Software Maintenance and Evolution

CSE 426

Evolving the Editor

Submitted by:

Omar Saad Ahmed Hegazy

ID: 17P6008

Senior-2 CESS

Submitted to:

Dr. Ayman Bahaa

Table of Contents

List of Figures

1.Introduction

Figure 1: Anubis IDE

In this report I will explain how to evolve Anubis IDE; This is a simple IDE which will allow you to perform some basic functionalities, created by Anubis Graduation project team in faculty of engineering Ain shams university supervisor (Professor: Ayman Bahaa).

The report will contain all the evolution details, new design [class diagram and sequence diagram], testing sample code for both python and c# and finally screenshots from the evolved code.

The GitHub repository can be found here:

https://github.com/Omar-Saad72/Anubis-IDE-Evolution

2. Evolution Details

A new feature is added to Anubis IDE which is supporting the **C# format** as the editor can now automatically recognize which format to use based on the file extension.

To add this feature, we need to make the program more general than being specific to only programming language. So, we need to add something to detect the programming language we are using [Python or c#].

MyPath.py:

A new file is added to the program. It is used to decide whether the selected file is Python or C# file, it contains a global variable that represents the path of the opened file to work on. It is used by coloring and Anubis files.

Anubis.py:

Three changes occurred to this file:

1. Import MyPath.
2. Call [create()] function to initiate the global variable with the file extension.
3. Get path from the globally shared variable that comes from MyPath module by modifying [on_clicked()] function.

Coloring.py:

Eight changes occurred to this file:

1. Import MyPath.
2. Change the name of the file from [PythonColoring] to [Coloring] as it is responsible now for coloring of both python and C#.
3. Change the name of Class [pythonHighlighter] to [Highlighter] as it is more general now.
4. Add 2 array CsRules[] and pyRules[] instead of rules[], with each array modifications to match its language's keywords.
5. Add 2 arrays CsKeywords[] and PyKeywords[] instead of Keywords[], where each one having keywords of its programming language in [Highlighter] Class.
6. Modify highlightBlock function by checking whether the current path extension matched Python language (.py) or C# language (.cs).
7. Function matchMultiline() has been divided into two functions, one for comments in C# and the other for comments in Python.
8. The parameters of C# function have been modified, since 2 extra params are added which are ending and beginning params.

3.New Design

3.1 Static View [Class Diagram]

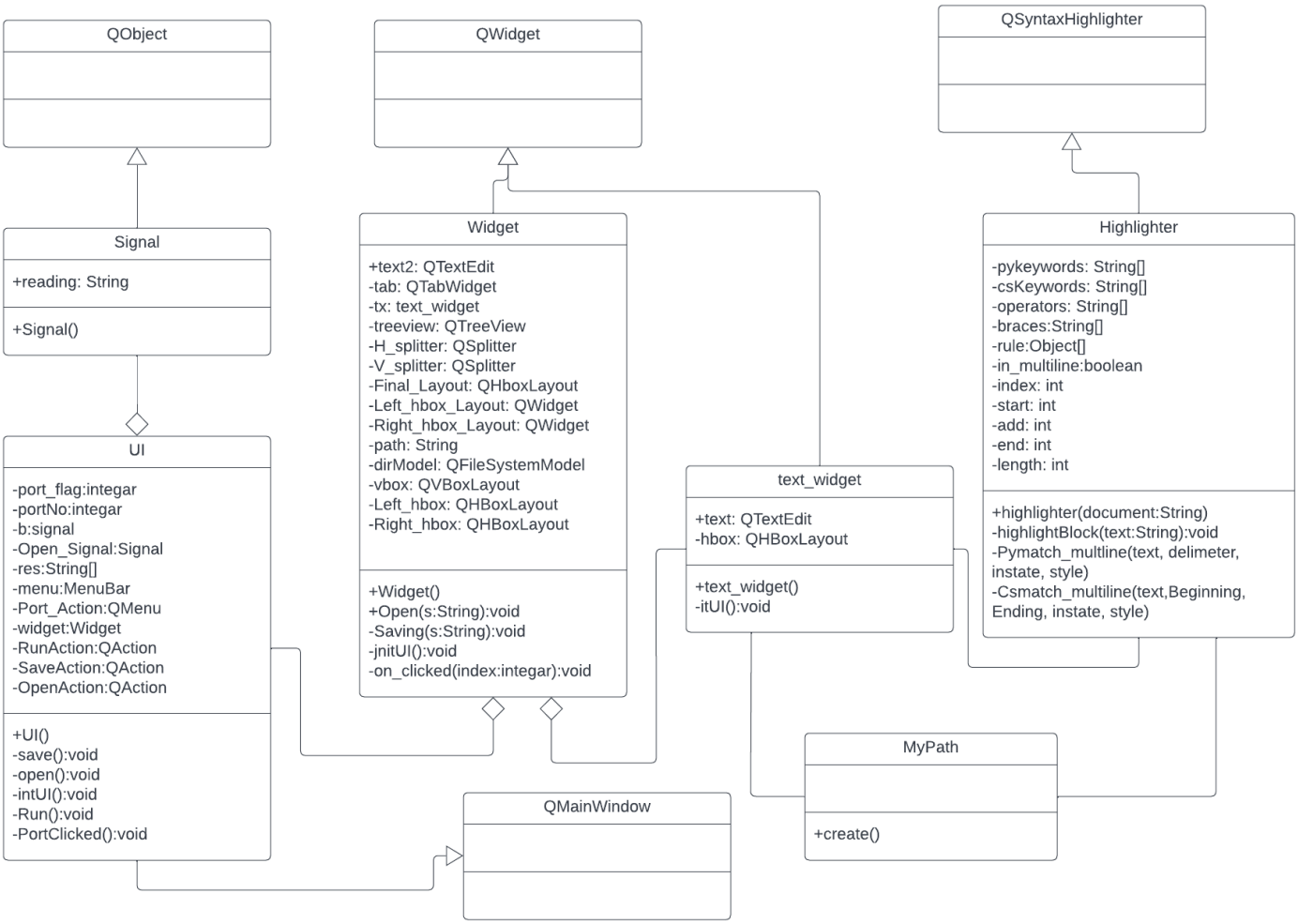

Figure 2: Class Diagram

A new class [Mypath] was added and class [Python Highlighter] name is changed to [Highlighter] to be more general to able to handle highlighting of both python and C# also its attributes and methods are modified.

3.2 Dynamic View [Sequence Diagram]

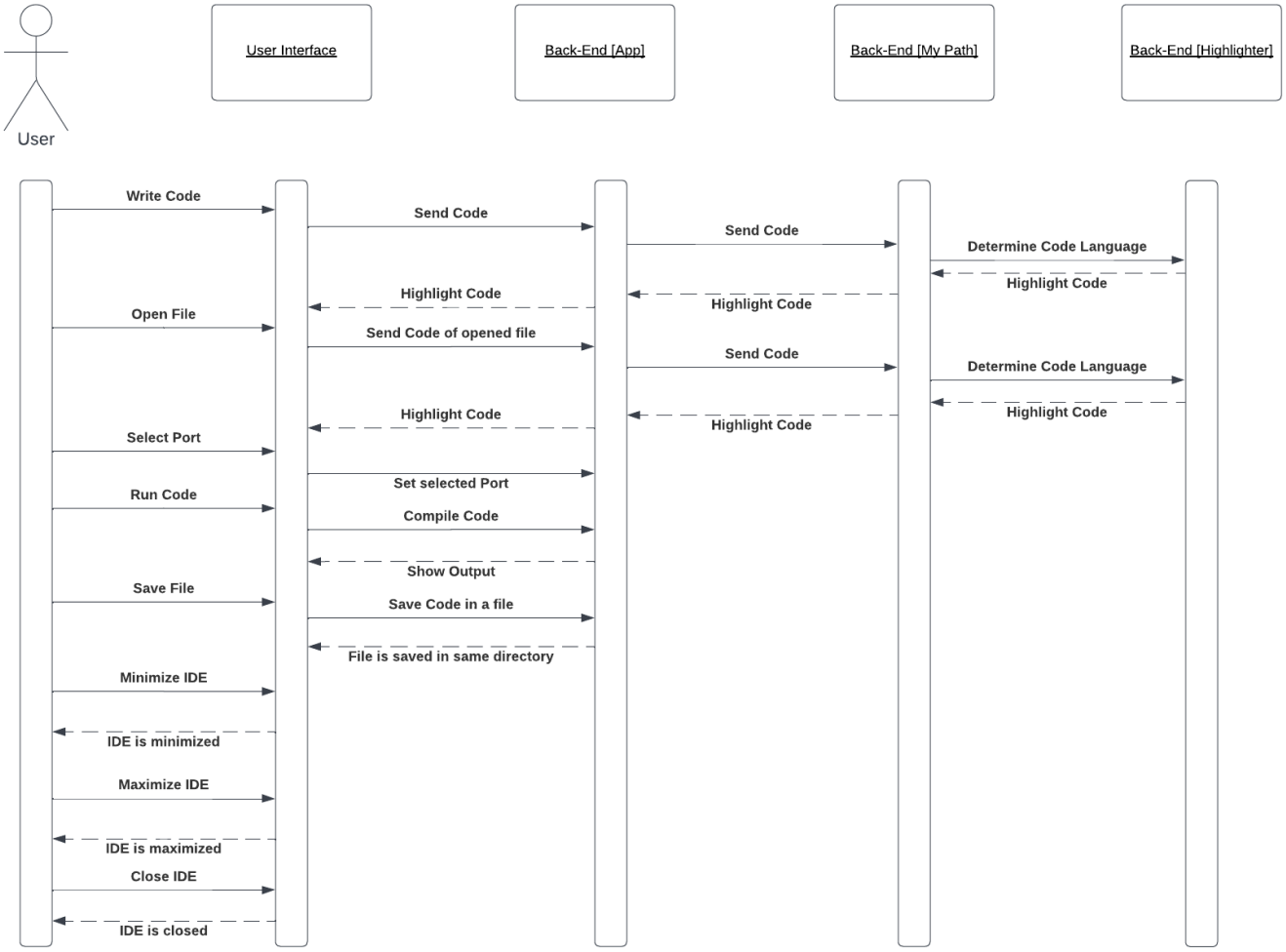

Figure 3: Sequence Diagram

A new component is added to my sequence diagram which is [My Path] which determine the language that the editor contain if it is python or C#.

4.Screenshots

4.1 Test Sample C# code

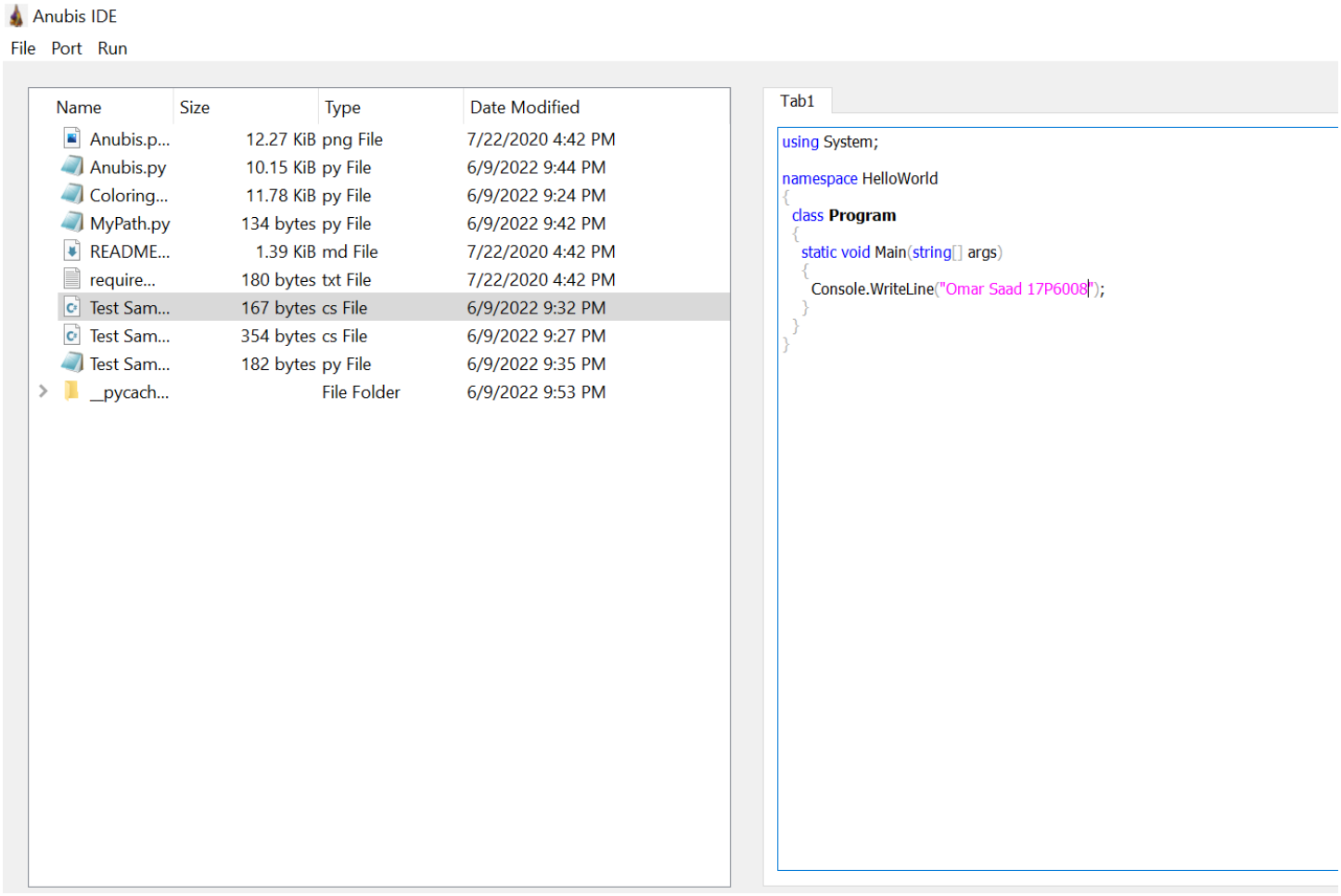

Figure 4: C# Test Code

Here we test a sample C# code:

1. We can see that syntax coloring is working correctly.
2. Class.
3. Function.
4. Console statement.

4.2 Test Sample Python code

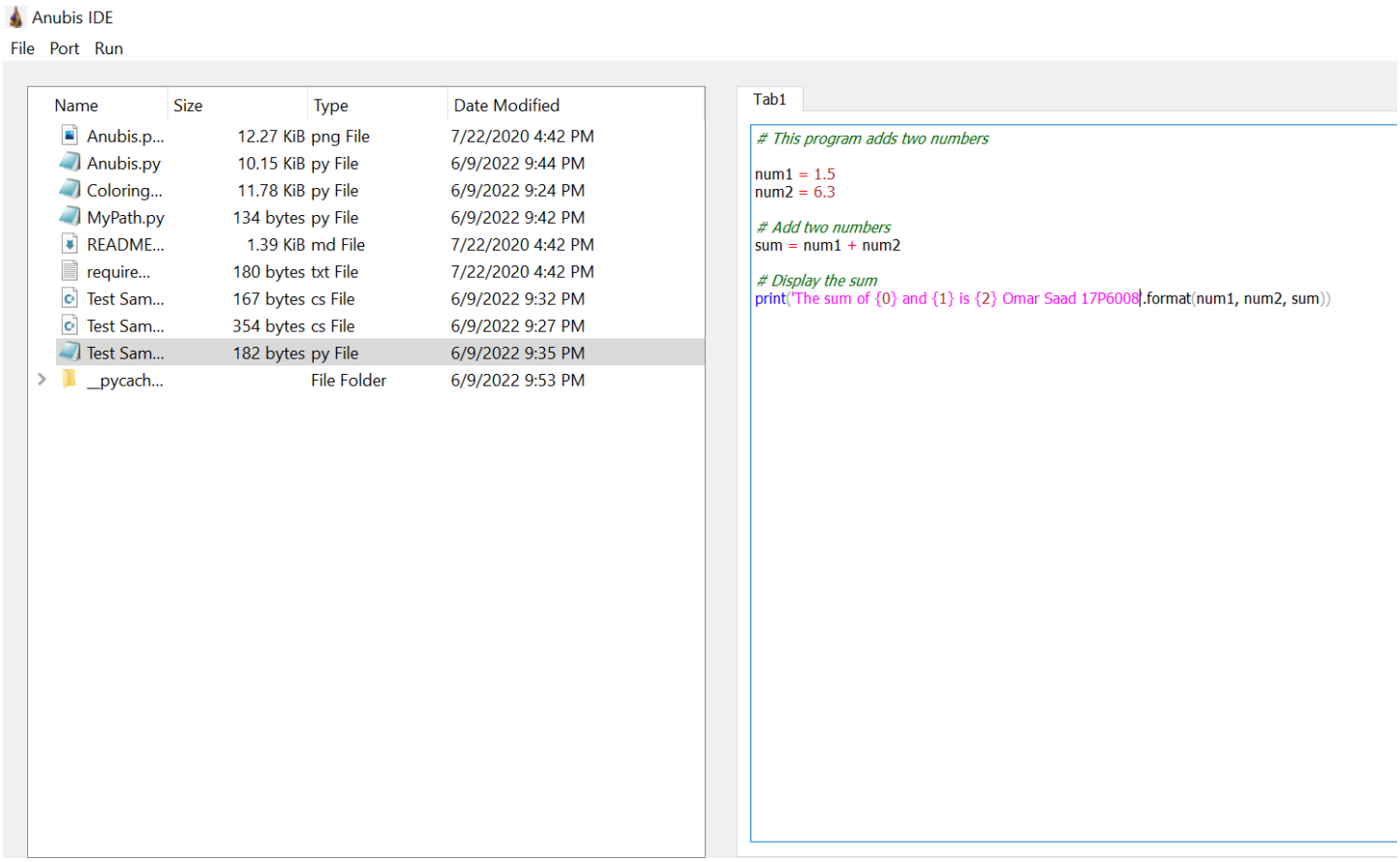

Figure 5: Python Test Code

Here we test a sample Python code:

1. We can see that syntax coloring is working correctly.
2. Addition of two variables.
3. Print Statement.

5. Code

5.1 MyPath.py

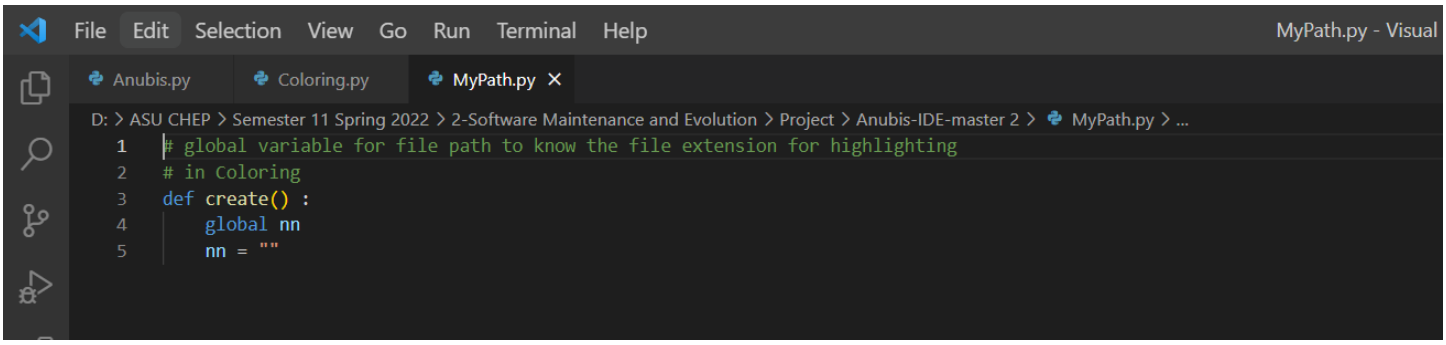

Figure 6: MyPath [1]

5.2 Coloring.py

File Edit Selection View Go Run Terminal Help Coloring.py - Visual

Anubis.py | Coloring.py

D: > ASU CHEP > Semester 11 Spring 2022 > 2-Software Maintenance and Evolution > Project > Anubis-IDE-master 2 > Coloring.py > ...

```
import sys
from PyQt5.QtCore import QRegExp
from PyQt5.QtGui import QColor, QTextCharFormat, QFont, QSyntaxHighlighter
import MyPath
# A Module created for sharing a Global Variable for Path to be used based
# on extension in Coloring


def format(color, style=''):
    """
    Return a QTextCharFormat with the given attributes.
    """
    _color = QColor()
    if type(color) is not str:
        _color.setRgb(color[0], color[1], color[2])
    else:
        _color.setNamedColor(color)

    _format = QTextCharFormat()
    _format.setForeground(_color)
    if 'bold' in style:
        _format.setFontWeight(QFont.Bold)
    if 'italic' in style:
        _format.setFontItalic(True)

    return _format


# Syntax styles that can be shared by all languages

STYLES2 = {
    'keyword': format([200, 120, 50], 'bold'),
    'operator': format([150, 150, 150]),
    'brace': format('darkGray'),
    'defclass': format([220, 220, 255], 'bold'),
    'string': format([20, 110, 100]),
    'string2': format([30, 120, 110]),
    'comment': format([128, 128, 128]),
    'self': format([150, 85, 140], 'italic'),
    'numbers': format([100, 150, 190]),
}
STYLES = {
    'keyword': format('blue'),
    'operator': format('red'),
    'brace': format('darkGray'),
    'defclass': format('black', 'bold'),
    'string': format('magenta'),
    'string2': format('darkMagenta'),
    'comment': format('darkGreen', 'italic'),
    'self': format('black', 'italic'),
    'numbers': format('brown'),
}


class Highlighter(QSyntaxHighlighter):
    """Syntax highlighter for the Python and C# languages.
    """
    # Python keywords
    pyKeywords = [
```

Figure 7: Coloring [1]

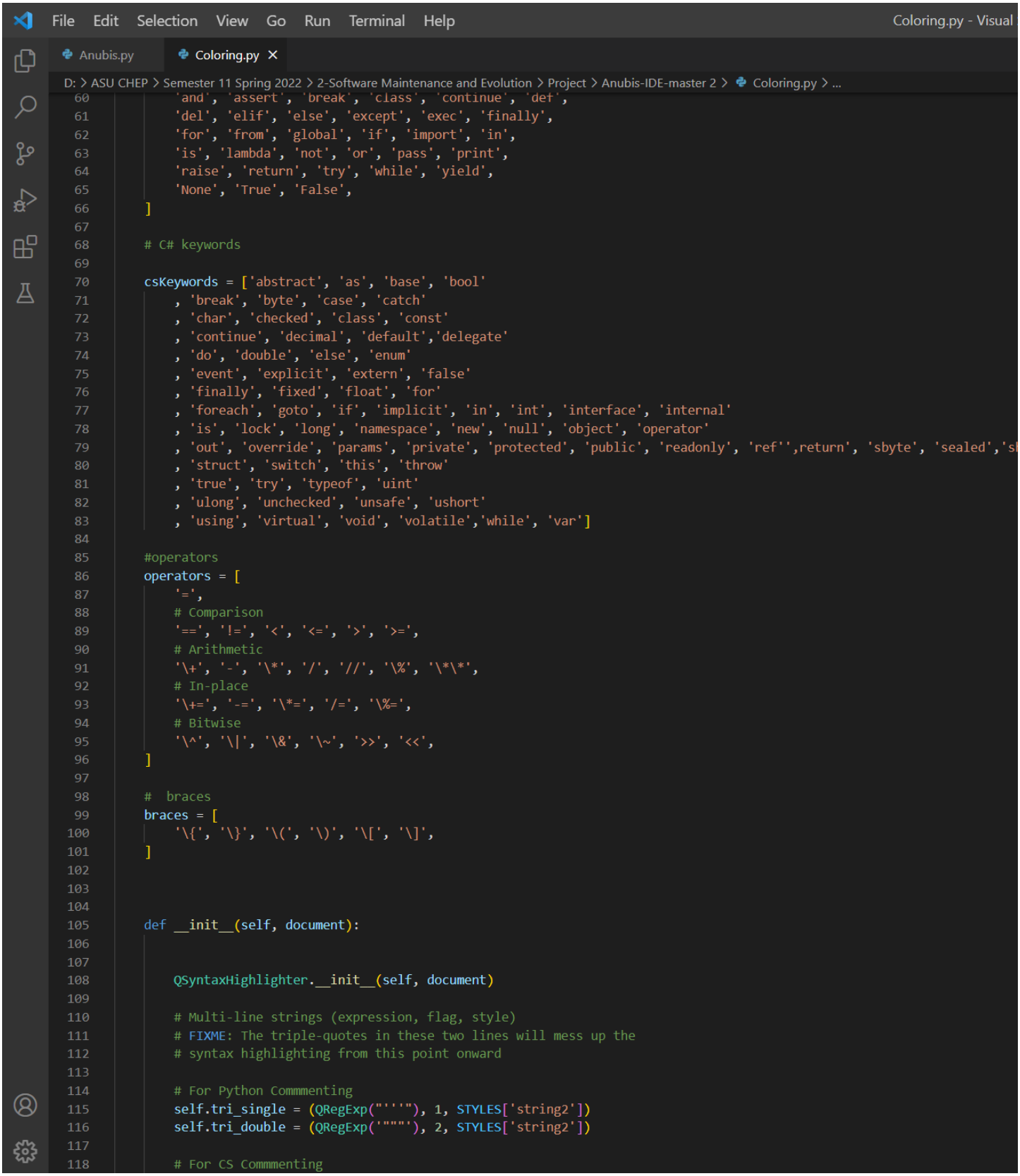

Figure 8: Coloring [2]

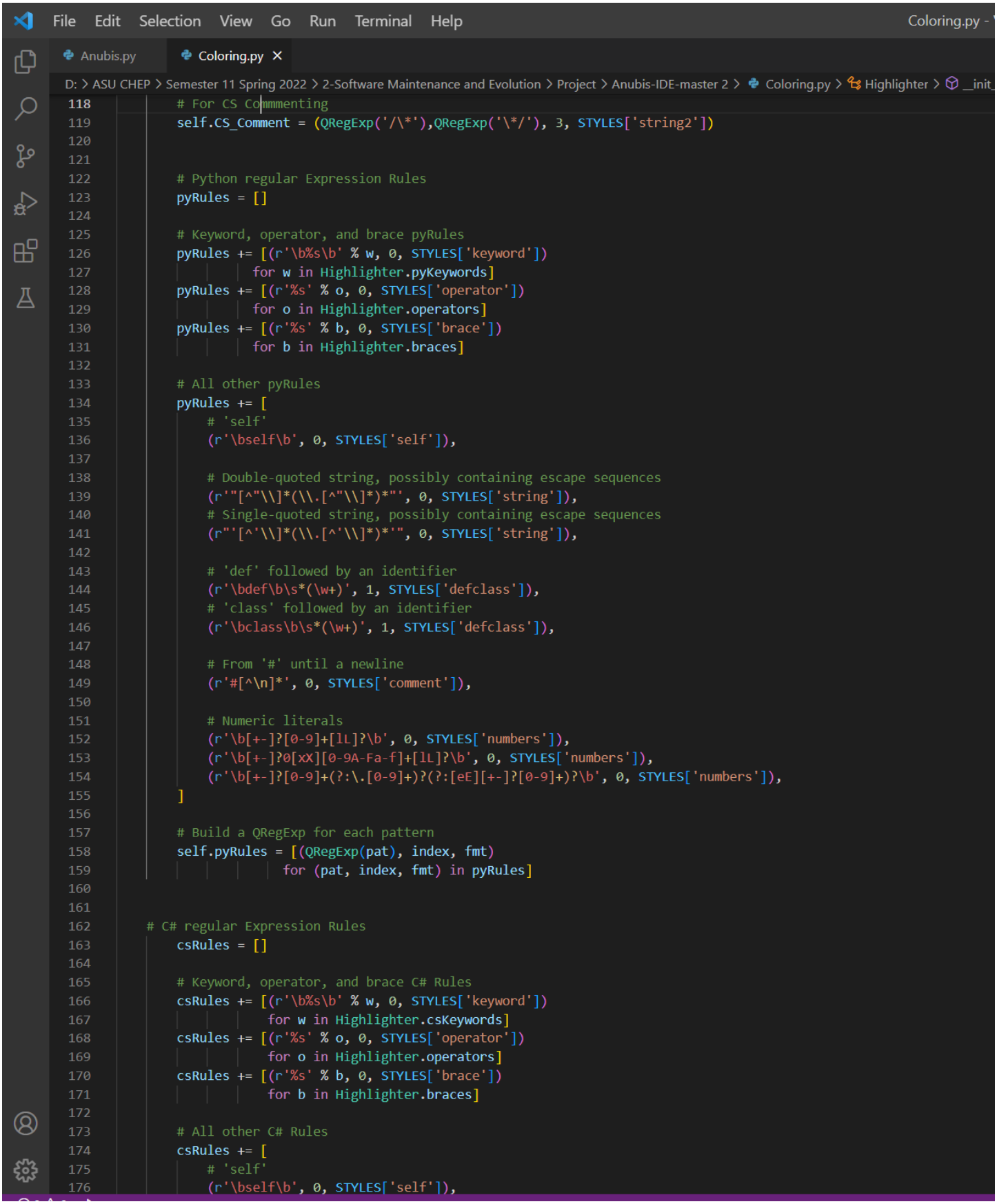

Figure 9: Coloring [3]

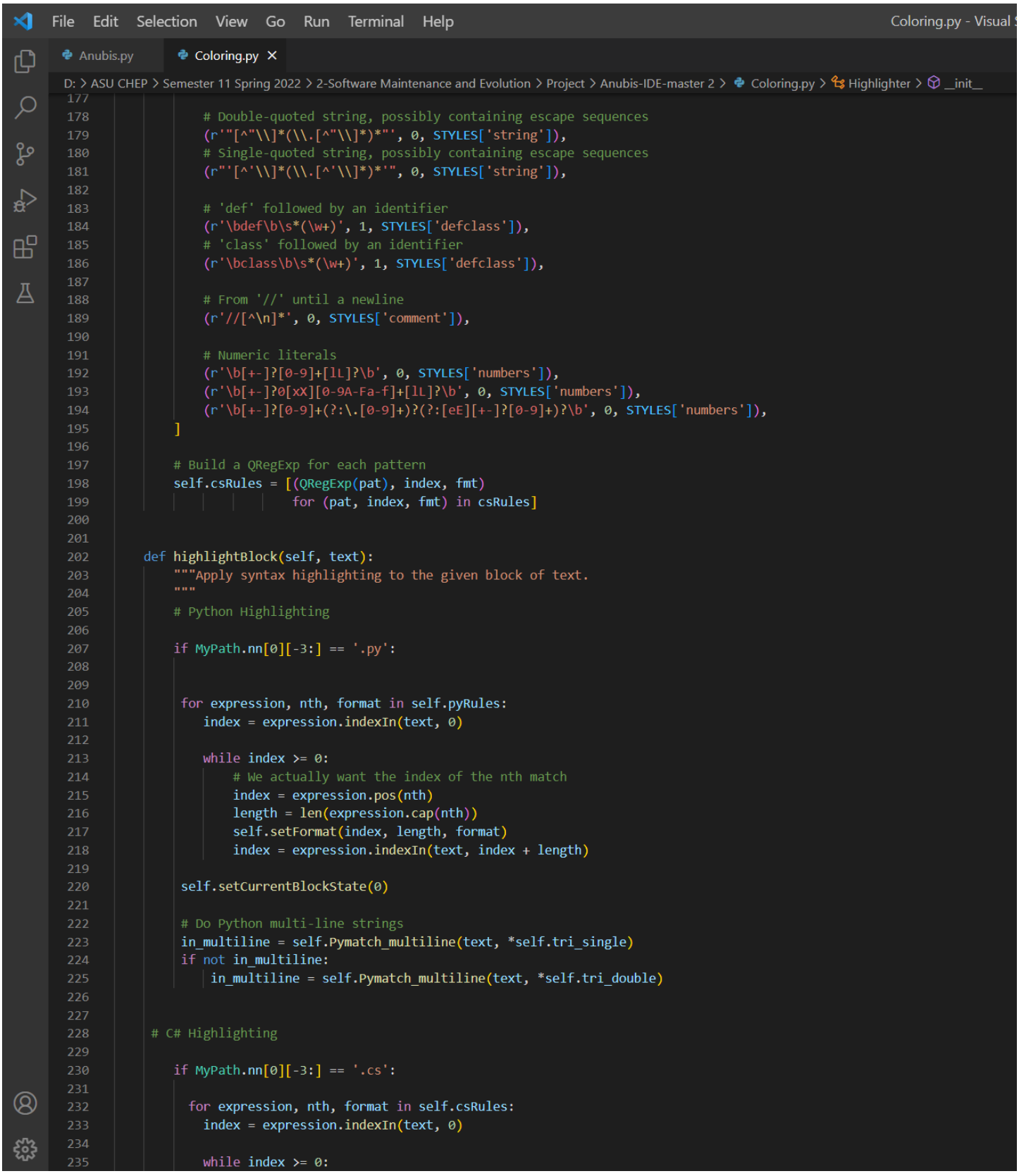

Figure 10: Coloring [4]

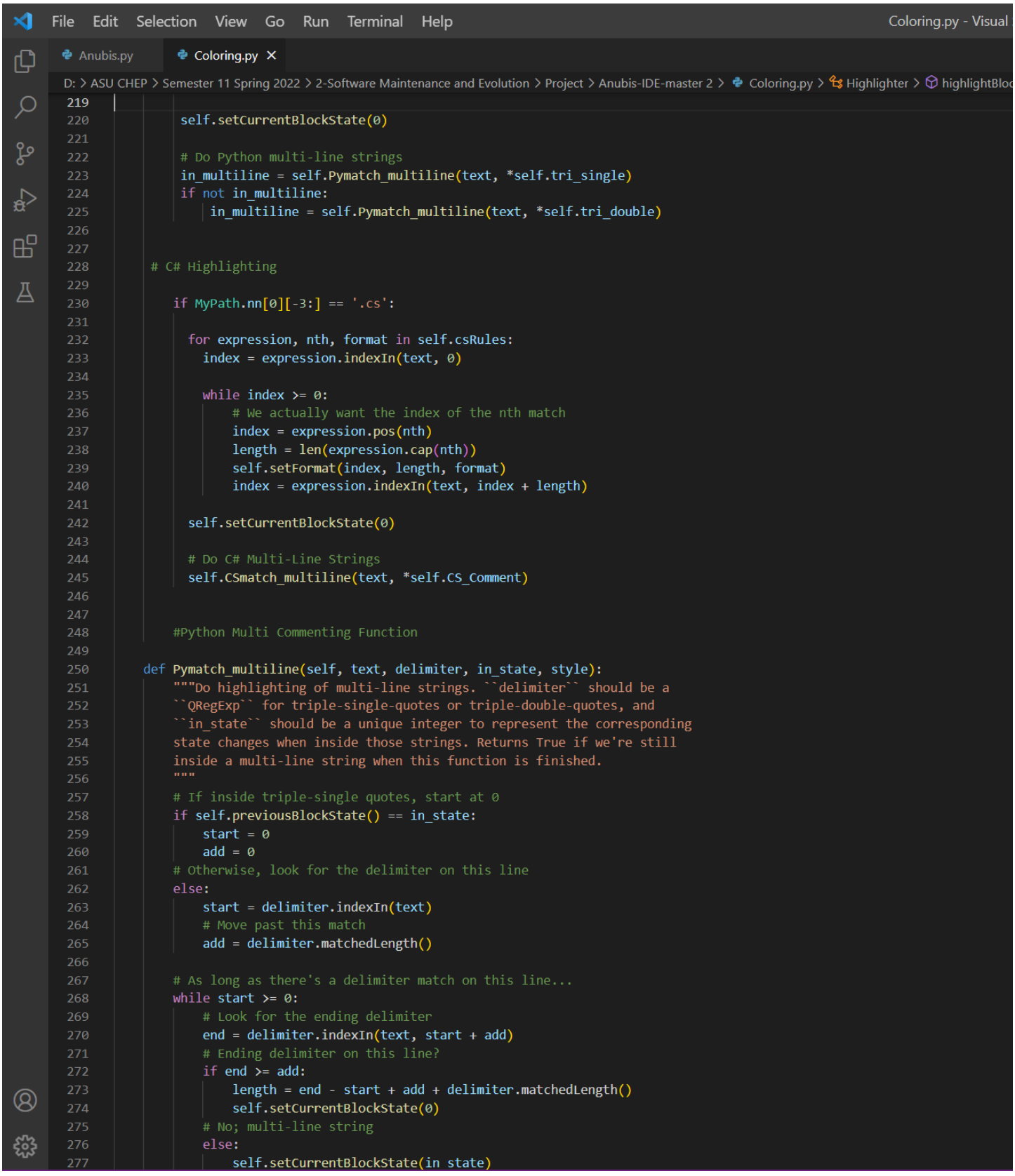

Figure 11: Coloring [5]

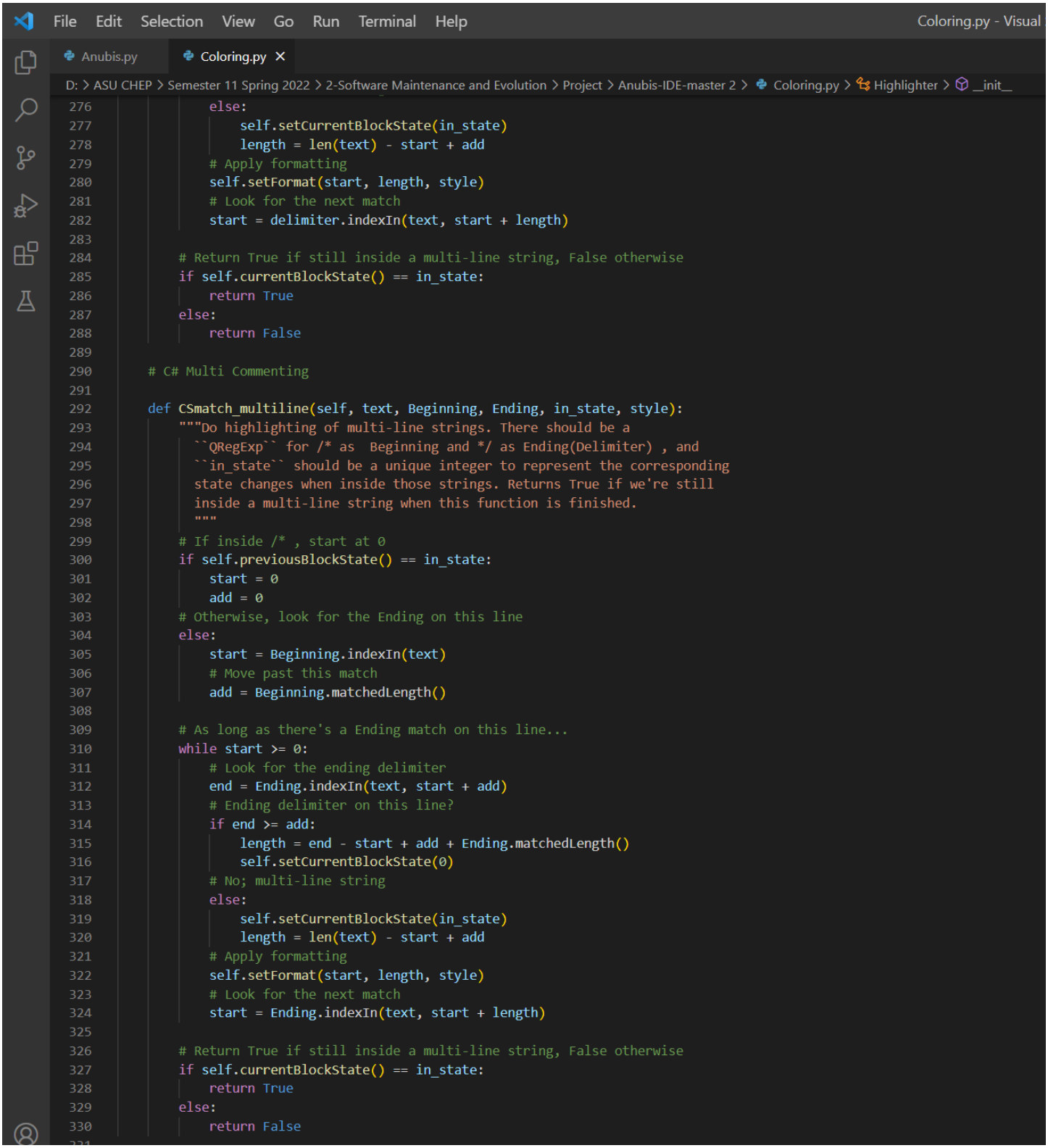

Figure 12: Coloring [6]

5.3 Anubis.py

File Edit Selection View Go Run Terminal Help Anubis.py - Visual

Anubis.py

D: > ASU CHEP > Semester 11 Spring 2022 > 2-Software Maintenance and Evolution > Project > Anubis-IDE-master 2 > Anubis.py > serial_ports

```
import sys
import glob
import serial

import Coloring
from PyQt5 import QtCore
from PyQt5 import QtGui
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from pathlib import Path
import MyPath

def serial_ports():
    """ Lists serial port names
        :raises EnvironmentError:
            On unsupported or unknown platforms
        :returns:
            A list of the serial ports available on the system
    """
    if sys.platform.startswith('win'):
        ports = ['COM%s' % (i + 1) for i in range(256)]
    elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):
        # this excludes your current terminal "/dev/tty"
        ports = glob.glob('/dev/tty[A-Za-z]*')
    elif sys.platform.startswith('darwin'):
        ports = glob.glob('/dev/tty.*')
    else:
        raise EnvironmentError('Unsupported platform')

    result = []
    for port in ports:
        try:
            s = serial.Serial(port)
            s.close()
            result.append(port)
        except (OSError, serial.SerialException):
            pass
    return result


#
#
#
#
############ Signal Class ############
#
#
#
#
class Signal(QObject):

    # initializing a Signal which will take (string) as an input
    reading = pyqtSignal(str)

    # init Function for the Signal class
    def __init__(self):
        QObject.__init__(self)

#
```

Figure 13: Anubis [1]

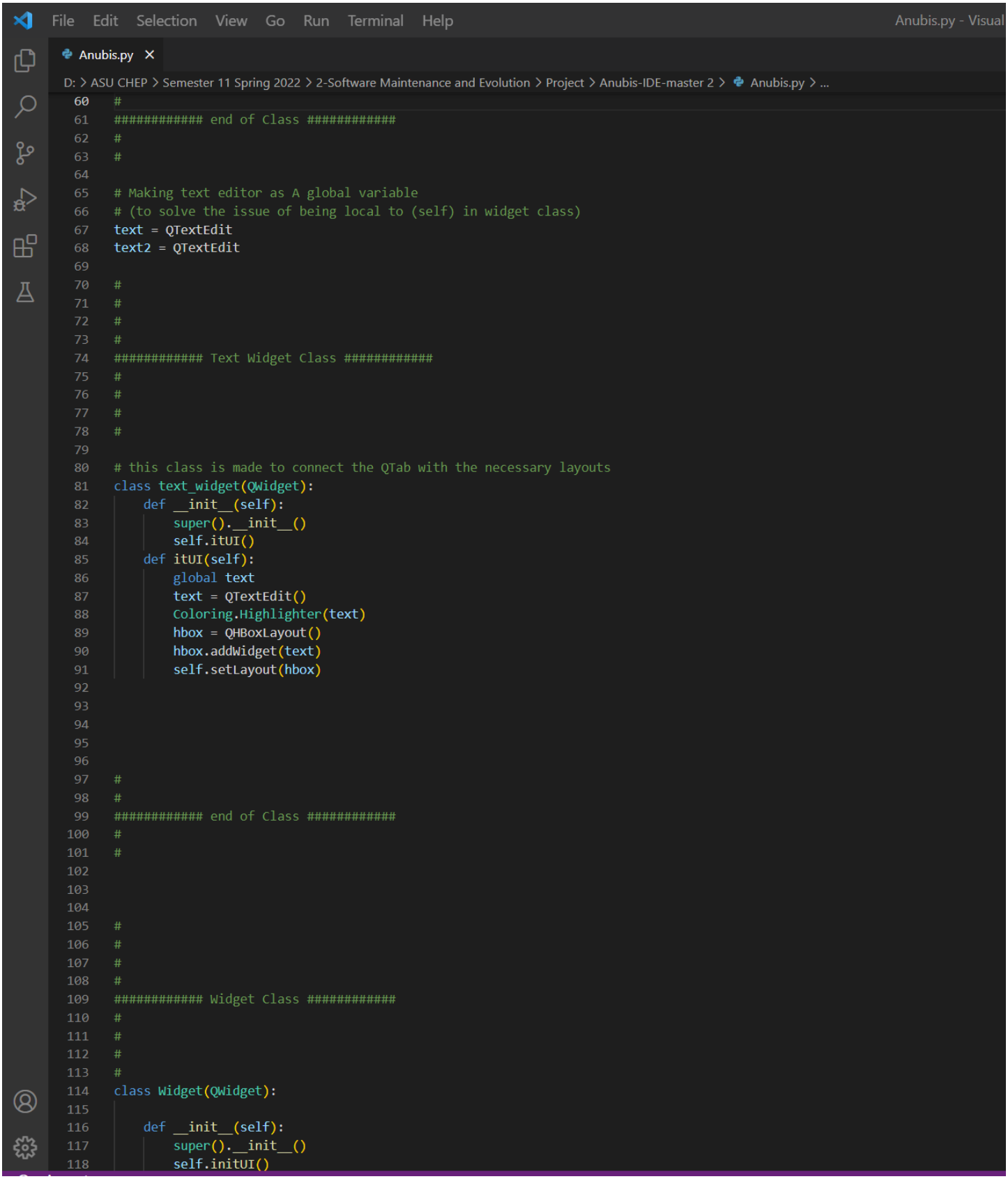

Figure 14: Anubis [2]

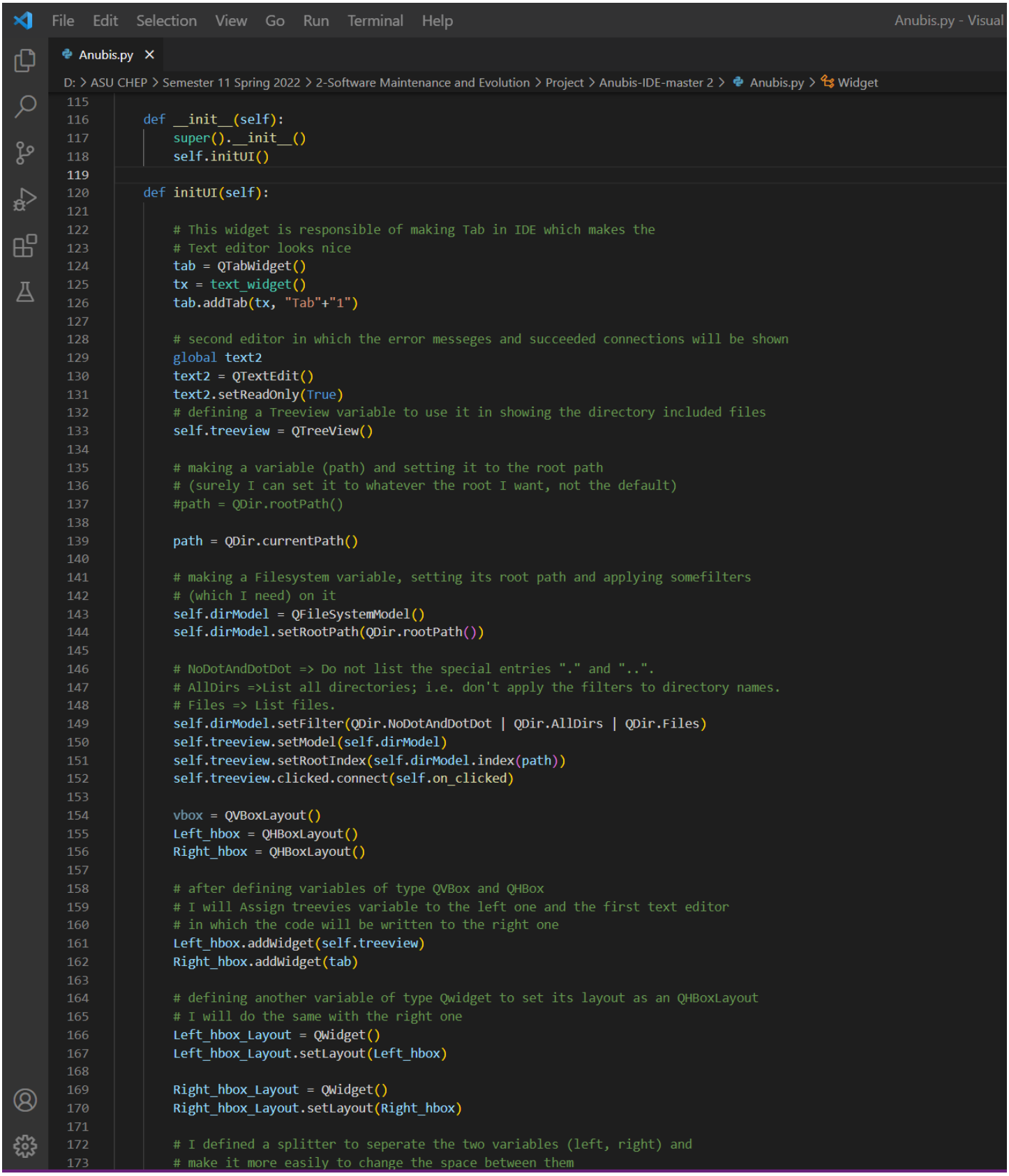

Figure 15: Anubis [3]

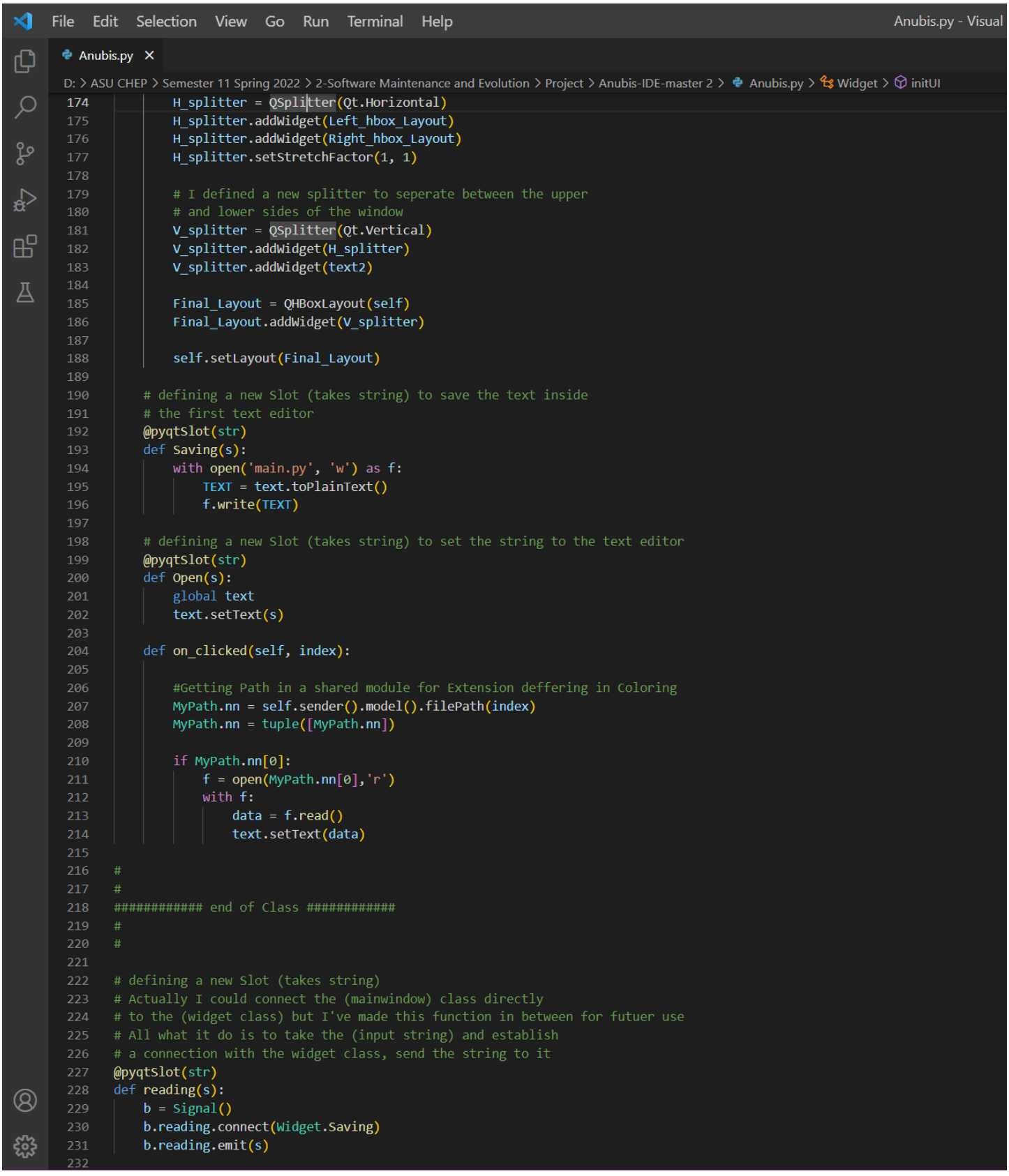

Figure 16: Anubis [4]

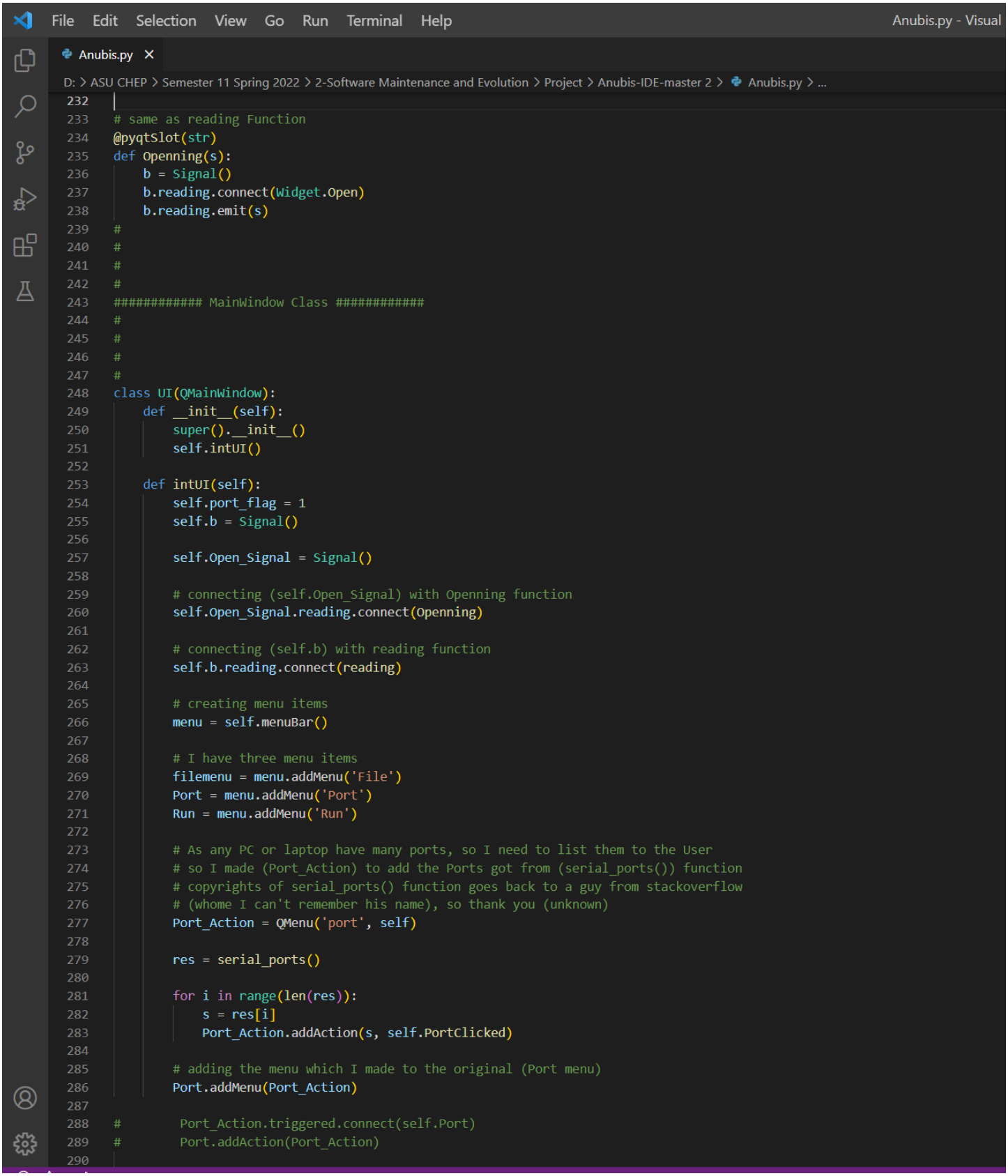

Figure 17: Anubis [5]

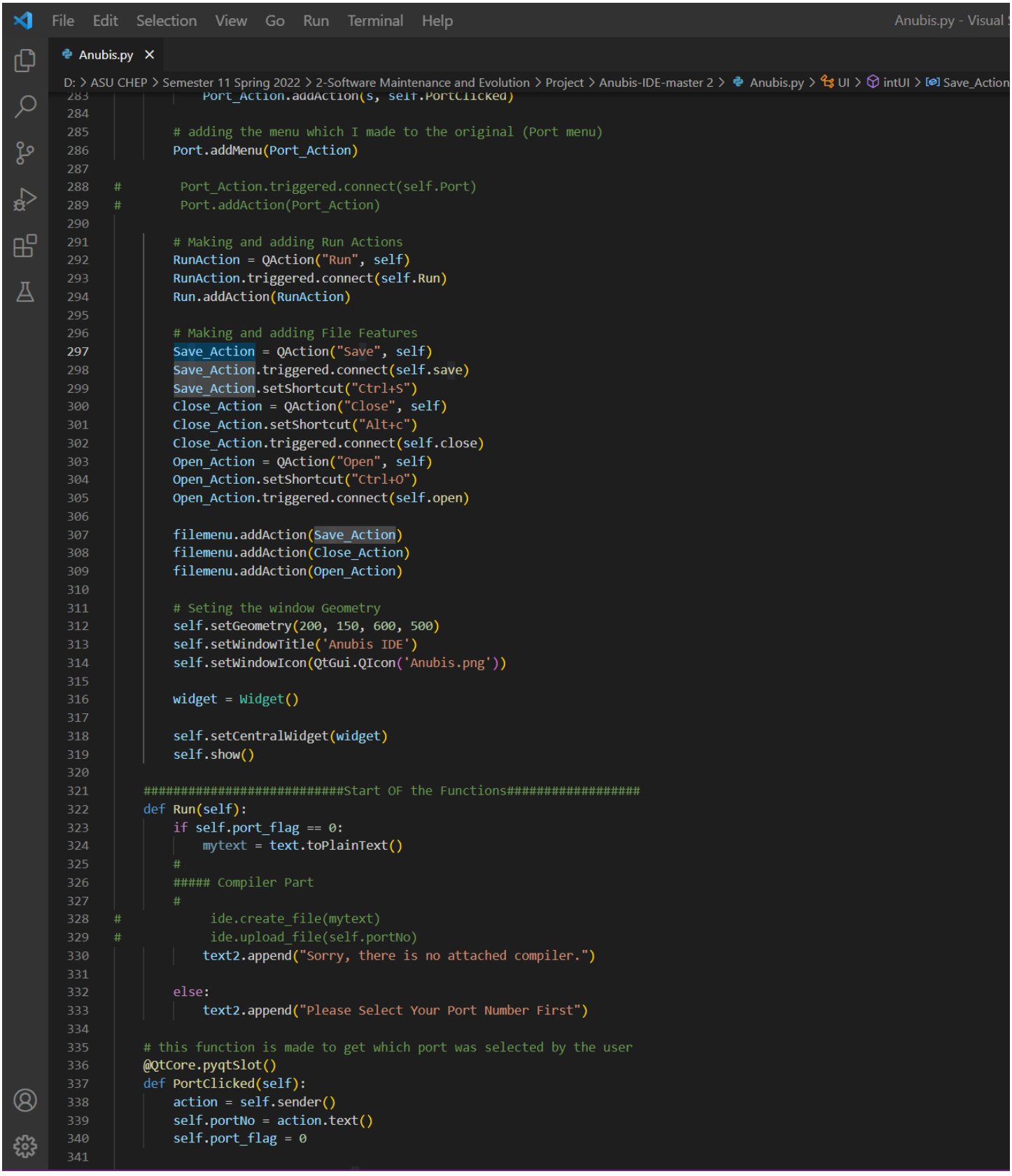

Figure 18: Anubis [6]

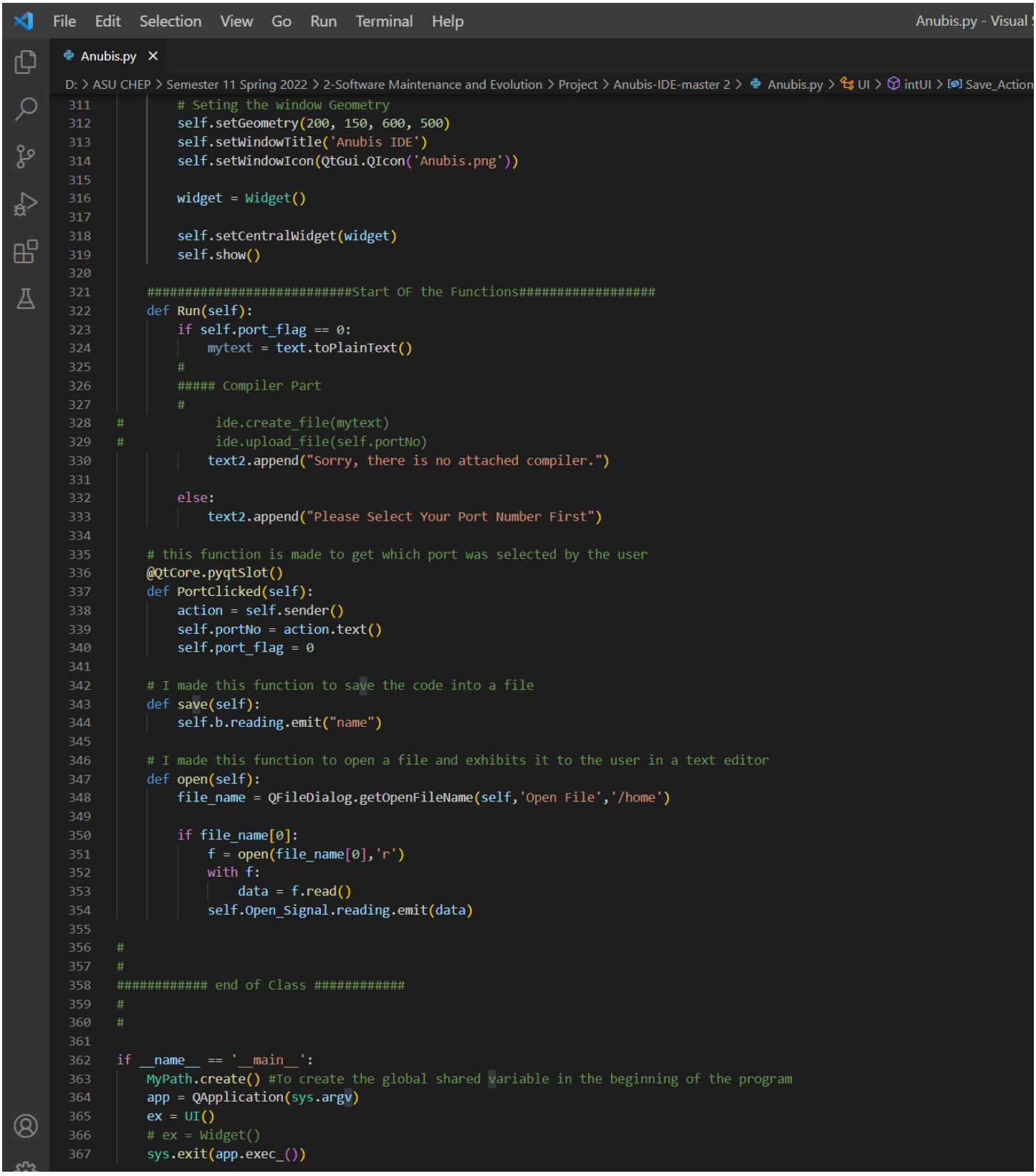

Figure 19: Anubis [7]