Review: *import* statement

- Two ways to use *import* statement:
  - **import** `<modulename>`
    - Then, need to prepend modulename to each constant or function
    - Ex: `math.ceil`, `math.pi`
  - **from** `<modulename> import <defn>`
    - Can then just use function or constant
    - Ex: `ceil`, `pi`

- Note: when have `xrange(5)`
  - `i` gets values `(0, 1, 2, 3, 4)`
  - Which means that loop executes 5 times

---

**Summing 5 Numbers from User**

```python
# keep track of running total
total = 0

# repeat: get user input for 5 numbers,
# keep running total
for i in xrange(1, 6):
    userNum = input("Enter number " + str(i) + ": ")
    # update running total
    total = total + userNum

# display total
print "The total is", total
```

---

**Accumulator Design Pattern**

- Initialize accumulator variable
- Loop until done
  - Update the value of the accumulator
- Display result

---

**Programming Practice**

- Average 5 numbers inputted by the user

```python
# keep track of running total
total = 0

# repeat: get user input for 5 numbers,
# keep running total
for i in xrange(1, 6):
    userNum = input("Enter number " + str(i) + ": ")
    # update running total
    total = total + userNum

# display total
print "The total is", total
```

---

**Lab 1 Feedback**

- Good test cases
  - Ex: Use well-known values for F→C conversion
- Good variable naming
- Good high-level descriptions
  - I use to make sure you understand the purpose of the program
### Lab 1 Feedback: Common Issues

- **Common mistakes**
  - Not executing program more than one time if have input from user
  - Unlabeled output
    - Tell user what is being output
  - Area of triangle result: not a float
    - Would find if test with two odd numbers
    - Common correct solutions: /2.0 or * .5
- **Common issue**
  - "Over floating"
  - Only need to ensure floats when doing division

### Lab 2 Overview

- **Practice Python programming**
  - String operations
  - Using Functions, Modules
  - for loops