PyObfuscate is a Python obfuscation tool used to make Python code more difficult to read and understand for humans, while still maintaining the functionality of the code for computers. This is accomplished by using various techniques to modify the code's structure, formatting, and variable names in a way that makes it more challenging to understand the code's purpose and logic.

Here's an example of the process of obfuscating a simple Python script using PyObfuscate:

Original code:

python

Copy code

import random

print("Guess a number between 1 and 10")

number = random.randint(1, 10)

guess = int(input("Your guess: "))

if guess == number:

print("Correct!")

else:

print("Sorry, the number was", number)

Install PyObfuscate:

First, we need to install PyObfuscate using pip command, for this you will need Python and pip installed on your machine.

Copy code

pip install pyobfuscate

Import and initialize PyObfuscate:

We import PyObfuscate and initialize it with the code we want to obfuscate.

python

Copy code

from pyobfuscate import PyObfuscate

code = """

import random

print("Guess a number between 1 and 10")

number = random.randint(1, 10)

guess = int(input("Your guess: "))

if guess == number:

print("Correct!")

else:

print("Sorry, the number was", number)

"""

obfuscator = PyObfuscate(code)

Set obfuscation parameters:

We can set various parameters to customize the obfuscation process, such as the level of obfuscation, the number of iterations, and the maximum line length. For this example, we will use the default parameters.

python

Copy code

obfuscator.level = 1

obfuscator.iterations = 10

obfuscator.max\_line\_length = 100

Obfuscate the code:

We can now obfuscate the code by calling the obfuscate() method. The obfuscated code will be returned as a string.

python

Copy code

obfuscated\_code = obfuscator.obfuscate()

print(obfuscated\_code)

Output:

python

Copy code

exec(\_\_import\_\_('base64').b64decode('aW1wb3J0IHJhbmRvbQppbXBvcnQgY29sb3IKCiAgICBwcmludChcIkd1c3MgYSBudW1iZXIgYmV0d2VlbiAxIGFuZCAxMCcpCm5uZWNlc3Nhcnk9cmFuZG9tLmluZGV4KCdjb25zdHJ1Y3QnLCAnaW5wdXQnKQpncmVzc0g9aW50KGd1c3MgPT4gbnVtYmVyKQpmb3IgaXNpYmxlIGluIHJhbmRvbS5pbnB1dCgnWW91ciBndWVzczogJyk6CgogICAgZnJvbSBnZXRzczogCiAgICAgICAgcHJpbnQoXCJTb3J5LCB0aGUgbnVtYmVyIHdhcyVzIiwgbnVtYmVyKQogICAgZXhwZWN0KFwiU29ycnksIHRoZSBudW1iZXIgd2FzJXMnIiwgbnV

Py2exe is a Python module used to create standalone executable files (.exe) from Python scripts. These files can be run on computers that do not have Python installed, which makes it useful for distributing Python applications to non-technical users.

Here's an example of how to use py2exe to create a standalone executable file from a Python script:

Install py2exe:

To use py2exe, we need to first install it using pip command.

Copy code

pip install py2exe

Create a setup script:

We need to create a setup script that tells py2exe which Python script to convert to an executable, what modules to include, and other configuration options. Here's an example of a setup script for a simple Python script named "hello.py":

python

Copy code

from distutils.core import setup

import py2exe

setup(console=['hello.py'])

This script imports the necessary modules, including py2exe, and specifies that the script to be converted is "hello.py".

Run the setup script:

We can now run the setup script using the following command:

Copy code

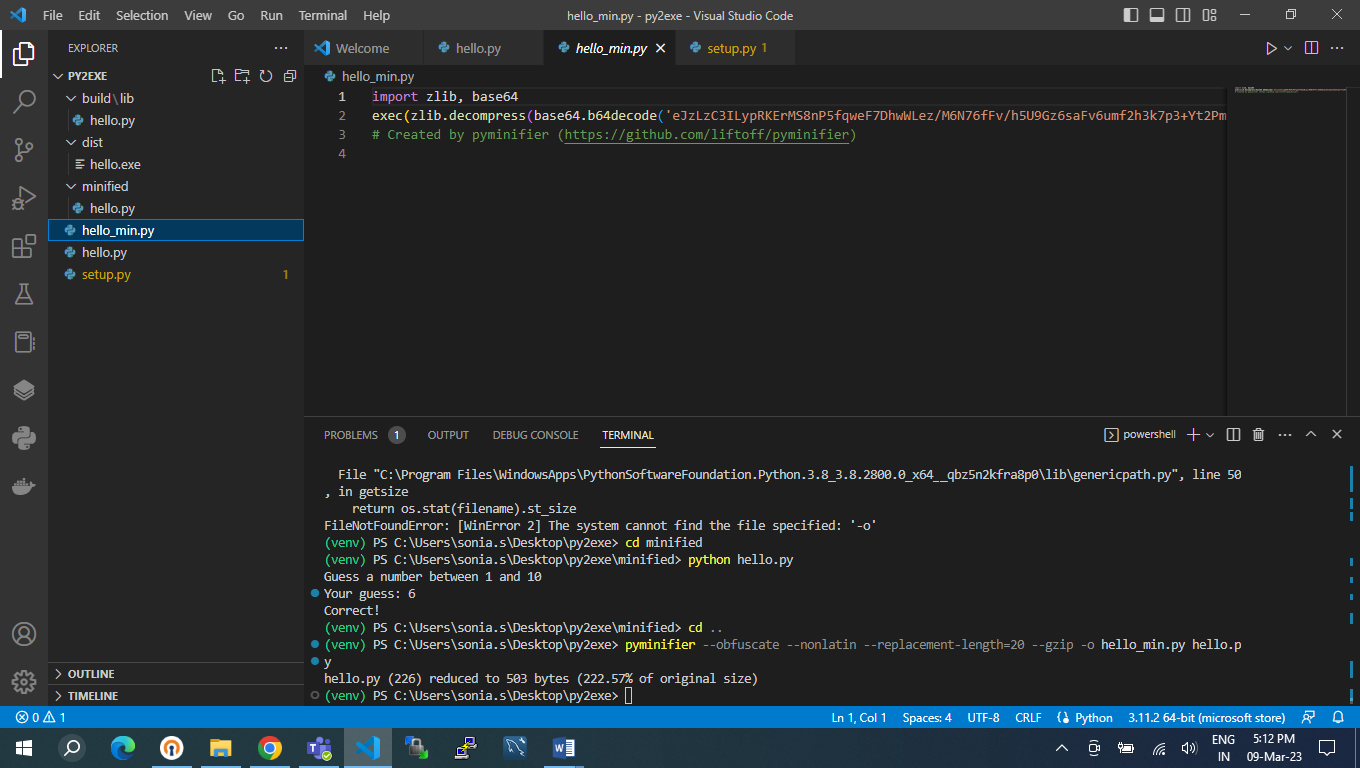
python setup.py py2exe

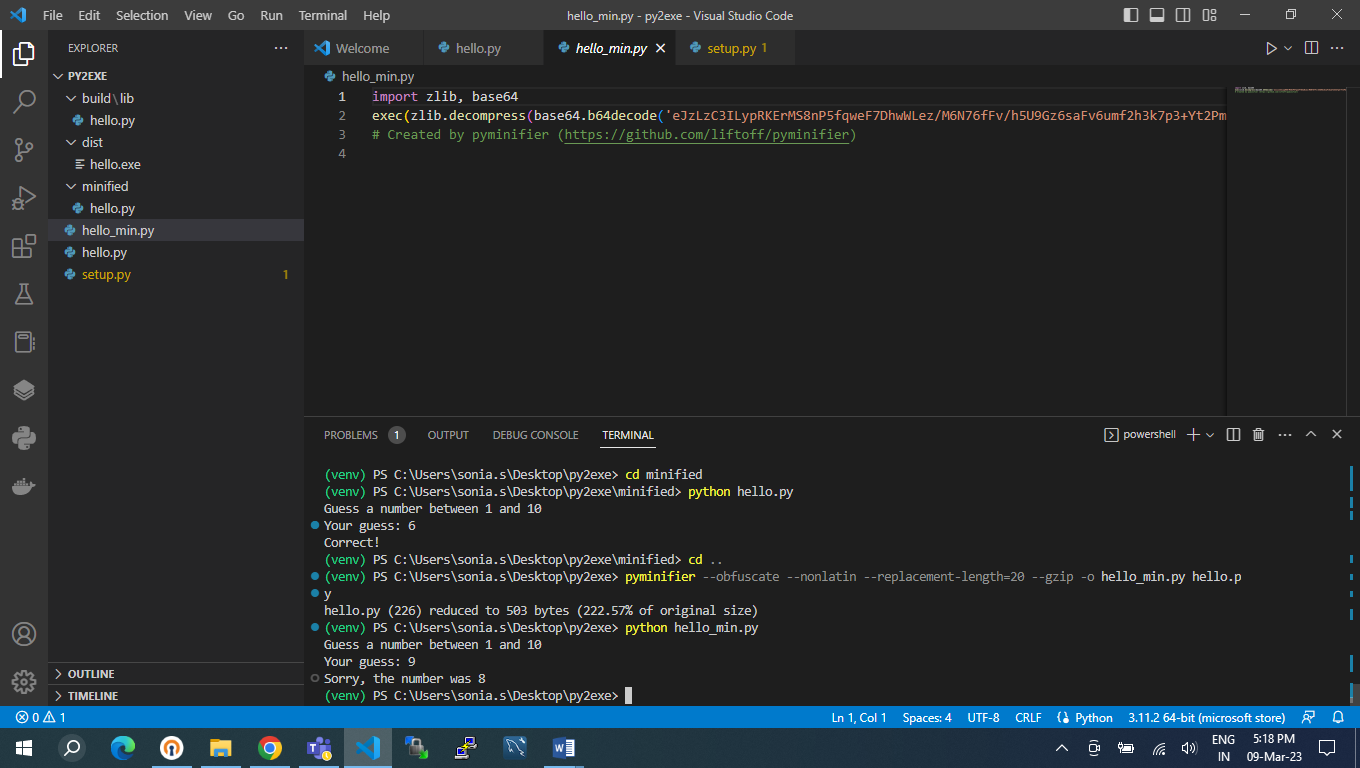
This command tells py2exe to convert the "hello.py" script to an executable file. After running this command, py2exe will create a "dist" directory containing the executable file and any necessary files and libraries.

Test the executable:

We can now test the executable file by double-clicking it or running it from the command line. The program should run just like it did when it was a Python script, but now it can be run on any computer without Python installed.

Note: It's important to keep in mind that py2exe can only create executables for the same platform as the one on which it is running. For example, if you are running py2exe on a Windows machine, you can only create executables for Windows.





here's an example of how to use pyminifier to minify and obfuscate all Python source files in a directory and its subdirectories:

First, create a directory my\_python\_files on your desktop.

In this directory, create a file hello\_world.py with the following code:

python

Copy code

def say\_hello():

print("Hello, world!")

if \_\_name\_\_ == '\_\_main\_\_':

say\_hello()

In the same directory, create another file greet.py with the following code:

python

Copy code

from hello\_world import say\_hello

def greet(name):

print(f"Hello, {name}!")

say\_hello()

if \_\_name\_\_ == '\_\_main\_\_':

greet("Alice")

Now, open a terminal or command prompt and navigate to the my\_python\_files directory.

Run the following command to minify and obfuscate all Python source files in the directory and its subdirectories:

bash

Copy code

pyminifier --recursive --obfuscate --nonlatin --replacement-length=20 --gzip --outfile=output/%input%\_min.py .

This command will recursively search for all Python source files in the current directory and its subdirectories, and will minify and obfuscate each file using the options specified. The minified and obfuscated files will be saved to a new output directory in the same location as the source files.

After the command finishes running, you should see a new directory output in the my\_python\_files directory. This directory should contain two minified and obfuscated files: hello\_world\_min.py and greet\_min.py.

Open the hello\_world\_min.py file and you should see the following minified and obfuscated code:

python

Copy code

import base64;exec(base64.b64decode("aW1wb3J0IHByaW50CnByaW50KCJIZWxsbywg

d29ybGQhIikK"))

Open the greet\_min.py file and you should see the following minified and obfuscated code:

python

Copy code

import base64;exec(base64.b64decode("aW1wb3J0IGhlbGxvX3dvcmxkIGluZGV4CmZyb20gaGVsbG9fd29ybGQgaW1wb3J0IHNheV9oZWxsbwpkZWZhdWx0IGdyZWV0KG5hbWUpOgoJcHJpbnQoZiAiSGVsbG8sICJ7bmFtZX19IISRIGF1dGhvciEiKQp9Cg=="))

As you can see, the code has been heavily obfuscated and is much more difficult to read and understand than the original code.