Welcome to this CoGrammar lecture: Functions

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

Software Engineering Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: Questions

Software Engineering Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support

- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting

- We would love your **feedback** on lectures: Feedback on Lectures

Skills Bootcamp Progression Overview

To be eligible for a certificate of completion, students must fulfil three specific criteria. These criteria ensure a high standard of achievement and alignment with the requirements for the successful completion of a Skills Bootcamp.

✓ Criterion 1 - Meeting Initial Requirements

Criterion 1 involves specific achievements within the first two weeks of the program. To meet this criterion, students need to:

- Attend a minimum of 7-8 hours per week of guided learning (lectures, workshops, or mentor calls) within the initial two-week period, for a total minimum of 15 guided learning hours (GLH), by no later than 15 September 2024.
- Successfully complete the Initial Assessment by the end of the first 14 days, by no later than 15 September 2024.

CoGrammar

Skills Bootcamp Progression Overview

✓ Criterion 2 - Demonstrating Mid-Course Progress

Criterion 2 involves demonstrating meaningful progress through the successful completion of tasks within the first half of the bootcamp.
To meet this criterion, students should:

- Complete 42 guided learning hours and the first half of the assigned tasks by the end of week 7, no later than 20 October 2024.

CoGrammar

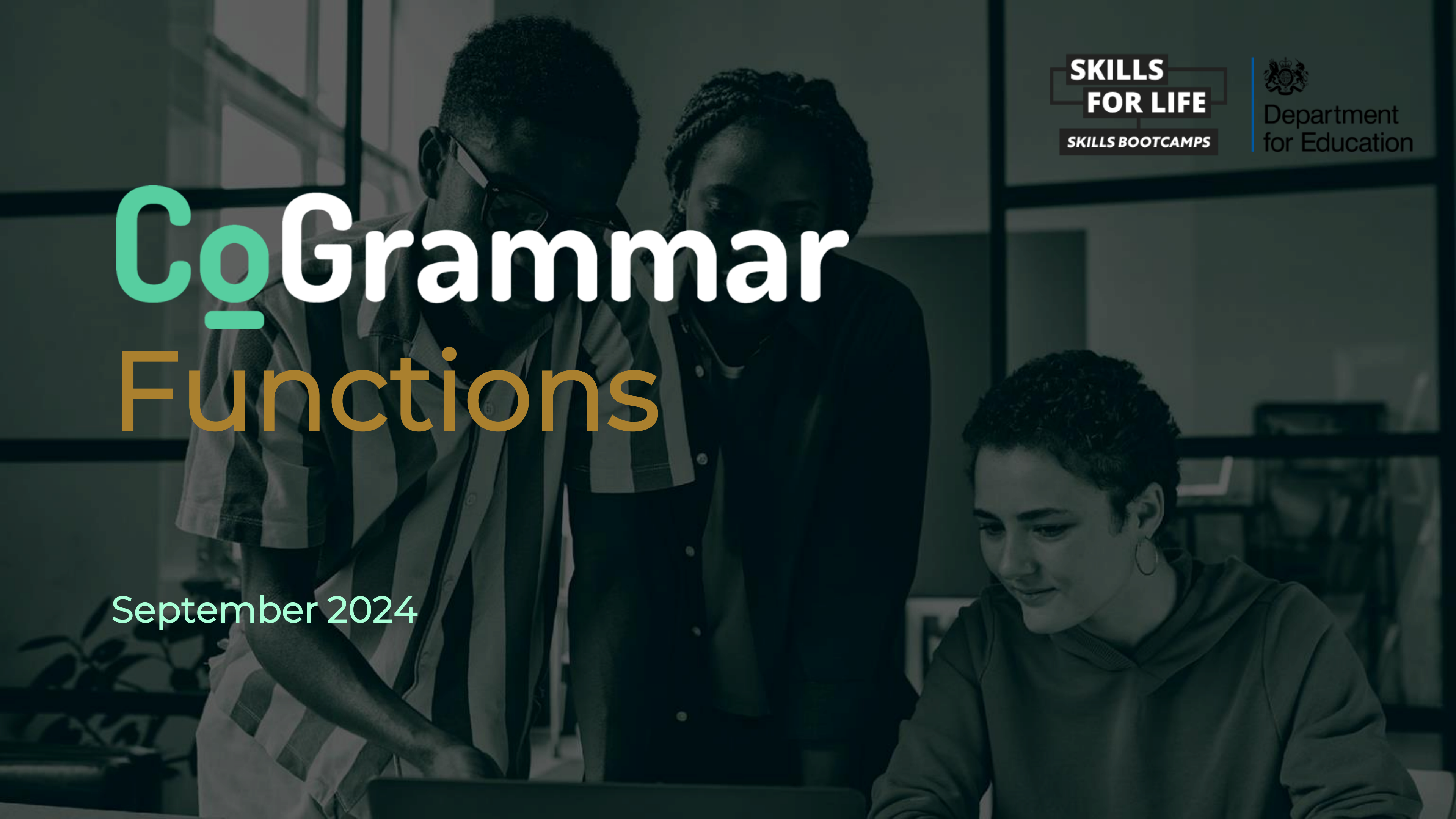
Skills Bootcamp Progression Overview

✓ Criterion 3 - Demonstrating Post-Course Progress

Criterion 3 involves showcasing students' progress after completing the course.
To meet this criterion, students should:

- Complete all mandatory tasks before the bootcamp's end date. This includes any necessary resubmissions, no later than 22 December 2024.
- Achieve at least 84 guided learning hours by the end of the bootcamp, 22 December 2024.

CoGrammar



SKILLS FOR LIFE
SKILLS BOOTCAMPS
Department for Education

CoGrammar Functions

September 2024

Learning Outcomes

- Understand the purpose and benefits of using functions in programming.
- Define and call functions with parameters and return values.
- Implement functions to modularise and organise code effectively.
- Understand the concept of variable scope and its importance in programming.
- Differentiate between local and global scope.

CoGrammar

Learning Outcomes

- Understand what a stack trace is and how it is generated during program execution.
- Interpret stack traces to debug and identify the source of errors in their code.
- Use stack traces to trace the flow of function calls and understand the sequence of execution.
- Implement the steps in debugging

CoGrammar

Poll

1. **What is the primary purpose of a function in programming?**
   A. To store large amounts of data
   B. To execute a specific task or set of tasks
   C. To improve the graphical interface of a program
   D. To connect to a database

CoGrammar

Poll

2. **Which of the following correctly defines a function in Python?**

   A. function my_function():

   B. def my_function:

   C. def my_function():

   D. my_function def():

CoGrammar

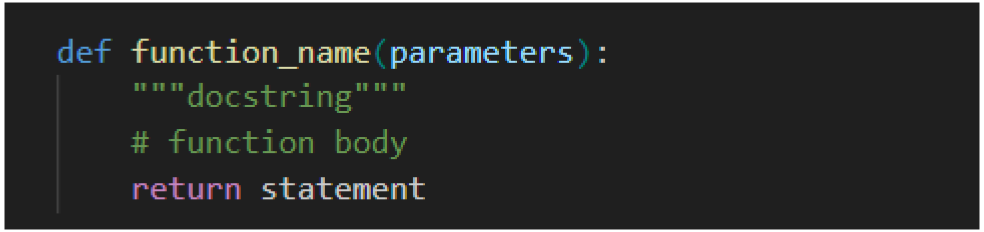
Functions

What are functions?

- Functions are reusable blocks of code that perform specific tasks.
- Called methods when used in OOP Classes.
- Functions help organise code, make it more readable, and facilitate debugging and maintenance.
- Useful for abstraction.
- Similarity to functions in maths, f(x) takes input x and produces some output.

CoGrammar

Function Syntax



```
def function_name(parameters):
    """docstring"""
    # function body
    return statement
```

- Function Name: A unique identifier for the function.
- Parameters : Inputs to the function, specified within parentheses.
- Function Body: The block of code that performs the desired task.
- Return Statement: Optional statement that specifies the value returned by the function.

Return Statement

- In Python, the `return` statement is like sending a message back to where you called a function from.
- It's a way for the function to finish its job and share its final answer with the rest of the program.

Imagine you ask a friend to solve a math problem for you. After working on it, your friend comes back to you with the solution. In Python, the `return` statement is like your friend giving you that solution. It's the way the function tells the rest of the program what answer it found.

CoGrammar

Calling functions

- Functions with one required positional input:
  - my_function(input1)
- Functions with two required positional inputs:
  - my_function(input1, input2)
- Functions with one required positional input and one optional keyword input:
  - my_function(input1, keyword_arg=input2)

CoGrammar

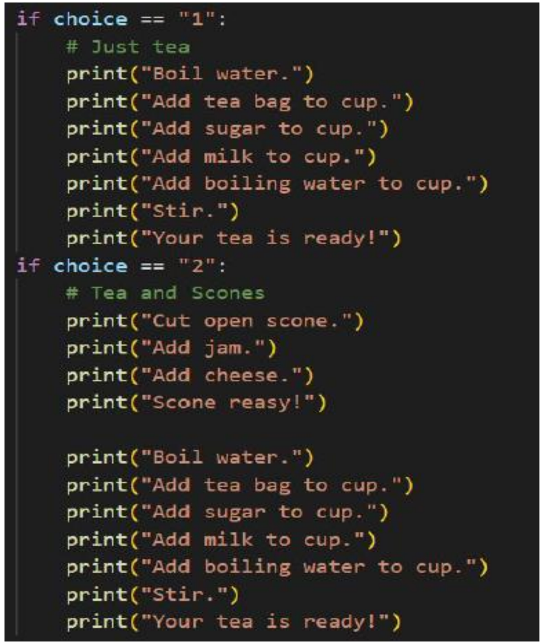
Why use functions?

- Code Reusability: Write once, use multiple times.
- Modularity: Break down complex problems into simpler pieces.
- Maintainability: Easier to update and fix issues in a modular codebase.
- Abstraction: Hide complexity and expose simple interfaces.
- Error checking/validation: Makes this easier, as you can define all rules in one place.

CoGrammar

How Functions Change Code

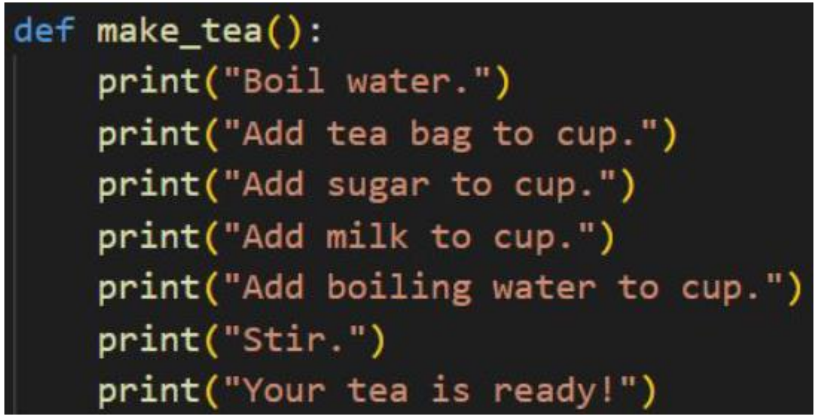
- Some original code example:



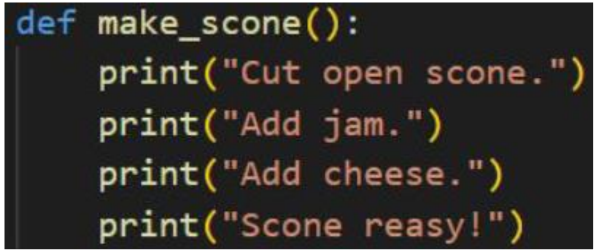
```
if choice == "1":
    # Just tea
    print("Boil water.")
    print("Add tea bag to cup.")
    print("Add sugar to cup.")
    print("Add milk to cup.")
    print("Add boiling water to cup.")
    print("Stir.")
    print("Your tea is ready!")
if choice == "2":
    # Tea and Scones
    print("Cut open scone.")
    print("Add jam.")
    print("Add cheese.")
    print("Scone reasy!")

    print("Boil water.")
    print("Add tea bag to cup.")
    print("Add sugar to cup.")
    print("Add milk to cup.")
    print("Add boiling water to cup.")
    print("Stir.")
    print("Your tea is ready!")
```

How Functions Change Code

- Abstract code into functions:



```python
def make_tea():
    print("Boil water.")
    print("Add tea bag to cup.")
    print("Add sugar to cup.")
    print("Add milk to cup.")
    print("Add boiling water to cup.")
    print("Stir.")
    print("Your tea is ready!")
```

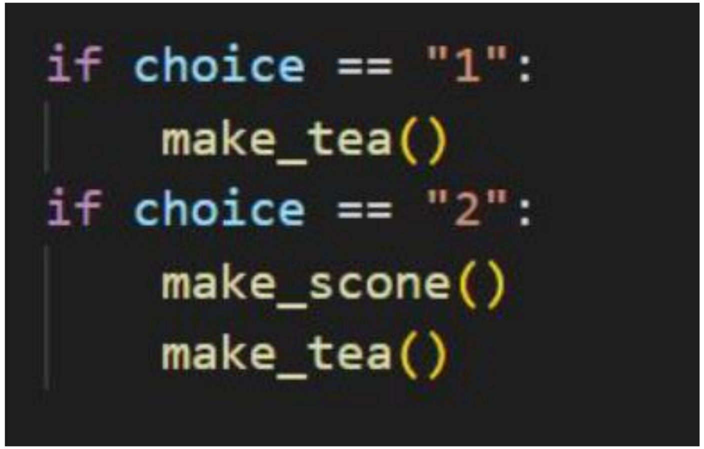


```python
def make_scone():
    print("Cut open scone.")
    print("Add jam.")
    print("Add cheese.")
    print("Scone reasy!")
```

CoGrammar

How Functions Change Code

- Updated code that is organised, more readable with code reuse implemented:

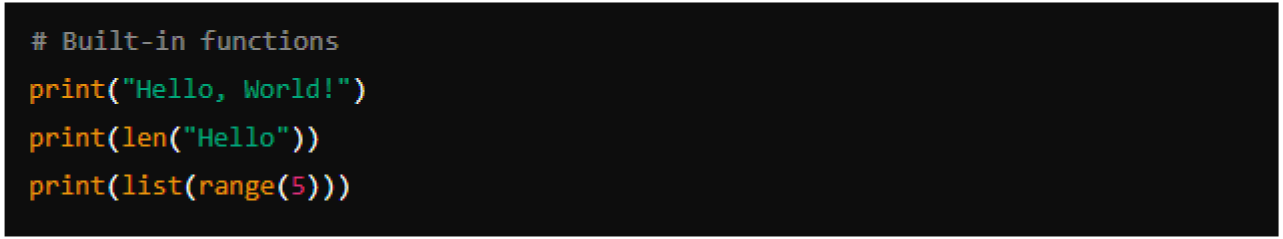


```
if choice == "1":
    make_tea()
if choice == "2":
    make_scone()
    make_tea()
```

CoGrammar

Using Built-In Functions

- Python provides numerous built-in functions for common tasks.
- Examples: print(), len(), input(), max(), min() and range()



```python
# Built-in functions
print("Hello, World!")
print(len("Hello"))
print(list(range(5)))
```

More Python Functions

- The list of functions that you can use in Python doesn’t just stop with what is built in.
- Using Pip (python package manager), you can install various packages containing modules.
- To search for packages, visit https://pypi.org/
- Some packages are already installed by default in Python, such as the Math package.
- These modules can be imported into your script using an import statement.

More Python Functions

- Let’s take a look at the maths module. Let’s say that you want to use round(), which rounds a number off.
- There are multiple ways to access this:
  - import math –or- from math import *

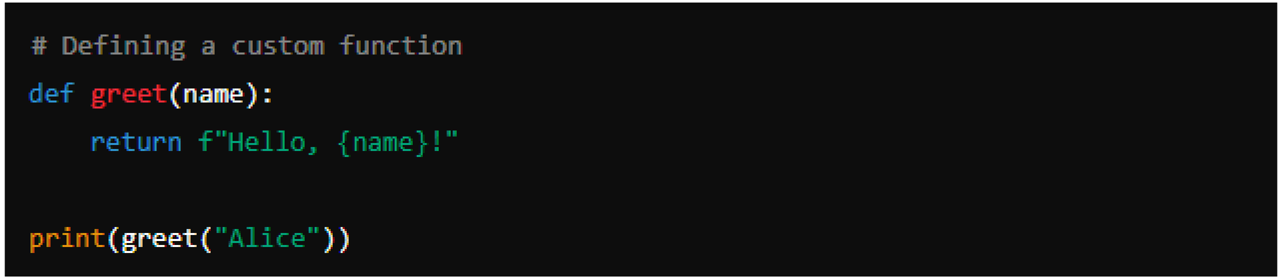
    my_result = math round my_num, 2)
  - from math import round

    my_result = round my_num, 2)

CoGrammar

Creating Custom Functions

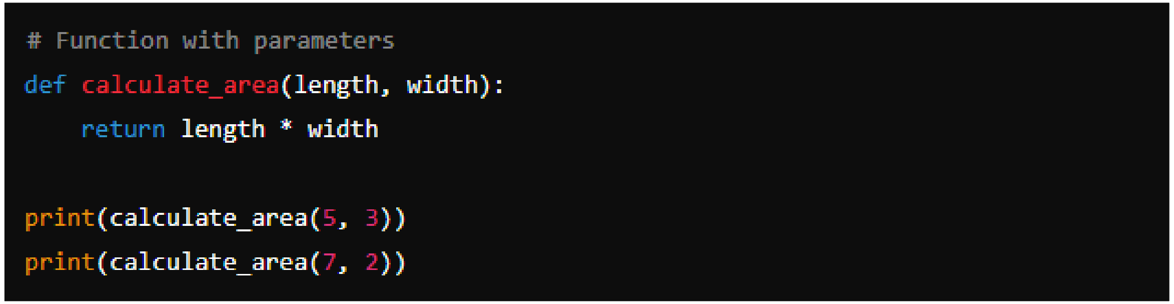
- Use the def keyword to define a function.
- Define a function to greet a user.



```python
# Defining a custom function
def greet(name):
    return f"Hello, {name}!"

print(greet("Alice"))
```

Functions with Parameters

- Functions can accept inputs through parameters.
- Length and width in the function definition are referred to as parameters, serving as placeholders that are assigned values when the function is called. 5 and 3 in the functions call, are referred to as arguments.
- Example: Calculate the area of a rectangle.

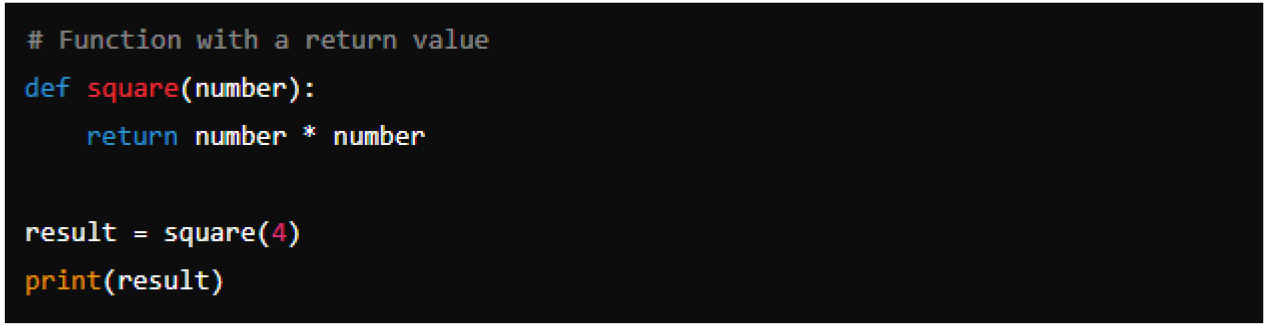


```python
# Function with parameters
def calculate_area(length, width):
    return length * width

print(calculate_area(5, 3))
print(calculate_area(7, 2))
```

CoGrammar

Functions with Return Values

- Use the return statement to return a result from a function.
- Example: Return the square of a number.



```python
# Function with a return value
def square(number):
    return number * number

result = square(4)
print(result)
```

CoGrammar

Scope

CoGrammar

What is a Scope?

- Scope refers to the accessibility of variables in different parts of a program. Variables defined within a function are typically scoped to that function, meaning they can only be accessed within that function.

CoGrammar

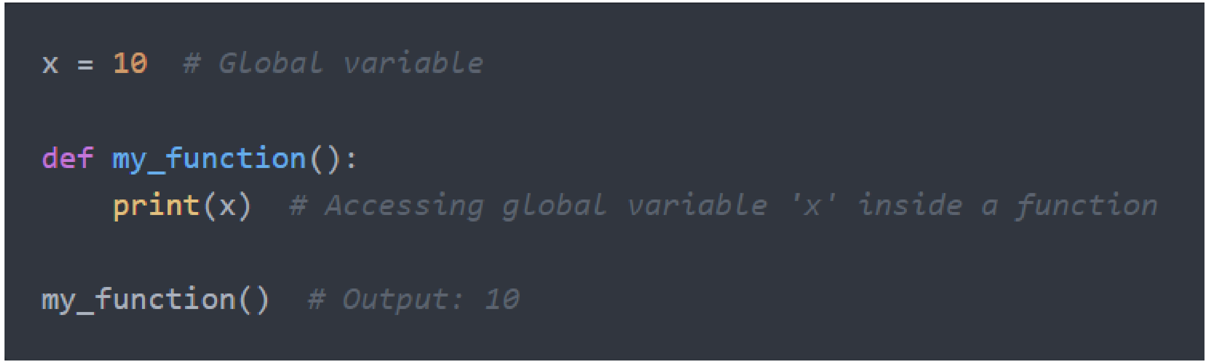
Types of Scope

- Global scope: Variables defined outside of any function, accessible throughout the entire program.
- Local scope: Variables defined within a function, only accessible within that function.
- Block scope: Some languages have block scope, where variables defined within a block (e.g., within an ‘if’ statement or a loop) are only accessible within that block.

CoGrammar

Global Scope

- Variables and functions defined outside of any function or class have global scope. They can be accessed from anywhere in the program, including inside functions and classes.

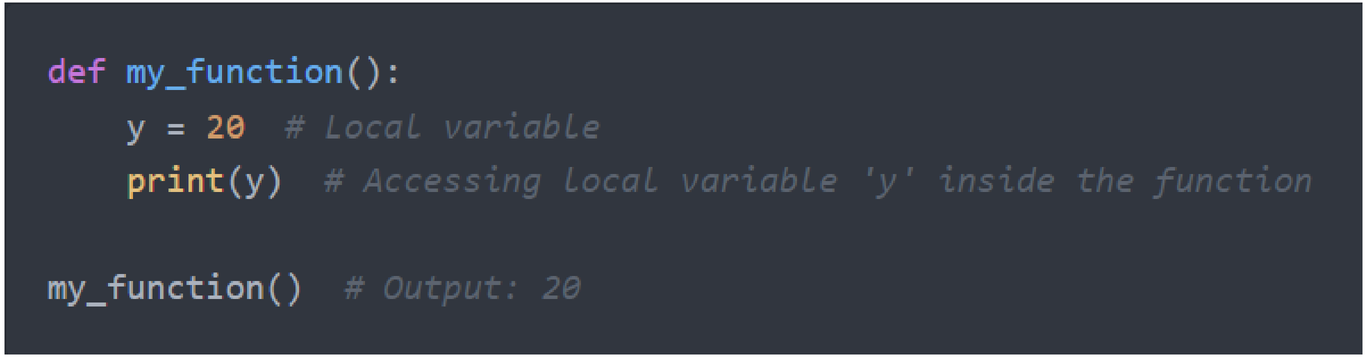


```
x = 10  # Global variable

def my_function():
    print(x)  # Accessing global variable 'x' inside a function

my_function()  # Output: 10
```

CoGrammar

Local Scope

- Variables and functions defined inside a function have local scope. They can only be accessed from within that function and are not visible outside of it.



```python
def my_function():
    y = 20  # Local variable
    print(y)  # Accessing local variable 'y' inside the function

my_function()  # Output: 20
```

Stack Traces

Stack-Traces

Stack traces are automatically generated when an error occurs in Python. They show the sequence of function calls leading up to the error.

```python
def divide(a, b):
    return a / b

def main():
    result = divide(10, 0)

main()
```

```
Traceback (most recent call last):
  File "example.py", line 7, in <module>
    main()
  File "example.py", line 5, in main
    result = divide(10, 0)
  File "example.py", line 2, in divide
    return a / b
ZeroDivisionError: division by zero
```

CoGrammar

Studying Error Messages

When an error occurs, Python captures the current state of the program's execution stack and prints a stack trace to the console. This stack trace includes:

- Error Type: The type of error that occurred (e.g., SyntaxError, NameError, TypeError).
- Error Message: A description of the error, providing additional details about what went wrong.
- Traceback: A list of function calls, starting from the point where the error occurred and going back to the initial entry point of the program.

CoGrammar

Debugging

CoGrammar

Debugging in Python

- Debugging is the process of identifying and fixing errors or bugs in a program.
- It involves analysing the behaviour of the code to understand why it is not working as expected and making the necessary corrections to resolve the issues.
- Debugging is an essential skill for programmers, and mastering it can greatly improve your ability to write reliable and efficient code.

CoGrammar

Steps in Debugging

- **Reproduce the Issue:** Start by reproducing the problem. Run the program and observe the behaviour that leads to the unexpected output.
- **Examine Error Messages:** Pay attention to any error messages or exceptions that are raised.
- **Use Print Statements:** Insert print statements at different points in the code to track the values of variables and understand the flow of execution. Print statements can help you identify which parts of the code are executing and what values are being used.

CoGrammar

Steps in Debugging...

- **Inspect Data:** Check the input data and intermediate values to ensure they are as expected. Verify that variables contain the correct data and are being manipulated correctly.
- **Use Debugging Tools:** Python provides debugging tools in the integrated development environments (IDEs) such as VS Code, which offer features like breakpoints, step-through debugging, and variable inspection to help you analyse and troubleshoot code.
- **Isolate the Problem:** Narrow down the scope of the problem by identifying the specific section of code where the error occurs. Focus on isolating the root cause of the issue rather than trying to fix everything at once.

CoGrammar

Steps in Debugging...

- **Fix the Issue:** Once you have identified the problem, make the necessary corrections to fix it. This may involve rewriting code, adjusting logic, or updating data structures to resolve the issue.
- **Test the Fix:** After making changes, test the program again to verify that the issue has been resolved. Ensure that the program behaves as expected and that the error no longer occurs.
- **Document Changes:** Document any changes made during the debugging process, including the steps taken to identify and fix the problem. This documentation can be helpful for future reference and for communicating with other developers.

Let’s take a short break

CoGrammar

Demo Time!

CoGrammar

Poll

1. **What is the difference between a positional argument and a keyword argument in a Python function?**
   A. Positional arguments must be integers, while keyword arguments must be strings.
   B. Positional arguments are passed based on their position, while keyword arguments are passed by explicitly specifying the parameter name.
   C. Positional arguments can be omitted, while keyword arguments cannot.
   D. There is no difference; they are the same.

CoGrammar

Poll

2. Which of the following is NOT a built-in Python function?

A. print()

B. len()

C. range()

D. execute()

CoGrammar

Poll

3. **How does Python handle a function that returns multiple values?**
   A. Python automatically combines them into a list.
   B. Python returns them as a single tuple.
   C. Python can only return one value; returning multiple values causes an error.
   D. Python ignores all but the last return value.

Summary

- Importance of Functions:

  Functions are essential in Python programming for organising code into reusable blocks. They improve modularity, making your code easier to read, maintain, and debug.

- Using Built-in Functions:

  Python provides a rich set of built-in functions, such as print(), len(), and sum(), that allow you to perform common tasks without needing to write extra code.

CoGrammar

Summary

- Creating and Using Custom Functions:

  You can create custom functions using the def keyword, enabling you to encapsulate specific tasks or calculations that can be reused throughout your program. Custom functions help reduce code duplication and improve overall program structure.

- Function Parameters and Return Values:

  Parameters allow you to pass information into functions, making them more flexible and reusable. Return values enable functions to output data back to the caller, allowing you to capture and use the results of computations or processes performed within the function.

Tasks To Complete

- T06 - Programming with User-defined Functions

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar