Announcements

- MyProgrammingLab
  - Assignment 1 due Sept. 10, 2014 by 11:59pm
  - myprogramminglab.com

- Program 1 has been assigned
  - Due Sept. 11th by 11:55pm via Moodle
  - Details on Course Website

Performing Calculations

- Math expression: performs calculation and gives a value
  - Math operator: tool for performing calculation
  - Operands: values surrounding operator
    - Variables can be used as operands
    - Resulting value typically assigned to variable

- Two types of division:
  - / operator performs floating point division
  - // operator performs integer division
    - Positive results truncated, negative rounded away from zero

Operator Precedence and Grouping with Parentheses

- Python operator precedence:
  1. Operations enclosed in parentheses
    - Forces operations to be performed before others
  2. Exponentiation (**)
  3. Multiplication (*), division (/ and //), and remainder (%)
  4. Addition (+) and subtraction (-)

- Higher precedence performed first
  - Same precedence operators execute from left to right
The Exponent Operator and the Remainder Operator

- **Exponent operator (**):** Raises a number to a power
  - \( x \) ** \( y \) = \( x^y \)
- **Remainder operator (%):** Performs division and returns the remainder
  - a.k.a. modulus operator
  - e.g., \( 4 \% 2 = 0 \), \( 5 \% 2 = 1 \)
  - Typically used to convert times and distances, and to detect odd or even numbers

Converting Math Formulas to Programming Statements

- Operator required for any mathematical operation
- When converting mathematical expression to programming statement:
  - May need to add multiplication operators
  - May need to insert parentheses

Mixed-Type Expressions and Data Type Conversion

- Data type resulting from math operation depends on data types of operands
  - Two int values: result is an int
  - Two float values: result is a float
  - int and float: int temporarily converted to float, result of the operation is a float
  - Mixed-type expression
  - Type conversion of float to int causes truncation of fractional part

Performing Calculation Practice

You’re working at a fast food restaurant where a burger costs $3.99 and French fries cost $1.99.

- Write a program (save this as a script) that uses 2 variables to store these two prices.
- Your program should then print out the cost of buying 2 burgers and 3 fries.
- If you finish early, make your program add in 9.25% sales tax.
Reading Input from the Keyboard

- Most programs need to read input from the user
- Built-in `input` function reads input from keyboard
  - Returns the data as a string
  - Format: `variable = input(prompt)`
  - `prompt` is typically a string instructing the user to enter a value
  - Does not automatically display a space after the prompt

Example:
```python
>>> name = input("Please enter your name: ")
```

Reading Numbers with the `input` Function

- `input` function always returns a string
- Built-in functions convert between data types
  - `int(item)` converts `item` to an int
  - `float(item)` converts `item` to a float
- Nested function call: general format:
  ```python
  function1(function2(argument))
  ```
  - Value returned by function2 is passed to function1
  - Type conversion only works if `item` is valid numeric value, otherwise, throws exception

Input from Keyboard

- For integers:
  ```python
  variable = int(input("Prompt"))
  ```
- For floats:
  ```python
  variable = float(input("Prompt"))
  ```
- For strings:
  ```python
  variable = input("Prompt")
  ```

Breaking Long Statements into Multiple Lines

- Long statements cannot be viewed on screen without scrolling and cannot be printed without cutting off
- Multi-line continuation character (`\`): Allows to break a statement into multiple lines
  ```python
  print('my first name is', first_name)
  ```
  ** You should use this if your code is going to word-wrap (more than 80 characters on a line)
More About Data Output

• **print** function displays line of output
  – Newline character at end of printed data
  – Special argument `end='delimiter'` causes `print` to place `delimiter` at end of data instead of newline character

• **print** function uses space as item separator
  – Special argument `sep='delimiter'` causes `print` to use `delimiter` as item separator

More About Data Output (cont’d.)

• Special characters appearing in string literal
  – Preceded by backslash (\)
    • Examples: newline (\n), horizontal tab (\t)
  – Treated as commands embedded in string

• When + operator used on two strings in performs string concatenation
  – Useful for breaking up a long string literal

Formatting Numbers

• Can format display of numbers on screen using built-in `format` function
  – Two arguments:
    • Numeric value to be formatted
    • Format specifier
  – Returns string containing formatted number
  – Format specifier typically includes precision and data type
    • Can be used to indicate scientific notation, comma separators, and the minimum field width used to display the value

Formatting Numbers (cont’d.)

• The `%` symbol can be used in the format string of `format` function to format number as percentage

• To format an integer using `format` function:
  – Use `d` as the type designator
  – Do not specify precision
  – Can still use `format` function to set field width or comma separator
**Demo**

Code will be posted in public directory after class.

**Practice**

1. Modify your food program (burger and fries) to prompt the user for the price of a burger and the price of fries instead of having those values hard-coded in.

2. Write a short program (new file) that will calculate the proper tip on a bill.
   - Prompt the user for the total cost of the bill.
   - Assume you are leaving an 18% tip.
   - Calculate the total tip for the bill.
   - Output the tip amount
   - If you’re done early, prompt the user for the tax rate, then use it to calculate the total cost of the bill with tax and tip.

**Next Time**

- Control Statements
- Sections 3.1 - 3.3