Prithu Adhikari

Week 5

**TASK**: Use an editor to input the Python program shown below then save it to a file called first\_prog.py. Once that has been done, execute the program from the command line.

number = input("Enter a number: ")

number = int(number)

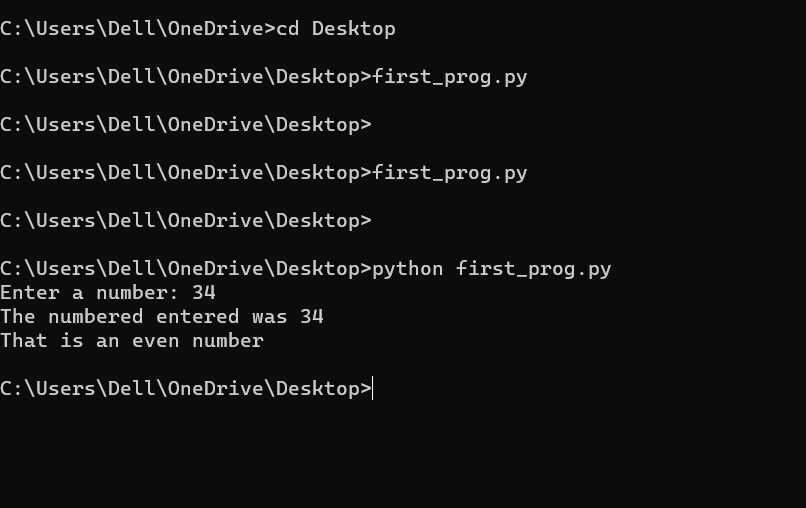
print("The numbered entered was", number)

if (number % 2) == 0:

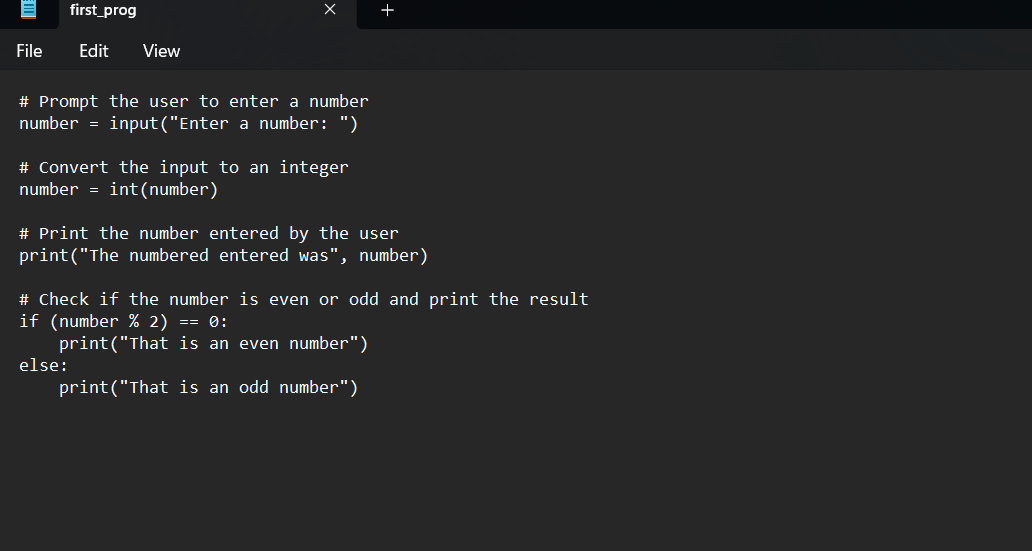
print("That is an even number")

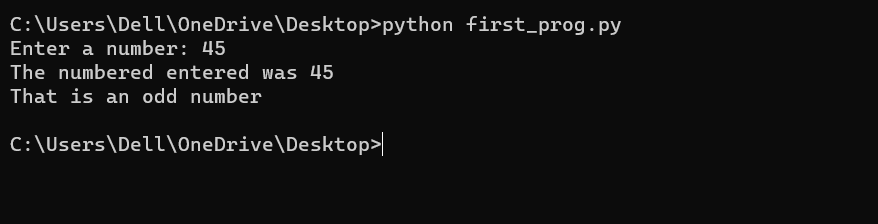
else:

print("That is an odd number")

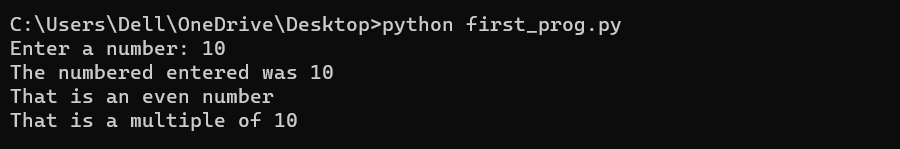


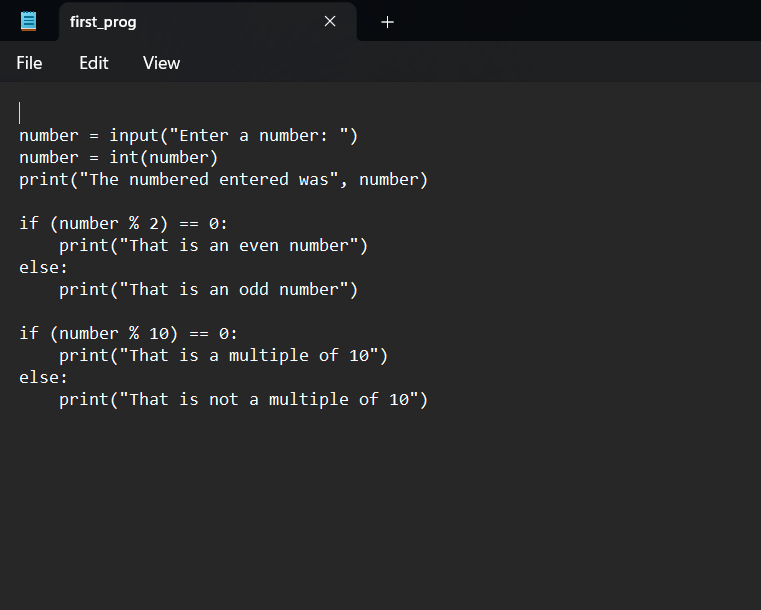
**TASK**: Open the first\_prog.py and add comments above each statement within the file describing what that statement does (using a ‘#’ at the beginning of the line) . Save the file and execute it again for testing purposes.





**TASK**: Open the first\_prog.py and add some extra code that identifies and prints a message stating whether the entered number is divisible by 10. You should be able to base the new code on the if statement already provided. Once completed, save the file and execute it again for testing.





**TASK**: Use an editor to input the Python program shown below then save it to a file called total.py. Once that has been done, execute the program from the command line, passing several numeric values for testing.

import sys

count = len(sys.argv)

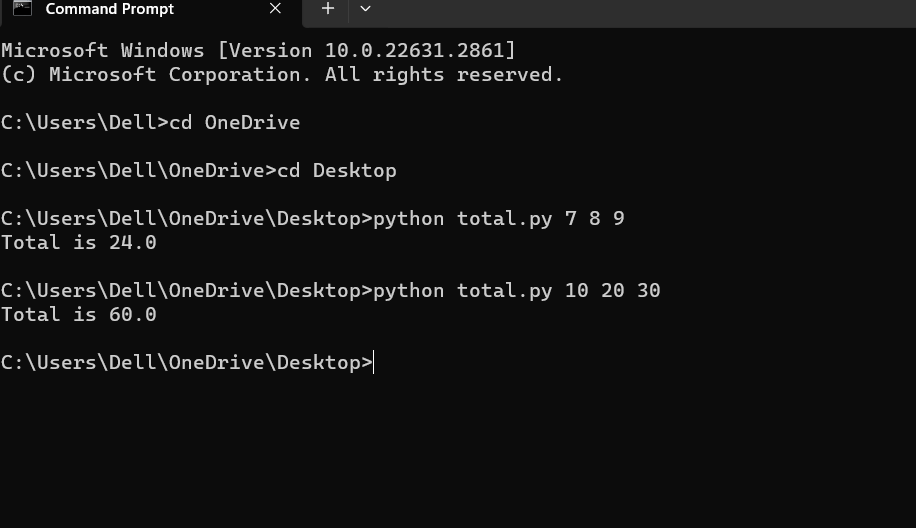
total = 0

while count > 1:

count -= 1

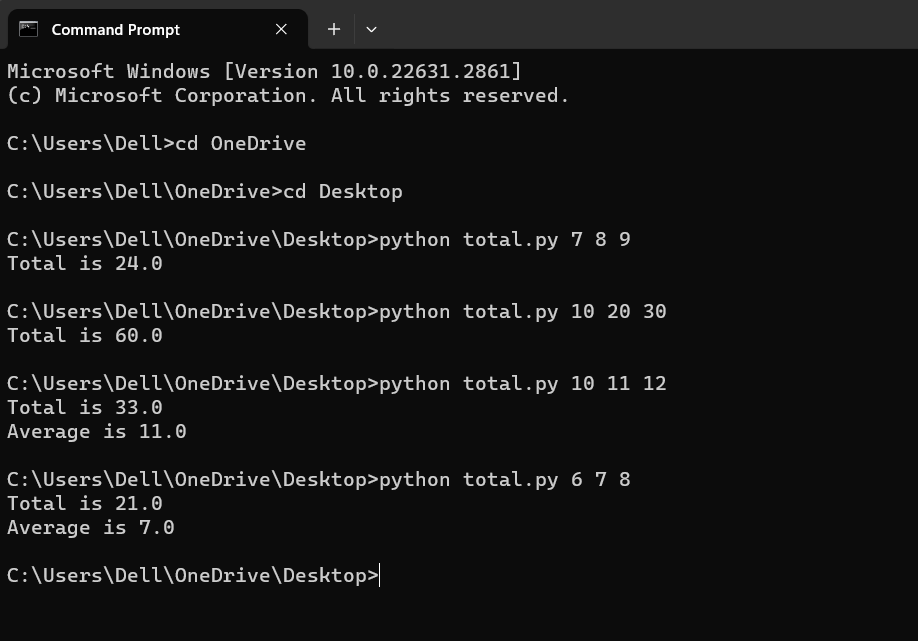
total += float(sys.argv[count])

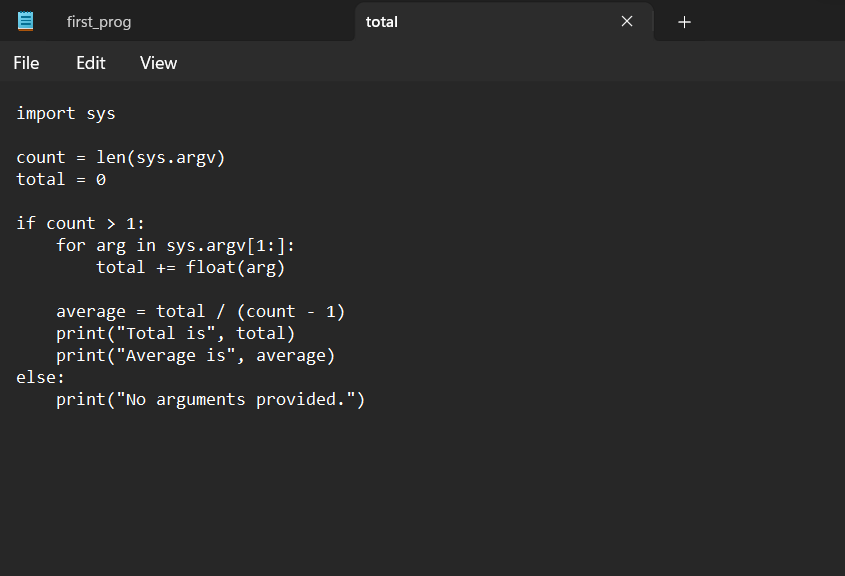
print("Total is", total)



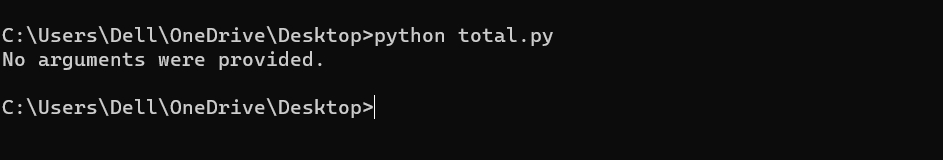
**TASK**: Improve the previous program by adding additional code that not only prints the total of any passed arguments, but also calculates and prints the average. Execute the program several times for testing. What happens if no arguments are passed?

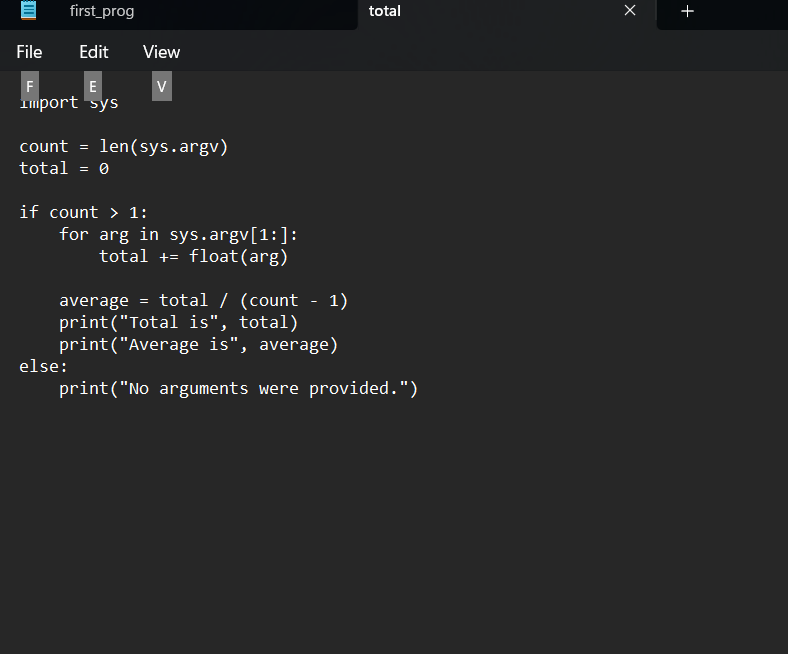
Remember, the first element of the sys.argv list is the program name itself, so it is not uncommon to ignore or skip processing of that first element.





**TASK**: Improve the program once more by adding a check to see whether no arguments have been passed, if so print a message saying “no arguments were provided”. Also add comments to the program. Execute the program several times for testing.





**TASK**: Use an editor to input the Python program shown below then save it to a file called my\_utils.py.

def average(values):

""" Calculates the average of the given list. """

total = 0;

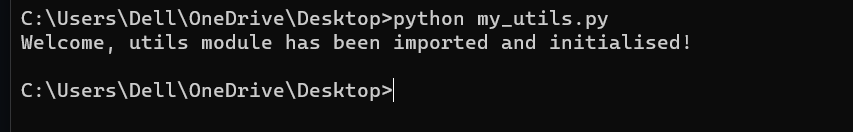
for n in values: # total the given values

total += float(n)

return total/len(values) # return calculated average

# initialisation statement

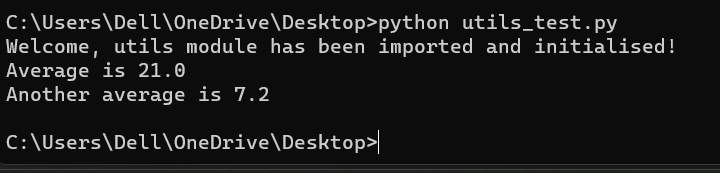
print("Welcome, utils module has been imported and initialised!")



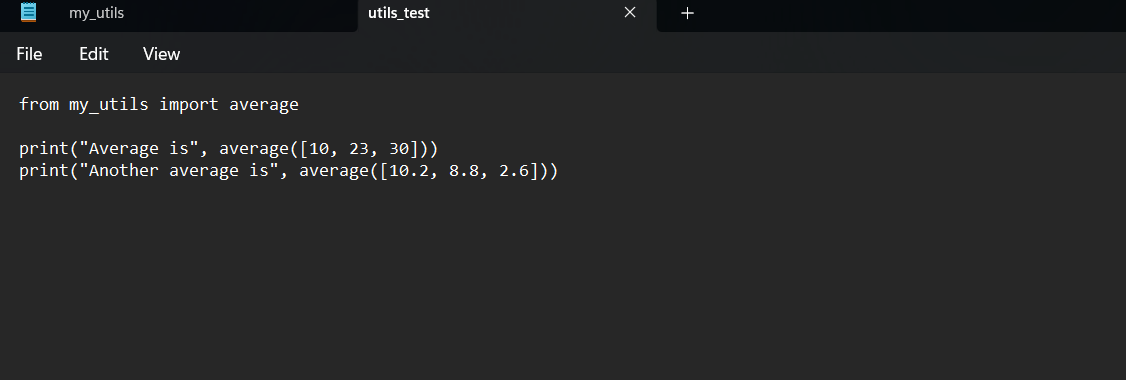
**TASK**: Use an editor to input another Python program utils\_test.py. This program should import my\_utils then call the average() function several times, passing a list of values as a parameter, e.g.

print("Average is", my\_utils.average([10, 23, 30]))

print("Another average is", my\_utils.average([10.2, 8.8, 2.6]))



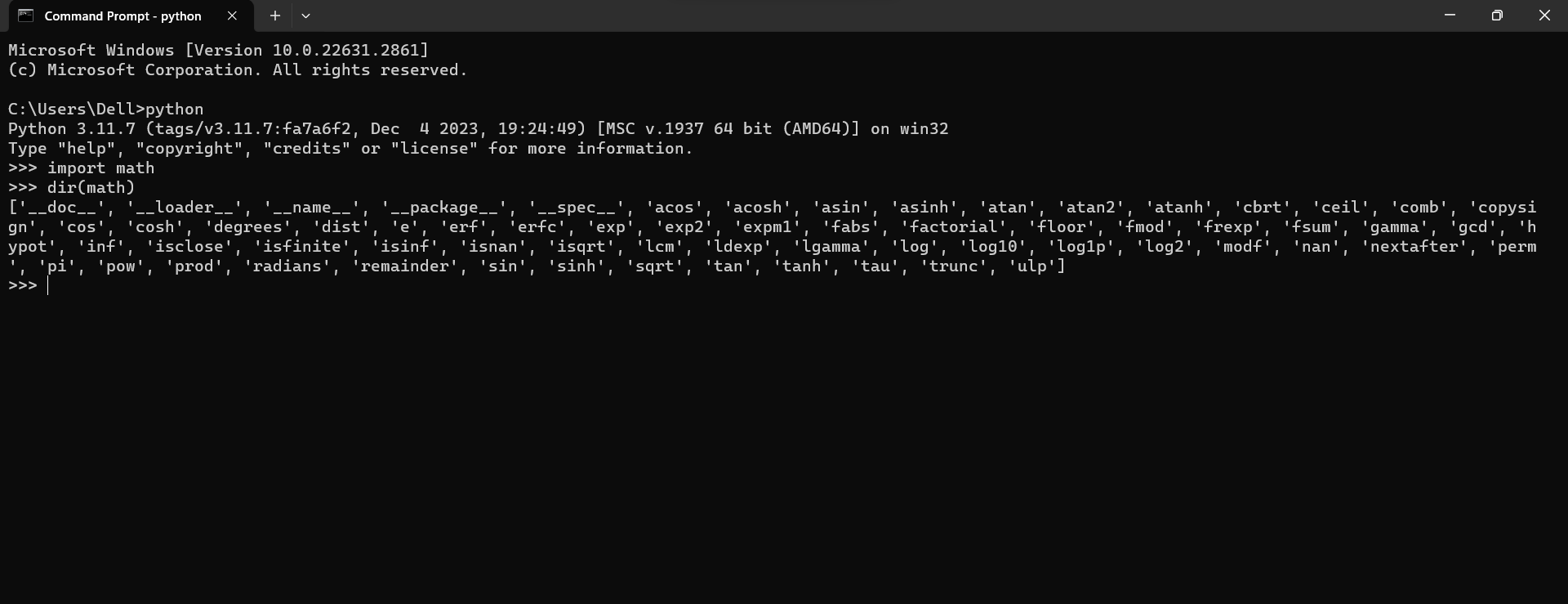
**TASK**: Update the previous program utils\_test.py, so that the import statement explicitly imports the average() function directly into the program’s *symbol table*, allowing the prefix to be removed from the later function calls.



**TASK**: Start Python in interactive mode and input the following statements.

>>> import math

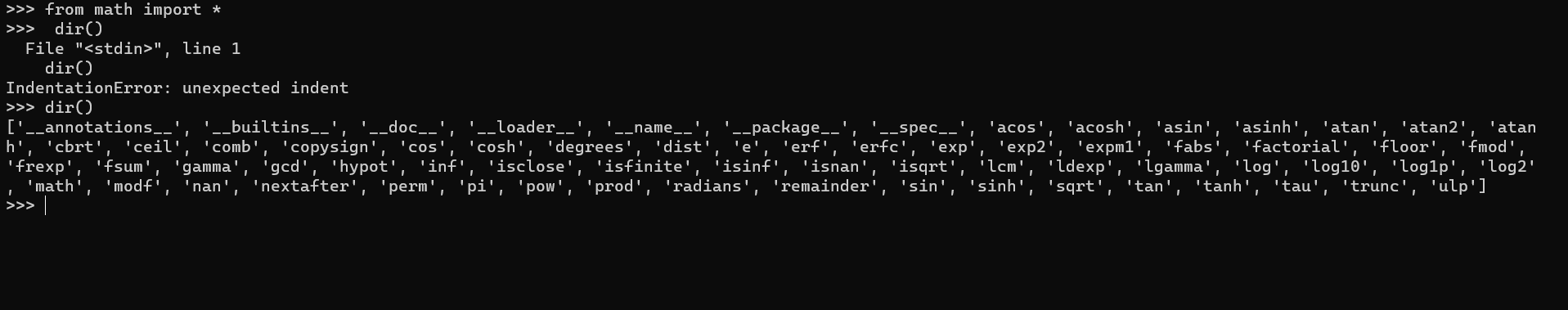
>>> dir(math)



]**TASK**: Now enter the following statements:

>>> from math import \*

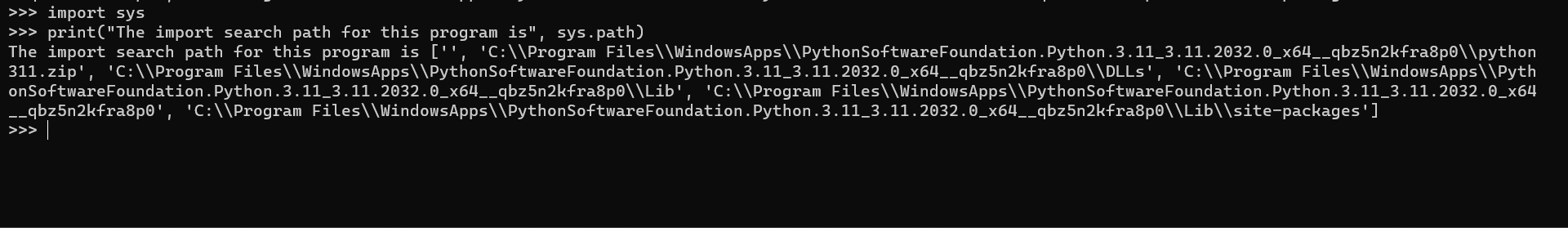
>>> dir()



**TASK**: Use an editor to input a Python program show\_path.py. Execute the script and examine the results. Notice the first entry will be the directory in which the program file itself is stored.

import sys

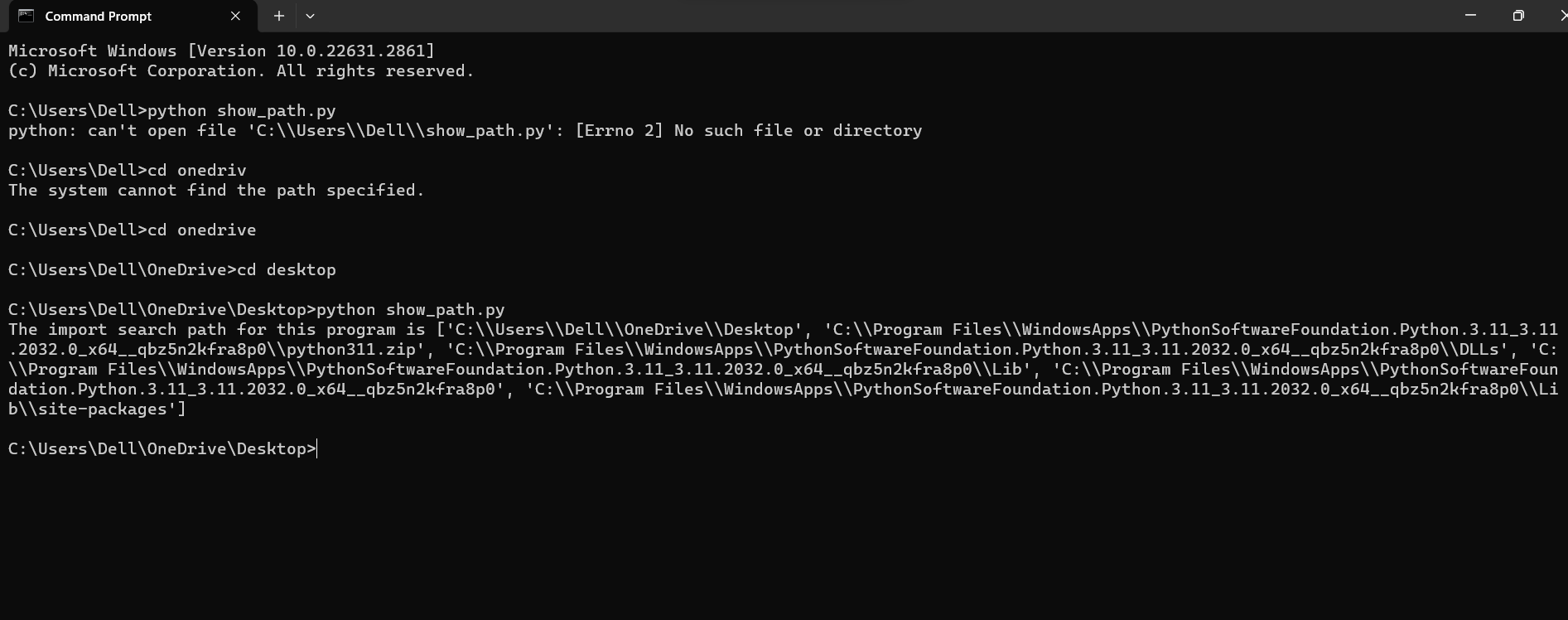
print("The import search path for this program is", sys.path)



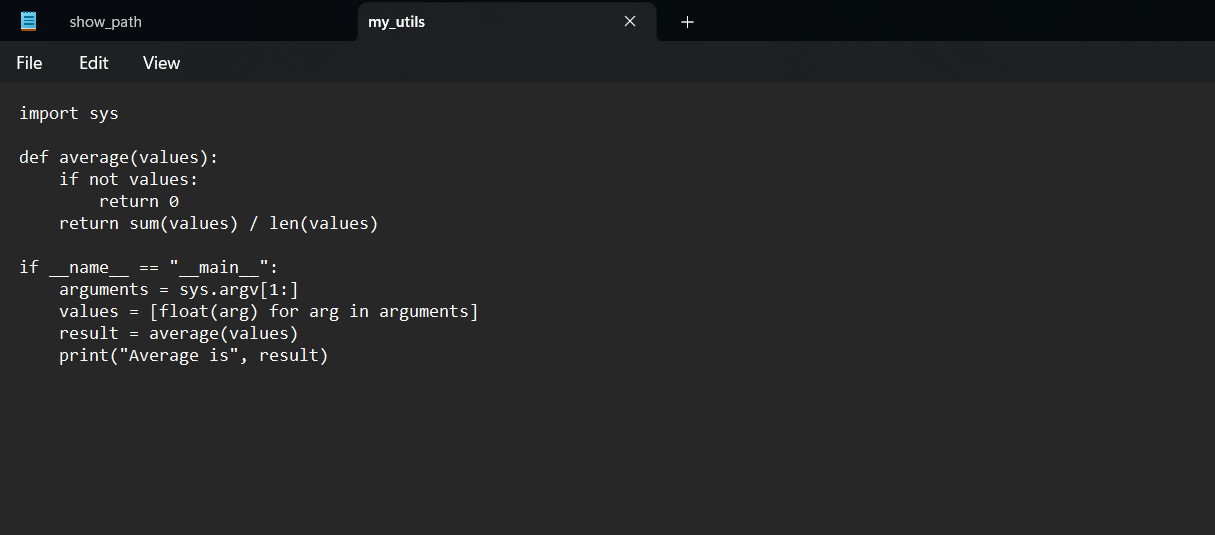
**TASK**: Use an editor to input a Python program show\_path.py. Execute the script and examine the results. Notice the first entry will be the directory in which the program file itself is stored.

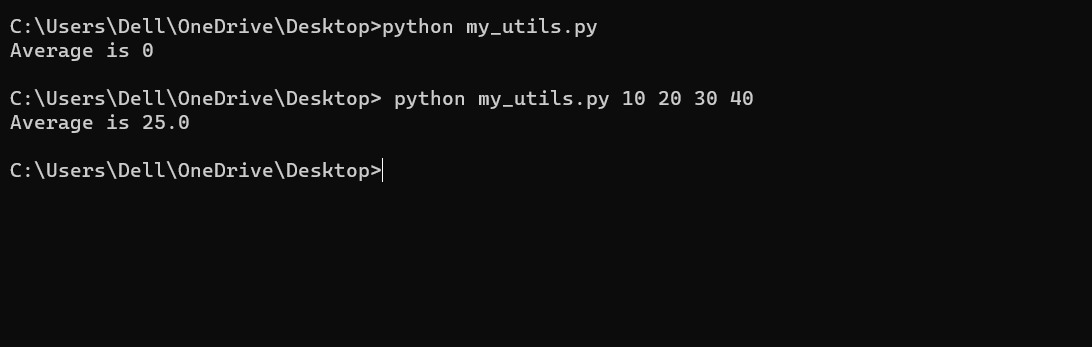
import sys

print("The import search path for this program is", sys.path)



**TASK**: Update the earlier program my\_utils.py, so that when executed directly from the command line it displays the average of any provided *command-line arguments*. However when imported by another program, nothing is displayed until the average() function is explicitly called.





**TASK**: Look at each of the phrases below and ensure you understand what each of these means. For any that you do not understand, do a little research to find a definition of each term. This research may involve looking back over these notes, or the associated lecture notes. It may also involve searching for these terms on the Internet.

* IDE
* Module
* Command Line Arguments
* Symbol-table
* Search path

IDE (Integrated Development Environment):

A software tool that provides a comprehensive environment for programming, including a code editor, debugger, and compiler.

Module:

A file in Python containing reusable code, including functions, classes, and variables, that can be imported into other scripts.

Command Line Arguments:

Values provided to a program when it is run from the command line, enabling customization and additional input.

Symbol-table:

A data structure in a compiler or interpreter that keeps track of program entities like variables and functions, associating names with their attributes.

Search Path:

A list of directories that an operating system or Python uses to locate executable files, libraries, or Python modules and packages.