calculating grades for a class that has three homework assignments and three tests. Your final grade in the class is weighted so that
  - 75% of the final grade is from the **average of the three tests**, and
  - 25% is from the **average of the three homework assignments.**
- 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()
```

```python
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.

```python
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)

```python
    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()
```
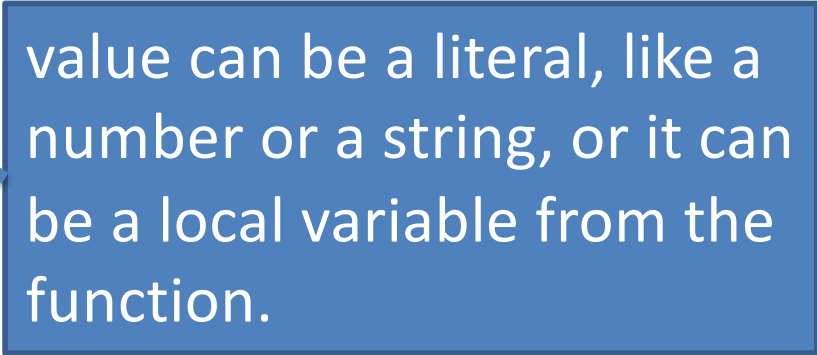
Return values to the rescue!

```
def function(param1, param2, …):
    statement
    statement
    [ more statements if desired ]
    return value
```
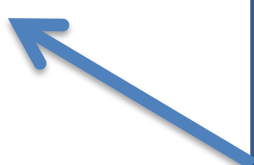
value can be a literal, like a number or a string, or it can be a local variable from the function.

Return values to the rescue!

```
def function(param1,
    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!

- Syntax:

`variable`` = function(arg1, arg2, ...)`

This variable "captures" the return value from the function. The variable will be set to whatever is after the word "`return`" in the function definition.

When Python sees a line like this, the function is called normally. However, when the function ends with `return` and a value is "sent back" to the caller, the value is copied 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.

```
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.

```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()
```

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`.

```python
def square(x):
    return x * x


def larger(a, b):
    if a > b:
        return a
    else:
        return b
```

Compare:

```
def square(x):
    return x * x
```

```
def square():
  x = int(input("What is x?"))
  print(x * x)
```

Take-away:

- If a function needs information from outside the function to do its job, it's usually better to receive that input via a parameter, rather than use an input statement inside the function.

- OK to use an input statement outside the function and then send the variable via an argument into the function.

Take-away:

- If a function wants to communicate its answer to the outside world, it's usually better to send that answer back via a return value rather than use a print statement inside the function.

- OK to capture the return value outside the function and then use a print statement outside the function.

- 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 >>>)

- Write a function called **salary** that takes two arguments: your *hourly wage* and your *tax bracket percent* (e.g., 0.15). This function should ***return*** your total income for the year, after taxes are deducted. Assume you are paid for 40 hours/week, 52 weeks/year.
  - The definition line will be `def salary(wage, bracket):`
  - Do not write a main() function. Test this from the Python shell.

- 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. Assume x and y will never be zero.
  - The def line will be: `def direction(x, y):`