IT314:Software Engineering
Lab: 5(Static Analysis)

Name: Vivek Godhasara

Id: 202001451

Date: 24/03/2023

QUESTION

Static Analysis:
Static analysis is a method of examining the source code of a software program without executing it. Static analysis can help detect errors, bugs, vulnerabilities, and other quality issues in the code. Static analysis tools can perform various tasks such as checking syntax, style, logic, data flow, control flow, and security. Static analysis can improve the reliability, performance, and maintainability of software by identifying and correcting defects early in the development process.

Static Analysis Tools:
Static analysis tools are software tools that analyze the source code of a program without executing it. They can help developers find and fix errors, bugs,Vulnerabilities, code smells, and other quality issues in their code. Static analysis tools can also measure various metrics of the code, such as complexity, readability, maintainability, test coverage, and documentation. Static analysis tools can be integrated into the development process as part of the code editor, the version control system, or the continuous integration pipeline. Some examples of static analysis tools are SonarQube, PMD, ESLint, and Pylint.

List of tools:
Python:
- Mypy
- Pylint
- Pyflakes
- Pycodestyle (pep8)
- Flake8
- Prospector
- Bandit

Java:
- FindBugs
- PMD
- Checkstyle
- Error Prone
- Spoon
- Spotbugs

Goal : Select the tool of your choice. Select a git repository, use the selected tool and analyze the files from the selected repository. Submit the tool output and understanding of the errors.

CODE 1:

This is the code of loading the cifar 10 database for doing analysis.This code is in python programming language.

```
import numpy as np
import matplotlib.pyplot as plt
import pickle

"""
The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes,
with 6000 images per class. There are 50000
training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each
with 10000 images. The test batch contains
exactly 1000 randomly-selected images from each class. The training
batches contain the remaining images in random
order, but some training batches may contain more images from one class
than another. Between them, the training
batches contain exactly 5000 images from each class.
"""


def unpickle(file):
    """load the cifar-10 data"""

    with open(file, 'rb') as fo:
        data = pickle.load(fo, encoding='bytes')
    return data


def load_cifar_10_data(data_dir, negatives=False):
    """
    Return train_data, train_filenames, train_labels, test_data,
test_filenames, test_labels
    """

    # get the meta_data_dict
    # num_cases_per_batch: 1000
```

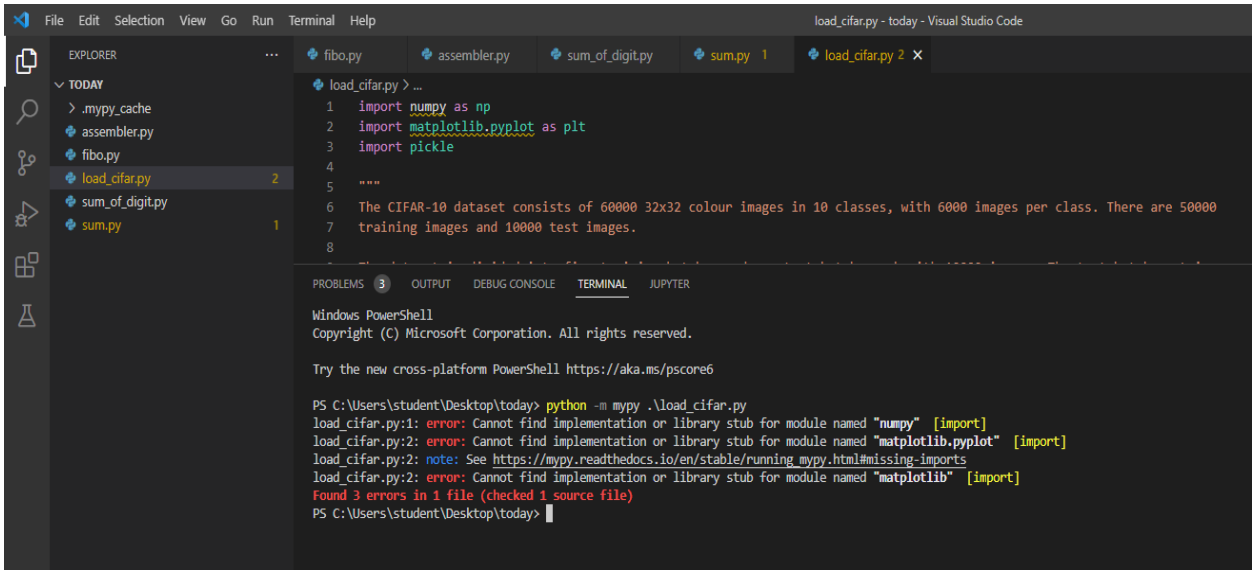
```
    # label_names: ['airplane', 'automobile', 'bird', 'cat', 'deer',
'dog', 'frog', 'horse', 'ship', 'truck']
    # num_vis: :3072

    meta_data_dict = unpickle(data_dir + "/batches.meta")
    cifar_label_names = meta_data_dict[b'label_names']
    cifar_label_names = np.array(cifar_label_names)

    # training data
    cifar_train_data = None
    cifar_train_filenames = []
    cifar_train_labels = []

    # cifar_train_data_dict
    # 'batch_label': 'training batch 5 of 5'
    # 'data': ndarray
    # 'filenames': list
    # 'labels': list

    for i in range(1, 6):
        cifar_train_data_dict = unpickle(data_dir +
"/data_batch_{}".format(i))
        if i == 1:
            cifar_train_data = cifar_train_data_dict[b'data']
        else:
            cifar_train_data = np.vstack((cifar_train_data,
cifar_train_data_dict[b'data']))
        cifar_train_filenames += cifar_train_data_dict[b'filenames']
        cifar_train_labels += cifar_train_data_dict[b'labels']

    cifar_train_data = cifar_train_data.reshape((len(cifar_train_data), 3,
32, 32))
    if negatives:
        cifar_train_data = cifar_train_data.transpose(0, 2, 3,
1).astype(np.float32)
    else:
        cifar_train_data = np.rollaxis(cifar_train_data, 1, 4)
    cifar_train_filenames = np.array(cifar_train_filenames)
    cifar_train_labels = np.array(cifar_train_labels)
```

```
    # test data
    # cifar_test_data_dict
    # 'batch_label': 'testing batch 1 of 1'
    # 'data': ndarray
    # 'filenames': list
    # 'labels': list

    cifar_test_data_dict = unpickle(data_dir + "/test_batch")
    cifar_test_data = cifar_test_data_dict[b'data']
    cifar_test_filenames = cifar_test_data_dict[b'filenames']
    cifar_test_labels = cifar_test_data_dict[b'labels']

    cifar_test_data = cifar_test_data.reshape((len(cifar_test_data), 3,
32, 32))
    if negatives:
        cifar_test_data = cifar_test_data.transpose(0, 2, 3,
1).astype(np.float32)
    else:
        cifar_test_data = np.rollaxis(cifar_test_data, 1, 4)
    cifar_test_filenames = np.array(cifar_test_filenames)
    cifar_test_labels = np.array(cifar_test_labels)

    return cifar_train_data, cifar_train_filenames, cifar_train_labels, \
        cifar_test_data, cifar_test_filenames, cifar_test_labels,
cifar_label_names


if __name__ == "__main__":
    """show it works"""

    cifar_10_dir = 'cifar-10-batches-py'

    train_data, train_filenames, train_labels, test_data, test_filenames,
test_labels, label_names = \
        load_cifar_10_data(cifar_10_dir)

    print("Train data: ", train_data.shape)
    print("Train filenames: ", train_filenames.shape)
    print("Train labels: ", train_labels.shape)
    print("Test data: ", test_data.shape)
```

```
    print("Test filenames: ", test_filenames.shape)
    print("Test labels: ", test_labels.shape)
    print("Label names: ", label_names.shape)

    # Don't forget that the label_names and filesnames are in binary and
need conversion if used.

    # display some random training images in a 25x25 grid
    num_plot = 5
    f, ax = plt.subplots(num_plot, num_plot)
    for m in range(num_plot):
        for n in range(num_plot):
            idx = np.random.randint(0, train_data.shape[0])
            ax[m, n].imshow(train_data[idx])
            ax[m, n].get_xaxis().set_visible(False)
            ax[m, n].get_yaxis().set_visible(False)
    f.subplots_adjust(hspace=0.1)
    f.subplots_adjust(wspace=0)
    plt.show()
```

Error:

Here we will try to fetch the errors using previously downloaded mypy static analysis tool.

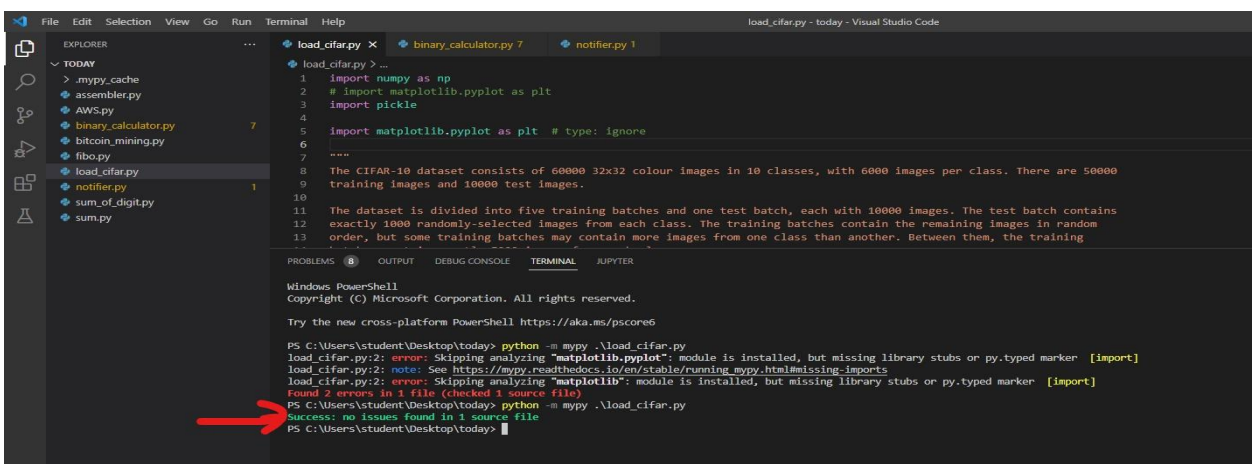


Upper Error message is suggesting that "matplotlib.pyplot" module is missing. Specifically, the error indicates that the import statement for "matplotlib.pyplot" is being skipped during the analysis process because the module is installed but missing library stubs or a "py.typed" marker.

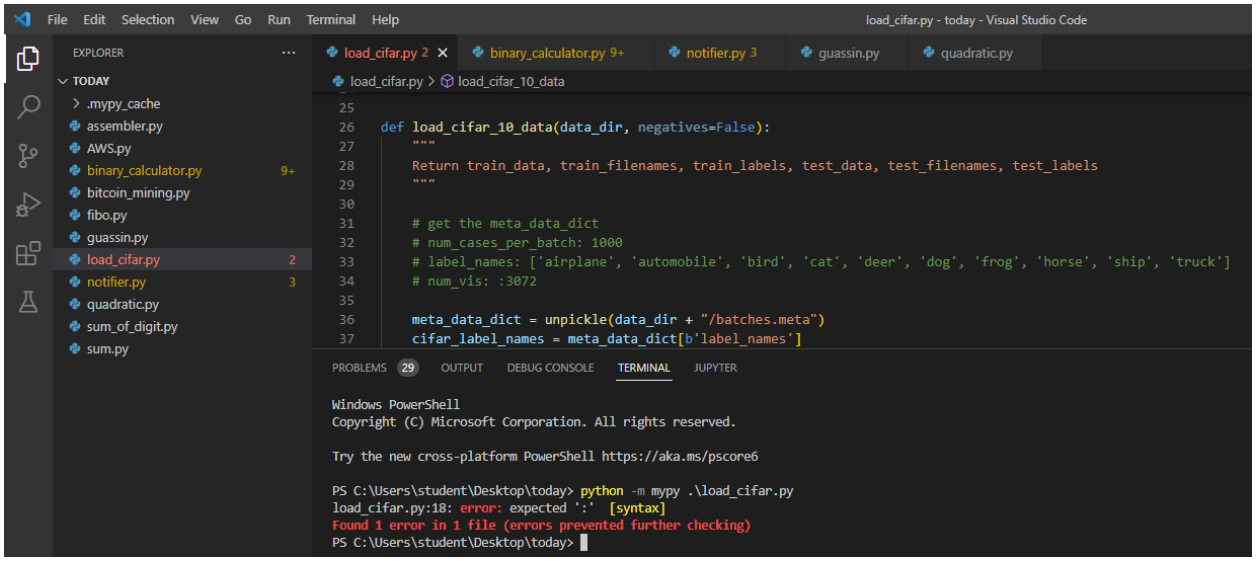
To fix this error, we have to install missing library stubs or a "py.typed" marker.Or, you could suppress the error by adding a comment to the import statement like this:

```
import matplotlib.pyplot as plt  # type: ignore
```

Now There is no erro left:



Detecting the Syntax Error.



CODE 2:

This is the code for a binary calculator..This code is in python programming language.

```
from tkinter import *
window = Tk()
window.title("Standard Binary Calculator")
window.resizable(0, 0)


def f1():
    s = e1_val.get()
    e1.delete(first=0, last=len(s))


def f2():
    s = e1_val.get()
    e1.insert(END, "1")


def f3():
    s = e1_val.get()
```

```
    e1.insert(END, "0")


def f4():
    x = 0
    s = e1_val.get()
    for i in range(0, len(s)):
        if s[i] == '/' or s[i] == 'X' or s[i] == '+' or s[i] == '-':
            a = s[0:i]
            b = s[i + 1:len(s)]
            if s[i] == '-':
                x = sub(int(a), int(b))
            elif s[i] == '/':
                x = int(int(a) / int(b))
            elif s[i] == 'X':
                x = int(int(a) * int(b))
            elif s[i] == '+':
                x = int(add(int(a), int(b)))

    e1.delete(first=0, last=len(s))
    e1.insert(END, "")
    e1.insert(END, str(x))


def bin_to_dec(n):
    num = n
    dec_value = 0
    base = 1
    temp = num
    while (temp):
        last_digit = temp % 10
        temp = int(temp / 10)

        dec_value += last_digit * base
        base = base * 2
    return dec_value


def ad(x, y):
    a = bin_to_dec(x)
```

```
    b = bin_to_dec(y)
    c = a + b
    d = bin(c).replace("0b", "")
    return d


def sub(x, y):
    a = bin_to_dec(x)
    b = bin_to_dec(y)
    c = a - b
    d = bin(c).replace("0b", "")
    return d


def f5():
    x = 0
    s = e1_val.get()
    flag = 1
    for i in range(0, len(s)):
        if s[i] == '/' or s[i] == 'X' or s[i] == '+' or s[i] == '-':
            flag = 0
            a = s[0:i]
            b = s[i + 1:len(s)]
            if s[i] == '-':
                x = sub(int(a), int(b))
            elif s[i] == '/':
                x = int(int(a) / int(b))
            elif s[i] == 'X':
                x = int(int(a) * int(b))
            elif s[i] == '+':
                x = int(add(int(a), int(b)))
    if flag == 0:
        e1.delete(first=0, last=len(s))
        e1.insert(END, str(x))
    e1.insert(END, "+")


def f():
    x = 0
    s = e1_val.get()
```

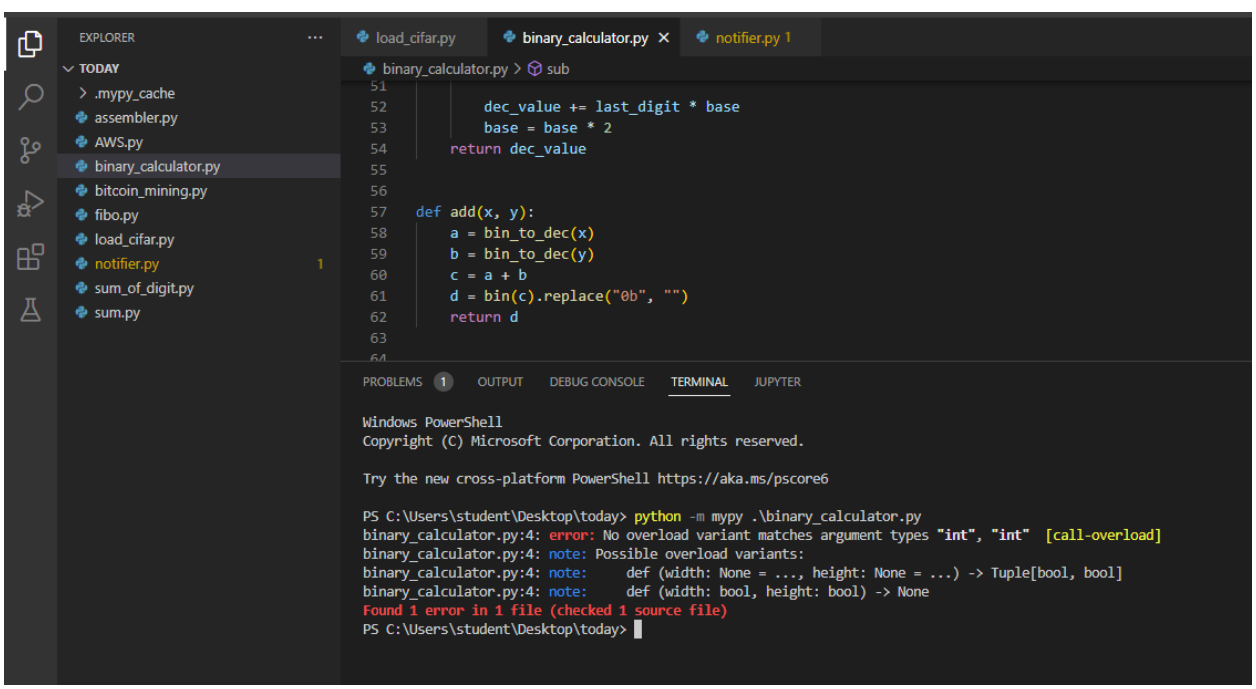
```
    flag = 1
    for i in range(0, len(s)):
        if s[i] == '/' or s[i] == 'X' or s[i] == '+' or s[i] == '-':
            flag = 0
            a = s[0:i]
            b = s[i + 1:len(s)]
            if s[i] == '-':
                x = sub(int(a), int(b))
            elif s[i] == '/':
                x = int(int(a) / int(b))
            elif s[i] == 'X':
                x = int(int(a) * int(b))
            elif s[i] == '+':
                x = int(add(int(a), int(b)))
    if flag == 0:
        e1.delete(first=0, last=len(s))
        e1.insert(END, str(x))
    e1.insert(END, "-")


def f7():
    x = 0
    s = e1_val.get()
    flag = 1
    for i in range(0, len(s)):
        if s[i] == '/' or s[i] == 'X' or s[i] == '+' or s[i] == '-':
            flag = 0
            a = s[0:i]
            b = s[i + 1:len(s)]
            if s[i] == '-':
                x = sub(int(a), int(b))
            elif s[i] == '/':
                x = int(int(a) / int(b))
            elif s[i] == 'X':
                x = int(int(a) * int(b))
            elif s[i] == '+':
                x = int(add(int(a), int(b)))
    if flag == 0:
        e1.delete(first=0, last=len(s))
        e1.insert(END, str(x))
```

```
    e1.insert(END, "/")


def f8():
    x = 0
    s = e1_val.get()
    flag = 1
    for i in range(0, len(s)):
        if s[i] == '/' or s[i] == 'X' or s[i] == '+' or s[i] == '-':
            flag = 0
            a = s[0:i]
            b = s[i + 1:len(s)]
            if s[i] == '-':
                x = sub(int(a), int(b))
            elif s[i] == '/':
                x = int(int(a) / int(b))
            elif s[i] == 'X':
                x = int(int(a) * int(b))
            elif s[i] == '+':
                x = int(add(int(a), int(b)))
    if flag == 0:
        e1.delete(first=0, last=len(s))
        e1.insert(END, str(x))
    e1.insert(END, "X")


e1_val = StringVar()
e1 = Entry(window, textvariable=e1_val, width=50)
e1.grid(row=0, column=0, columnspan=4)

b1 = Button(windo, text="1", width=8, height=2, command=f2)
b1.grid(row=1, column=0)

b0 = Button(window, text="0", width=8, height=2, command=f3)
b0.grid(row=1, column=1)

clear = Button(window, text="C", width=8, height=2, command=f1)
clear.grid(row=1, column=2)

beq = Button(window, text="=", width=8, height=2, command=f4)
```

```
beq.grid(row=1, column=3)

badd = Button(window, text="+", width=8, height=2, command=f5)
badd.grid(row=2, column=0)

bsub = Button(window, text="-", width=8, height=2, command=f6)
bsub.grid(row=2, column=1)

bmul = Button(window, text="X", width=8, height=2, command=f8)
bmul.grid(row=2, column=2)

bdiv = Button(window, text="/", width=8, height=2, command=f7)
bdiv.grid(row=2, column=3)

window.mainloop()
```

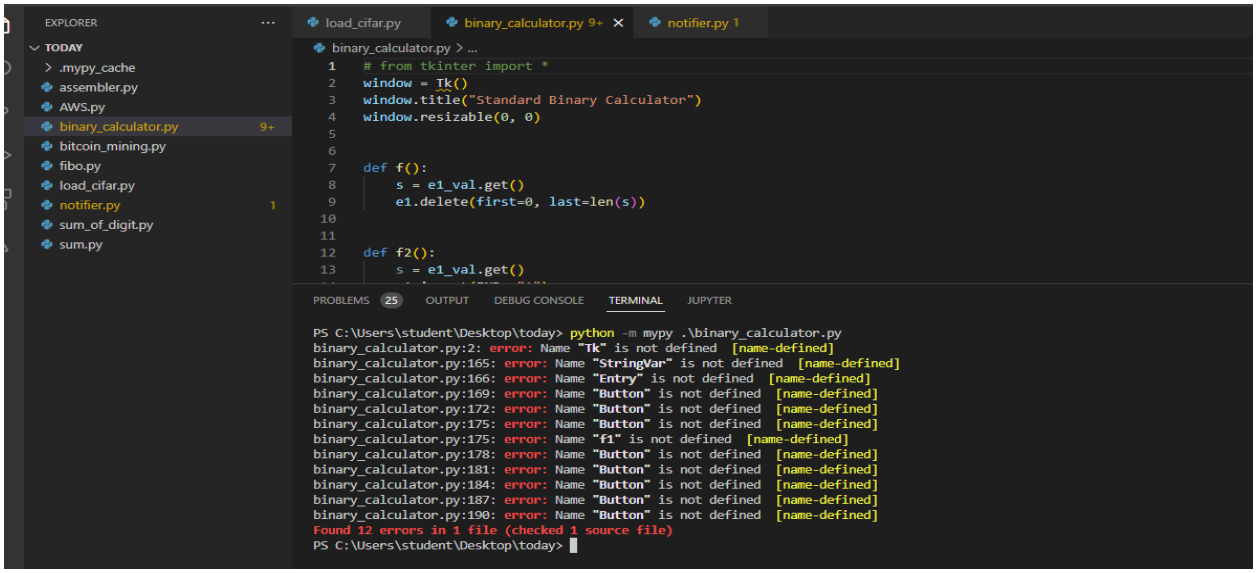
Error:

Here we will try to fetch the errors using previously downloaded mypy static analysis tool.



This error message is indicating that there is a problem with the function call on line 4 of the "binary_calculator.py" file. The error message suggests that there is no function overload variant that matches the argument types "int", "int".

Here after removing the import of library we are getting this type of error.



CODE 3:

This is the code for a notifier..This code is in python programming language.

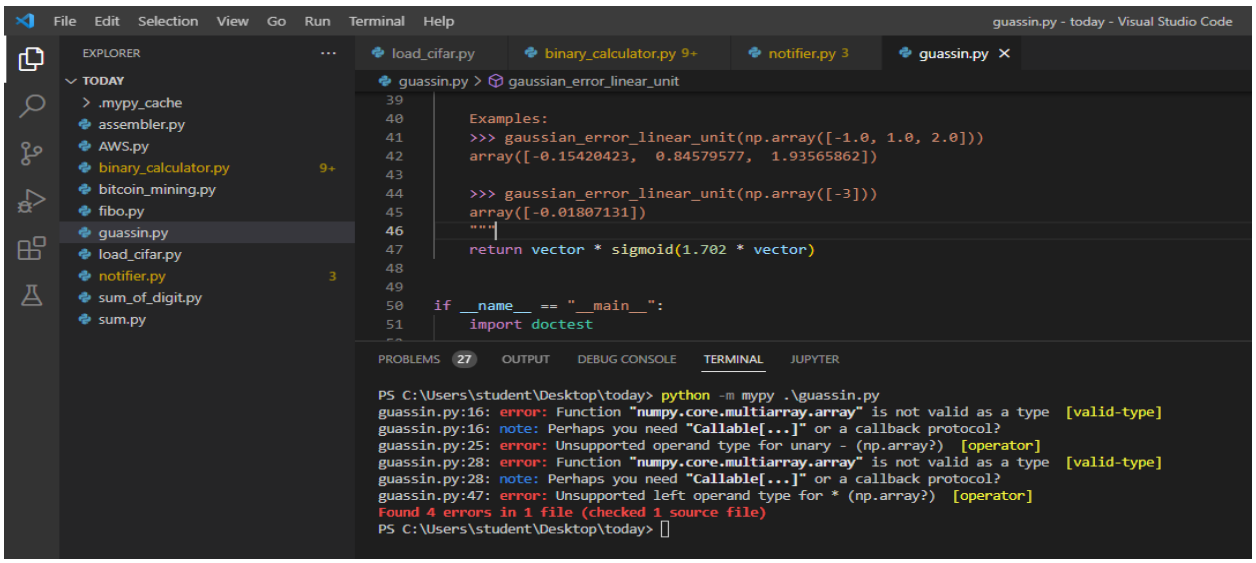
```
"""
This script demonstrates an implementation of the Gaussian Error Linear
Unit function.
*
https://en.wikipedia.org/wiki/Activation_function#Comparison_of_activation
_functions

The function takes a vector of K real numbers as input and returns x *
sigmoid(1.702*x).
Gaussian Error Linear Unit (GELU) is a high-performing neural network
activation
function.

This script is inspired by a corresponding research paper.
```

```
* https://arxiv.org/abs/1606.08415
"""

import numpy as np


def sigmoid(vector: np.array) -> np.array:
    """
    Mathematical function sigmoid takes a vector x of K real numbers as
input and
    returns 1/ (1 + e^-x).
    https://en.wikipedia.org/wiki/Sigmoid_function

    >>> sigmoid(np.array([-1.0, 1.0, 2.0]))
    array([0.26894142, 0.73105858, 0.88079708])
    """
    return 1 / (1 + np.exp(-vector))


def gaussian_error_linear_unit(vector: np.array) -> np.array:
    """
    Implements the Gaussian Error Linear Unit (GELU) function

    Parameters:
        vector (np.array): A  numpy array of shape (1,n)
        consisting of real values

    Returns:
        gelu_vec (np.array): The input numpy array, after applying
        gelu.

    Examples:
    >>> gaussian_error_linear_unit(np.array([-1.0, 1.0, 2.0]))
    array([-0.15420423,  0.84579577,  1.93565862])

    >>> gaussian_error_linear_unit(np.array([-3]))
    array([-0.01807131])
    """
    return vector * sigmoid(1.702 * vector)
```

```python
if __name__ == "__main__":
    import doctest

    doctest.testmod()
```

Error:

Here we will try to fetch the errors using previously downloaded mypy static analysis tool.



(i) guassin.py:16: error: Function "numpy.core.multiarray.array" is not valid as a type [valid-type]

In the upper error message there is an issue with the type of a function call to "numpy.core.multiarray.array". The error message suggests that the function call may require a type hint to specify the expected type of the return value.

Here is the hint given by **mypy**

guassin.py:16: note: Perhaps you need "Callable[...]" or a callback protocol?

(ii) guassin.py:25: error: Unsupported operand type for unary - (np.array?) [operator]

The second error message suggests that there is an issue with using the unary minus operator on a numpy array. It is likely that the array does not support this operation or there is an issue with the syntax used in the code.

(iii) guassin.py:47: error: Unsupported left operand type for * (np.array?) [operator]

This message suggests that there is an issue with using the multiplication operator on a numpy array. It is likely that the array does not support this operation or there is an issue with the syntax used in the code.