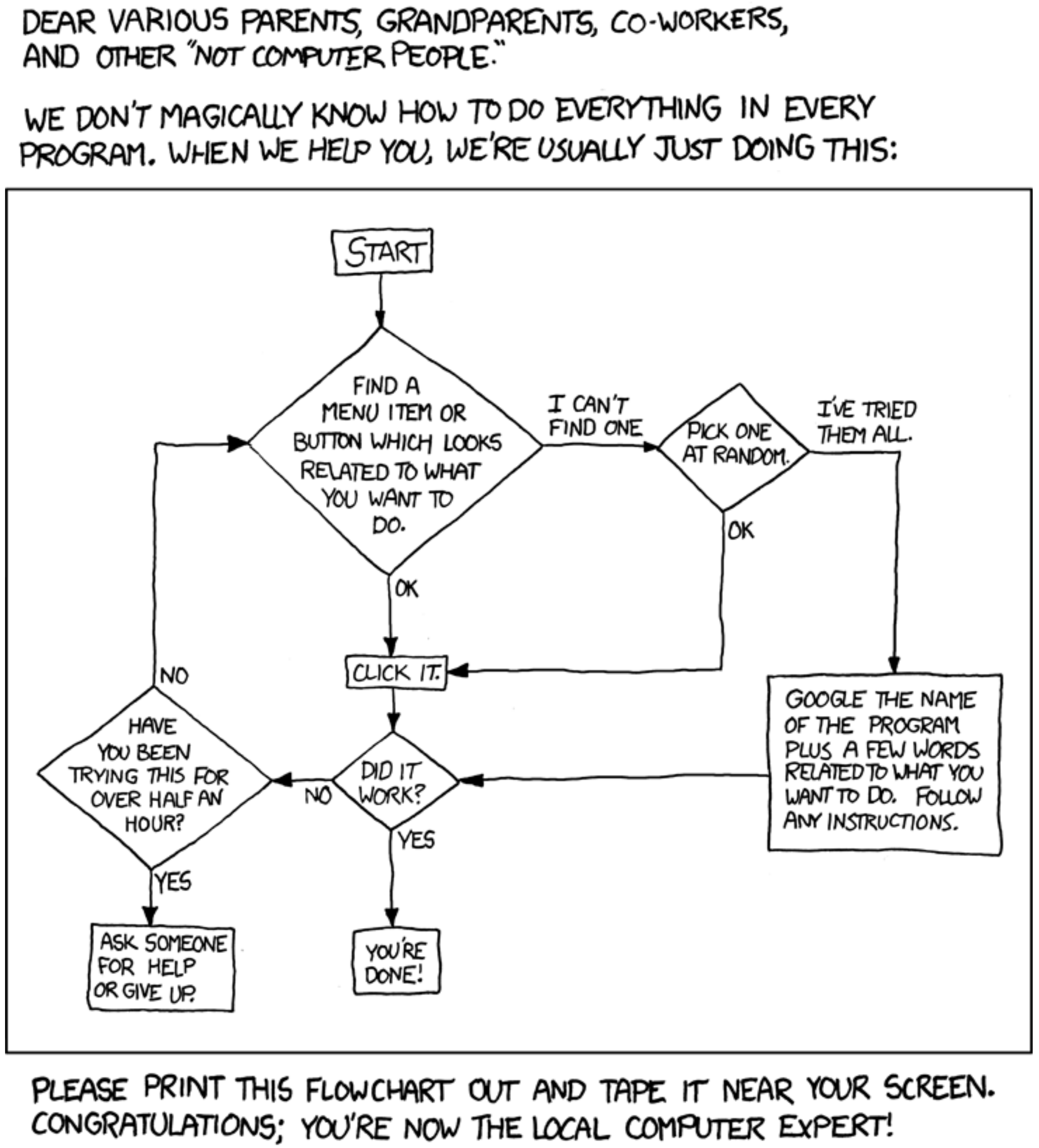
Debugging Strategies


DEAR VARIOUS PARENTS, GRANDPARENTS, CO-WORKERS, AND OTHER "NOT COMPUTER PEOPLE."
WE DON'T MAGICALLY KNOW HOW TO DO EVERYTHING IN EVERY PROGRAM. WHEN WE HELP YOU, WE'RE USUALLY JUST DOING THIS:
START
FIND A MENU ITEM OR BUTTON WHICH LOOKS RELATED TO WHAT YOU WANT TO DO.
I CAN'T FIND ONE
PICK ONE AT RANDOM.
I'VE TRIED THEM ALL.
OK
OK
CLICK IT.
GOOGLE THE NAME OF THE PROGRAM PLUS A FEW WORDS RELATED TO WHAT YOU WANT TO DO. FOLLOW ANY INSTRUCTIONS.
DID IT WORK?
NO
YES
HAVE YOU BEEN TRYING THIS FOR OVER HALF AN HOUR?
NO
YES
ASK SOMEONE FOR HELP OR GIVE UP.
YOU'RE DONE!
PLEASE PRINT THIS FLOWCHART OUT AND TAPE IT NEAR YOUR SCREEN. CONGRATULATIONS; YOU'RE NOW THE LOCAL COMPUTER EXPERT!

Problem-solving approach/aka how to write a script

1. Name and write down the basic problem you are trying to solve. Describe your input file format and desired output file format.
2. Write down the possible variable types, functions, and flow direction approaches that might apply to the goals of the script.
3. Write out a conceptual plan for the script, identify the major steps.
4. Translate the steps into code using the tools identified in step 2.
5. Run and debug your script.
6. Step back, and evaluate the approach, identify places for improvement, streamlining, etc.

- filestoXYYY.py
- sequence reader script
- latlon.py

General debugging strategies

- types of bugs?
  - validation
    - hand-checking subsets
    - build in redundancies
      - all files processed?
      - number of lines?
      - don’t make mistakes

General debugging strategies, contd.

- get modular
- print statements along the way, one at a time
  - comment out at end
- play in the sandbox
- work on the right file
- save it
- check line endings
- check your data
- comment out blocks of code
- end of loop comments

TABLE 13.1 Common errors running Python programs and some potential solutions

Example of reported error	Probable causes and solutions
`-bash: myscript.py: command not found`	If this error begins with `-bash` or with the name of your shell, then the program you are trying to run (`myscript.py` in this example) is either not in a folder listed in your `PATH`, or else its permissions are not set to executable. See Chapter 6 for setting the `PATH` variable, and try `chmod u+x myscript.py` at the command line to designate the file as executable.
`/Users/lucy/scripts/myprogram.py: line 3: import: command not found`	If the `command not found` error is reported from within your Python program, rather than from `bash` (you can tell based on the text that precedes it), then there may be a problem with your shebang line. Specifically, it may not be sending the contents of your script to the Python program or other suitable interpreter. Double-check the `#!` line in the file, or copy one out of a working program. This type of error can also be generated if you misspell a built-in Python function within your program. Carefully check lines that call such functions for typos.
`bad interpreter not a directory`	The shebang line has a slash after `/usr/bin/env/`, as if `env` were a directory. Remove the slash so the interpreter understands that `env` is a program.
`usr/bin/env: bad interpreter: No such file or directory`	The part of the statement in your shebang line after `env` wasn't found. Copy the contents of the shebang line after the `#!` characters and paste into a terminal window to see if this launches Python.
`permission denied`	Permissions not executable. Fix with `chmod u+x myscript.py`
`name 'x' is not defined`	There are several possibilities here: • You misspelled a variable name in the program. • You are trying to modify a variable that was not originally defined. For example, you may be adding to a list or string that was not first defined as empty. Fix by initializing the variable, e.g., `MyList=[]` or `MyString=""`. • You are trying to use a function that needs to be imported from a module first. • You are using a function without the required module name in dot notation, for example `randint(5)` instead of `random.randint(5)`. Either add the module name to the function name, or else rephrase your import using from (as in `from random import *`) to access all the mo[illegible] functions directly.

TABLE 13.1 (continued)

Example of reported error	Probable causes and solutions
`Indentation error:`	Take a guess! If the text has been pasted from a Web source, use Show Invisibles in your text editor to make sure the indentations are all just tabs or all just spaces. Also remember that invisible characters are sometimes introduced when copying and pasting.
`Attribute error`	Misspelling of a built-in function. For example, you would get this error if you have a string `MyString`, and use `MyString.lowercase()` instead of `MyString.lower()`. To fix, double-check the function or variable name that comes after the dot in your dot notation.
`type error 'xx' object is not callable`	Trying to retrieve values from a list using parentheses instead of square brackets, causing the list to be interpreted as a function name.
`traceback...zero division error`	Division by zero. This often happens when a function returns an unexpected zero, or when there is an unexpected zero in the input data. Check user and variable input to make sure that strings exist and are not blank. Test to make sure that a number is nonzero before using it as a denominator.
`Non-ASCII character '\xe2' in file`	One possibility is that the file contains one or more "curly quotes." Search for these and replace any found with "straight quotes." Another possibility is that some other symbol such as a bullet or degree sign has been used without specifying Unicode UTF-8. Add `# coding: utf-8` below the `#!` line.
`invalid syntax`	This could be many things, either in the line indicated or in preceding lines: • Missing colon after `if`, `else`, or `for` statement • Missing close parenthesis or close bracket • Using = instead of == in a logical test. • Mixing spaces and tabs when indenting

```
import sys
Debug = True

# (insert program statements here)

# wherever you want to give feedback, insert these lines
if Debug:
    print MyList
    # or you can use
    sys.stderr.write(MyList)
```

(A) Traditional

```
for Line in File:
  if Line[0]==">":
    Name=Line.strip()[1:]
  # lines with > are Names
  else:
    # check for a pre-existing key
    if Name in Dict.keys():
      Dict[Name] +=  Line.strip()
    # not a key so define
    else:
      Dict[Name] = Line.strip()
```

(B) Faster

```
for Line in File:
  if Line[0]==">":
    Name=Line.strip()[1:]
  else:
    try:
      # try to append with +=
      # assumes Name is a key
      Dict[Name] +=  Line.strip()
      # oops, not a key so define
    except KeyError:
      Dict[Name] = Line.strip()
```