As you arrive:

1. Start up your computer and plug it in
2. ***Log into Angel*** and go to CSSE 120
3. Do the ***Attendance Widget*** – the PIN is on the board
4. Go to the course ***Schedule Page***
   - From your *bookmark*, or from the *Lessons* tab in Angel
5. Open the ***Slides*** for today if you wish

Pair Programming

Review and Practice

Lots of time to work on Homeworks 3 and 4

Outline

- **Review**
  - Starting a program in *main*. How to *define* a function, *call* a function.
  - Organizing a program into *functions*
  - The *input-compute-output* pattern
  - Functions with *parameters* that *return* values
    - Parameters and variables defined in functions are *local* to that function
    - *Capture* the returned value in a variable, or use it directly
  - Definite *loops*, using a *range* statement
  - The *Accumulator Loop* pattern

- ***Pair Programming***
- **Most of today is *Practice, Practice, Practice* on:**
- **Homework 3**
  - *factorial*
  - *barChart*
  - *Conversion*
- **Homework 4**
  - *bullsEye*
  - *countIntegers*

Check out project for today

- Go to SVN Repository view, at bottom of the workbench
  - If it is not there,
    Window→Show View→Other→SVN → SVN Repositories
- Browse SVN Repository view for
    04-ReviewAndPractice project
- Right-click it, and choose Checkout
  - Accept options as presented
- Expand the 04-ReviewAndPractice project that appears in Package Explorer (on the left-hand-side)
  - Browse the modules.
  - We will start with ***1-mainStructure.py*** (next slide)

Review – in Eclipse

04-ReviewAndPractice

- **1-mainStructure:**
  - Starting a program in *main*.
  - How to *define* a function, *call* a function.
- **2-inputComputeOutput:**
  - *How to organize a program into functions.*
  - *How to input and convert the input to a float ('int' would convert to an int).*
  - *How to print (i.e., produce output).*
  - *How to use a FOR loop.*
    - *This FOR loop is a definite loop using RANGE.*
    - *It can be thought of as an example of the Accumulator Loop pattern.*
  - *How to use local variables*
    - *for the constant (3.9) and howManyToPrint (10), along with the input variable (x) and the loop variable (k).*

More review – in Eclipse

`04-ReviewAndPractice`

- **3-functionsWithParameters:**
  - Functions with *parameters* that *return* values
    - How to *define* a function with *parameters*
    - How to *call* that function, supplying *actual arguments* whose values the parameters are assigned to
    - Parameters and variables defined in functions are *local* to that function
    - *Capture* the returned value in a variable, or use it directly
  - ***It is all a question of SPECIFICATION – communicating with the user of your code.***
  - *Appending* to a list
- **4-accumulatorLoopPattern:**
  - What the Accumulator Loop pattern is, example

Pair Programming

- Working in pairs on a single computer
  - One person, the *driver*, uses the keyboard
  - The other person, the *navigator*, watches, thinks, and takes notes
- For hard (or new) problems, this technique
  - Reduces number of errors
  - Saves time in the long run
- Works best when partners have similar skill level
- If not, then student with most experience should navigate, while the other student drives.

Food tasting

- Suppose you are at food tasting show and are tasting 5 different dishes
- Sampling the dishes in different orders may affect how good they taste
- If you want to try out every possible ordering, how many different orders would there be?

  That number is the factorial of 5

  $n! = n\,(n-1)\,(n-2)\,\ldots\,(1)$
- What type of problem is this?

Accumulating results: factorial

- ***Work with a partner*** (pick a driver and navigator)
  - Follow the advice in

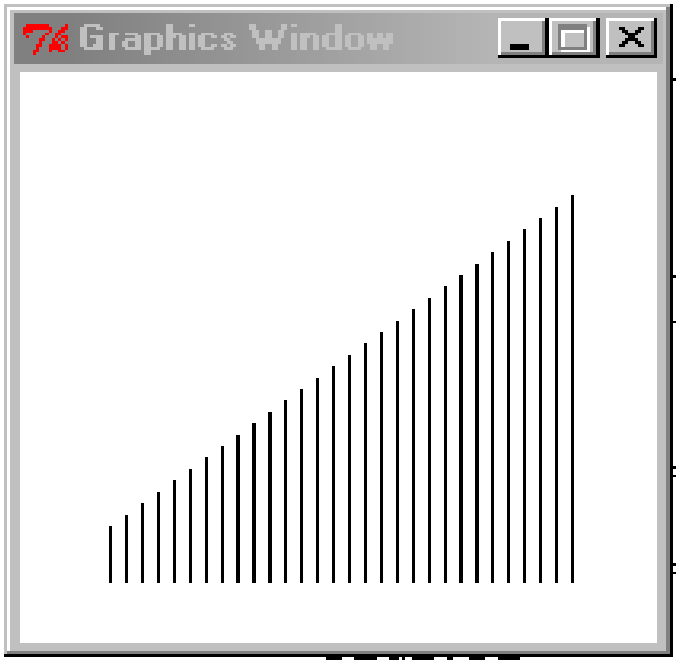
    www.rose-hulman.edu/class/csse/csse120/201110/Resources/pairProgramming.html

    for how to work together, whose machine you should use, how to turn in your collaborative work, and how you will both end up with a graded version of that work.
- Do the TODO's in **factorial.py**

  in **03-assignmentsAndLoops**
  - Those TODO's will direct you to write and test a function that computes n!, for any given n.

Graphics Exercise with loops

- Trade roles with partner—new driver, new navigator
- Do the TODO's in **barChart.py** in **03-assignmentsAndLoops**
  - They will direct you to draw a figure like this where the lengths of the lines increase by a constant amount
  - The ***graphicsExample*** module that we supplied and your previous graphics program may be useful.
  - You may want to use variables to hold current x-coordinate and current line length, and change the values of those variables each time through the loop



Rest of today

- Do your pair programming from Homework 3
  - ***factorial***
  - ***barChart***
- Do your individual work from Homework 3
  - ***conversion***
- And Homework 4
  - ***bullsEye***
  - ***sumAndCount***
- Don't forget the reading quizzes in both homework
- Both homework are due Monday