

**FURTHER READING-**

# **FUNCTIONS IN PYTHON**

**ASSOC. PROFESSOR IOANNIS KYPRAIOS**

# THIS LECTURE WILL COVER...

- What is a function?
- Defining a Python function
- Function parameters & arguments
- Scope
- Returning values
- Example: Write a function to return the Fahrenheit value of a Celsius temperature number.
- **Example 2: Write a Python function that compares two input Strings and returns True if they are identical or False if they are not identical.**



# WHAT IS A FUNCTION?

A function is a unit of code that is often defined by its role within a greater code structure. Specifically, a function contains a unit of code that works on various inputs, many of which are variables, and produces concrete results involving changes to variable values or actual operations based on the inputs. (*techopedia, 2023*)

# WHAT IS A FUNCTION?

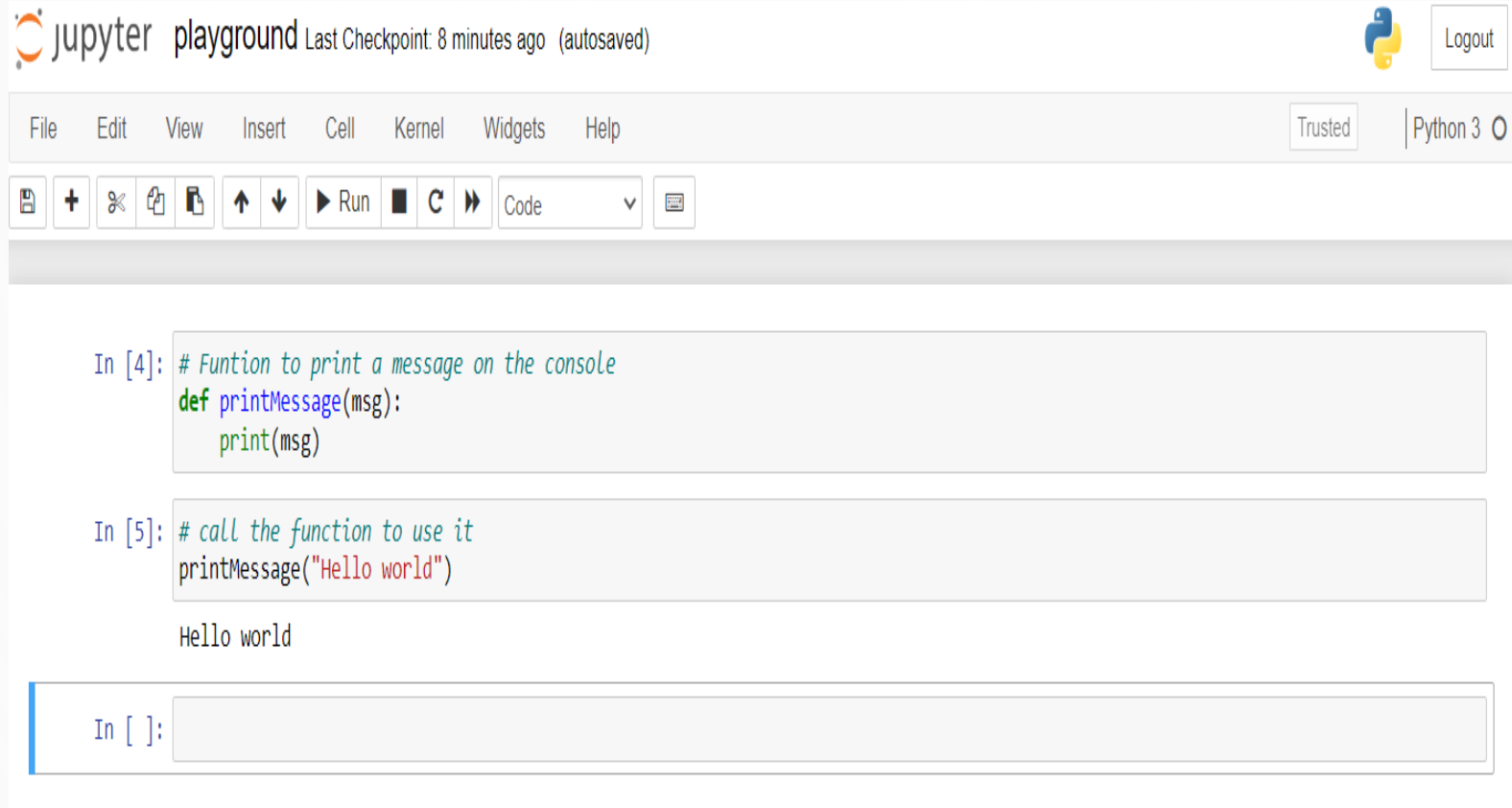
A function is a unit of code that is often defined by its role within a greater code structure. Specifically, a function contains a unit of code that works on various inputs, many of which are variables, and produces concrete results involving changes to variable values or actual operations based on the inputs. (*techopedia, 2023*)

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing. (*tutorialspoint, 2023*)

# HOW TO DEFINE A PYTHON FUNCTION

365 Data Science, “Defining a function in Python,” 4, December 2020.  
[Online]. Available: <https://www.youtube.com/watch?v=n9aDBOycsCY> .  
[Accessed September 11, 2023].

# HOW TO DEFINE A PYTHON FUNCTION



The screenshot displays the Jupyter Playground web interface. At the top, the header includes the 'jupyter playground' logo, a status message 'Last Checkpoint: 8 minutes ago (autosaved)', a Python logo, and a 'Logout' button. Below the header is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. To the right of the menu bar are 'Trusted' and 'Python 3' indicators. A toolbar below the menu bar contains icons for saving, adding, deleting, copying, pasting, undo, redo, and running code, along with a dropdown menu currently set to 'Code'. The main workspace shows two input cells. The first cell, labeled 'In [4]:', contains a comment '# Funtion to print a message on the console' (note the typo 'Funtion') followed by a function definition: `def printMessage(msg):` and an indented `print(msg)`. The second cell, labeled 'In [5]:', contains a comment '# call the function to use it' followed by the function call `printMessage("Hello world")`. Below the second cell, the output 'Hello world' is displayed. A third input cell, labeled 'In [ ]:', is currently empty and ready for input.

```
In [4]: # Funtion to print a message on the console
def printMessage(msg):
    print(msg)

In [5]: # call the function to use it
printMessage("Hello world")

Hello world

In [ ]:
```

# HOW TO DEFINE A PYTHON FUNCTION

- ❑ Start with the keyword ***def***.
- ❑ After the keyword ***def*** write the **name of your function**.
- ❑ Use brackets for writing the **function parameters** or leave them empty for no function parameters.
- ❑ Type in the next line the ***main body*** of the function i.e. the *code of what the function does*.
- ❑ To use then the function you defined, you need to ***call it*** i.e. type the name of the function with its arguments (if any).

# FUNCTION PARAMETERS

- ❑ To make better use of a function you need to include *parameters* when defining it.
- ❑ Create the function's parameters by writing them inside the brackets next to the name of the function.
- ❑ No need to specify the data type of the function parameters.
- ❑ You can have **none, one or many parameters** for a function.

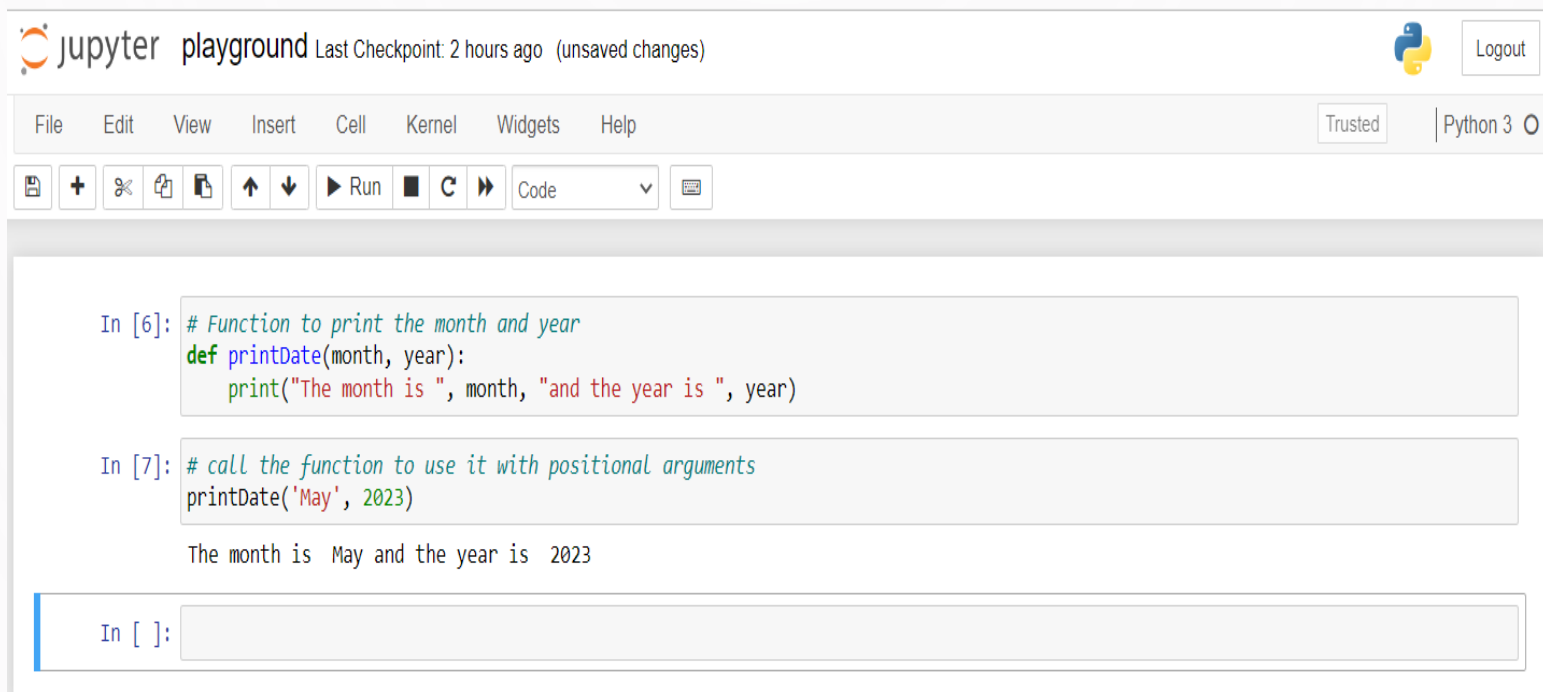


# FUNCTION ARGUMENTS

- ❑ To use a function you need to **call it by giving values to its parameters.**
- ❑ *Pass arguments to the function.*

# FUNCTION ARGUMENTS – POSITIONAL

- ❑ User provides the values for the parameters directly.
- ❑ Need to remember the order in which the positional arguments are passed to the function.



The screenshot shows a Jupyter Playground interface. At the top, it says "jupyter playground" with a status "Last Checkpoint: 2 hours ago (unsaved changes)" and a "Logout" button. Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are "Trusted" and "Python 3" indicators. Below the menu bar is a toolbar with icons for saving, adding, deleting, copying, pasting, undo, redo, and running code. The main area contains two input cells. The first cell, labeled "In [6]:", contains a function definition: 

```
# Function to print the month and year
def printDate(month, year):
    print("The month is ", month, "and the year is ", year)
```

 The second cell, labeled "In [7]:", contains a function call: 

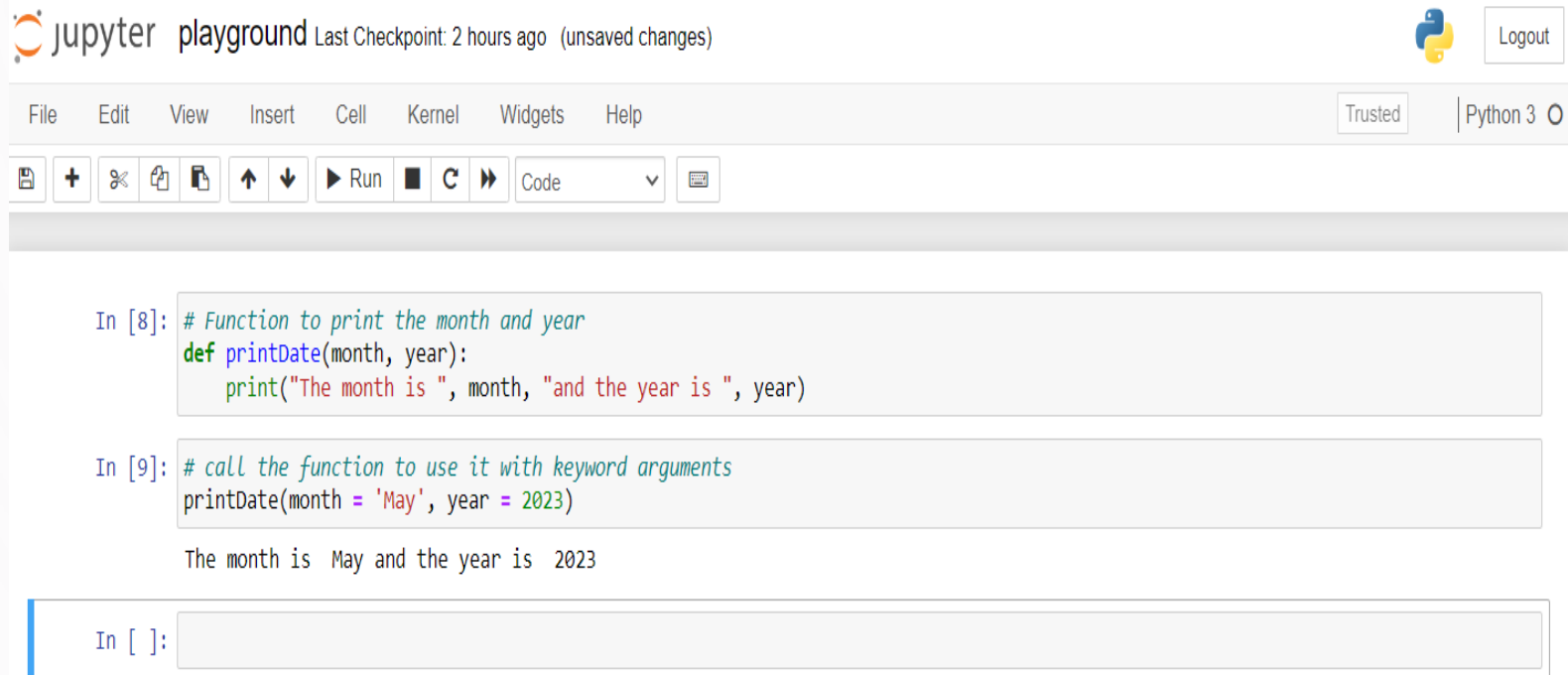
```
# call the function to use it with positional arguments
printDate('May', 2023)
```

 Below the second cell, the output is displayed: "The month is May and the year is 2023". At the bottom, there is an empty input cell labeled "In [ ]:".

# FUNCTION ARGUMENTS – KEYWORD

Pass arguments using the format:

*parameterName = value*



The image shows a Jupyter Playground interface. At the top, it says "jupyter playground" with a logo, followed by "Last Checkpoint: 2 hours ago (unsaved changes)". There is a "Logout" button and a "Python 3" indicator. Below the header is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". A toolbar contains icons for saving, adding cells, running, and other actions. The main area displays two code cells. The first cell, labeled "In [8]:", contains a function definition: 

```
# Function to print the month and year
def printDate(month, year):
    print("The month is ", month, "and the year is ", year)
```

. The second cell, labeled "In [9]:", contains a function call: 

```
# call the function to use it with keyword arguments
printDate(month = 'May', year = 2023)
```

. Below the code, the output of the second cell is displayed: "The month is May and the year is 2023". At the bottom, there is an input field for a new code cell, labeled "In [ ]:".

```
In [8]: # Function to print the month and year
def printDate(month, year):
    print("The month is ", month, "and the year is ", year)

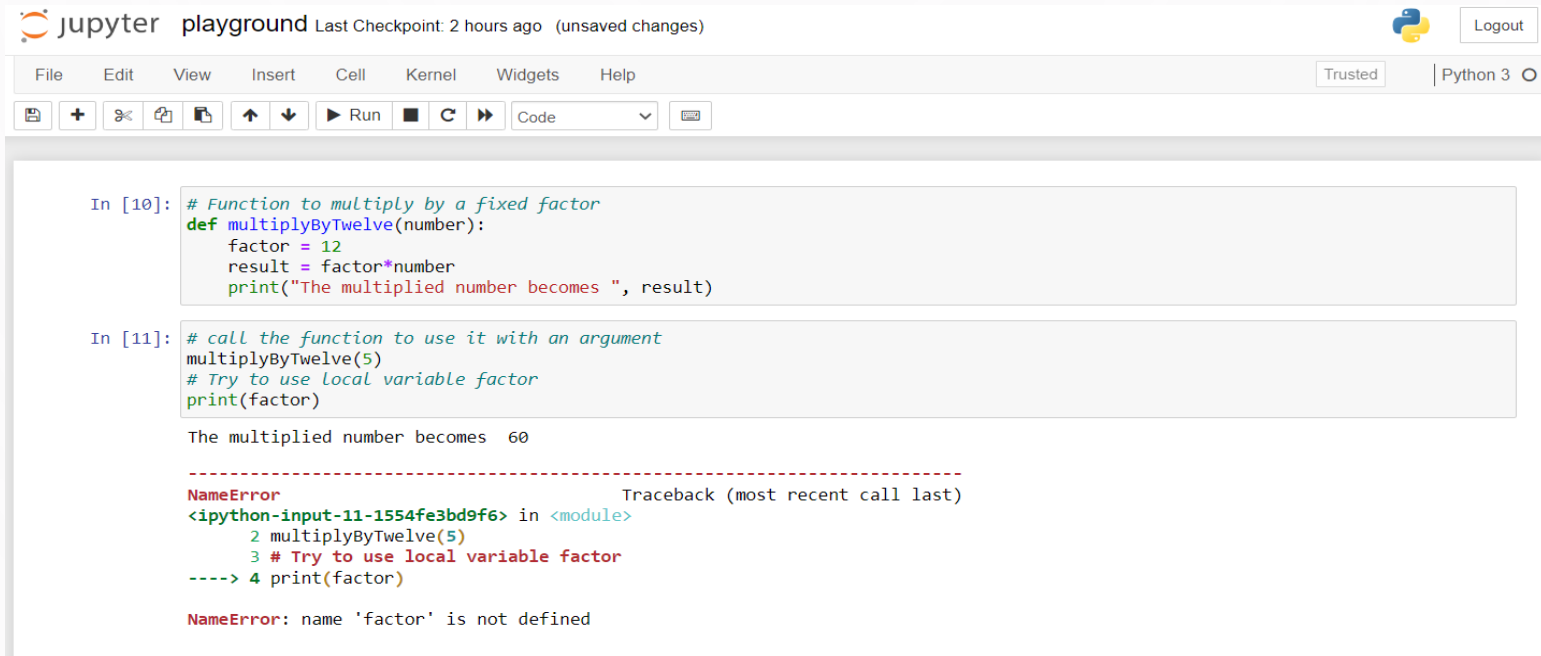
In [9]: # call the function to use it with keyword arguments
printDate(month = 'May', year = 2023)

The month is May and the year is 2023

In [ ]:
```

# SCOPE IN FUNCTIONS

- ❑ Functions have *local* and *global* scope.
- ❑ *Local* variables used by one function cannot be called by another function's body.
- ❑ ***Local* scope variables cannot be used in a *global* scope.**



The image shows a Jupyter playground interface. At the top, it says "jupyter playground" and "Last Checkpoint: 2 hours ago (unsaved changes)". There is a "Logout" button and a "Python 3" selector. Below the menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), there is a toolbar with icons for file operations, a "Run" button, and a "Code" dropdown. The main area contains two input cells. The first cell (In [10]) defines a function `multiplyByTwelve` that takes a number and returns the result of multiplying it by 12. The second cell (In [11]) calls the function with the argument 5 and then tries to print the local variable `factor`. The output shows the result of the function call (60) and then a `NameError` traceback indicating that the variable `factor` is not defined in the global scope.

```
In [10]: # Function to multiply by a fixed factor
def multiplyByTwelve(number):
    factor = 12
    result = factor*number
    print("The multiplied number becomes ", result)

In [11]: # call the function to use it with an argument
multiplyByTwelve(5)
# Try to use local variable factor
print(factor)

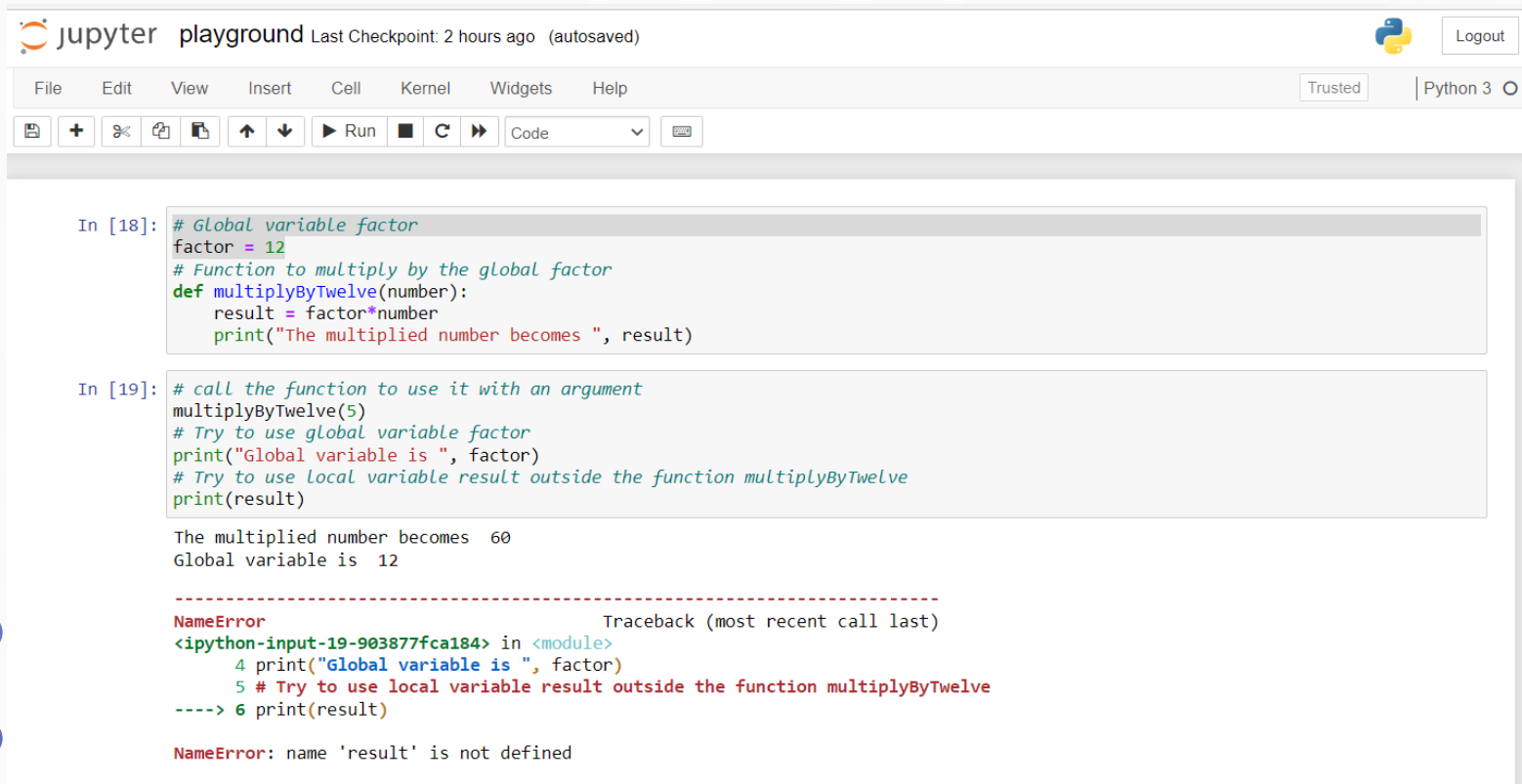
The multiplied number becomes 60

-----
NameError                                Traceback (most recent call last)
<ipython-input-11-1554fe3bd9f6> in <module>
      2 multiplyByTwelve(5)
      3 # Try to use local variable factor
----> 4 print(factor)

NameError: name 'factor' is not defined
```

# SCOPE IN FUNCTIONS

- ❑ *Global* variables can be used locally by all functions.



The image shows a Jupyter playground interface. At the top, it says "jupyter playground" and "Last Checkpoint: 2 hours ago (autosaved)". There is a "Logout" button and a "Python 3" selector. Below the menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), there is a toolbar with icons for saving, adding cells, undo, redo, and running code. The main area contains two code cells. The first cell (In [18]) defines a global variable "factor" and a function "multiplyByTwelve". The second cell (In [19]) calls the function and then tries to use the global variable "factor" and the local variable "result" outside the function. The output shows the function's result and the global variable's value, followed by a "NameError" because "result" is not defined outside the function.

```
In [18]: # Global variable factor
factor = 12
# Function to multiply by the global factor
def multiplyByTwelve(number):
    result = factor*number
    print("The multiplied number becomes ", result)

In [19]: # call the function to use it with an argument
multiplyByTwelve(5)
# Try to use global variable factor
print("Global variable is ", factor)
# Try to use local variable result outside the function multiplyByTwelve
print(result)

The multiplied number becomes 60
Global variable is 12

-----
NameError                                Traceback (most recent call last)
<ipython-input-19-903877fca184> in <module>
      4 print("Global variable is ", factor)
      5 # Try to use local variable result outside the function multiplyByTwelve
----> 6 print(result)

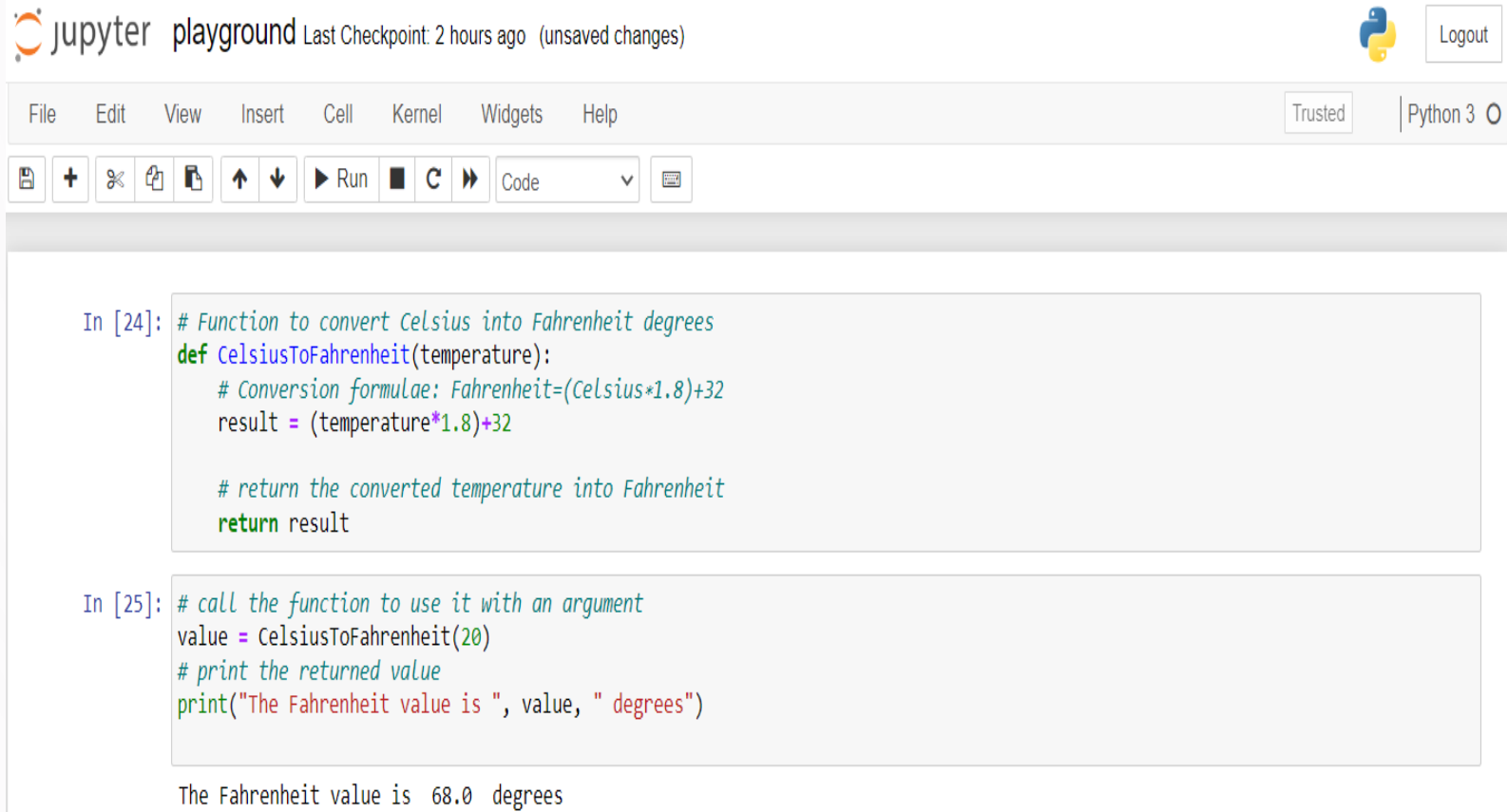
NameError: name 'result' is not defined
```

# FUNCTION RETURN

- ❑ A function does not need to always display its output directly.
- ❑ Instead, it can for example process some data and then return a value or set of values.
- ❑ The value that the function returns is called a *return value*.

## EXAMPLE:

WRITE A FUNCTION TO RETURN THE FAHRENHEIT VALUE OF A CELSIUS TEMPERATURE NUMBER



The image shows a Jupyter Playground interface. At the top, it says "jupyter playground" with a status "Last Checkpoint: 2 hours ago (unsaved changes)" and a "Logout" button. Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are "Trusted" and "Python 3" indicators. Below the menu bar is a toolbar with icons for saving, adding, deleting, and running code. The main area contains two code cells. The first cell, labeled "In [24]:", defines a function `CelsiusToFahrenheit` that takes a temperature and returns its Fahrenheit equivalent using the formula  $F = C \times 1.8 + 32$ . The second cell, labeled "In [25]:", calls the function with the argument 20 and prints the result. The output of the second cell is "The Fahrenheit value is 68.0 degrees".

```
In [24]: # Function to convert Celsius into Fahrenheit degrees
def CelsiusToFahrenheit(temperature):
    # Conversion formulae: Fahrenheit=(Celsius*1.8)+32
    result = (temperature*1.8)+32

    # return the converted temperature into Fahrenheit
    return result

In [25]: # call the function to use it with an argument
value = CelsiusToFahrenheit(20)
# print the returned value
print("The Fahrenheit value is ", value, " degrees")

The Fahrenheit value is 68.0 degrees
```

# SUMMARY

## THIS SESSION COVERED...

- What is a function?
- Defining a Python function
- Function parameters & arguments
- Scope
- Returning values
- Example: Write a function to return the Fahrenheit value of a Celsius temperature number.
- Example 2: Write a Python function that compares two input Strings and returns a message indicating whether Strings are equal or not





# BIBLIOGRAPHY & REFERENCES

- [1] P. Robbins, "Chapter 8: Functions and Modules," in Python programming for beginners, P. Robbins Ed., UK, Amazon, ISBN:979-8376161821, 2023, pp.73-85.
- [2] E. Matthes, "Chapter 8: Functions" in Python Crash Course: A hands-on, project-based introduction to programming, 2<sup>nd</sup> Edition, E. Matthes Ed., San Francisco, William Pollock No Starch Press Inc. 2019, pp.129-156.
- [3] 365 Data Science, "Defining a function in Python," 4, December 2020. [Online]. Available: <https://www.youtube.com/watch?v=n9aDBOyCsCY> . [Accessed May 11, 2023].
- [4] W3Schools, "Python Functions," [Online]. Available: [https://www.w3schools.com/python/python\\_functions.asp](https://www.w3schools.com/python/python_functions.asp) . [Accessed May 11, 2023].
- [5] M. Rouse, "What does function mean?," Techopedia, 30, March 2017. [Online]. Available: <https://www.techopedia.com/definition/25615/function> . [Accessed May 11, 2023].
- [6] Tutorialspoint, "Computer Programming – Functions," [Online]. Available: [https://www.tutorialspoint.com/computer\\_programming/computer\\_programming\\_functions.htm](https://www.tutorialspoint.com/computer_programming/computer_programming_functions.htm) . [Accessed May 11, 2023].