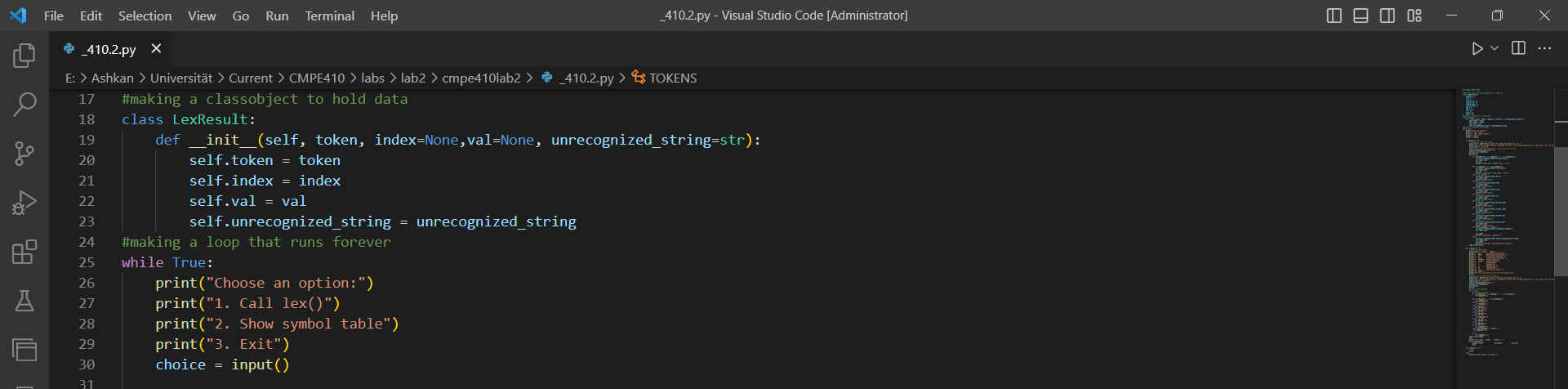
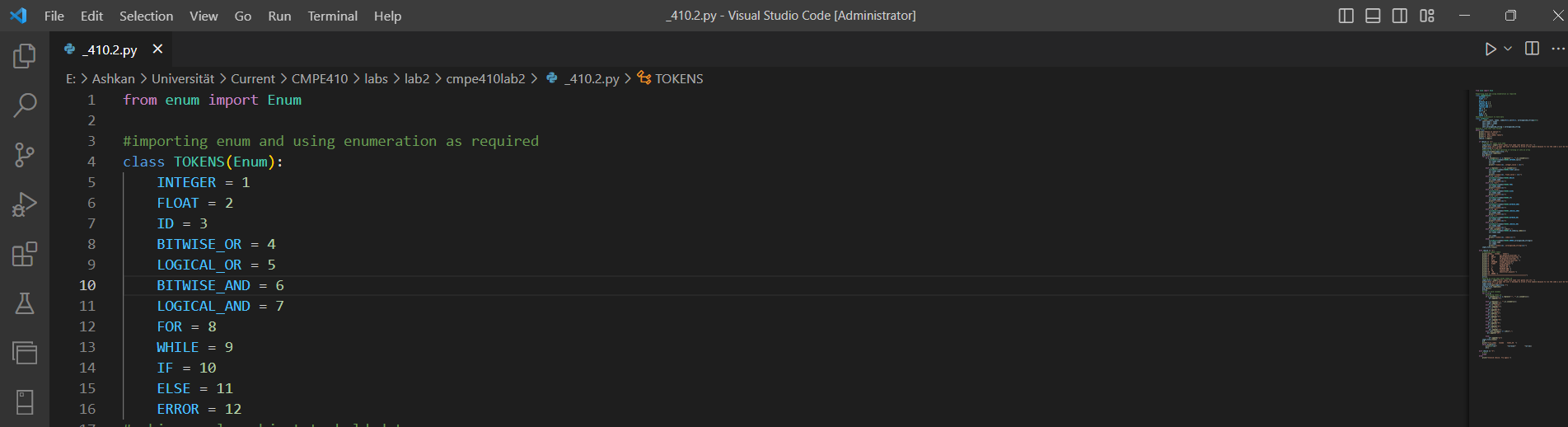
I wrote a lexical analyzer in python that recognizes integers, floating point numbers,  
identifiers, logic operators &, &&, | and ||, as well as reserved words “for”, “while”, “if” and  
“else”.   
This lexical analyzer function is reading the input from a file, and returning an object as a result  
each time it is called. The returned object should have attributes for 1 the token, 2 index into  
the symbol table, 3 value in case of an integer, 4 value in case of a floating point, and 5  
string of characters in case of an unrecognized lexeme.

Here I imported Enum to define a set of related constants that can be used in your code, for example to represent different states of a system or different options for a user interface. In this class, I set values for token attributes.



Here I made LexResult class, which has four attributes:

Token: Represents the type of token that was recognized during lexical analysis.

Index: Represents the index of the first character in the input string that was not recognized as part of the current token. This is an optional attribute, and if not provided, it will be set to none.

Val: Represents the value of the recognized token. This is also an optional attribute, and if not provided, it will be set to none.

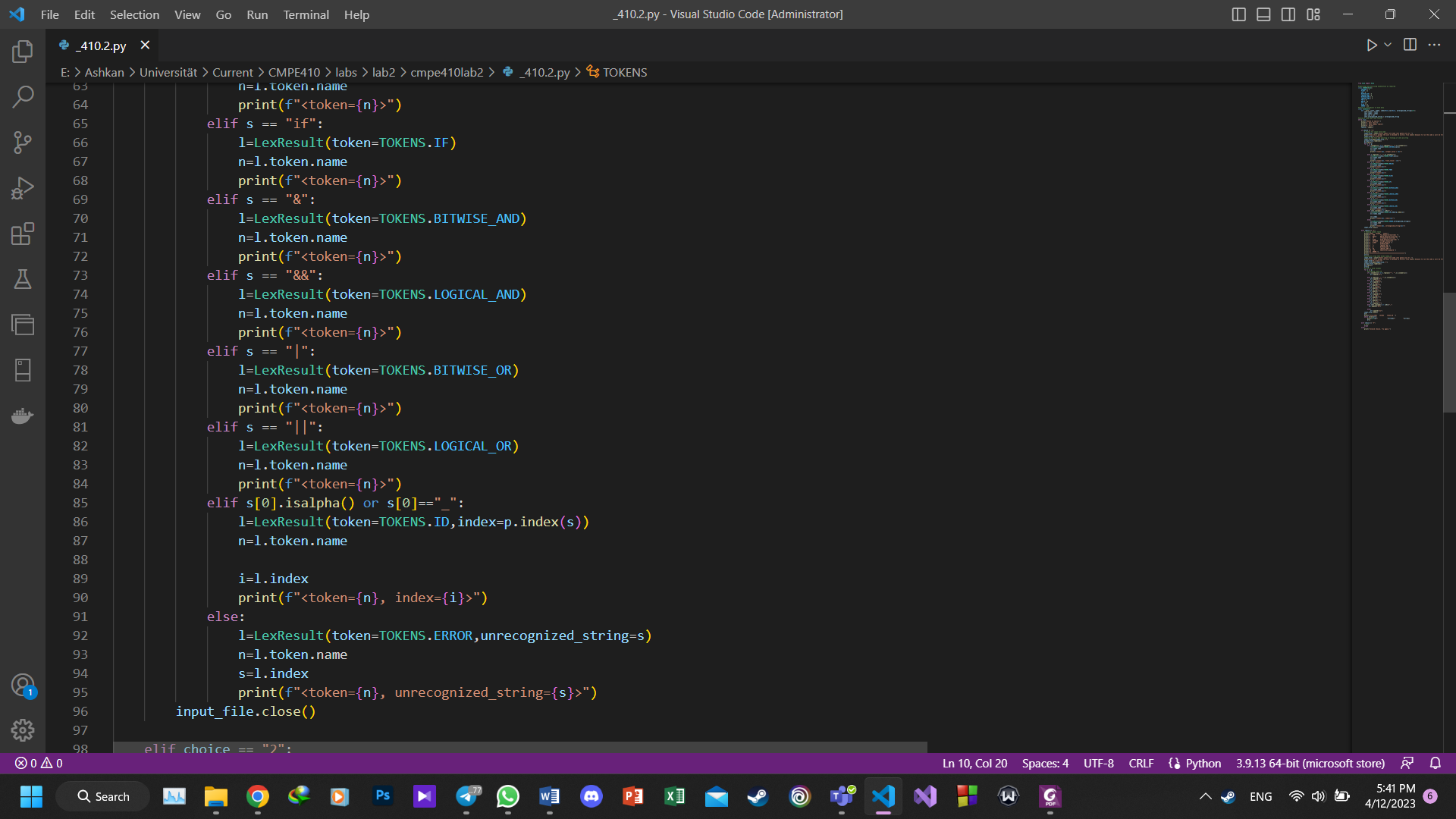
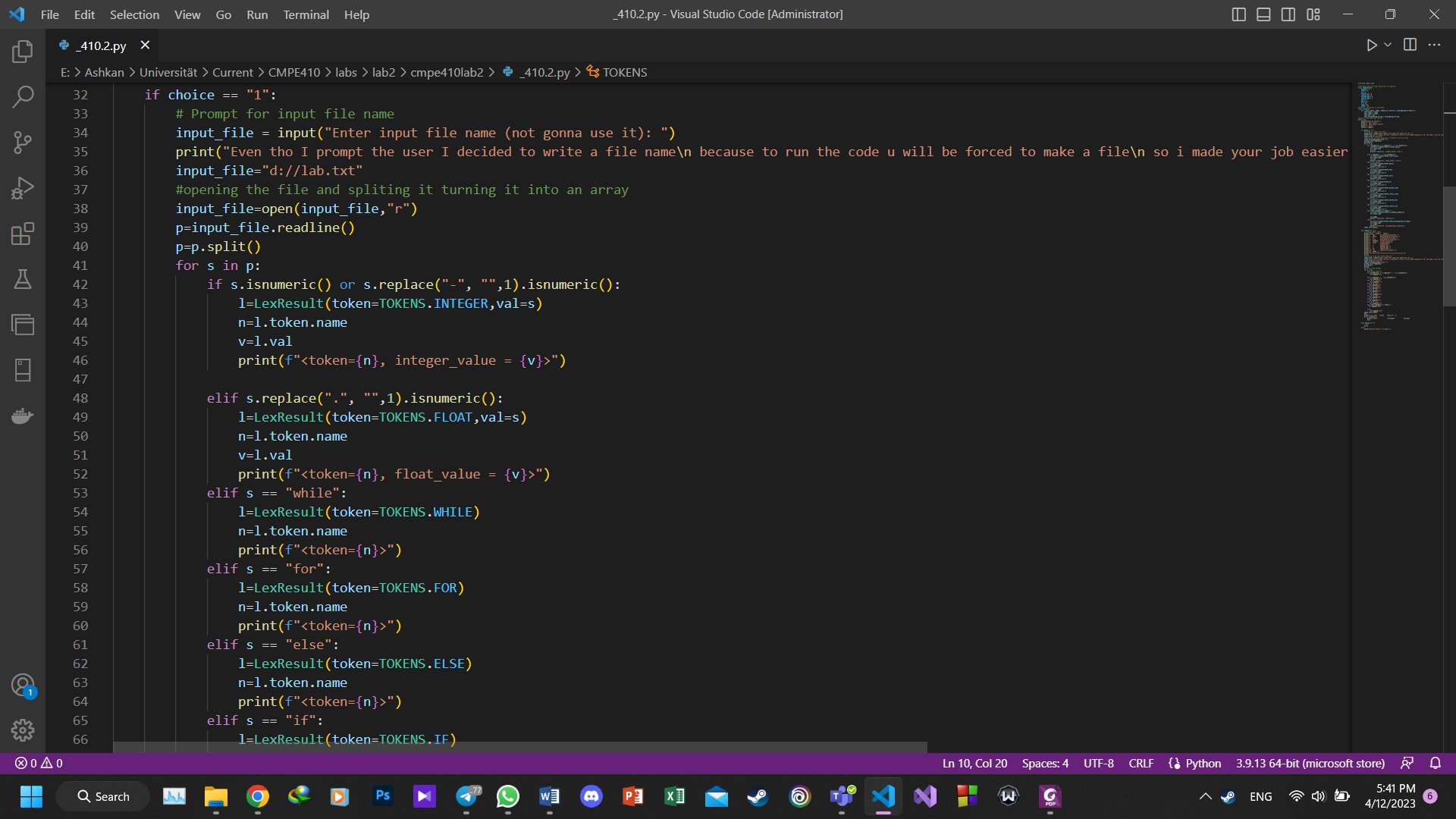
unrecognized string: Represents the portion of the input string that was not recognized as part of the current token. This is an optional attribute, and if not provided, it will be set to the empty string ("").

The while loop runs indefinitely until the user chooses the "Exit" option.

Inside the loop, three options are displayed using print() statements: "Call lex()", "Show symbol table", and "Exit".

The input() function is used to prompt the user to enter a choice, and the choice is stored in the choice variable.

Depending on the user's choice, different actions can be taken. The code for these actions is not shown in this snippet, but it would typically involve calling other functions or methods.



This code block corresponds to the "Call lex()" option in the menu. It reads input from a file, tokenizes the input, and outputs the resulting tokens.

The user is prompted to enter an input file name, but the code also provides a default file name in case the user does not enter one.

The file is opened and read line by line.

Each line is split into words using the split() method, which splits the line at whitespace characters and returns a list of words.

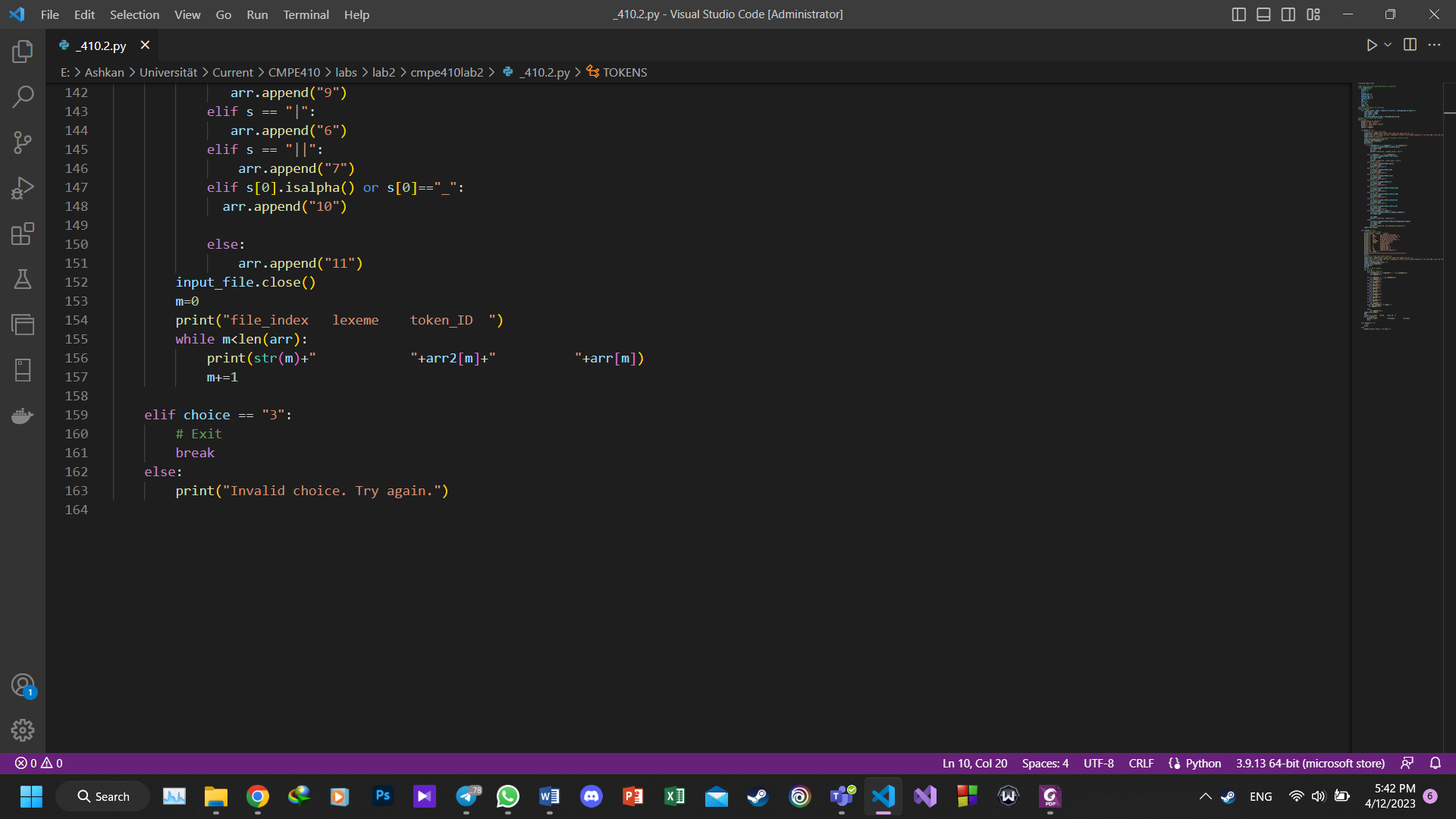
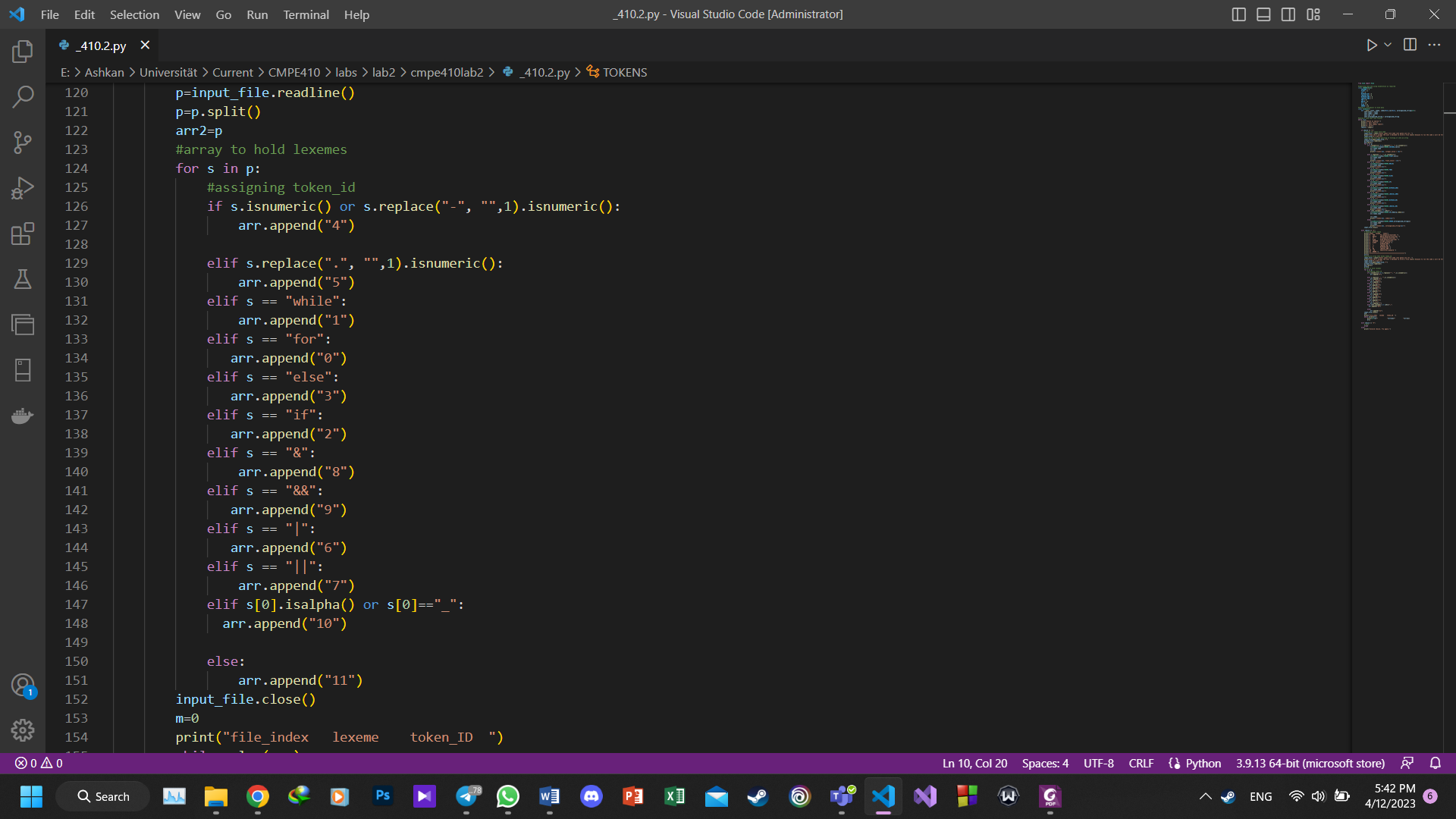
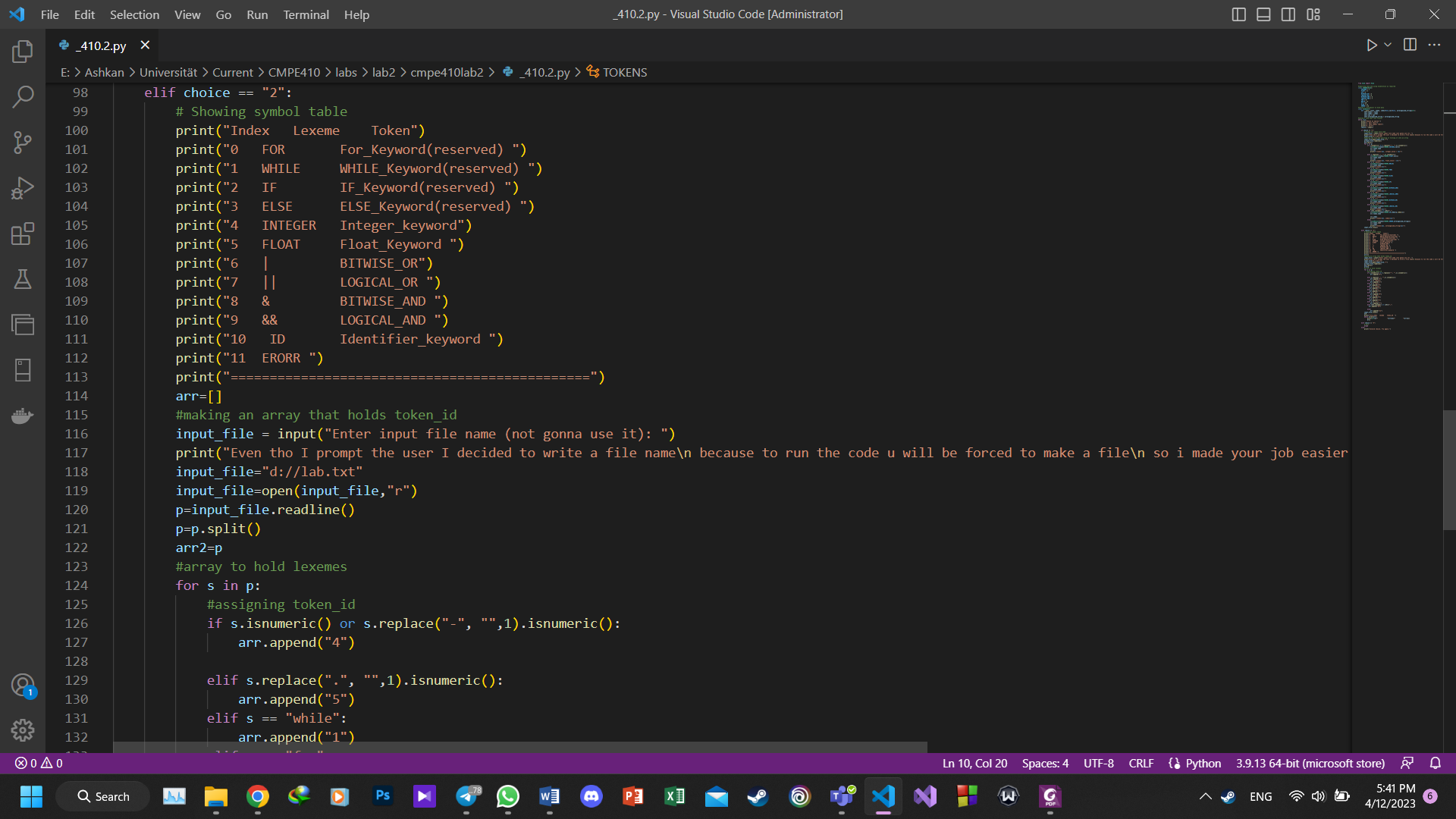
For each word in the list, the code checks if it matches any of the predefined tokens. If it does, a LexResult object is created with the appropriate token type and value.

If the word is an identifier (a sequence of letters or underscores), a LexResult object is created with the ID token type and the index of the identifier in the input file.

If the word is a number, a LexResult object is created with either the INTEGER or FLOAT token type, depending on whether the number contains a decimal point.

If the word is not recognized as any of the predefined tokens, a LexResult object is created with the ERROR token type and the unrecognized word as the unrecognized\_string attribute.

The resulting LexResult objects are printed to the console in a specific format using print() statements.



This code allows the user to choose between three options. If the user enters "2", the program will read from a file and output the index, lexeme, and token ID of each word in the file.

The program first prompts the user to enter the input file name, but then immediately overwrites that input with a hardcoded file name. The program then reads from the file and splits it into words, which are stored in a list called p.

The program then initializes an empty list called arr, which will hold the token IDs of each word. It then iterates over each word in p, using a series of if and elif statements to assign the appropriate token ID to each word. The token IDs are stored in arr.

Finally, the program iterates over the arr and p lists to print out the index, lexeme, and token ID of each word. The program prints out the hardcoded table of token IDs for the user's reference.