CS 112: Intro to Comp Prog

- Lecture Review
- Data Types
- String Operations
- Arithmetic Operators
- Variables
- In-Class Exercises
- Lab Assignment #2
- Upcoming
Lecture Topics

- Variables *
- Types
- Functions
  - Purpose
  - Parameters
  - Return
- Design Process **

* Will see in more depth, on Slide 6
** Will see in more depth, on Slide 8
Data Types

- **Integer**
  - `int()` (aka Fixed-Point)
  - `{ -1, 0, 7, 3, -5, ... }

- **Floating-Point**
  - `float()` (aka Decimal)
  - `{ -0.25, 12.41, -0.1, 0, 5.0, ... }

- **String**
  - `str()`
  - {“Hello World”, “CS 112 Rocks!”, “Pineapple”, ...}
  - Anything surrounded by quotes is considered a String
String Operations

- `+` and `*` are operators with Strings (both sides must be Strings)...they can be used to combine or repeat a string
  - `+` concatenation operator
  - `*` repetition operator

- `print "Hi There, " + "John"` >>> Hi There, John
- `print "Beetlejuice " * 3` >>> Beetlejuice Beetlejuice Beetlejuice
- `print "spider-" + "man"` >>> spider-man

- `print "what is " + "happening " + "my" + " friend?"` >>> what is happening my friend?
Arithmetic Operators

- Mathematical operators exist for both floats and ints
  - + addition
  - - subtraction
  - * multiplication
  - / division (for version less than 3: \(1/2 = 0\), v3+: \(1/2 = 0.5\))
  - % modulus (returns the remainder)

- (Version <3.0) When both dividend and divisor are integers then the result must also be an integer (one way to think of this is you do division normally and just hack off the decimals...no rounding)
  - \(8/3 = 2.6666667\) cut off the decimal part and it is just 2
  - \(3/4 = 0.75\) cut off the decimal part and it is just 0

- Modulus is what is left over (Let's say I want groups of 4 out of 23 students, how many full groups do I have, and what is the size of the group that is not 4)...if we do 23 % 4 this will result in 3.
Variables

- Variables denote blocks in memory that allow us (the programmers) to store and recall information at any point in the program. In order to create a variable we will need to specify a name and assign a value into it (whether it be text, numbers, etc.) this data specifies the variable's type.

- \( x = 5 \) - variable \( x \) is being created and assigned the integer value: 5
- \( \text{text} = \text{“hello”} \) - variable \( \text{text} \) is being created and assigned the string value: “hello”
- \( \text{data} = 8.1 \) - variable \( \text{data} \) is being created and assigned the floating-point value: 8.1
Naming Variables

- Variable Names can only contain numbers, letters, and underscores
- Variable Name cannot start with a number
- Use meaningful names.

- Good Names
  - first_name
  - fifteenPer
  - bill
  - LastName
  - amt
  - meters

- Bad Names (invalid & poorly named)
  - 1stName
  - fhdjsghg
  - name#1
  - feet&inches
  - purplepeopleater
A Top-Down Approach

• We can use a Top-Down approach to designing our program, these approach helps us keep the goal of the program in mind and “not lose the forest for the trees”

• Top-Down means looking from the big picture and breaking it down

• Example: An iron crowbar -> iron elements -> atoms -> protons, neutron, electrons -> ...

• Thus we can identify a broad set of steps needed to accomplish our goal. These steps can be broken down into smaller stages and eventually into code
Program Planning

- Below is a series of steps you should go through either in your head or on paper to help you code your program.

1) Identify the Main Steps (Parts)...later make these into separate functions

2) Identify the Variables (Information) needed for each part or the entire program

3) Begin writing pseudo-code for each of steps

4) Revision (may take several revisions)

5) Convert pseudo-code into Python

6) Debug your code
Programming Examples

• Write a program that will:
  
  1. Ask the user for two numbers
  2. Add the two numbers and output the result

• Write a program that will:
  
  1. Ask the user for a dollar amount (for now assume <$1.00)
  2. Calculate the optimal number of quarters, dimes, nickels, and pennies to make the change amount.
     
     ➢ Hint: Convert the dollar amount into number of pennies first and think of everything in terms of pennies...e.g. A quarter is 25 pennies.
     
     ➢ Think how modulus will work here
  
  3. Output the results

** Take a moment to think how to solve these, we will go over them in a minute. **
```python
def main():
    # the raw_input reads in user input as a string, so it must be
    # converted to an int value using the conversion function
    # it is then stored in a new variable x
    x = int(raw_input("Enter the first number : "))

    # same as above except that we read in a second number storing
    # the value into a new variable y
    y = int(raw_input("Enter the second number : "))

    # using the addition operator add the two values stored, and
    # store the result into a new variable z
    z = x + y

    # in order to combine the string with the value in z,
    # z must be converted to a string using the conversion function
    print "The sum of two numbers is " + str(z)

    # nice little pause to end
    raw_input("\nPress Enter to Exit")

main() # tell your program to run the function we just defined```
Change Program

```python
def main():
    change = float(input("Enter the dollar amount: "))
    pennies = int(change * 100)  # need to cast to int as you can't have part of a penny
    quarters = pennies / 25  # find the number of full quarters (again why we needed
                          # as pennies as int)
    pennies = pennies % 25  # set pennies to be the left over pennies, that do not
                          # make up part of the quarters
    # repeat the above process for each coin
    # ...
    dimes = pennies / 10
    pennies = pennies % 10
    nickels = pennies / 5
    pennies = pennies % 5

    # We do not need to divide by one since it will just be itself, so we are done with
    # the calculation steps

    print "$" + str(change) + " is "
    print \tquarters, "quarters",
    print \tdimes, "dimes",
    print \tnickels, "nickels",
    print \tpennies, "pennies"

main()  # Execute the defined function
```
Assignment #2

• Due next week before lab
• You need to turn in one file
  - Lab2.py
    • Contains the proper comment header
    • Reads input from the user
    • Does the proper conversion as outlined in the pdf
    • Outputs the result as shown in the example output
      (must cut-off decimal points...using either the conversion functions or the round function see last slide)

• Again make sure to verify that everything was submitted
**FOR YOU:** Write a Tipper program where the user enters a restaurant bill total. The program should then display two amounts: a 15-percent tip and a 20-percent tip:

Example Execution (Bolded and Italicized is user input)

```python
>>> tip.py
Enter the restaurant bill total: 32.78
15-Percent: 4.917, 20-Percent: 6.556
```
Upcoming

- Branching, WhileLoops, and Program Planning
- Finish Lab Assignment #2 and be sure to complete it before next week's lab
- Read Chapter 3 in the textbook
Rounding

Functions to truncate numeric data:

• `math.ceil(<number>)`
• `math.floor(<number>)`
• `round(<number> [, <places>])`
• `int()`

```python
import math

def main():
    x = 3.2
    y = 4.6
    print "Original:\tx=",x,"\ty=",y
    print "ceil():\t x=",math.ceil(x),"\ty=",math.ceil(y)
    print "floor():\t x=",math.floor(x),"\ty=",math.floor(y)
    print "round():\t x=",round(x),"\ty=",round(y)
    print "int():\t x=",int(x),"\ty=",int(y)

main()

>>>

Original:  x= 3.2  y= 4.6
ceil():   x= 4.0  y= 5.0
floor():  x= 3.0  y= 4.0
round():  x= 3.0  y= 5.0
int():    x= 3   y= 4
>>>
```