Software Engineering Bootcamp

Hyperiondev

Functions

Lecture - Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- No question is daft or silly - **ask them!**
- There are Q/A sessions at the end of the session, should you wish to ask any follow-up questions.
- For all non-academic questions, please submit a query: www.hyperiondev.com/support
- Report a safeguarding incident: http://hyperiondev.com/safeguardreporting

Learning Outcomes

- Define what functions are and explain their purpose in coding.
- Utilise built-in functions such as print(), len(), and range() in their own projects.
- Create functions, defining custom behaviours and logic to solve specific problems.
- Implement functions that accept input through parameter variables.
- Design functions that return data to the caller.

Polls

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

Polls

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():

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.

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:

```
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!")
```

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

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()
```

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)

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.

Using Built-In Functions

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

```
# 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)

Creating Custom Functions

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

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

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

Functions with Parameters

- Functions can accept inputs through parameters.
- Example: Calculate the area of a rectangle.

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

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

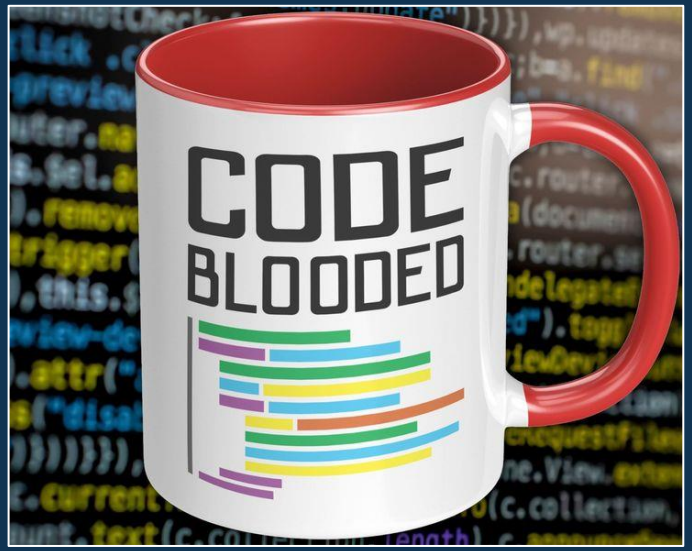
Functions with Return Values

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

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

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

Let’s get coding!



Hyperiondev

Q & A Section

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

Polls

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 argument name.

   C. Positional arguments can be omitted, while keyword arguments cannot.

   D. There is no difference; they are the same.

Polls

2. Which of the following is NOT a built-in Python function?
    A. print()
    B. len()
    C. range()
    D. execute()

Polls

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.

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.

Hyperiondev

Thank you for joining us

Take regular breaks.
Stay hydrated.
Avoid prolonged screen time.
Remember to have fun :)

Some useful links

- Official Python Documentation:
  https://docs.python.org/3/tutorial/controlflow.html#defining-functions

- Online Tutorials:
  https://realpython.com/defining-your-own-python-function/
  https://www.w3schools.com/python/python_functions.asp

- Additional Reading:
  "Automate the Boring Stuff with Python" by Al Sweigart