Practical Regex: Find Python Errors

Eckel, TJHSST AI2, Spring 2022

Background & Explanation

Another application of regular expressions is finding patterns, not words, within a larger document; this is often used in coding to check for particular patterns *in other code*. As one straightforward example of this, you're going to write code to search .py files for some common Python mistakes.

This assignment is new this year; any feedback is welcome!

Assignment

Choose five of the common mistakes below, and write code to detect them. You don't have to use only regex for this, but regex should help considerably.

Your code will receive a single command line argument – the name of a Python file. For each of the five mistakes you've chosen, output what the mistake is you're checking for, and the line number of each line in the file that contains that error. **WE WILL FOLLOW CONVENTION AND NUMBER LINES STARTING WITH 1, NOT 0.**

I understand that coding is complicated, and almost all of the options below have exceptions where that pattern might not actually be a problem. You can assume you're not dealing with high-level insanity here; if it helps, imagine you're writing a code checker for Foundations-level work – work from students who know functions, variables, loops, lists, and strings, but don't know eg list comprehensions.

Since you'll need to report line numbers, it may be helpful to break the .py file into lines by splitting on \n and search each line separately. Depending on what you're looking for, though, searching the whole file might be easier. You might even want to do both and use the two different representations for different tasks.

Feel free to do additional manipulation of the lines, or multiple regex searches, or whatever you need to do to get this to work; there isn't a right answer for how this is supposed to happen. Make it work however it feels right to you. If you don't think a regex is useful at all for one of the mistakes you're looking for, though, I'll be surprised.

Possible mistakes you can choose to look for:

- A variable in python can contain any word character, [A-Za-z0-9_], but can't begin with a digit. Find variable names that begin with digits. (Even cooler if you can exclude results between " or ' marks, to ignore strings. That doesn't have to be done using regex, though it can be.)
- Any variable in python followed by any number of spaces (including zero) and then an opening parenthesis can be assumed to be a function call or a function definition. Find any function calls to undefined functions.
- Find any function calls to defined functions that have the wrong number of arguments.
- Find situations where = has been used instead of ==. (Again, assume Foundations-level work; no need to go down too many rabbit holes for surprising uses. Consider if, elif, while, and return statements, and perhaps assigning Boolean values to variables like `x = y == 3`, which will set x to True or False based on the value of y.)
- Find situations where == has been used instead of =.
- Find situations where a method is called on a variable before that variable has been defined. (ie, `a.append(3)` before any line that says `a = something`, or a function definition where a is passed as an argument.)
- Find a situation where a student who is used to coding in Java maybe wrote some Java code in Python by mistake. Be specific about the situation you're looking for. This should not be something as simple as looking for, like, the word "public" – it's totally reasonable to make a variable that is just named "public" in Python!

- Global variables are often bad coding practice unless they are constants that are referenced without modification. Search for any global variables that are modified inside a function.
- Either in combination with the above bullet or separately, search for any local variables that are given the same name as global variables – also not an error, but inadvisable.
- Find indentation errors (that is, an increase in indentation level not preceded by something that forms a code block, or a decrease in indentation level that doesn't match a prior indentation level.)
- Find missing colons.
- I'm sure there are tons of these I haven't thought of. Feel free to propose any that I haven't thought of – I may add them to this assignment in the future! They can be mistakes or bad coding practices.

I have no public test cases or sample runs for this assignment yet; if you want to help me make some, I'd be delighted to work with you. Either way, you will also need to make a separate .py file with mistakes in it that you can use to demonstrate your error finding program's effectiveness; see the spec below.

Specification

This is a little bit different; read carefully.

Submit a single **folder** to the link on the course website. It should contain two files:

- One, `error_search.py`, should behave as written above – search a Python file given in a command line argument for all of the mistakes you have chosen. For each one, print a line explaining the mistake you're looking for, and then any line numbers of the given file that contain that error.
- The second, `error_example.py`, should contain a piece of Python code that you have modified to contain at least **two instances of each mistake** you're searching for. In `error_example.py`, comment the lines that contain the mistakes, so I can open up the output from your `error_search.py` next to your `error_example.py` and compare them.
- Note, though, that I will also be developing tests of my own to use and your code will need to give accurate results on those as well!

This assignment is **complete** if:

- The "Name" field on the Dropbox submission form contains your **class period**, then your **last name**, then your **first name**, in that order.
- Your code operates exactly as specified above.