Craig Muth

12/9/2022

IT FDN 110 A Au 22: Foundations Of Programming: Python

Lessons Learned

Pseudocode vs flowcharts:

Years ago, I was taught flowcharting as a way to plan things out before starting work – but have more recently gotten into the bad habit of staring to code and trying to develop the logic flow on the fly. That does not work very well. Because the final logic isn’t complete until the code is complete or nearly complete – when a flaw in the logic can lead to a complete rewrite.

Without the correct application for creating a flowchart (PowerPoint is not it) it can be difficult to make changes if the requirements are still in development when the work is expected to start. Many of the people I work with can’t read flowcharts, they certainly can’t draw them, and when they attempt to make mark ups on existing ones – the result is often incomprehensible.

However, pseudocode can be written with any word processor or text editor to describe what’s supposed to happen in more of an outline format - which is easier for non-technical people to understand. Even with code statements mixed in, proper context around those statements can make it possible to understand the intent without having to understand the exact statements. There is certainly no need to learn and remember the meaning behind the different shapes used in a flowchart, and in the text format it seems easier to get across the idea that a yes/no question cannot have 6 different versions of “it depends” as an answer.

I have not used pseudocode for years, but as I recall it worked very well for the project it was used on. Three people were involved, each doing part of the design – handing off, integrating, and coordinating the pseudocode (sometimes over the phone, I said it had been years) seemed to have fewer issues than if we had tried via flowchart.

Plus, if you don’t delete the pseudocode, you have a running commentary of what each subsection of code is intended to do.

Benefits of GitHub:

By using GitHub, I have been able to access my code and working documents from multiple locations, using at least four different machines. Two of those machines have no ports for removable media, and the other two the ability to write to removable media has been disabled.

So, whether I’m at home, travelling, or have some spare time at work – I can access the files without having to carry a laptop everywhere I go. Using it to compare versions of the script to verify what changes have occurred, rather than attempting to remember or document every change as I go along, or manually comparing the code line by line to find the changes makes it much simpler to go back and document changes resulting from an “I just want to test this one thing” moment resulting from a random idea.

Code reuse:

Not something I learned in this class, but something I was glad to see being discussed. A surprising number of coders I’ve worked with seem to avoid use of subroutines, functions, or other forms of reusable code. Instead, they hard code full operations each time the need it a program – making it harder to diagnose errors and update code as necessary. Engineers (non-software) seem to be more likely to do this.

The worst case I’ve seen was a VBA macro in an Excel workbook that was 35 pages (8.5 x 11) printed out, most of which was repeated code. A single outdated variable was preventing it from running, but as it declared in at least nine places along with repeating the code that was using it – the user spent three years using a manual work around. After recoding to use subroutines it was reduced to about seven pages.

Separation of Concerns:

This was something new – and I really like the way it helps to organize the code into something easier to follow by grouping similar elements together within the program structure.

Referring again to the example under code reuse – had this kind of structuring been used, would not have taken days to decipher the code and locate all the instances of the single (but constantly redeclared) variable that was causing the script to fail.