

Fundamentals of Computer Programming



Submitted by

Rishav Rauniyar

Section E 8711

BSc(Computing)

Level 4 1st Semester

Submitted to

Krishna Sir

Introduction to Programming

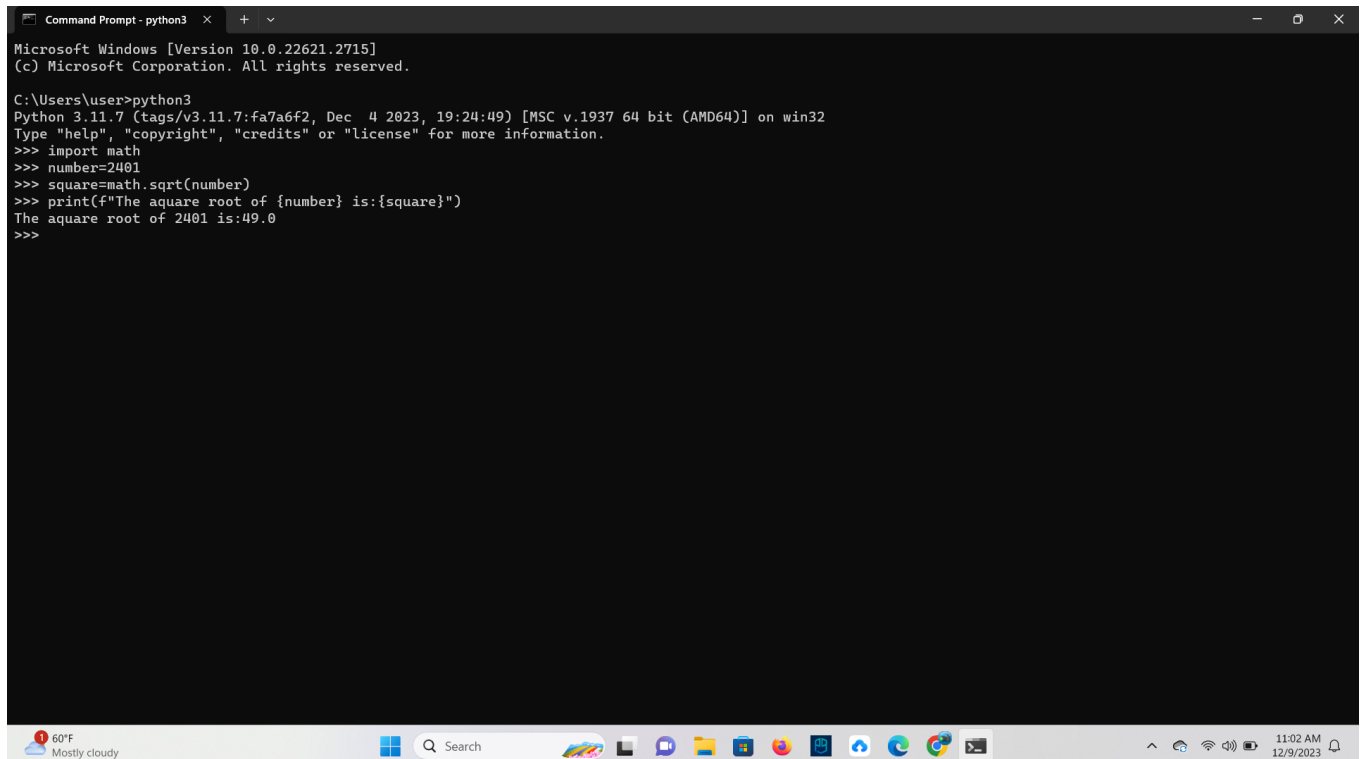
Lab Worksheet

Week 4

Importing and Using Functions

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.

Ans

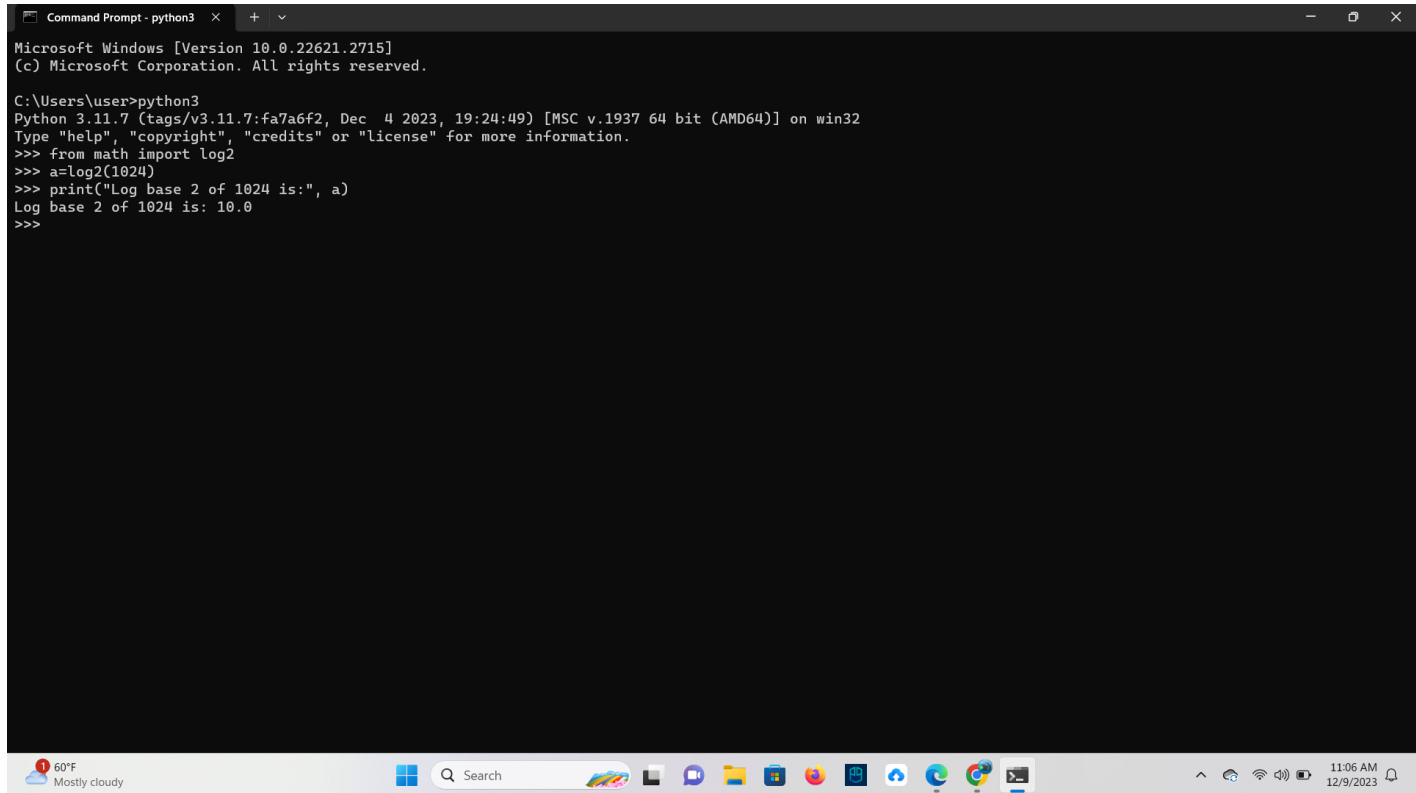
A screenshot of a Windows Command Prompt window titled "Command Prompt - python3". The window shows the execution of Python code to calculate the square root of 2401. The output is "The square root of 2401 is:49.0". The taskbar at the bottom shows the date and time as 11:02 AM on 12/9/2023.

```
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import math
>>> number=2401
>>> square=math.sqrt(number)
>>> print(f"The square root of {number} is:{square}")
The square root of 2401 is:49.0
>>>
```

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.

Ans

A screenshot of a Windows Command Prompt window titled "Command Prompt - python3". The window shows the execution of Python code. The prompt is at the C:\Users\user directory. The user has run 'python3', which has opened a Python 3.11.7 shell. Inside the shell, the user has imported the 'log2' function from the 'math' module, assigned it to 'a', and then printed a message along with the value of 'a'. The output shows that the log base 2 of 1024 is 10.0. The Windows taskbar is visible at the bottom, showing the time as 11:06 AM on 12/9/2023.

```
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from math import log2
>>> a=log2(1024)
>>> print("Log base 2 of 1024 is:", a)
Log base 2 of 1024 is: 10.0
>>>
```

Defining Functions

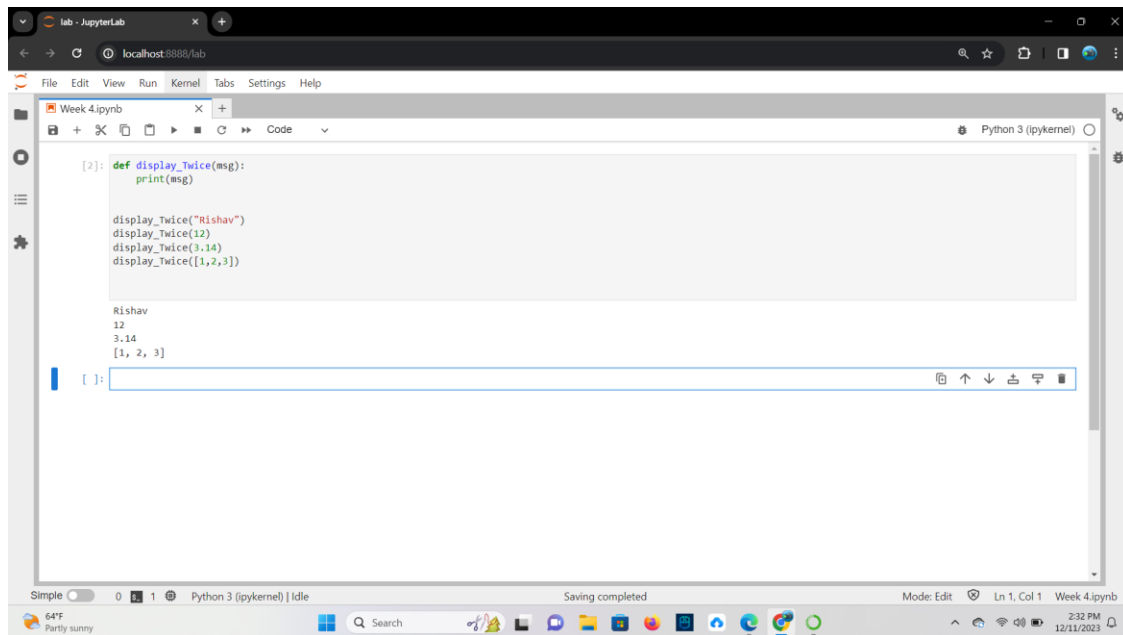
```
def display_Twice(msg):

    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.

Ans



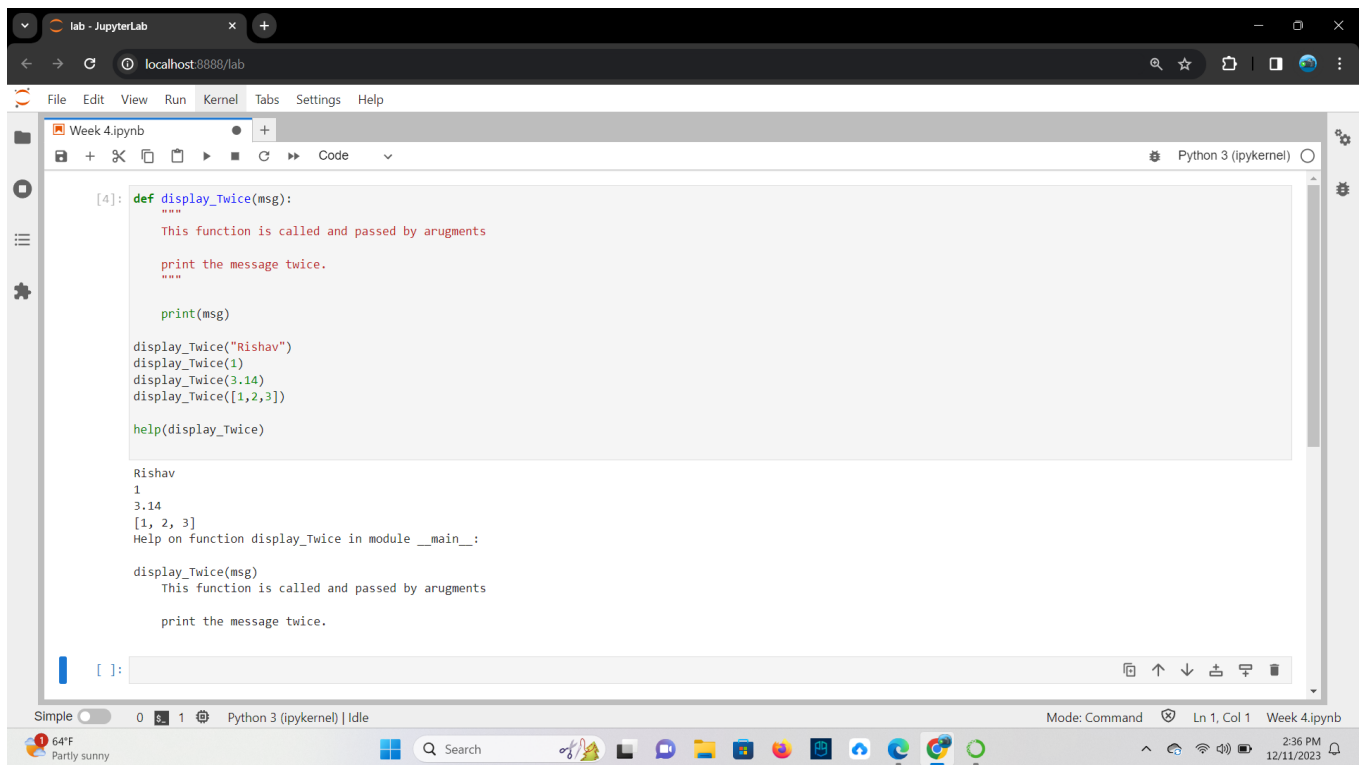
The screenshot shows a JupyterLab window with a single tab titled 'Week 4.ipynb'. The interface includes a top menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a left sidebar with icons for file explorer, search, and settings. The main area is a code editor with a light gray background. It contains a Python function definition and its execution output. The function is named 'display_twice' and takes a message as input. It prints the message twice. The output shows the function being called with 'Rishav', '12', '3.14', and a list '[1, 2, 3]'. The status bar at the bottom indicates 'Simple' mode, 'Python 3 (ipykernel) | Idle', and 'Saving completed'. The system tray at the very bottom shows the date and time as '2:32 PM 12/11/2023'.

```
[2]: def display_twice(msg):  
      print(msg)  
  
      display_twice("Rishav")  
      display_twice(12)  
      display_twice(3.14)  
      display_twice([1,2,3])  
  
      Rishav  
      12  
      3.14  
      [1, 2, 3]
```

Docstrings

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 (display Twice)` and see what it displays.

Ans

The screenshot shows a JupyterLab window with a browser address bar at localhost:8888/lab. The main area contains a code editor with a Python script. The script defines a function 'display_twice(msg)' with a docstring, prints the message twice, and then calls the function with various arguments: 'Rishav', '1', '3.14', '[1, 2, 3]', and the function itself. The output shows the results of these calls, including the function's docstring and the message being printed twice for each call. The bottom status bar indicates 'Python 3 (ipykernel) | Idle' and 'Mode: Command'.

Formal and Actual Parameters

When a function is being defined, the arguments we specify are usually referred to as the formal parameters of the function. In the above example the formal parameter is called msg. When a function is being called, the argument values provided are called the actual parameters. The formal parameter names act like local variables within the function and can only be accessed within that specific function block. Any variables declared within the function block can also only be accessed from within that block and cease to exist once the function ends. This idea of local-scope means functions do not have to worry about using variable names that may already exist elsewhere in the program or within other function

Returning a value

```
def findMax(a,b):
```

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

```
    if ( a > b ):
```

```
        max = a
```

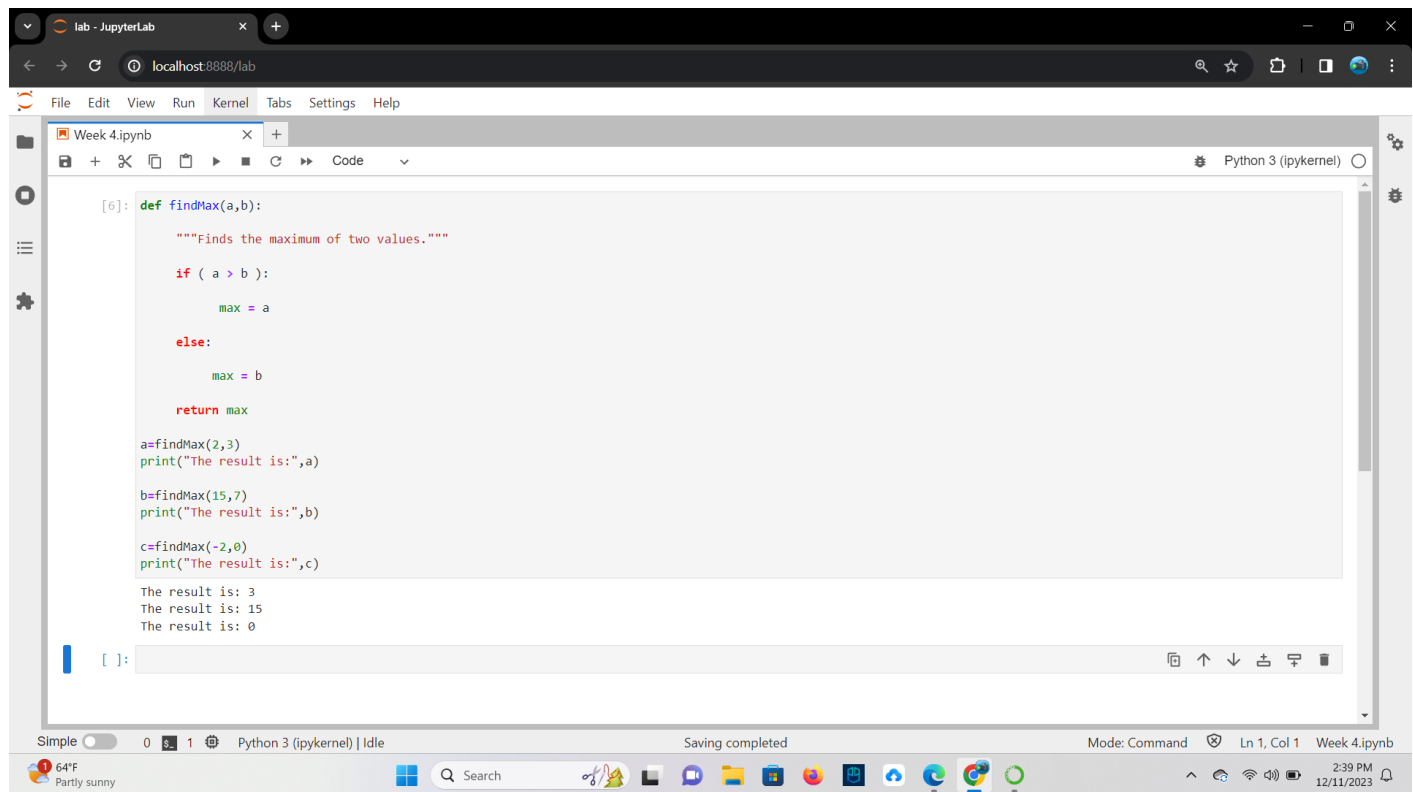
else:

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.

Ans



The screenshot shows a JupyterLab window with a browser address bar at localhost:8888/lab. The interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a toolbar. A file named 'Week 4.ipynb' is open. The code editor contains the following Python code:

```
[6]: def findMax(a,b):  
    """Finds the maximum of two values."""  
    if ( a > b ):  
        max = a  
    else:  
        max = b  
    return max  
  
a=findMax(2,3)  
print("The result is:",a)  
  
b=findMax(15,7)  
print("The result is:",b)  
  
c=findMax(-2,0)  
print("The result is:",c)
```

The output of the code is displayed below the code cells:

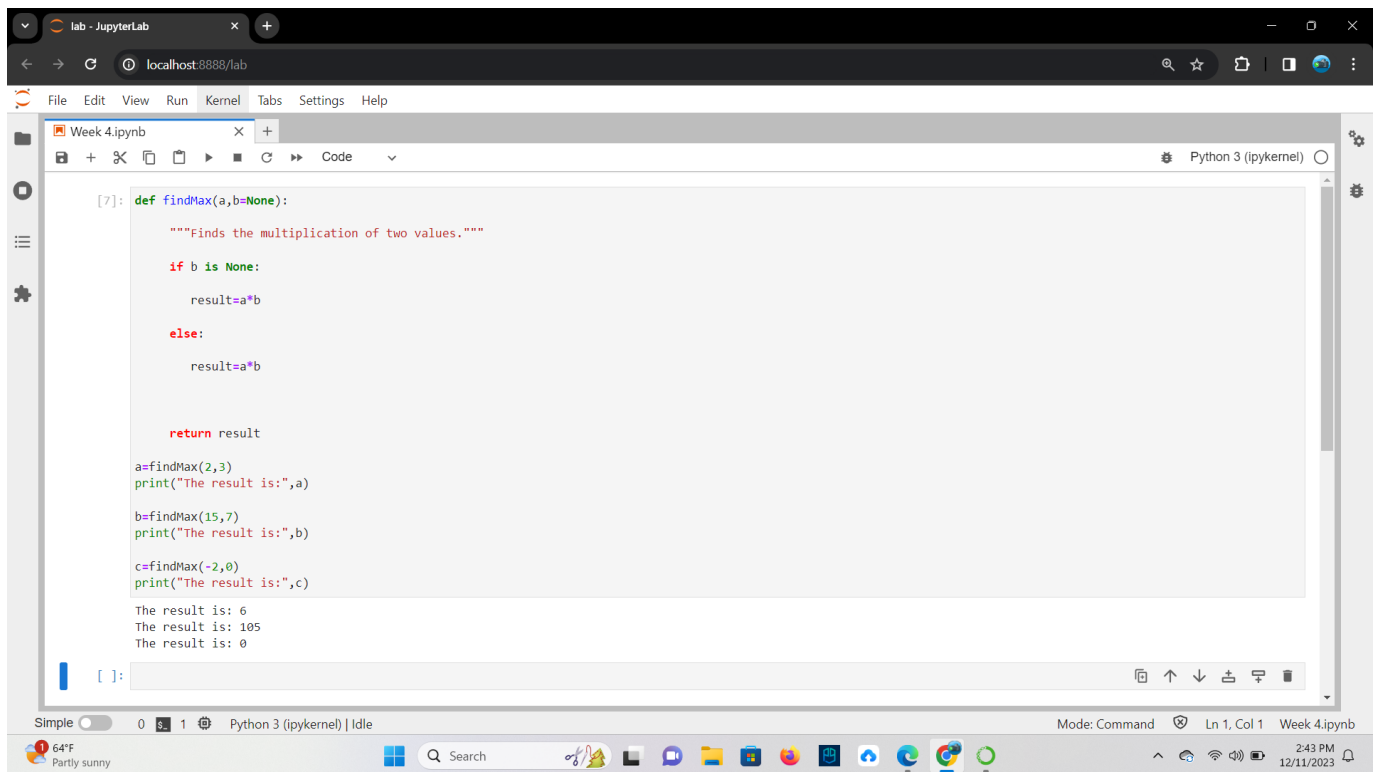
```
The result is: 3  
The result is: 15  
The result is: 0
```

The bottom status bar shows 'Simple' mode, 'Python 3 (ipykernel) | Idle', 'Saving completed', 'Mode: Command', 'Ln 1, Col 1', and 'Week 4.ipynb'. The Windows taskbar at the bottom shows the date and time as 2:39 PM on 12/11/2023.

Default Arguments

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.

Ans



The screenshot shows a JupyterLab window with a file named 'Week 4.ipynb'. The code editor contains the following Python code:

```
[7]: def findMax(a,b=None):  
    """Finds the multiplication of two values."""  
    if b is None:  
        result=a*b  
    else:  
        result=a*b  
  
    return result  
  
a=findMax(2,3)  
print("The result is:",a)  
  
b=findMax(15,7)  
print("The result is:",b)  
  
c=findMax(-2,0)  
print("The result is:",c)
```

The output of the code is displayed below the code cell:

```
The result is: 6  
The result is: 105  
The result is: 0
```

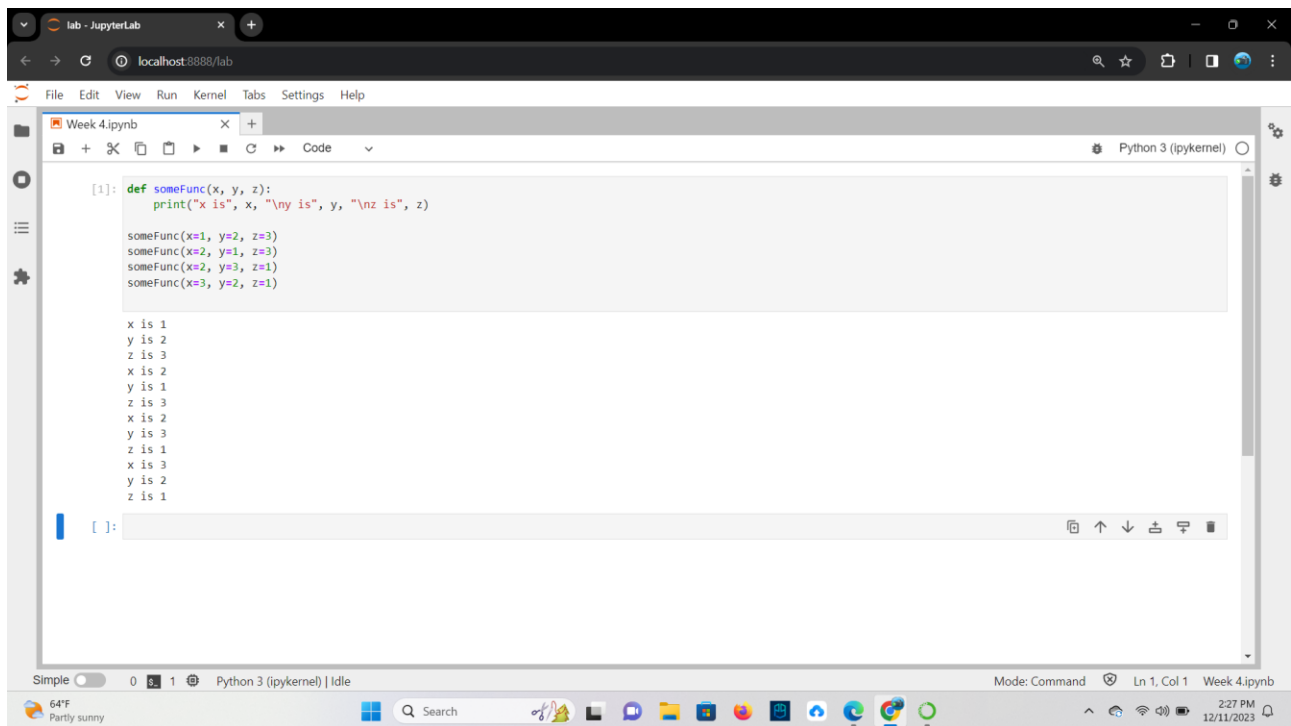
The JupyterLab interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help), a toolbar with icons for file operations, and a status bar at the bottom showing 'Simple' mode, 'Python 3 (ipykernel) | Idle', and system information (64°F, Partly sunny, 2:43 PM, 12/11/2023).

Keyword Arguments

```
def someFunc(x, y, z):  
    print("x is", x, "\ny is", y, "\nz is", z)
```

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

Ans



The screenshot shows a JupyterLab window with a browser address bar at localhost:8888/lab. The notebook file is named 'Week 4.ipynb'. The code cell contains the following Python code:

```
[1]: def someFunc(x, y, z):  
      print("x is", x, "\ny is", y, "\nz is", z)  
  
      someFunc(x=1, y=2, z=3)  
      someFunc(x=2, y=1, z=3)  
      someFunc(x=2, y=3, z=1)  
      someFunc(x=3, y=2, z=1)
```

The output of the code cell shows the following text:

```
x is 1  
y is 2  
z is 3  
x is 2  
y is 1  
z is 3  
x is 2  
y is 3  
z is 1  
x is 3  
y is 2  
z is 1
```

The JupyterLab interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help), a toolbar with icons for file operations, and a status bar at the bottom showing 'Simple' mode, 'Python 3 (ipykernel)', and the current file 'Week 4.ipynb'.

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)

Ans


```
Command Prompt - python3 x + v
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

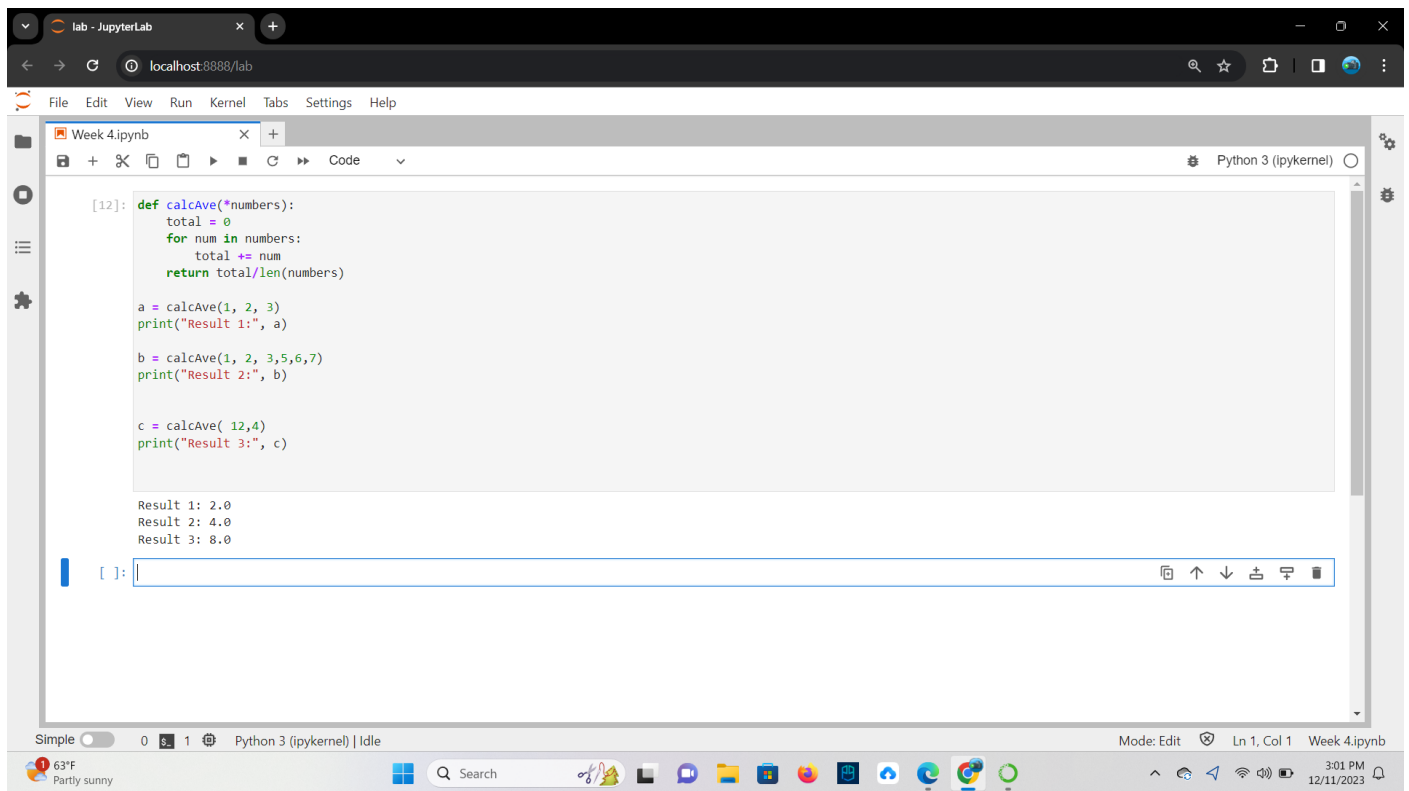
C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello", "World", sep="-")
Hello-World
>>> print("Rishav", "One", "Five", sep=",")
Rishav,One,Five
>>> print("Item", "Quantity", "Price", sep=":")
Item:Quantity:Price
>>> print("Line 1", "Line 2", "Line 3", sep="\n")
Line 1
Line 2
Line 3
>>> print("Hello" , "Python", sep=" ")
HelloPython
>>>
```

Arbitrary Length Argument Lists

```
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. Note: variadic arguments are normally defined last in the formal parameter list (and can only be followed by keyword type parameters).

Ans



Lambda Expressions

hypot = lambda a,b : math.sqrt(a * a + b * b)

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.

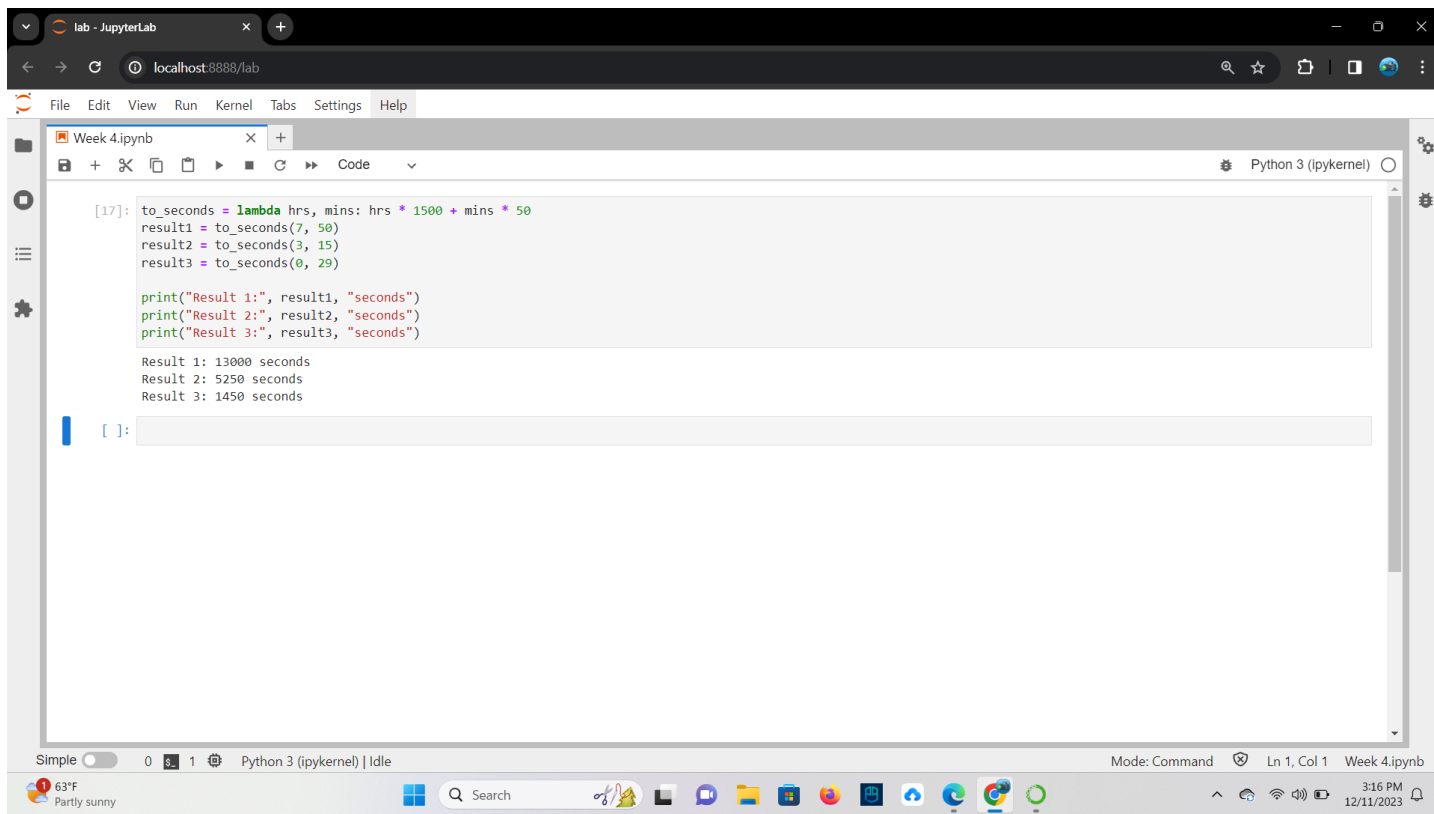
Ans

```
Command Prompt - python3 x + v
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec  4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import math
>>> hypot = lambda a, b: math.sqrt(a * a + b * b)
>>> a = type(hypot)
>>> print(a)
<class 'function'>
>>>
```

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.

Ans



The screenshot shows a JupyterLab window with a single code cell. The code defines a lambda function `to_seconds` that takes hours and minutes as arguments and returns the total seconds. It then calls this function with three different inputs: (7, 50), (3, 15), and (0, 29). The output of the cell shows the results of these calls: 13000 seconds, 5250 seconds, and 1450 seconds respectively.

```
[17]: to_seconds = lambda hrs, mins: hrs * 1500 + mins * 50
result1 = to_seconds(7, 50)
result2 = to_seconds(3, 15)
result3 = to_seconds(0, 29)

print("Result 1:", result1, "seconds")
print("Result 2:", result2, "seconds")
print("Result 3:", result3, "seconds")

Result 1: 13000 seconds
Result 2: 5250 seconds
Result 3: 1450 seconds
```

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
```

Ans

The screenshot shows a JupyterLab window with a single code cell. The code defines a lambda function `to_seconds` that takes `hrs` and `mins` as arguments and returns `hrs * 1500 + mins * 0`. It then calls this function with `1`, `2`, and `5` as arguments, storing the results in `result1`, `result2`, and `result3` respectively. The code also prints these results. A red traceback error is displayed below the code, indicating a `TypeError` because the lambda function is missing a required positional argument, `'mins'`. The error message is: `TypeError: <lambda>() missing 1 required positional argument: 'mins'`. The JupyterLab interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help), a toolbar, and a status bar at the bottom showing the current mode (Command), line and column numbers (Ln 1, Col 1), and the file name (Week 4.ipynb).

```
[20]: to_seconds = lambda hrs, mins: hrs * 1500 + mins * 0
      result1 = to_seconds(1)
      result2 = to_seconds(2)
      result3 = to_seconds(5)

      print("Result 1:", result1, "seconds")
      print("Result 2:", result2, "seconds")
      print("Result 3:", result3, "seconds")

-----
TypeError                                Traceback (most recent call last)
Cell In[20], line 2
      1 to_seconds = lambda hrs, mins: hrs * 1500 + mins * 0
----> 2 result1 = to_seconds(1)
      3 result2 = to_seconds(2)
      4 result3 = to_seconds(5)

TypeError: <lambda>() missing 1 required positional argument: 'mins'
```

Key Terminology

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.

- Module

Ans They are simply file with `.py` extension containing python code that can be imported inside another python program.

- The Python Standard Library

Ans The Python Standard Library is a collection of script modules accessible to a Python program.

- Formal Parameters

Ans Formal parameters are the variables defined by the function that receives values when the function is called.

- Actual Parameters (argument values)

Ans the actual value that is passed into the method by a caller.

- Default and Keyword Arguments

Ans Default Arguments in Python represent the function arguments that will be used if no arguments are passed to the function call. Keyword Arguments are values that, when passed into a function, are identifiable by specific parameter names.

- Lambda Expression

Ans A lambda expression is a short block of code which takes in parameters and returns a value.