1 Importing modules

The module, **our_math**, contains two functions that may be useful in other programs. It also contains some calls to those functions that the writer of that module wanted to execute (lines 13-15). [**our_math.py**]

```
1.  def add(x, y):
2.      '''(number, number) -> number
3.      Return the sum of x and y.'''
4.
5.      return x + y
6.
7.  def square(x):
8.      '''(number) -> number
9.      Return x squared.'''
10.
11.     return x ** 2
12.
13. print square(add(34, 23.9))
14. sum = add(23.5, 34)
15. print square(sum / 3)
```

Now let's use the **add** function from the **our_math** module in another module:

[**calculations.py**]

```
import our_math
def average(grade1, grade2):
    '''(number, number) -> float
    return the average of grade1 and grade2.'''

    return our_math.add(grade1, grade2) / 2.0

avg = average(74.3, 85)
print "Your test average is :" + str(avg)
```

Q. What do we see in the shell when we run **calculations.py**?

1.1 if __name__ == '__main__'

When importing a module, we do not want the entire program to execute, just the function definitions. Let's modify **our_math** so that the code on lines 13-15 (referring to line numbers in **our_math.py**) won't be executed when the module is imported:

[**our_math_improved.py**]

```
def add(x, y):
    '''(number, number) -> number
    Return the sum of x and y.'''

    return x + y

def square(x):
    '''(number) -> number
    Return x squared.'''

    return x ** 2

if __main__ == '__main__':
        print square (add(34, 23.9)
        sum = (23.5, 34)
        print square (sum / 3)
```

this is the very module being run, do the code
below. But if it's just being imported, don't!

Now let's make **calculations_improved.py**, which has the same code as **calculations.py**, except it imports **our_math_improved** instead of **our_math**.

Q. Now what do we see in the shell when we Run **calculations.py**?

__name__ is a built-in variable that every module has. If the module is the one that is being run, then **__name__** has the value **"__main__"**, otherwise it has the module name.

The statement **if __name__ == "__main__"** says:

Any "loose" code (code not inside a function) should be collected within **if __name__ == "__main__":** at the end of the module, after all the function definitions. From now on, we will include **if __name__ == "__main__":** in every module we create.

1.2 Tracing: calculations.py and calculations_improved.py

Trace both versions of the **calculations** module. Be sure to click "Step Into" on the **import** statement to see what happens in each version. Also, keep a close eye on the **__name__** variable in the Stack Data and make note of how it changes depending on which module is currently being executed.