Shivank Joshi

TP1 Project Proposal

Project Description:

    Name: 'Code Style-Checker and Compressor'

    As the name indicates, my program will have a user interface that will

    prompt the user to select the file they wish to compress or fix style of.

    The user will be able to see their edited code and then decide if they

    want to write it to a file or not.

Competitive Analysis:

    There are various programs available that will edit style as you write your

    code but these mostly take into account whitespace and other basic style guidelines.

    Of course, there are also many ways to compress a file. My project will have a unique

    criteria for style suggestions: it will take into account spacing but also things

    like magic numbers and placement of comments and ambiguous variable names (largely

    derived from 112 style guidelines).

Structural Format:

    One user interface file named product that will run on 112 graphics. This will call

    the files 'compress' and 'style' depending on which output the user wants. Compress

    and style have lots of functions within them and also call the 'helpers' file which

    stores some common functions, data and module imports.

Algorithmic Plan:

    To gather information from input code I am using a mixture of information from the AST

    module, redbaron module and string parsing.

    To edit the code, I will use my gathered information, further string parsing, and node

    transforming techniques for both ast/redbaron modules. The astor module is only used

    for one function that is useful for turning a tree back into a string of code.

    I am using an approach that gives me code closer and closer to the 'goal' code at the end

    of each series of simple steps. By making one sweeping change at a time through code,

    I will eventually be able to produce the final product code. The most difficult portion

    of my project will be determining duplicate code that can be shortened into helper

    functions in both the style and compress changes. I am considering using a language

    processor for this if it is not required for mvp.

Timeline Plan:

    I will set aside one day this weekend to get my user interface working. I plan to spend

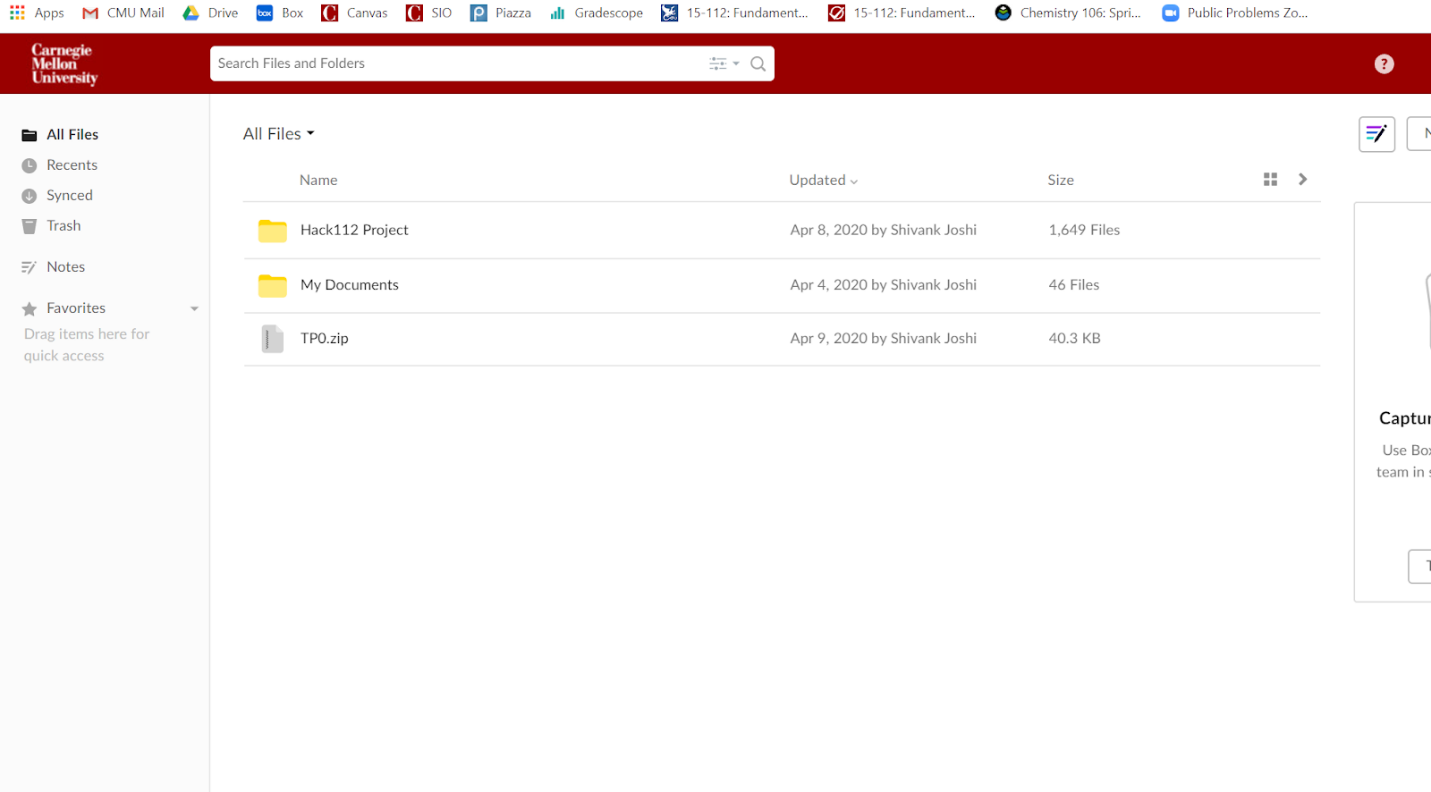
    a lot of time between TP1 and TP2 testing my code with lots of different samples and

    find gaps in my code to make it as robust as possible. Then I will work on more complex

    examples of editing: fixing repeated code and making assignments on one line in compress.

Version Control:

    I am putting different versions of my code into my CMU Box (mostly every TP)



Modules:

    ast

        redbaron, astor (essentially ast helpers)

    pretty print (not really using it, just for debugging)

    PyDictionary (for smart comment suggestions)

TP2 Updates:

* New name: ‘Code Modifier’
* The product file now calls a separate GUI file (product\_helper.py) to run
* Very minimal characters would be saved by implementing same-line assignment of variables in the compression algorithm so I probably will not do it
* My main goal from this point forward will be implementing in-app modification of code based on the style suggestions and improving UX

TP3 Updates:

* Additional file product\_style\_editor.py added to store the editor’s information
* Some big fixes implemented for both compress and style functions
* Unfixable errors:
  + Each time redbaron parses sample code it does this differently, so there is no way for me to be sure that certain code will not break upon initially entry of a source file
  + Triple string quotes will not parse and be put back together into code correctly
* Style suggestions improved and different structure used for style suggestions data (two -for viewing and user editing)
* Loading bar added

Updated Version Control:

