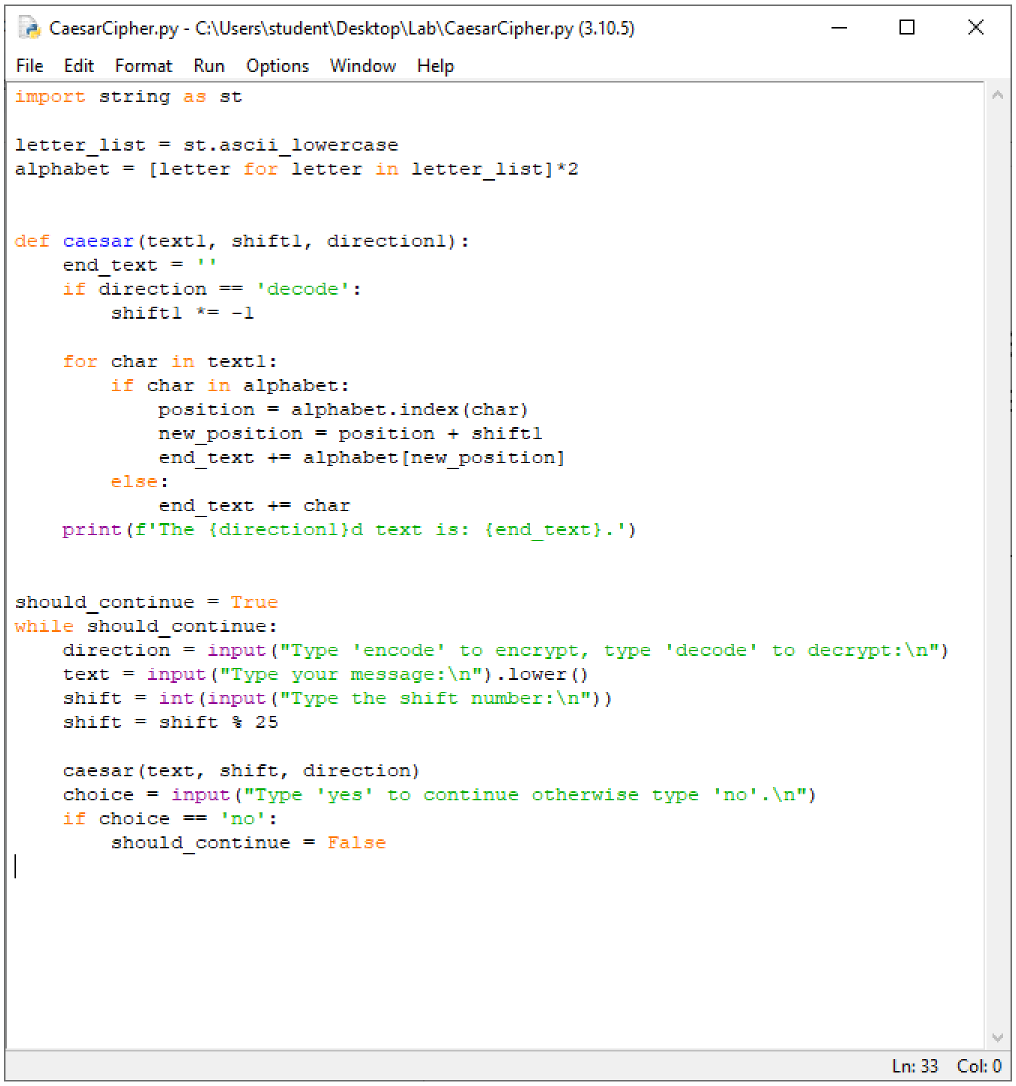
IT314: Software Engineering

Lab 5 Static Analysis

Name: Prashasti Srivastava
Student ID: 202001125

Static Analysis Tool: Pylint

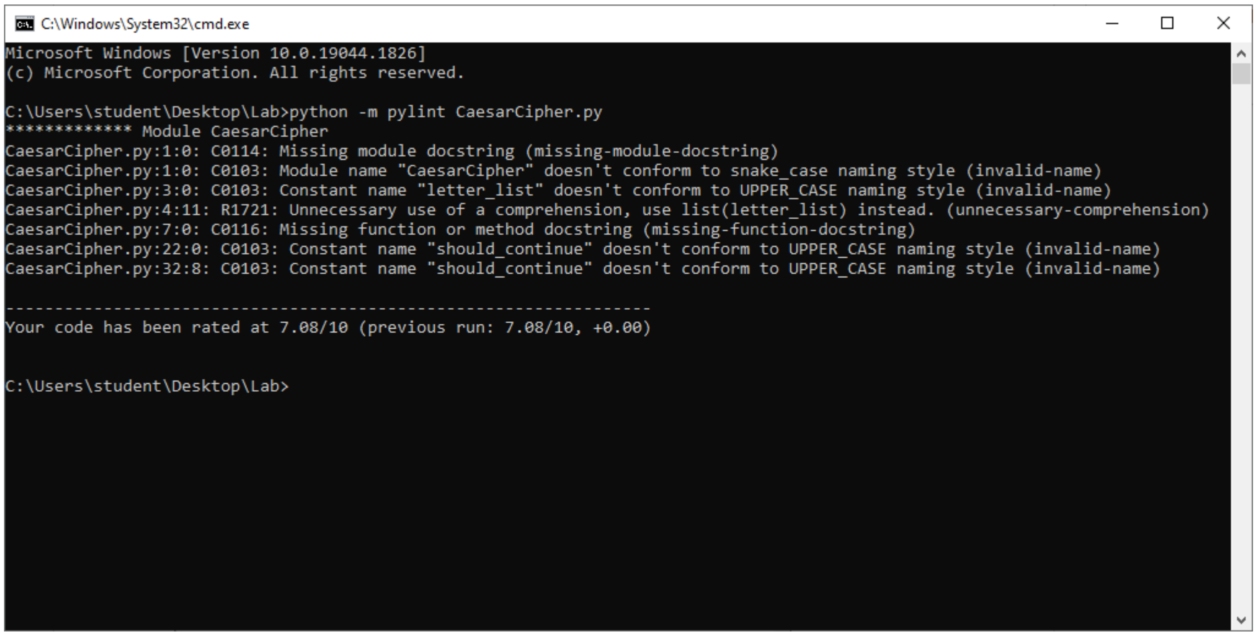
Module 1: CaesarCipher.py



CaesarCipher.py - C:\Users\student\Desktop\Lab\CaesarCipher.py (3.10.5)

```python
import string as st

letter_list = st.ascii_lowercase
alphabet = [letter for letter in letter_list]*2


def caesar(text1, shift1, direction1):
    end_text = ''
    if direction == 'decode':
        shift1 *= -1

    for char in text1:
        if char in alphabet:
            position = alphabet.index(char)
            new_position = position + shift1
            end_text += alphabet[new_position]
        else:
            end_text += char
    print(f'The {direction1}d text is: {end_text}.')


should_continue = True
while should_continue:
    direction = input("Type 'encode' to encrypt, type 'decode' to decrypt:\n")
    text = input("Type your message:\n").lower()
    shift = int(input("Type the shift number:\n"))
    shift = shift % 25

    caesar(text, shift, direction)
    choice = input("Type 'yes' to continue otherwise type 'no'.\n")
    if choice == 'no':
        should_continue = False
```

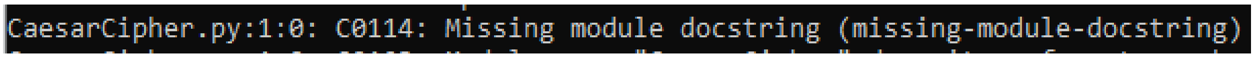
Analysis:



C:\Windows\System32\cmd.exe

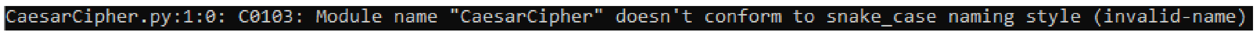
```
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\student\Desktop\Lab>python -m pylint CaesarCipher.py
************* Module CaesarCipher
CaesarCipher.py:1:0: C0114: Missing module docstring (missing-module-docstring)
CaesarCipher.py:1:0: C0103: Module name "CaesarCipher" doesn't conform to snake_case naming style (invalid-name)
CaesarCipher.py:3:0: C0103: Constant name "letter_list" doesn't conform to UPPER_CASE naming style (invalid-name)
CaesarCipher.py:4:11: R1721: Unnecessary use of a comprehension, use list(letter_list) instead. (unnecessary-comprehension)
CaesarCipher.py:7:0: C0116: Missing function or method docstring (missing-function-docstring)
CaesarCipher.py:22:0: C0103: Constant name "should_continue" doesn't conform to UPPER_CASE naming style (invalid-name)
CaesarCipher.py:32:8: C0103: Constant name "should_continue" doesn't conform to UPPER_CASE naming style (invalid-name)

------------------------------------------------------------------
Your code has been rated at 7.08/10 (previous run: 7.08/10, +0.00)


C:\Users\student\Desktop\Lab>
```

Errors:



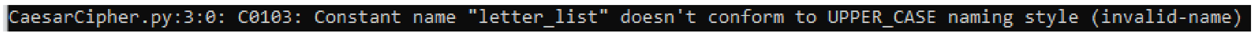
```
CaesarCipher.py:1:0: C0114: Missing module docstring (missing-module-docstring)
```

Python documentation strings (or docstrings) provide a convenient way of associating documentation with Python modules, functions, classes, and methods and this module has no docstring at the starting of the file



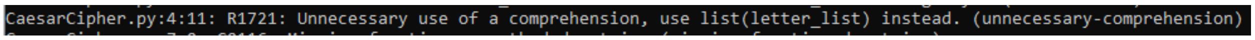
```
CaesarCipher.py:1:0: C0103: Module name "CaesarCipher" doesn't conform to snake_case naming style (invalid-name)
```

Snake case style of writing in which each space is replaced with an underscore (_) character, and the first letter of each word is written in lowercase. This convention is not followed by the particular name.



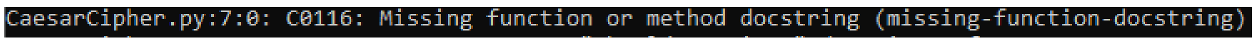
```
CaesarCipher.py:3:0: C0103: Constant name "letter_list" doesn't conform to UPPER_CASE naming style (invalid-name)
```

Upper Case is used for variables that should be constant. This convention is not followed by the particular name.



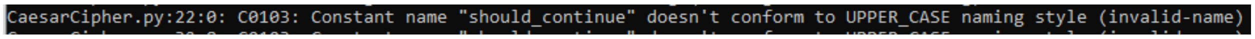
```
CaesarCipher.py:4:11: R1721: Unnecessary use of a comprehension, use list(letter_list) instead. (unnecessary-comprehension)
```

It states that instead of using an identity comprehension, we should consider using the list, dict or set constructor for faster and simpler code.



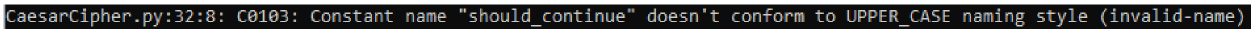
```
CaesarCipher.py:7:0: C0116: Missing function or method docstring (missing-function-docstring)
```

Python documentation strings (or docstrings) provide a convenient way of associating documentation with Python modules, functions, classes, and methods and this module function has no docstring.



```
CaesarCipher.py:22:0: C0103: Constant name "should_continue" doesn't conform to UPPER_CASE naming style (invalid-name)
```

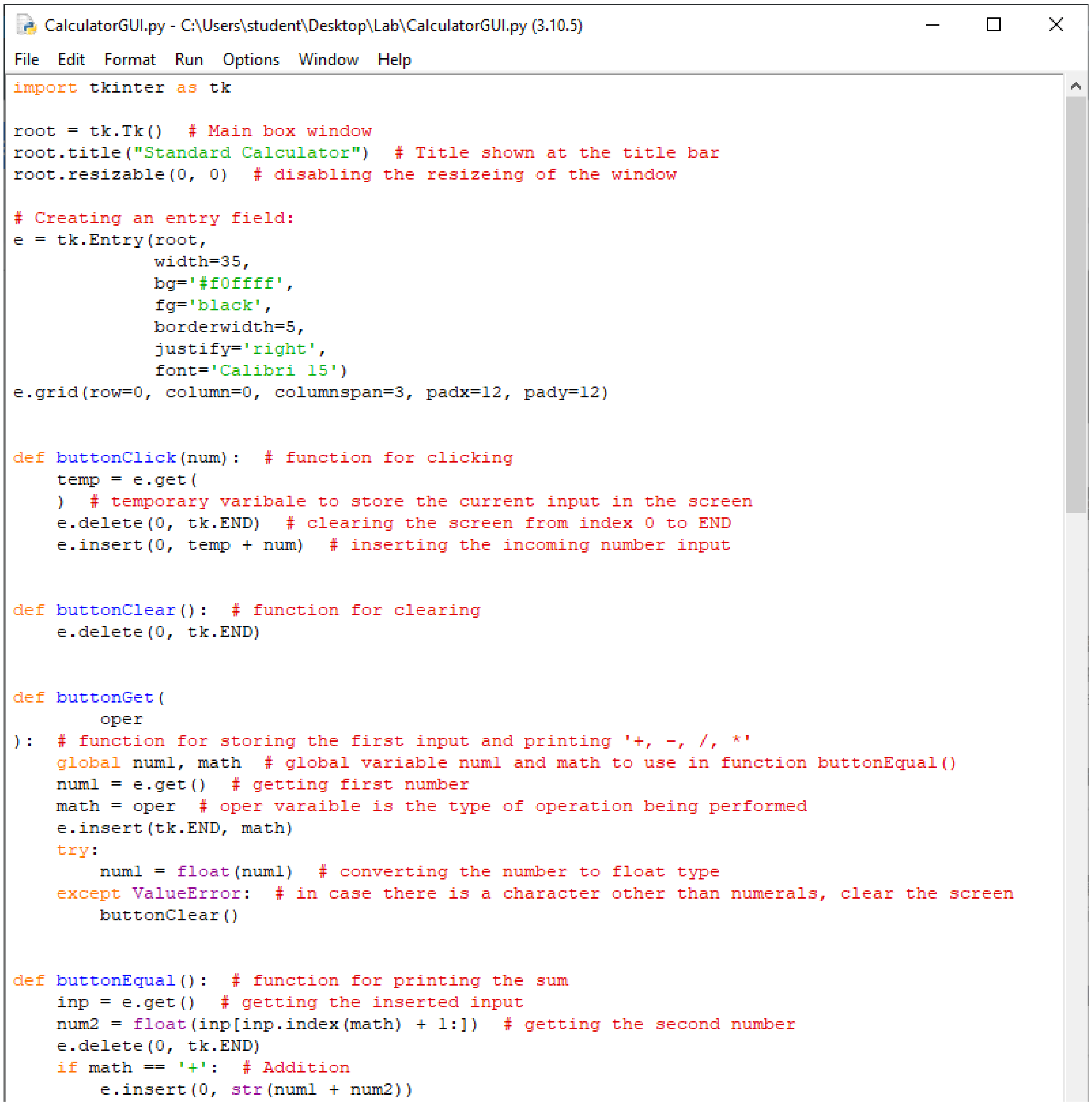
Upper Case is used for variables that should be constant. This convention is not followed by the particular name.



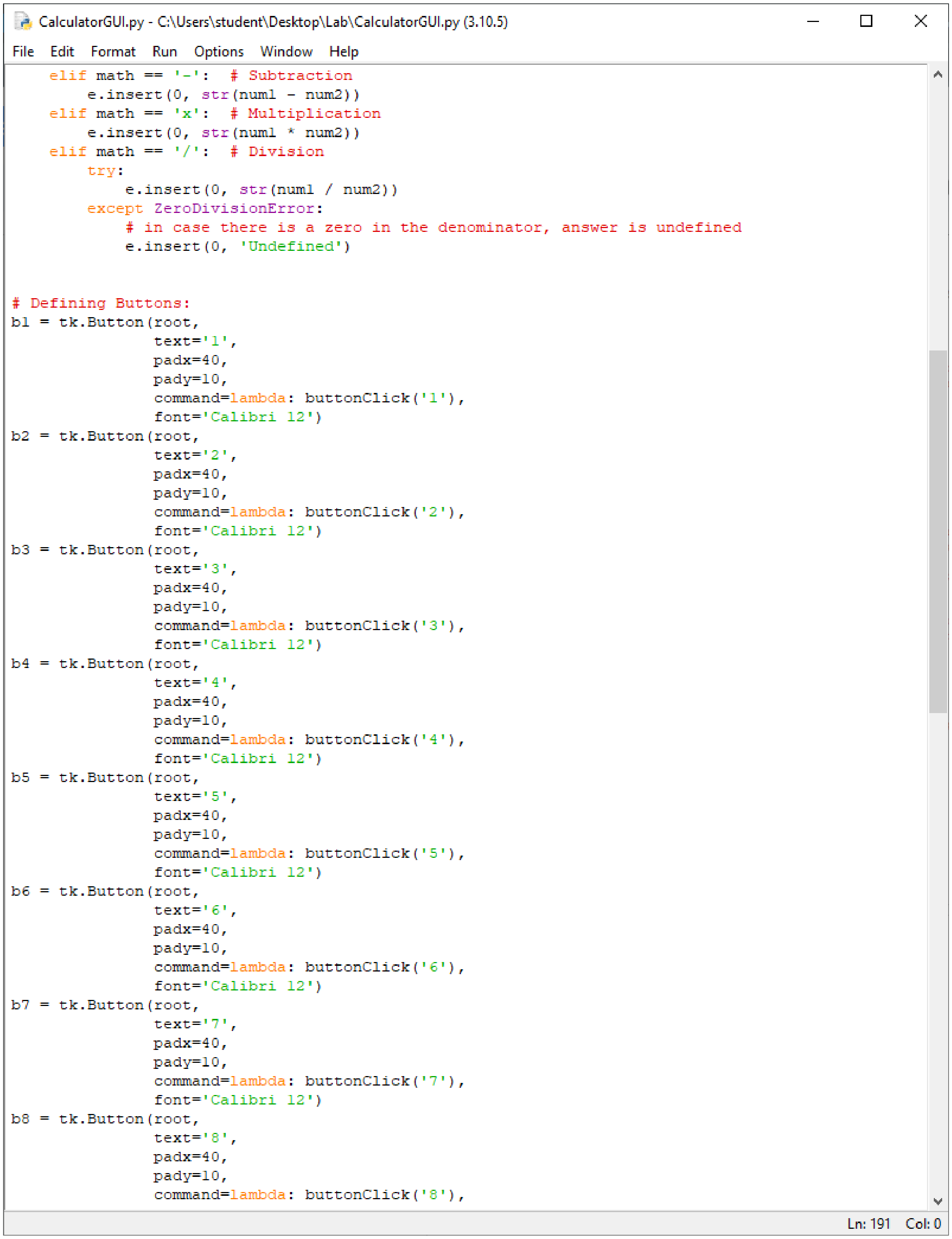
```
CaesarCipher.py:32:8: C0103: Constant name "should_continue" doesn't conform to UPPER_CASE naming style (invalid-name)
```

Upper Case is used for variables that should be constant. This convention is not followed by the particular name.

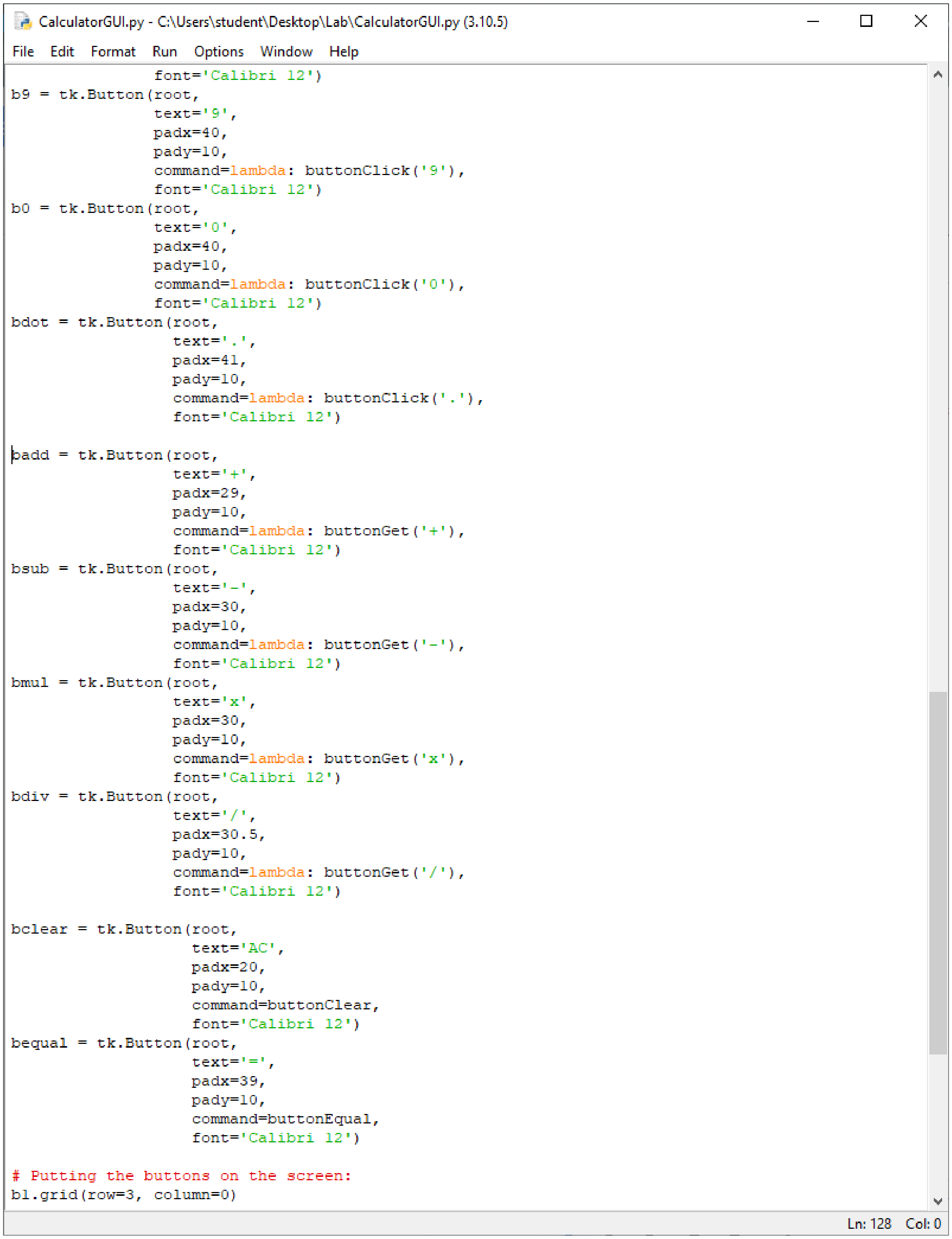
Module 2: CalculatorGUI.py


CalculatorGUI.py - C:\Users\student\Desktop\Lab\CalculatorGUI.py (3.10.5)
File Edit Format Run Options Window Help

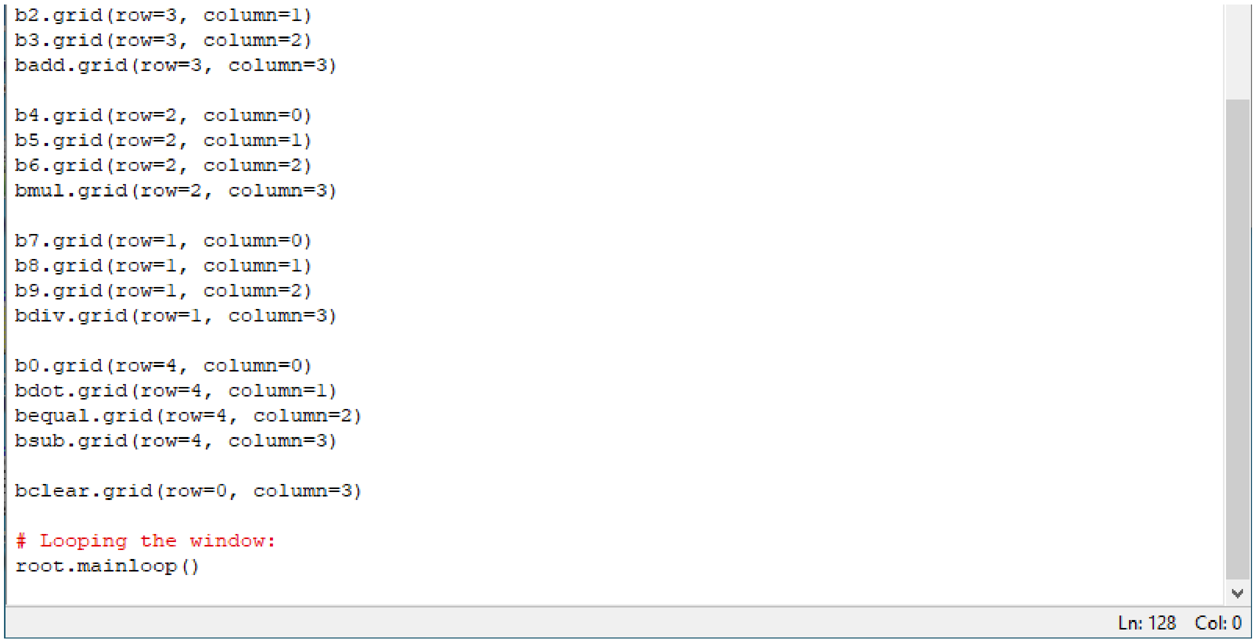
```python
import tkinter as tk

root = tk.Tk()  # Main box window
root.title("Standard Calculator")  # Title shown at the title bar
root.resizable(0, 0)  # disabling the resizeing of the window

# Creating an entry field:
e = tk.Entry(root,
             width=35,
             bg='#f0ffff',
             fg='black',
             borderwidth=5,
             justify='right',
             font='Calibri 15')
e.grid(row=0, column=0, columnspan=3, padx=12, pady=12)


def buttonClick(num):  # function for clicking
    temp = e.get(
    )  # temporary varibale to store the current input in the screen
    e.delete(0, tk.END)  # clearing the screen from index 0 to END
    e.insert(0, temp + num)  # inserting the incoming number input


def buttonClear():  # function for clearing
    e.delete(0, tk.END)


def buttonGet(
        oper
):  # function for storing the first input and printing '+, -, /, *'
    global num1, math  # global variable num1 and math to use in function buttonEqual()
    num1 = e.get()  # getting first number
    math = oper  # oper varaible is the type of operation being performed
    e.insert(tk.END, math)
    try:
        num1 = float(num1)  # converting the number to float type
    except ValueError:  # in case there is a character other than numerals, clear the screen
        buttonClear()


def buttonEqual():  # function for printing the sum
    inp = e.get()  # getting the inserted input
    num2 = float(inp[inp.index(math) + 1:])  # getting the second number
    e.delete(0, tk.END)
    if math == '+':  # Addition
        e.insert(0, str(num1 + num2))
```


CalculatorGUI.py - C:\Users\student\Desktop\Lab\CalculatorGUI.py (3.10.5)

```
    elif math == '-':  # Subtraction
        e.insert(0, str(num1 - num2))
    elif math == 'x':  # Multiplication
        e.insert(0, str(num1 * num2))
    elif math == '/':  # Division
        try:
            e.insert(0, str(num1 / num2))
        except ZeroDivisionError:
            # in case there is a zero in the denominator, answer is undefined
            e.insert(0, 'Undefined')


# Defining Buttons:
b1 = tk.Button(root,
               text='1',
               padx=40,
               pady=10,
               command=lambda: buttonClick('1'),
               font='Calibri 12')
b2 = tk.Button(root,
               text='2',
               padx=40,
               pady=10,
               command=lambda: buttonClick('2'),
               font='Calibri 12')
b3 = tk.Button(root,
               text='3',
               padx=40,
               pady=10,
               command=lambda: buttonClick('3'),
               font='Calibri 12')
b4 = tk.Button(root,
               text='4',
               padx=40,
               pady=10,
               command=lambda: buttonClick('4'),
               font='Calibri 12')
b5 = tk.Button(root,
               text='5',
               padx=40,
               pady=10,
               command=lambda: buttonClick('5'),
               font='Calibri 12')
b6 = tk.Button(root,
               text='6',
               padx=40,
               pady=10,
               command=lambda: buttonClick('6'),
               font='Calibri 12')
b7 = tk.Button(root,
               text='7',
               padx=40,
               pady=10,
               command=lambda: buttonClick('7'),
               font='Calibri 12')
b8 = tk.Button(root,
               text='8',
               padx=40,
               pady=10,
               command=lambda: buttonClick('8'),
```


CalculatorGUI.py - C:\Users\student\Desktop\Lab\CalculatorGUI.py (3.10.5)
File Edit Format Run Options Window Help
Ln: 128 Col: 0

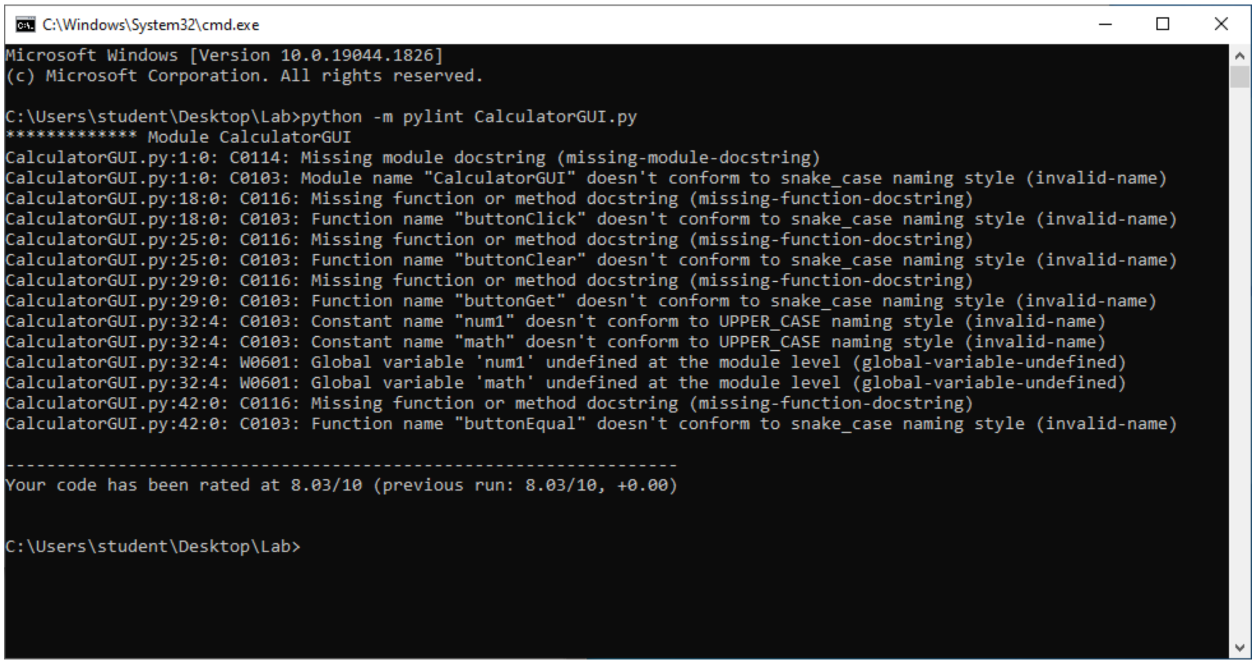
```
               font='Calibri 12')
b9 = tk.Button(root,
               text='9',
               padx=40,
               pady=10,
               command=lambda: buttonClick('9'),
               font='Calibri 12')
b0 = tk.Button(root,
               text='0',
               padx=40,
               pady=10,
               command=lambda: buttonClick('0'),
               font='Calibri 12')
bdot = tk.Button(root,
                 text='.',
                 padx=41,
                 pady=10,
                 command=lambda: buttonClick('.'),
                 font='Calibri 12')

badd = tk.Button(root,
                 text='+',
                 padx=29,
                 pady=10,
                 command=lambda: buttonGet('+'),
                 font='Calibri 12')
bsub = tk.Button(root,
                 text='-',
                 padx=30,
                 pady=10,
                 command=lambda: buttonGet('-'),
                 font='Calibri 12')
bmul = tk.Button(root,
                 text='x',
                 padx=30,
                 pady=10,
                 command=lambda: buttonGet('x'),
                 font='Calibri 12')
bdiv = tk.Button(root,
                 text='/',
                 padx=30.5,
                 pady=10,
                 command=lambda: buttonGet('/'),
                 font='Calibri 12')

bclear = tk.Button(root,
                   text='AC',
                   padx=20,
                   pady=10,
                   command=buttonClear,
                   font='Calibri 12')
bequal = tk.Button(root,
                   text='=',
                   padx=39,
                   pady=10,
                   command=buttonEqual,
                   font='Calibri 12')

# Putting the buttons on the screen:
b1.grid(row=3, column=0)
```



```
b2.grid(row=3, column=1)
b3.grid(row=3, column=2)
badd.grid(row=3, column=3)

b4.grid(row=2, column=0)
b5.grid(row=2, column=1)
b6.grid(row=2, column=2)
bmul.grid(row=2, column=3)

b7.grid(row=1, column=0)
b8.grid(row=1, column=1)
b9.grid(row=1, column=2)
bdiv.grid(row=1, column=3)

b0.grid(row=4, column=0)
bdot.grid(row=4, column=1)
bequal.grid(row=4, column=2)
bsub.grid(row=4, column=3)

bclear.grid(row=0, column=3)

# Looping the window:
root.mainloop()
```

Ln: 128   Col: 0

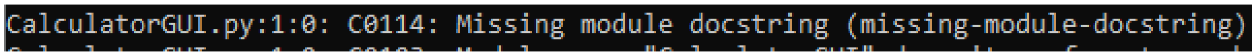
Analysis:



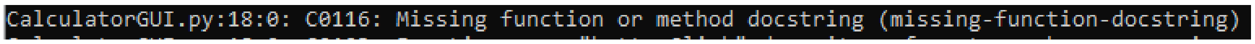
C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\student\Desktop\Lab>python -m pylint CalculatorGUI.py
************* Module CalculatorGUI
CalculatorGUI.py:1:0: C0114: Missing module docstring (missing-module-docstring)
CalculatorGUI.py:1:0: C0103: Module name "CalculatorGUI" doesn't conform to snake_case naming style (invalid-name)
CalculatorGUI.py:18:0: C0116: Missing function or method docstring (missing-function-docstring)
CalculatorGUI.py:18:0: C0103: Function name "buttonClick" doesn't conform to snake_case naming style (invalid-name)
CalculatorGUI.py:25:0: C0116: Missing function or method docstring (missing-function-docstring)
CalculatorGUI.py:25:0: C0103: Function name "buttonClear" doesn't conform to snake_case naming style (invalid-name)
CalculatorGUI.py:29:0: C0116: Missing function or method docstring (missing-function-docstring)
CalculatorGUI.py:29:0: C0103: Function name "buttonGet" doesn't conform to snake_case naming style (invalid-name)
CalculatorGUI.py:32:4: C0103: Constant name "num1" doesn't conform to UPPER_CASE naming style (invalid-name)
CalculatorGUI.py:32:4: C0103: Constant name "math" doesn't conform to UPPER_CASE naming style (invalid-name)
CalculatorGUI.py:32:4: W0601: Global variable 'num1' undefined at the module level (global-variable-undefined)
CalculatorGUI.py:32:4: W0601: Global variable 'math' undefined at the module level (global-variable-undefined)
CalculatorGUI.py:42:0: C0116: Missing function or method docstring (missing-function-docstring)
CalculatorGUI.py:42:0: C0103: Function name "buttonEqual" doesn't conform to snake_case naming style (invalid-name)

------------------------------------------------------------------
Your code has been rated at 8.03/10 (previous run: 8.03/10, +0.00)


C:\Users\student\Desktop\Lab>
```

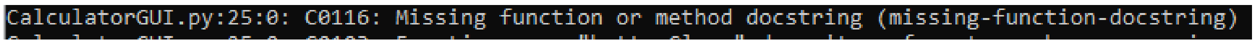
Error:



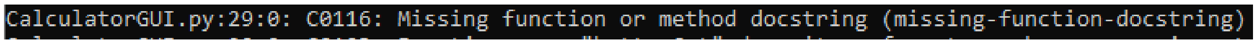
```
CalculatorGUI.py:1:0: C0114: Missing module docstring (missing-module-docstring)
```



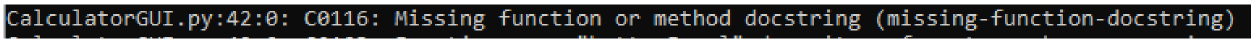
```
CalculatorGUI.py:18:0: C0116: Missing function or method docstring (missing-function-docstring)
```



```
CalculatorGUI.py:25:0: C0116: Missing function or method docstring (missing-function-docstring)
```

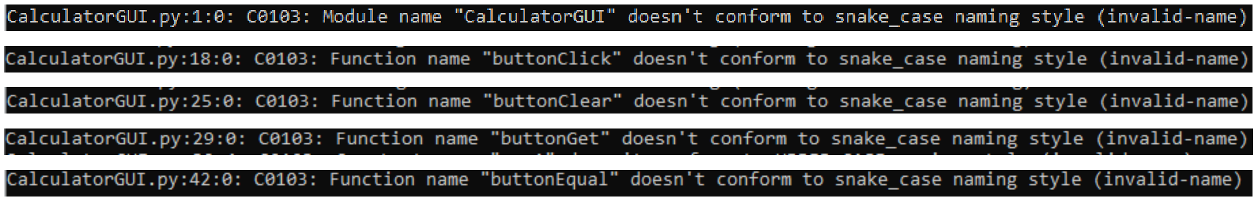


```
CalculatorGUI.py:29:0: C0116: Missing function or method docstring (missing-function-docstring)
```



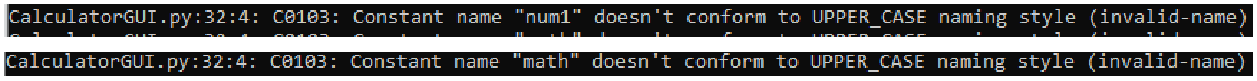
```
CalculatorGUI.py:42:0: C0116: Missing function or method docstring (missing-function-docstring)
```

Python documentation strings (or docstrings) provide a convenient way of associating documentation with Python modules, functions, classes, and methods and this module or function has no docstring at the starting of the file



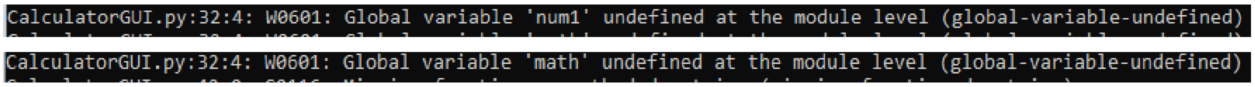
```
CalculatorGUI.py:1:0: C0103: Module name "CalculatorGUI" doesn't conform to snake_case naming style (invalid-name)
CalculatorGUI.py:18:0: C0103: Function name "buttonClick" doesn't conform to snake_case naming style (invalid-name)
CalculatorGUI.py:25:0: C0103: Function name "buttonClear" doesn't conform to snake_case naming style (invalid-name)
CalculatorGUI.py:29:0: C0103: Function name "buttonGet" doesn't conform to snake_case naming style (invalid-name)
CalculatorGUI.py:42:0: C0103: Function name "buttonEqual" doesn't conform to snake_case naming style (invalid-name)
```

Snake case style of writing in which each space is replaced with an underscore (_) character, and the first letter of each word is written in lowercase. This convention is not followed by the particular name.



```
CalculatorGUI.py:32:4: C0103: Constant name "num1" doesn't conform to UPPER_CASE naming style (invalid-name)
CalculatorGUI.py:32:4: C0103: Constant name "math" doesn't conform to UPPER_CASE naming style (invalid-name)
```

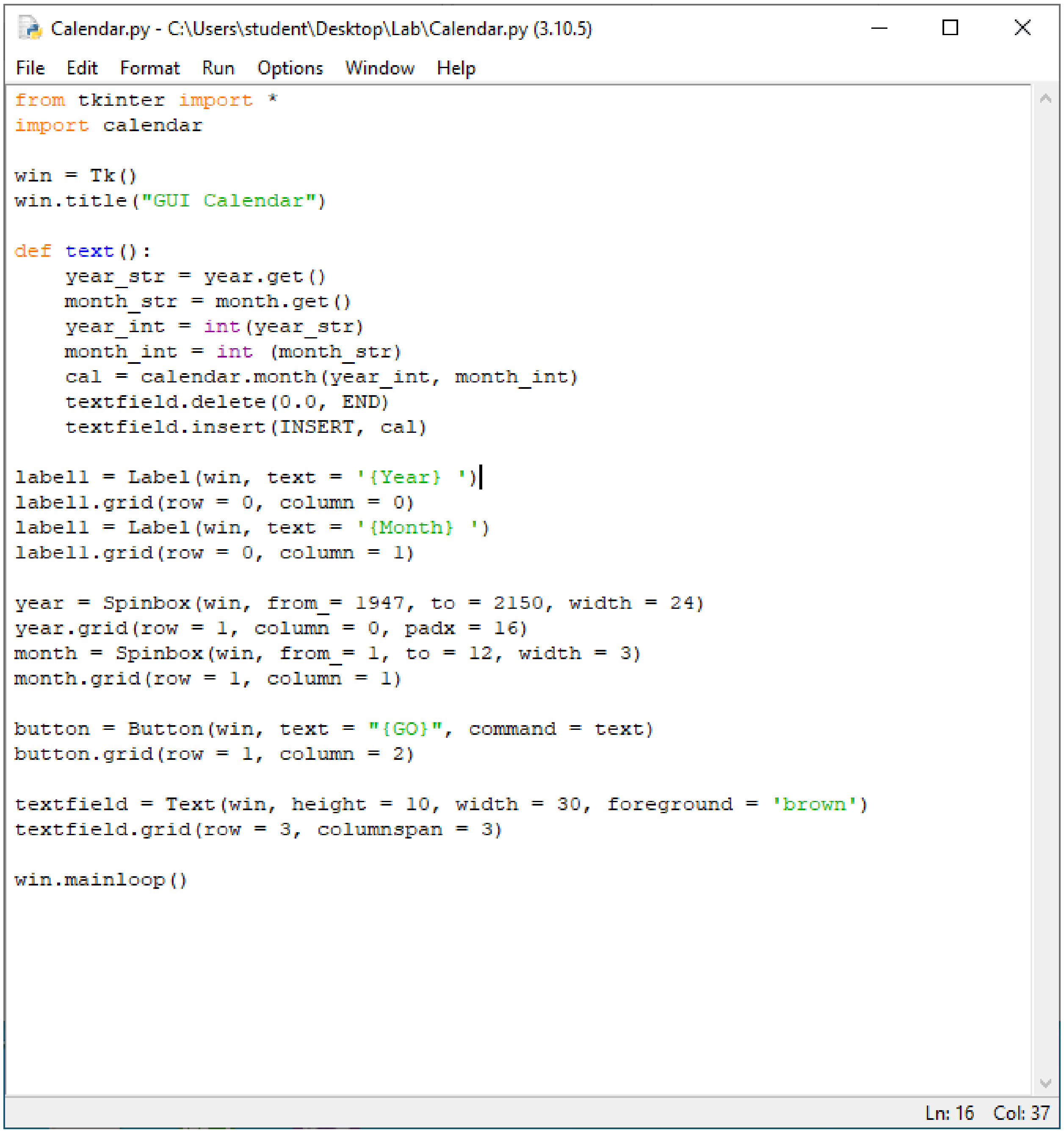
Upper Case is used for variables that should be constant. This convention is not followed by the particular name.



```
CalculatorGUI.py:32:4: W0601: Global variable 'num1' undefined at the module level (global-variable-undefined)
CalculatorGUI.py:32:4: W0601: Global variable 'math' undefined at the module level (global-variable-undefined)
```

It needs a line at the top-level of the module (outside of any function) defining the variable which is not present in this file

Module 3: Calendar.py

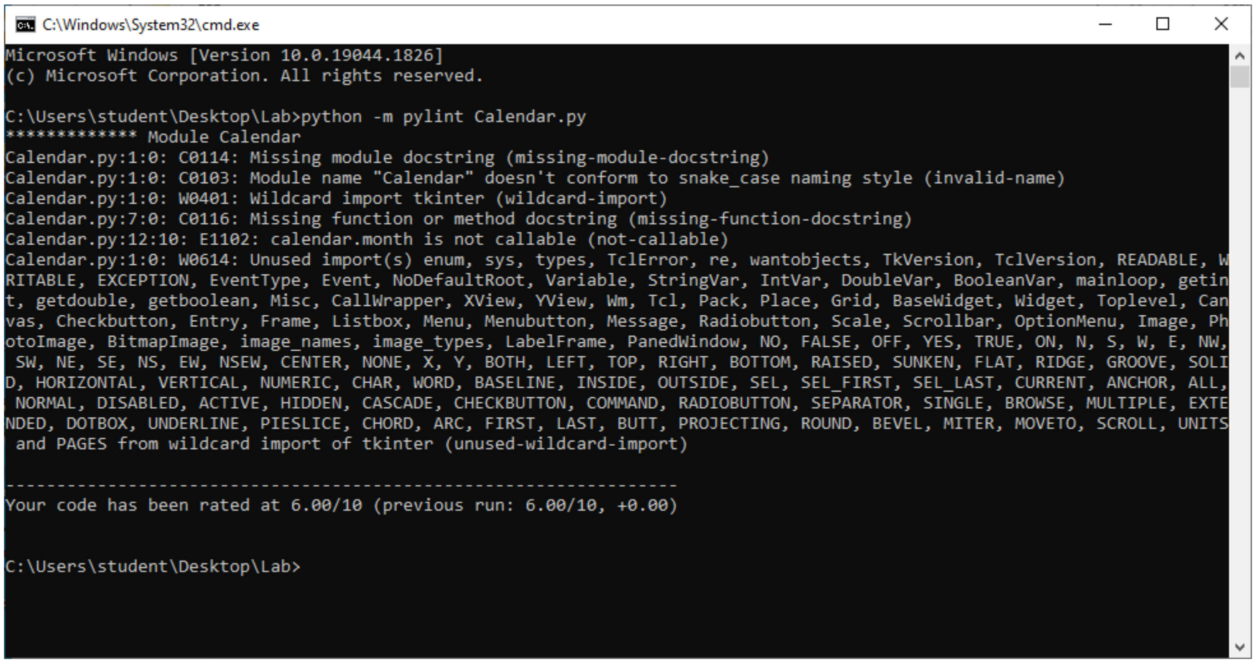

Calendar.py - C:\Users\student\Desktop\Lab\Calendar.py (3.10.5)

File Edit Format Run Options Window Help

```python
from tkinter import *
import calendar

win = Tk()
win.title("GUI Calendar")

def text():
    year_str = year.get()
    month_str = month.get()
    year_int = int(year_str)
    month_int = int (month_str)
    cal = calendar.month(year_int, month_int)
    textfield.delete(0.0, END)
    textfield.insert(INSERT, cal)

label1 = Label(win, text = '{Year} ')
label1.grid(row = 0, column = 0)
label1 = Label(win, text = '{Month} ')
label1.grid(row = 0, column = 1)

year = Spinbox(win, from_= 1947, to = 2150, width = 24)
year.grid(row = 1, column = 0, padx = 16)
month = Spinbox(win, from_= 1, to = 12, width = 3)
month.grid(row = 1, column = 1)

button = Button(win, text = "{GO}", command = text)
button.grid(row = 1, column = 2)

textfield = Text(win, height = 10, width = 30, foreground = 'brown')
textfield.grid(row = 3, columnspan = 3)

win.mainloop()
```

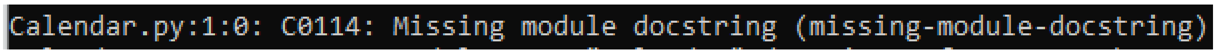
Ln: 16 Col: 37

Analysis:

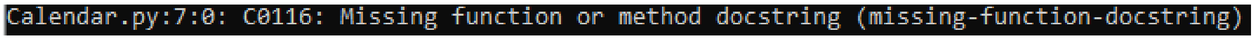


```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\student\Desktop\Lab>python -m pylint Calendar.py
************* Module Calendar
Calendar.py:1:0: C0114: Missing module docstring (missing-module-docstring)
Calendar.py:1:0: C0103: Module name "Calendar" doesn't conform to snake_case naming style (invalid-name)
Calendar.py:1:0: W0401: Wildcard import tkinter (wildcard-import)
Calendar.py:7:0: C0116: Missing function or method docstring (missing-function-docstring)
Calendar.py:12:10: E1102: calendar.month is not callable (not-callable)
Calendar.py:1:0: W0614: Unused import(s) enum, sys, types, TclError, re, wantobjects, TkVersion, TclVersion, READABLE, W
RITABLE, EXCEPTION, EventType, Event, NoDefaultRoot, Variable, StringVar, IntVar, DoubleVar, BooleanVar, mainloop, getin
t, getdouble, getboolean, Misc, CallWrapper, XView, YView, Wm, Tcl, Pack, Place, Grid, BaseWidget, Widget, Toplevel, Can
vas, Checkbutton, Entry, Frame, Listbox, Menu, Menubutton, Message, Radiobutton, Scale, Scrollbar, OptionMenu, Image, Ph
otoImage, BitmapImage, image_names, image_types, LabelFrame, PanedWindow, NO, FALSE, OFF, YES, TRUE, ON, N, S, W, E, NW,
 SW, NE, SE, NS, EW, NSEW, CENTER, NONE, X, Y, BOTH, LEFT, TOP, RIGHT, BOTTOM, RAISED, SUNKEN, FLAT, RIDGE, GROOVE, SOLI
D, HORIZONTAL, VERTICAL, NUMERIC, CHAR, WORD, BASELINE, INSIDE, OUTSIDE, SEL, SEL_FIRST, SEL_LAST, CURRENT, ANCHOR, ALL,
 NORMAL, DISABLED, ACTIVE, HIDDEN, CASCADE, CHECKBUTTON, COMMAND, RADIOBUTTON, SEPARATOR, SINGLE, BROWSE, MULTIPLE, EXTE
NDED, DOTBOX, UNDERLINE, PIESLICE, CHORD, ARC, FIRST, LAST, BUTT, PROJECTING, ROUND, BEVEL, MITER, MOVETO, SCROLL, UNITS
 and PAGES from wildcard import of tkinter (unused-wildcard-import)

------------------------------------------------------------------
Your code has been rated at 6.00/10 (previous run: 6.00/10, +0.00)


C:\Users\student\Desktop\Lab>
```

Errors:

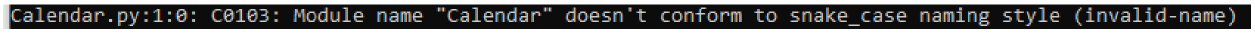


```
Calendar.py:1:0: C0114: Missing module docstring (missing-module-docstring)
```



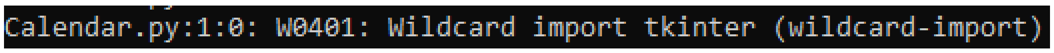
```
Calendar.py:7:0: C0116: Missing function or method docstring (missing-function-docstring)
```

Python documentation strings (or docstrings) provide a convenient way of associating documentation with Python modules, functions, classes, and methods and this module or function has no docstring at the starting of the file



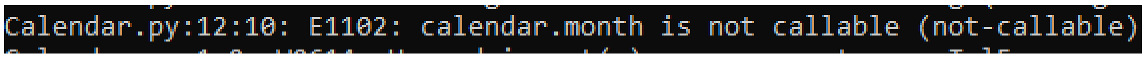
```
Calendar.py:1:0: C0103: Module name "Calendar" doesn't conform to snake_case naming style (invalid-name)
```

Snake case style of writing in which each space is replaced with an underscore (_) character, and the first letter of each word is written in lowercase. This convention is not followed by the particular name.



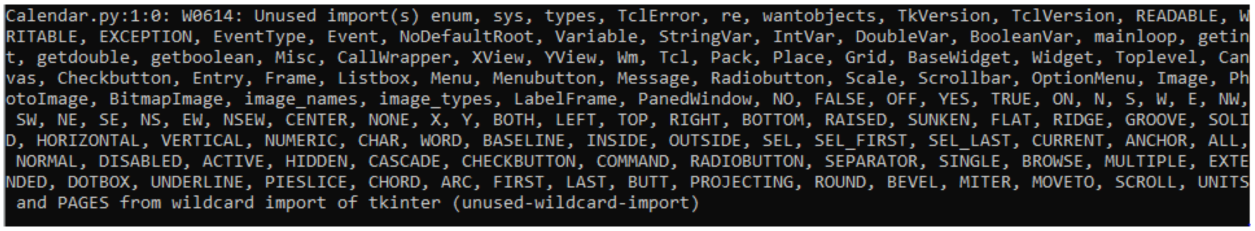
```
Calendar.py:1:0: W0401: Wildcard import tkinter (wildcard-import)
```

Used when `from module import *` is detected. This is a bad practice because it clutters namespace with unneeded modules, packages, variables, etc. Moreover, it takes time to load them too.



```
Calendar.py:12:10: E1102: calendar.month is not callable (not-callable)
```

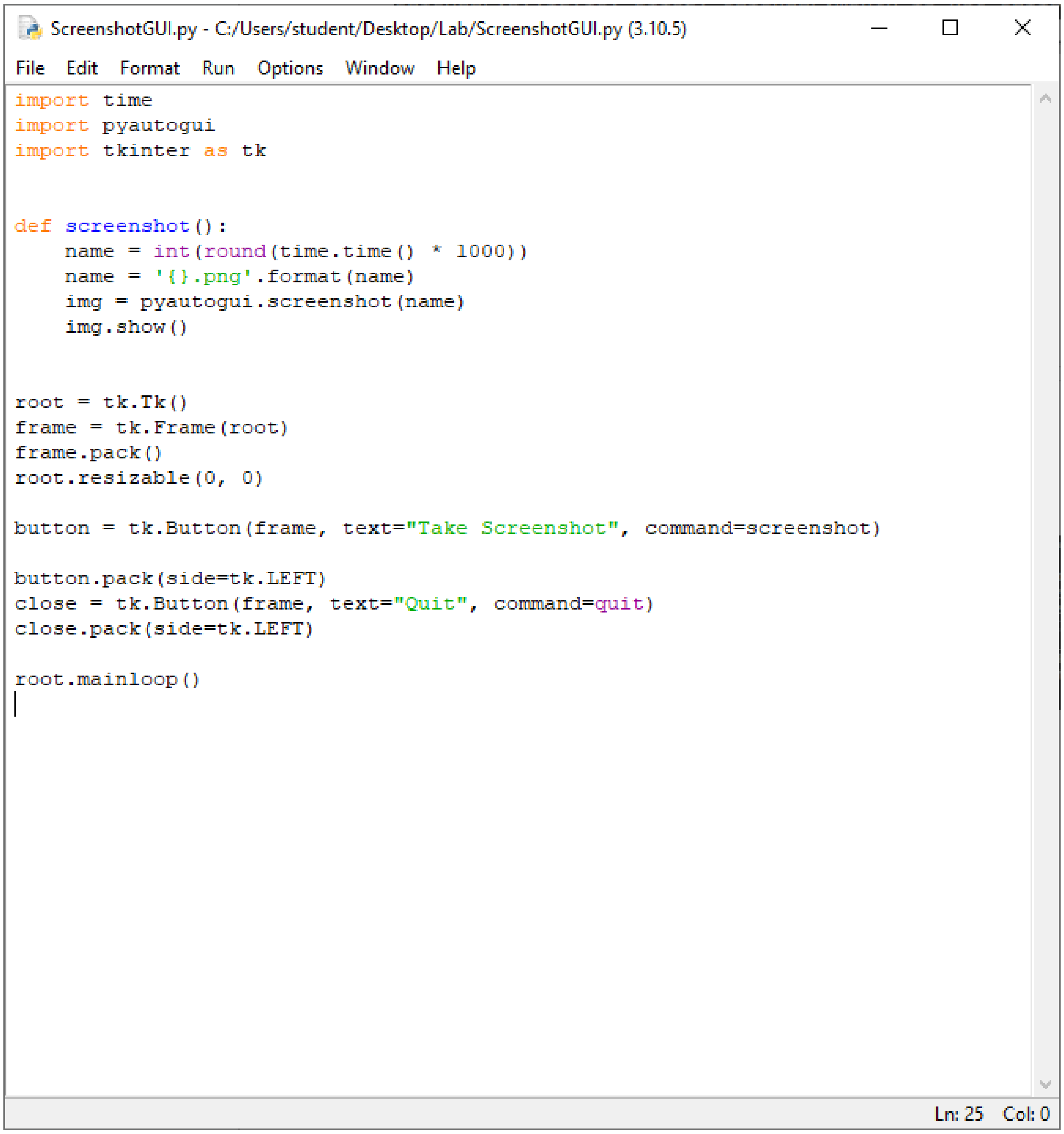
Used when an object being called has been inferred to a non callable object.



```
Calendar.py:1:0: W0614: Unused import(s) enum, sys, types, TclError, re, wantobjects, TkVersion, TclVersion, READABLE, W
RITABLE, EXCEPTION, EventType, Event, NoDefaultRoot, Variable, StringVar, IntVar, DoubleVar, BooleanVar, mainloop, getin
t, getdouble, getboolean, Misc, CallWrapper, XView, YView, Wm, Tcl, Pack, Place, Grid, BaseWidget, Widget, Toplevel, Can
vas, Checkbutton, Entry, Frame, Listbox, Menu, Menubutton, Message, Radiobutton, Scale, Scrollbar, OptionMenu, Image, Ph
otoImage, BitmapImage, image_names, image_types, LabelFrame, PanedWindow, NO, FALSE, OFF, YES, TRUE, ON, N, S, W, E, NW,
 SW, NE, SE, NS, EW, NSEW, CENTER, NONE, X, Y, BOTH, LEFT, TOP, RIGHT, BOTTOM, RAISED, SUNKEN, FLAT, RIDGE, GROOVE, SOLI
D, HORIZONTAL, VERTICAL, NUMERIC, CHAR, WORD, BASELINE, INSIDE, OUTSIDE, SEL, SEL_FIRST, SEL_LAST, CURRENT, ANCHOR, ALL,
 NORMAL, DISABLED, ACTIVE, HIDDEN, CASCADE, CHECKBUTTON, COMMAND, RADIOBUTTON, SEPARATOR, SINGLE, BROWSE, MULTIPLE, EXTE
NDED, DOTBOX, UNDERLINE, PIESLICE, CHORD, ARC, FIRST, LAST, BUTT, PROJECTING, ROUND, BEVEL, MITER, MOVETO, SCROLL, UNITS
 and PAGES from wildcard import of tkinter (unused-wildcard-import)
```

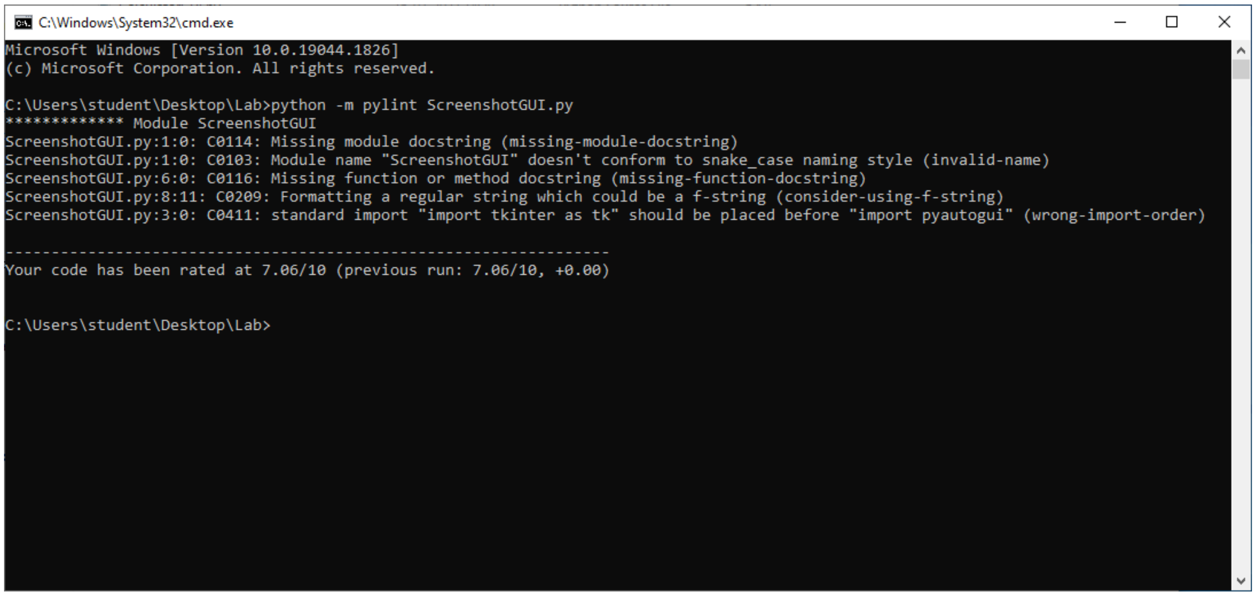
Used when an imported module or variable is not used.

Module 4: ScreenshotGUI.py


ScreenshotGUI.py - C:/Users/student/Desktop/Lab/ScreenshotGUI.py (3.10.5)
File Edit Format Run Options Window Help
Ln: 25 Col: 0

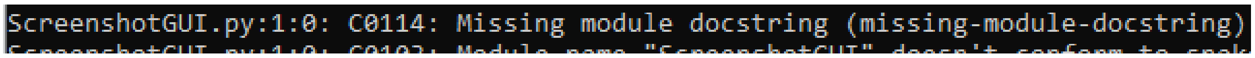
```python
import time
import pyautogui
import tkinter as tk


def screenshot():
    name = int(round(time.time() * 1000))
    name = '{}.png'.format(name)
    img = pyautogui.screenshot(name)
    img.show()


root = tk.Tk()
frame = tk.Frame(root)
frame.pack()
root.resizable(0, 0)

button = tk.Button(frame, text="Take Screenshot", command=screenshot)

button.pack(side=tk.LEFT)
close = tk.Button(frame, text="Quit", command=quit)
close.pack(side=tk.LEFT)

root.mainloop()
```

Analysis:

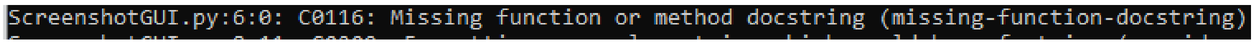


```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\student\Desktop\Lab>python -m pylint ScreenshotGUI.py
************* Module ScreenshotGUI
ScreenshotGUI.py:1:0: C0114: Missing module docstring (missing-module-docstring)
ScreenshotGUI.py:1:0: C0103: Module name "ScreenshotGUI" doesn't conform to snake_case naming style (invalid-name)
ScreenshotGUI.py:6:0: C0116: Missing function or method docstring (missing-function-docstring)
ScreenshotGUI.py:8:11: C0209: Formatting a regular string which could be a f-string (consider-using-f-string)
ScreenshotGUI.py:3:0: C0411: standard import "import tkinter as tk" should be placed before "import pyautogui" (wrong-import-order)

------------------------------------------------------------------
Your code has been rated at 7.06/10 (previous run: 7.06/10, +0.00)


C:\Users\student\Desktop\Lab>
```

Analysis:

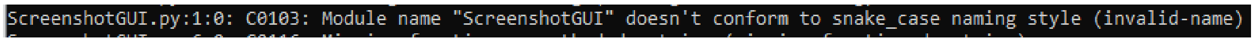


```
ScreenshotGUI.py:1:0: C0114: Missing module docstring (missing-module-docstring)
```



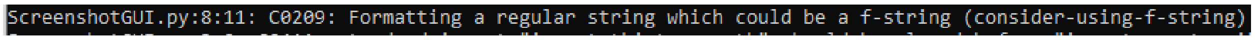
```
ScreenshotGUI.py:6:0: C0116: Missing function or method docstring (missing-function-docstring)
```

Python documentation strings (or docstrings) provide a convenient way of associating documentation with Python modules, functions, classes, and methods and this module or function has no docstring at the starting of the file



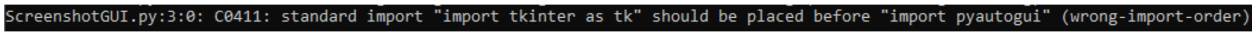
```
ScreenshotGUI.py:1:0: C0103: Module name "ScreenshotGUI" doesn't conform to snake_case naming style (invalid-name)
```

Snake case style of writing in which each space is replaced with an underscore (_) character, and the first letter of each word is written in lowercase. This convention is not followed by the particular name.



```
ScreenshotGUI.py:8:11: C0209: Formatting a regular string which could be a f-string (consider-using-f-string)
```

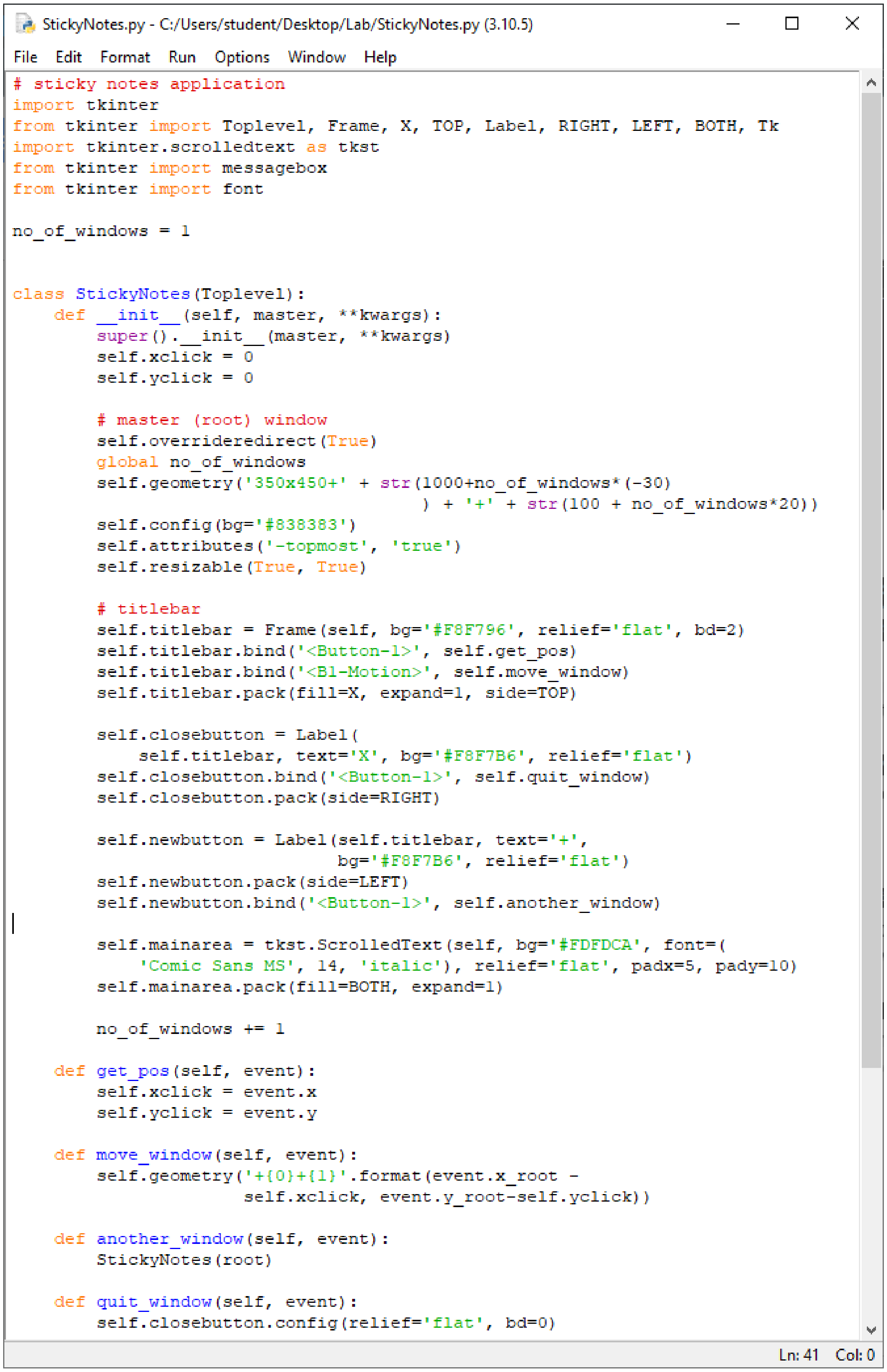
The use of f-strings is preferred in python, and this error is indicated when we detect a string that is being formatted with format() or % which could potentially be a f-string.



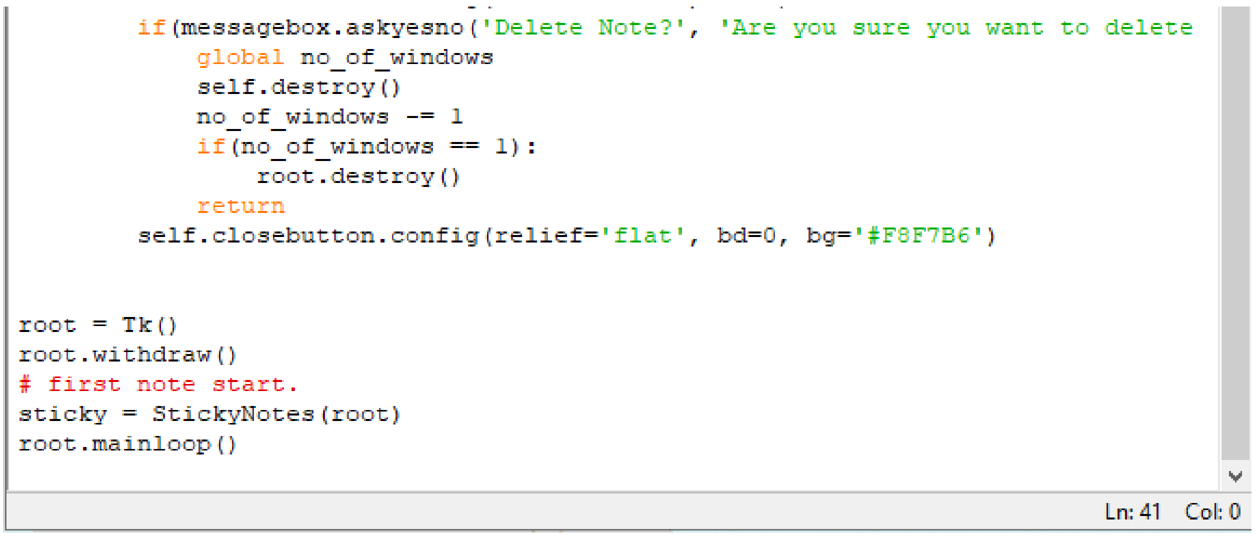
```
ScreenshotGUI.py:3:0: C0411: standard import "import tkinter as tk" should be placed before "import pyautogui" (wrong-import-order)
```

It is warned when PEP8 import order is not respected, which is standard imports first, then third-party libraries, then local imports

Module 5: StickyNotes.py


StickyNotes.py - C:/Users/student/Desktop/Lab/StickyNotes.py (3.10.5)
File Edit Format Run Options Window Help

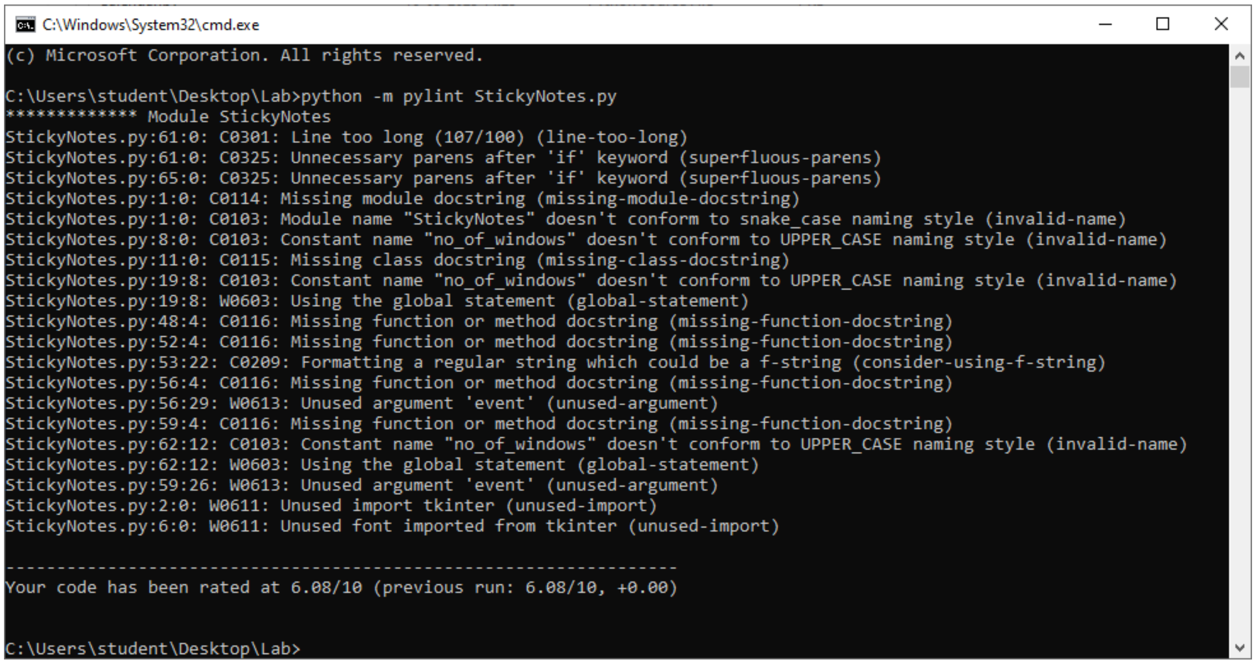
```python
# sticky notes application
import tkinter
from tkinter import Toplevel, Frame, X, TOP, Label, RIGHT, LEFT, BOTH, Tk
import tkinter.scrolledtext as tkst
from tkinter import messagebox
from tkinter import font

no_of_windows = 1


class StickyNotes(Toplevel):
    def __init__(self, master, **kwargs):
        super().__init__(master, **kwargs)
        self.xclick = 0
        self.yclick = 0

        # master (root) window
        self.overrideredirect(True)
        global no_of_windows
        self.geometry('350x450+' + str(1000+no_of_windows*(-30)
                                       ) + '+' + str(100 + no_of_windows*20))
        self.config(bg='#838383')
        self.attributes('-topmost', 'true')
        self.resizable(True, True)

        # titlebar
        self.titlebar = Frame(self, bg='#F8F796', relief='flat', bd=2)
        self.titlebar.bind('<Button-1>', self.get_pos)
        self.titlebar.bind('<B1-Motion>', self.move_window)
        self.titlebar.pack(fill=X, expand=1, side=TOP)

        self.closebutton = Label(
            self.titlebar, text='X', bg='#F8F7B6', relief='flat')
        self.closebutton.bind('<Button-1>', self.quit_window)
        self.closebutton.pack(side=RIGHT)

        self.newbutton = Label(self.titlebar, text='+',
                               bg='#F8F7B6', relief='flat')
        self.newbutton.pack(side=LEFT)
        self.newbutton.bind('<Button-1>', self.another_window)

        self.mainarea = tkst.ScrolledText(self, bg='#FDFDCA', font=(
            'Comic Sans MS', 14, 'italic'), relief='flat', padx=5, pady=10)
        self.mainarea.pack(fill=BOTH, expand=1)

        no_of_windows += 1

    def get_pos(self, event):
        self.xclick = event.x
        self.yclick = event.y

    def move_window(self, event):
        self.geometry('+{0}+{1}'.format(event.x_root -
                      self.xclick, event.y_root-self.yclick))

    def another_window(self, event):
        StickyNotes(root)

    def quit_window(self, event):
        self.closebutton.config(relief='flat', bd=0)
```



```
        if(messagebox.askyesno('Delete Note?', 'Are you sure you want to delete
            global no_of_windows
            self.destroy()
            no_of_windows -= 1
            if(no_of_windows == 1):
                root.destroy()
            return
        self.closebutton.config(relief='flat', bd=0, bg='#F8F7B6')


root = Tk()
root.withdraw()
# first note start.
sticky = StickyNotes(root)
root.mainloop()
```

Ln: 41 Col: 0

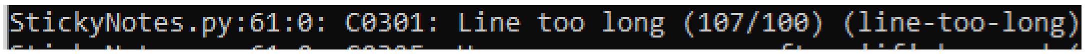
Analysis:



C:\Windows\System32\cmd.exe

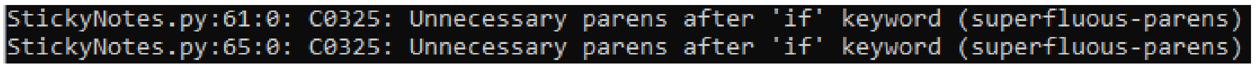
```
(c) Microsoft Corporation. All rights reserved.

C:\Users\student\Desktop\Lab>python -m pylint StickyNotes.py
************* Module StickyNotes
StickyNotes.py:61:0: C0301: Line too long (107/100) (line-too-long)
StickyNotes.py:61:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
StickyNotes.py:65:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
StickyNotes.py:1:0: C0114: Missing module docstring (missing-module-docstring)
StickyNotes.py:1:0: C0103: Module name "StickyNotes" doesn't conform to snake_case naming style (invalid-name)
StickyNotes.py:8:0: C0103: Constant name "no_of_windows" doesn't conform to UPPER_CASE naming style (invalid-name)
StickyNotes.py:11:0: C0115: Missing class docstring (missing-class-docstring)
StickyNotes.py:19:8: C0103: Constant name "no_of_windows" doesn't conform to UPPER_CASE naming style (invalid-name)
StickyNotes.py:19:8: W0603: Using the global statement (global-statement)
StickyNotes.py:48:4: C0116: Missing function or method docstring (missing-function-docstring)
StickyNotes.py:52:4: C0116: Missing function or method docstring (missing-function-docstring)
StickyNotes.py:53:22: C0209: Formatting a regular string which could be a f-string (consider-using-f-string)
StickyNotes.py:56:4: C0116: Missing function or method docstring (missing-function-docstring)
StickyNotes.py:56:29: W0613: Unused argument 'event' (unused-argument)
StickyNotes.py:59:4: C0116: Missing function or method docstring (missing-function-docstring)
StickyNotes.py:62:12: C0103: Constant name "no_of_windows" doesn't conform to UPPER_CASE naming style (invalid-name)
StickyNotes.py:62:12: W0603: Using the global statement (global-statement)
StickyNotes.py:59:26: W0613: Unused argument 'event' (unused-argument)
StickyNotes.py:2:0: W0611: Unused import tkinter (unused-import)
StickyNotes.py:6:0: W0611: Unused font imported from tkinter (unused-import)

------------------------------------------------------------------
Your code has been rated at 6.08/10 (previous run: 6.08/10, +0.00)


C:\Users\student\Desktop\Lab>
```

Errors:



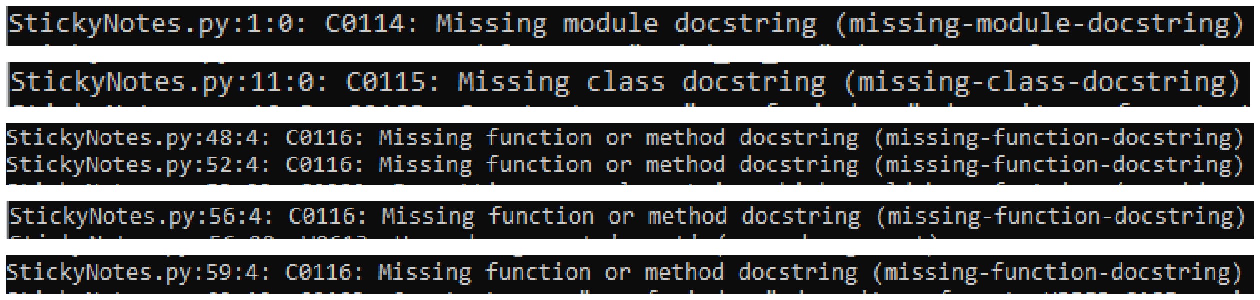
```
StickyNotes.py:61:0: C0301: Line too long (107/100) (line-too-long)
```

The line is longer than a given number of characters.



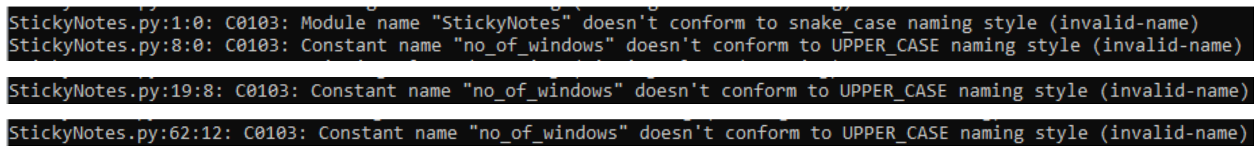
```
StickyNotes.py:61:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
StickyNotes.py:65:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
```

Used when a single item in parentheses follows an if, for, or other keyword or the use of parenthesis is not required



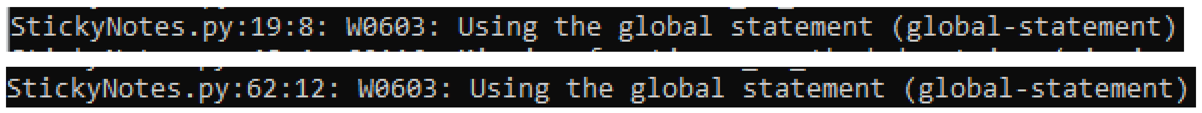
```
StickyNotes.py:1:0: C0114: Missing module docstring (missing-module-docstring)
StickyNotes.py:11:0: C0115: Missing class docstring (missing-class-docstring)
StickyNotes.py:48:4: C0116: Missing function or method docstring (missing-function-docstring)
StickyNotes.py:52:4: C0116: Missing function or method docstring (missing-function-docstring)
StickyNotes.py:56:4: C0116: Missing function or method docstring (missing-function-docstring)
StickyNotes.py:59:4: C0116: Missing function or method docstring (missing-function-docstring)
```

Python documentation strings (or docstrings) provide a convenient way of associating documentation with Python modules, functions, classes, and methods and this module or function or class has no docstring at the starting of the file



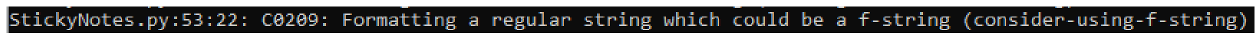
```
StickyNotes.py:1:0: C0103: Module name "StickyNotes" doesn't conform to snake_case naming style (invalid-name)
StickyNotes.py:8:0: C0103: Constant name "no_of_windows" doesn't conform to UPPER_CASE naming style (invalid-name)
StickyNotes.py:19:8: C0103: Constant name "no_of_windows" doesn't conform to UPPER_CASE naming style (invalid-name)
StickyNotes.py:62:12: C0103: Constant name "no_of_windows" doesn't conform to UPPER_CASE naming style (invalid-name)
```

Snake case style of writing in which each space is replaced with an underscore (_) character, and the first letter of each word is written in lowercase. This convention is not followed by the particular name.



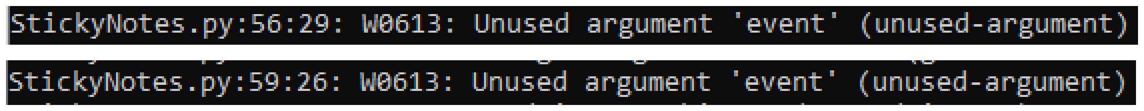
```
StickyNotes.py:19:8: W0603: Using the global statement (global-statement)
StickyNotes.py:62:12: W0603: Using the global statement (global-statement)
```

Used when you use the "global" statement to update a global variable. Pylint discourages its usage.



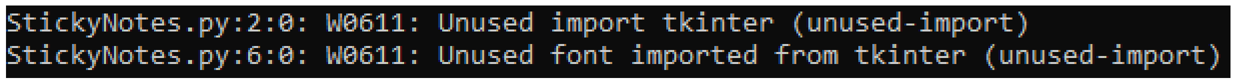
```
StickyNotes.py:53:22: C0209: Formatting a regular string which could be a f-string (consider-using-f-string)
```

The use of f-strings is preferred in python, and this error is indicated when we detect a string that is being formatted with format() or % which could potentially be a f-string.



```
StickyNotes.py:56:29: W0613: Unused argument 'event' (unused-argument)
StickyNotes.py:59:26: W0613: Unused argument 'event' (unused-argument)
```

Here, a function or method argument is not used.



```
StickyNotes.py:2:0: W0611: Unused import tkinter (unused-import)
StickyNotes.py:6:0: W0611: Unused font imported from tkinter (unused-import)
```

Here, imported module or variable is not used.