# Introduction to Programming

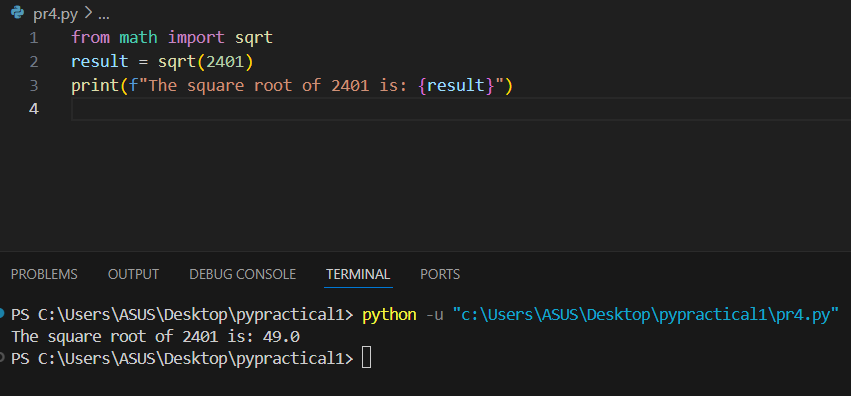
## Lab Worksheet

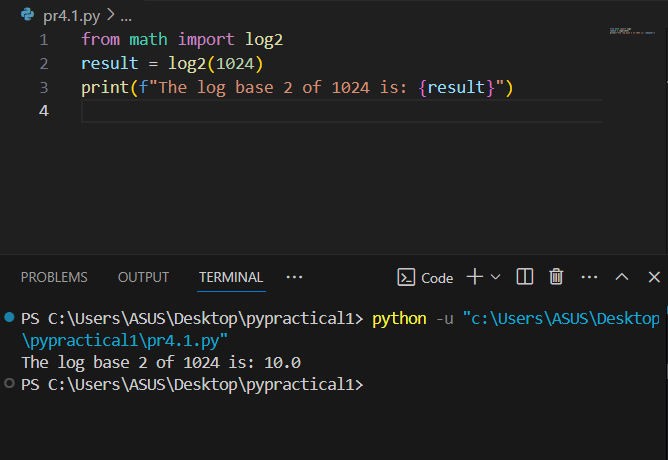
Week 4

**TASK**: Write some code that imports the math *module*, then calculates and prints the square root of the number 2401. Use the sqrt() function provided by the math module.

Individual functions can also be imported from a module. For example, we could import the sqrt() function only using the following code:

from math import sqrt



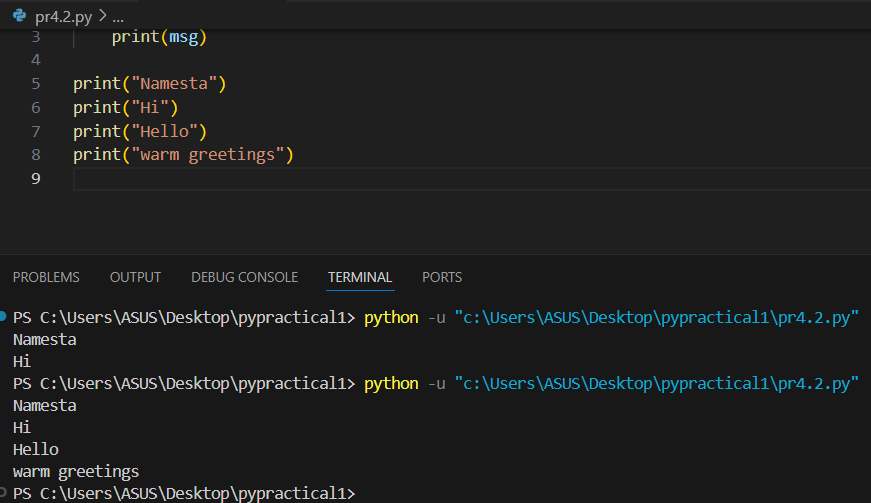
**TASK**: Write some code that imports only the log2() function from the math *module*, then call this function to calculate the log base 2 value of 1024. Print the result to the screen.

def displayTwice(msg):

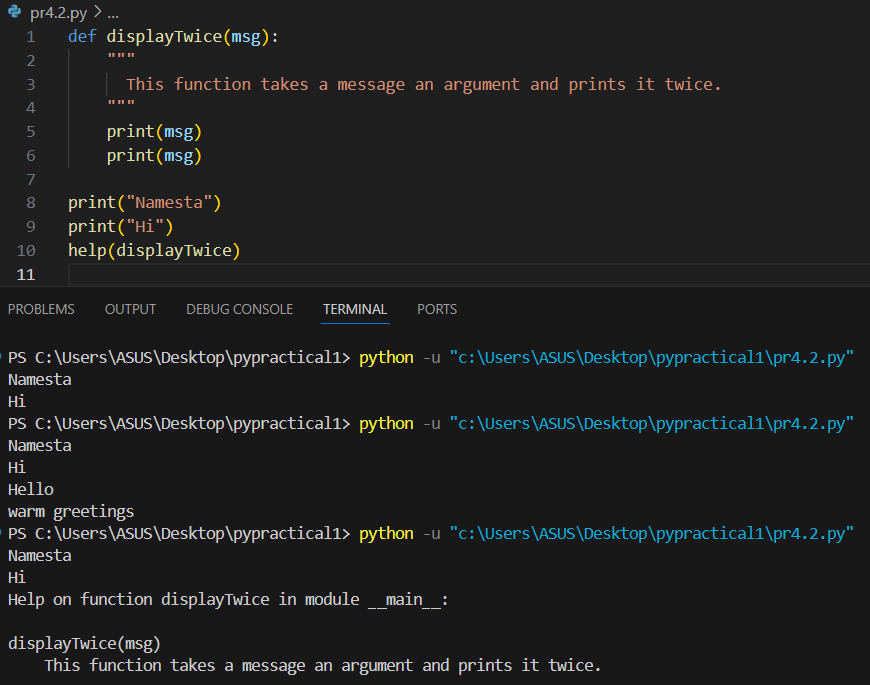
print(msg)

print(msg)

**TASK**: Input the above function definition. Once that is done make several calls to the function passing different argument values for testing.



**TASK**: Re-Input the above function definition, but this time add a docstring that includes a description of the function’s purpose. Once that is done enter a command such as help(displayTwice) and see what it displays.



def findMax(a,b):

"""Finds the maximum of two values."""

if ( a > b ):

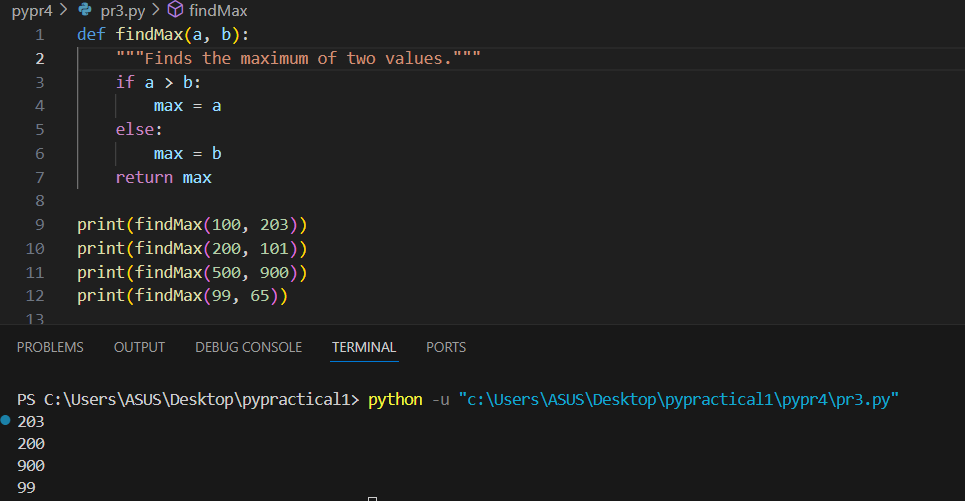
max = a

else:

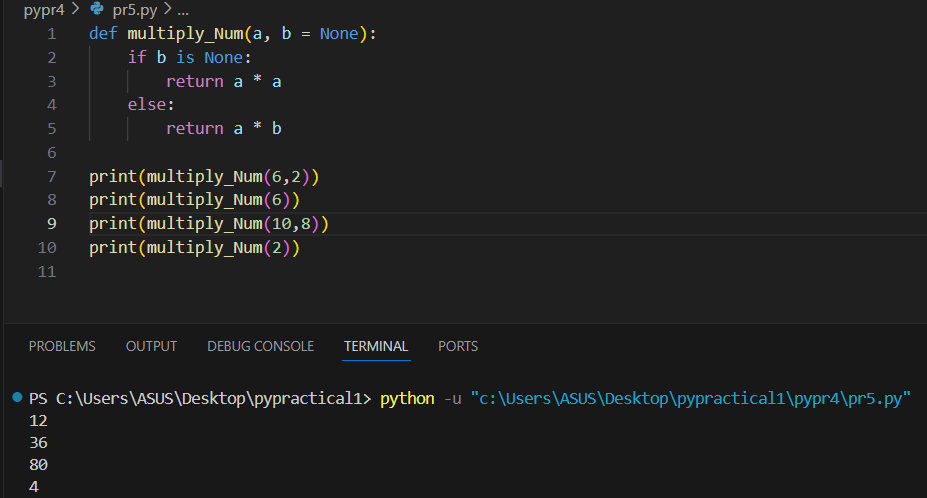
max = b

return max

**TASK**: Input the above function definition. Once that is done make several calls to the function passing different argument values and displaying the returned value.



**TASK**: Define a function that takes two numeric values, multiplies them together then returns the result. If the function is called with only a single argument however, then the value should be multiplied by itself. Once the function is defined, call it several times and display the returned values for testing purposes.



someFunc(y=2, z=3, x=1)

x is 1

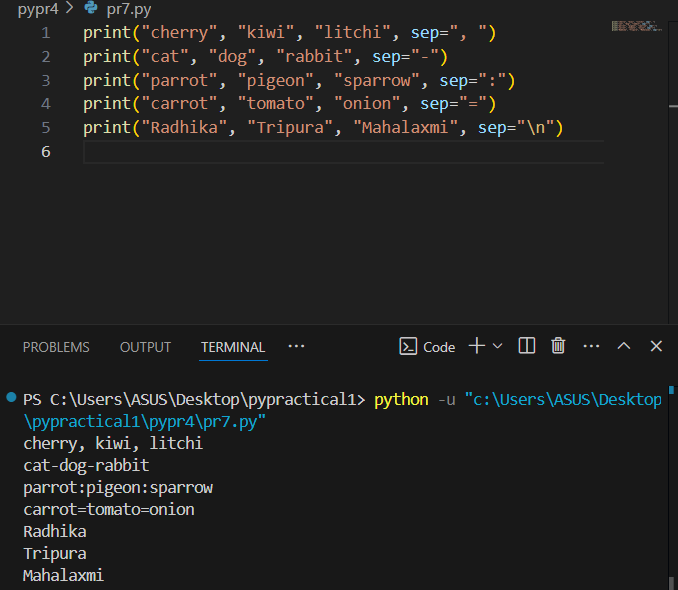
y is 2

z is 3

**TASK**: Enter the example function shown above, then try calling it using the *keyword arguments* in several different orders.



**TASK**: The built-in print() function supports a keyword argument called sep. This is used to decide what character to display between each of the provided positional parameters. Write some code that makes several calls to the print() function while setting the sep argument to values other than a space (which is the default)



def calcAve(\*numbers):

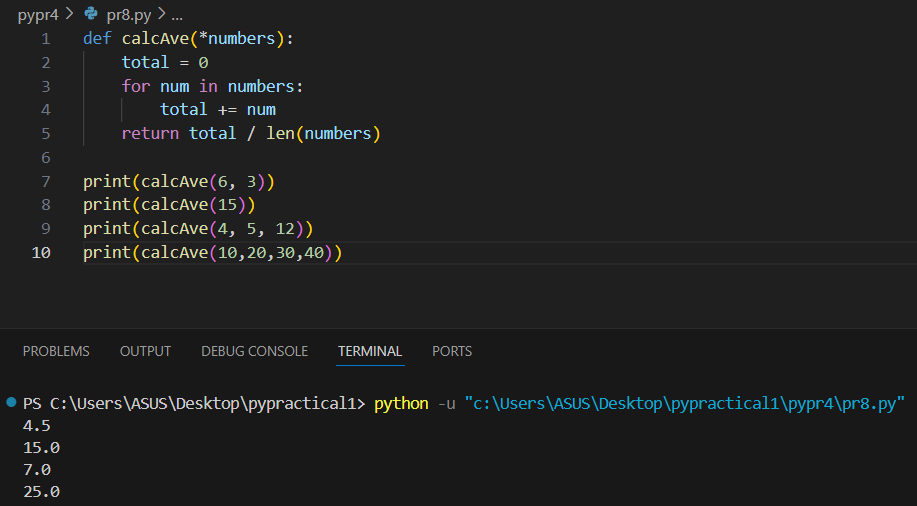
total = 0

for num in numbers:

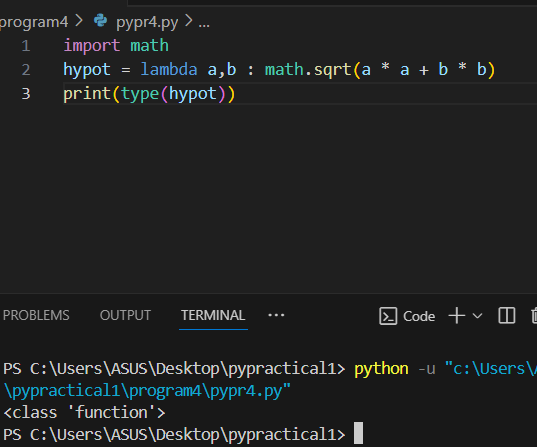
total += num

return total/len(numbers)

**TASK**: Enter the example function shown above, then try calling it several times, passing a different number of numeric arguments each time.



**TASK**: Enter the example lambda expression shown above, then find out the *data type* of the hypot variable using a call to the type() function. Notice the result.



**TASK**: Write a lambda expression that takes two formal parameters, hours and minutes. The expression should calculate and return the total number of equivalent seconds. Assign the expression to a variable called to\_seconds, then call the function several times for testing.

Given the sample input shown below, your solution should display the same results -

>>> to\_seconds(0,2)

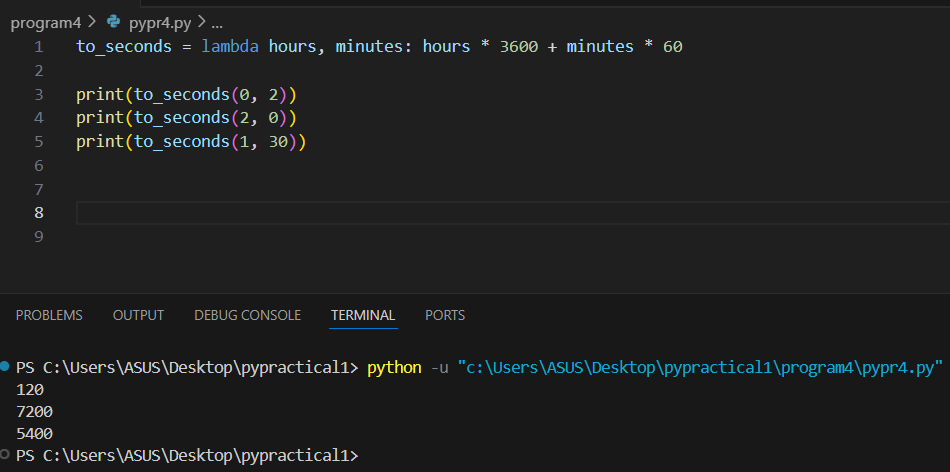
120

>>> to\_seconds(2,0)

7200

>>> to\_seconds(1,30)

5400



**TASK**: Improve your previous lambda expression so that if only one argument is passed within a call, then the number of minutes defaults to 0, as detailed below:

>>> to\_seconds(1)

3600

>>> to\_seconds(2)

7200

