



Welcome to this session: Task Walkthrough - Programming with User-defined Functions

The session will start shortly...

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



Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:



Ian Wyles
Designated Safeguarding
Lead



Simone Botes



Nurhaan Snyman



Rafiq Manan



Ronald Munodawafa



Tevin Pitts

Scan to report a
safeguarding concern



or email the Designated
Safeguarding Lead:
Ian Wyles

safeguarding@hyperiondev.com

Skills Bootcamp Data Science

- 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](#)

Skills Bootcamp Data Science

- 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](#)**
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

Learning Outcomes

- ❖ **Design reusable functions** for solving specific computational problems.
- ❖ **Apply conditional statements** to make decisions based on user input and data.
- ❖ **Perform dynamic calculations** to address real-world challenges.
- ❖ **Combine modular programming** concepts to extend the functionality of solutions.
- ❖ **Document and troubleshoot** programs effectively.

Lecture Overview

- Presentation of the Task
- Parts of a Function
- Built-in Functions
- User-defined Functions
- Scope
- Task Walkthrough



Functions Task

Imagine you're helping a client plan their next big event, whether it's a wedding, birthday party, or corporate gathering. You'll build a Python program that calculates the total cost of an event, factoring in venue, catering, and entertainment options. In addition, your program will allow the client to compare their budget with the estimated cost and provide suggestions to optimize their spending.

- ❖ **Create Functions to Calculate Costs:** Venue, Catering, Entertainment
- ❖ **Add New Features:** Budget Comparison, Custom Recommendations
 - ❖ **Present the Results:** Breakdown of Cost




What is the purpose of defining a function in Python?

- A. To execute a block of code once
- B. To store data permanently
- C. To output data to the console
- D. To organize and reuse code efficiently



Which statement is true about if-else in Python?

- A. It enables decision-making based on specific criteria.
 - B. It allows the program to run without conditions.
 - C. It repeats code multiple times.
 - D. It is used for mathematical operations only.
- 

Functions

A block of organised, reusable code that accomplishes a specific task.

- ❖ A function can be **called repeatedly** throughout your code.
- ❖ Functions can either be **user-defined** or **built-in**.
- ❖ This helps us **minimise repeating lines of code** unnecessarily.
- ❖ The main benefits of using functions are:
 - It improves code **modularity, management** and **maintenance**.
 - It makes our code more **readable**.
 - It **reduces potential errors**.



Functions

- ❖ We've already used some functions in our code so far:
 - **print("Message")**: outputs message
 - **input("Message to user")**: Displays message to user, returns input
 - **type(value)**: returns the type of our value
 - **int(value)**: converts our value to an Integer
 - **float(value)**: converts our value to a Float
 - **bool(value)**: converts our value to a Boolean
 - **str(value)**: converts our value to a String
 - **range(start, stop, step)**: returns a list of numbers, starting at *start* and stopping before *stop* and using a *step* between the numbers
 - See more in the function dictionary...

Functions

- ❖ To declare a function in Python, we use the **def** keyword.
- ❖ We have to provide a **name** for our function (using variable naming conventions), a list of **parameters** (placeholders for function inputs) in brackets, a colon and **body** of the function indented.
- ❖ We also need to add a **return statement** for functions that return a value. This is not necessary for all functions e.g. functions that modify a state.

```
# Syntax of a user-defined function
def functionName(parameter1, parameter2):
    # function block containing statements
    # which accomplishes a specific task
    result = "Output"
    return result
```

Functions

- ❖ After defining a function, we **call or invoke** it to use it in our code.
- ❖ We call a function with its name followed by a list of **arguments** enclosed in brackets, if required by the functions.
- ❖ **Arguments** are the input values provided to the function and take the place of the **parameters** defined in the function in the **same position**.

```
# Function which calculates the sum of two numbers
def calculateSum(a, b):
    return a + b

sum1 = calculateSum(800982390, 247332) # 801229722
sum2 = calculateSum(sum1, 3) # 801229725
```

Scope

The area of visibility and accessibility of a variable in a program.

- ❖ The **scope** of a variable determines **where in the code it can be seen**.
- ❖ Functions in Python have **local scope**, meaning variables declared **inside a function** are only **accessible within** that function.
- ❖ Variables declared outside of a function, known as **global variables**, can be accessed anywhere.
- ❖ Python has different **types of scope**:

- Global Scope
- Local Scope
- Block Scope

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.

Functions Task

Imagine you're helping a client plan their next big event, whether it's a wedding, birthday party, or corporate gathering. You'll build a Python program that calculates the total cost of an event, factoring in venue, catering, and entertainment options. In addition, your program will allow the client to compare their budget with the estimated cost and provide suggestions to optimize their spending.

- ❖ **Create Functions to Calculate Costs:** Venue, Catering, Entertainment
- ❖ **Add New Features:** Budget Comparison, Custom Recommendations
 - ❖ **Present the Results:** Breakdown of Cost



What will return do in a Python function?

- A. Print the result to the console
- B. Stop the execution of a function and send a value back to the caller
- C. Execute the function again
- D. Save the function to a file



How can you pass data to a Python function?

- A. By using the `print()` statement inside the function
- B. By assigning a value to the function name
- C. By providing arguments in parentheses when calling the function
- D. By declaring variables inside the function body

Summary

- ★ **Function Design:**
Modular programming with reusable functions.
- ★ **Conditional Logic:**
Using if-else to implement dynamic pricing and discounts.
- ★ **User Interaction:**
Gathering inputs to personalize calculations.
- ★ **Lists and Loops:**
Using lists to store and calculate activity costs dynamically.
- ★ **Budget Analysis:**
Comparing costs against user-defined budgets and providing feedback.

CoGrammar

Q & A SECTION

**Please use this time to ask
any questions relating to the
topic, should you have any.**

Thank you for attending



CoGrammar



Department
for Education