Outline

• We emphasize programming language concepts for wide current and future use!
  – 201-> 310,210-> 333,311-> 409,402->compilers 417
  CREATE YOUR OWN PROGRAMMING LANG.!

• Dataflow programming eg. Spreadsheet (Excel)
  – Also LabView (for labs like nano-college's), MathLab (for math/engineering), & for electronic
    music synthesis: Moog-->Max/MSP or (or free pd)
  – Test your Loan HW programs yourself!

• Debug tracing in IDLE—SEE the SEQUENCE
Dataflow programming

- Each node has a FORMULA, which could be a constant.
- A formula might use VALUE(s) from other nodes, which it refer to by node name (like A, B, C, D)
First Special Spreadsheet Features

- The nodes are in a rectangular grid.
- The node names are <Letter><Number>, where the Letter is the column and the Number is the row. So, A2 names first column, 2\textsuperscript{nd} row node.
- The FORMULA (except when it's a constant) does not appear in the node. When you select a node, its FORMULA appears in the edit box above the spreadsheet.
Second, the UNUSUAL feature:

- Copy/Paste is a software feature that is almost universal in software today..everyone knows it. The pasted data is the same as the original...

- BUT, when a SPREADSHEET node is copied and pasted to a different node, the FORMULA is modified in the copy! Here's how:

- The DISPLACEMENT VECTOR from the original node to the different node is “ADDED” to every (plain) node name in the FORMULA.
Other tips for using application SW without spending time/money on courses: (notes of what I said in the lecture and afterthoughts)

- Ask a friend who's used it.

- Look at and explore the buttons and menus for symbols or names related to what you want to do. Example: to show a number as a money value, use Excel's money button (shows a $).

- Click the ? (help) button/search and read as necessary. Example: search for “money.” Read near the beginning of the help eventually.

- Ask: What computer science ideas can help? Example: Excel formula expressions have a syntax like, but not exactly the same, as Python's. What are the the operation symbols? In Excel, = at the beginning of a formula says “What follows is a formula, not a constant.” +, -, *, / are arithmetic operators like in Python. Numeric constants also have syntax similar to Python's. What's DIFFERENT is how a Node (commonly called a CELL) name behaves. Now, Excel has library FUNCTIONS, like Python. Since you know about Python functions, ask “shouldn't Excel support you programming your own functions? If so how?” The answer can be found in the help!
Individual In-Class Exercise:

- Start Microsoft Office -> Excel
- Label a month column. Put 0 under the label.
- Put in the formula to make the cell below the 0 compute the previous month plus (+) 1.
- Select that formula cell and activate “Copy”
- Select a lot of cells below that formula cell, then do “Paste” on them.
- See that successive month numbers are computed! (1, 2, 3, .... etc.)
Other columns..

- Label a “Principal” column. Put the loan amount, say 20000.00, as the initial principal next to the 0 for month 0.

- Label a “Monthly Rate” column. Put the initial monthly rate in its row for month 0. Write the formula to make the next month's rate be the previous month's. Copy and paste to get the rate in each month's row.
More columns..

- Set up a column for “payment” and arrange for the same payment to be used each month.
- Code the formula that computes the interest for month 1 to be (the principal for month 0) * (the rate for month 0). Copy and paste this formula into the rows below.
- Code the formula that computes the principal for month 1 in terms of month 1's payment, month 0's interest, and month 0's principal.
.....Finish

- Copy and paste the principal formula from the cell where you typed it in to many of the cells below it.
- What that is done, the correct loan table will appear!
- If there are not enough rows filled in with the correct formulas to see how long the loan lasts, just copy and paste to fill in more rows.
- NOW: Change the payment in the top row's payment cell and Excel will recompute the whole thing!
from sys import stdout

def plot():
    for x in range(0,5):
        for y in range(0,4):
            A = x + y
            stdout.write(str(x)+','+str(y)+','+str(A)+'  ')
            print

def main():
    plot()

str( <any value of any type>  ) returns a string that expresses the given value in a way appropriate for the type.
  + on strings concatenates [POLYMORPHIC operator..]
(some people used these in their homework..Great!)
Why a debugger is cool.

- The key to success in computer science is being able to imagine from a body of code (perhaps with the help of pencil and paper) what the computer does in EACH small STEP in SEQUENCE.
- The debugger SHOWS you each step and what happens when it is taken.
- It shows beginners what to imagine.
- It helps experts analyze programs and find bugs.
How to use IDLE's debugger.

- First, prepare and run your program from the edit window as we have instructed.
- From the IDLE shell window, select Debugger from the Debug menu. In the Debugger Control window, make sure Stack, Locals and Source are selected.
- Re-run program, typically by calling main( ) from the shell window. Notice it DOESN'T fully run.
Control Window: What to look at

- The line outlined in blue is the Python code line that WILL be executed when you click the Step button.

- The bottom sub-window shows the VARIABLES and their CURRENT values (BEFORE the blue Python code line is executed.)

- Think of what the blue line will do. Then, click Step. You will see a variable change value, or get defined, if the blue line is an assignment. If the blue line prints, you will see its output in the shell window (as usual).
WORD TO THE WISE..

There will be a question on the first midterm that probes how experienced you are with the IDLE debugger. Using it is a course requirement, not just an option!
Other tips.

- If the blue line calls a function, **Step** will show the function body execute step by step.
- If the function is in a library or a different file, a new edit window will pop up. In that case, click **Out** to get back to your own program.
- Avoid stepping “into” internal functions by clicking **Over** (to jump “over” function calls) instead of **Step** for lines like

  ```python
  stdout.write(str(x)+','+str(y)+','+str(A)+' ')
  ```
More tips..

- The control window shows the function calls that LED TO executing the **outlined line** but which HAVE NOT YET returned.

- On other systems, the **outlined lines** may be outlined in **a different color** from blue.

- On other systems, the **line to execute next** will be outlined in the edit window (where you see the whole file containing the function currently executing) in addition to the control window.