

# Built-in functions

INTERMEDIATE PYTHON FOR DEVELOPERS



**George Boorman**

Curriculum Manager, DataCamp

# What we'll cover

- Functions
  - Custom functions
- Modules
- Packages

# Functions we know

```
# Printing  
print("Display this as an output")
```

```
'Display this as an output'
```

```
# Checking data types  
type(print)
```

```
builtin_function_or_method
```

```
# Looping through a range of numbers  
for num in range(1, 5):  
    print(num)
```

```
1  
2  
3  
4
```

# max() and min()

```
sales = [125.97, 84.32, 99.78, 154.21, 78.50, 83.67, 111.13]
```

```
# Find the largest sale  
max(sales)
```

```
154.21
```

```
# Find the smallest sale  
min(sales)
```

```
78.5
```

# sum() and round()

```
sum(sales)
```

```
737.5799999999999
```

```
# Store total sales
```

```
total_sales = sum(sales)
```

```
# Round to two decimal places
```

```
round(total_sales, 2)
```

```
737.58
```

# Nested functions

- Call a function then call another function

```
# Store total sales
total_sales = sum(sales)

# Round to two decimal places
round(total_sales, 2)
```

737.58

- Call a function within a function

```
# Store total sales
total_sales = round(sum(sales), 2)

# Round to two decimal places
print(total_sales)
```

737.58

# len()

- Counts the number of elements

```
# Count the number of sales  
len(sales)
```

```
7
```

```
# Calculate average sales  
sum(sales) / len(sales)
```

```
105.36857142857141
```

# len()

```
# Length of a string  
len("Introduction to Programming for Developers")
```

42

```
# Length of dictionary  
len({"a": 1, "b": 2, "c": 3})
```

3

- Also works with sets and tuples
- Does not work with floats, integers, or booleans



# sorted()

```
# Sort the sales list in ascending order  
sorted(sales)
```

```
[78.5, 83.67, 84.32, 99.78, 111.13,  
125.97, 154.21]
```

```
# Sort a string alphabetically  
sorted("George")
```

```
['G', 'e', 'e', 'g', 'o', 'r']
```

# help()

```
# Get information about the sorted() function  
help(sorted)
```

Help on built-in function sorted in module builtins:

```
sorted(iterable, /, *, key=None, reverse=False)
```

Return a new list containing all items from the iterable in ascending order.

A custom key function can be supplied to customize the sort order, and the reverse flag can be set to request the result in descending order.

- Works with `int`, `str`, `{}`, `[]`, `list`, etc.

# Benefits of functions

- Perform complex tasks with less code

```
# Find total sales  
sum(sales)
```

```
737.57999999999999
```

# Benefits of functions

```
# Find total sales
# Create a variable to increment
sales_count = 0

# Loop through sales
for sale in sales:
    # Increment sales_count by each sale
    sales_count += sale
    print(sales_count)
```

- `sum()` is reusable, shorter, cleaner, and less prone to errors!

```
125.97
210.29
310.07
464.28
542.78
626.44999999999999
737.57999999999999
```

# Functions cheat sheet

Function	Returns
<code>print()</code>	Display an output, e.g., variable's values
<code>max()</code>	Find the largest value in a data structure
<code>min()</code>	Find the smallest value in a data structure
<code>sum()</code>	Add up all elements in a data structure
<code>round()</code>	Trim a float to a specified number of decimal places
<code>len()</code>	Count the number of elements in a data structure
<code>sorted()</code>	Sort elements in a data structure in ascending order
<code>help()</code>	Get information about a function, variable, or value

<sup>1</sup> <https://docs.python.org/3/library/functions.html>

# Let's practice!

INTERMEDIATE PYTHON FOR DEVELOPERS

# Modules

INTERMEDIATE PYTHON FOR DEVELOPERS



**George Boorman**

Curriculum Manager, DataCamp

# What are modules?

- Modules are Python scripts
  - Files ending with `.py`
  - Contain functions and attributes
  - Can contain other modules
- Python comes with several modules
- Help us avoid writing code that already exists!





# Python modules

- There are around 200 built-in modules
- Popular modules include:
  - `os` - for interpreting and interacting with your operating system
  - `collections` - advanced data structure types and functions
  - `string` - performing string operations
  - `logging` - to log information when testing or running software
  - `subprocess` - to run terminal commands

```
# List all files in a directory  
ls
```

- Full list of Python modules:  
<https://docs.python.org/3/py-modindex.html>

# Importing a module

```
# General syntax  
import <module_name>
```

```
# Import the os module  
import os
```

```
# Check the type  
type(os)
```

```
<class 'module'>
```

# Finding a module's functions

- Look at the documentation

```
# Call help()  
# Warning - will return a very large output!  
help(os)
```

```
Help on module os:
```

```
NAME
```

```
os - OS routines for NT or Posix depending on what system we're on.
```

```
MODULE REFERENCE
```

```
https://docs.python.org/3.10/library/os.html
```

<sup>1</sup> <https://docs.python.org/3/library/os.html#module-os>

# Getting the current working directory

```
# Using an os function  
os.getcwd()
```

```
'/home/georgeboorman/intermediate_python_for_developers'
```

- Useful if we need to refer to the directory repeatedly

```
# Assign to a variable  
work_dir = os.getcwd()
```

# Changing directory

```
# Changing directory  
os.chdir("/home/georgeboorman")
```

```
# Check the current directory  
os.getcwd()
```

```
'/home/georgeboorman'
```

```
# Confirm work_dir has not changed  
work_dir
```

```
'/home/georgeboorman/intermediate_python_for_developers'
```

# Module attributes

- Attributes have values
- Functions perform tasks
- Don't use parentheses with attributes

```
# Get the local environment  
os.environ
```

```
environ{'PATH': '/usr/local/bin',  
        'TERM': 'xterm',  
        'HOSTNAME': '097a0fe4-d6ce-4325-a6e2-1d0ce2800c2b',  
        'TZ': 'Europe/Brussels',  
        'PYTHONWARNINGS': 'ignore',  
        'LANG': 'en_US.UTF-8'  
        ...}
```

# Importing a single function from a module

- Importing a whole module can require a lot of memory
- Can import a specific function from a module

```
# Import a function from a module  
from os import chdir
```

# Importing multiple functions from a module

```
# Import multiple functions from a module
from os import chdir, getcwd

# No need to include os.
getcwd()
```

```
'/home/georgeboorman'
```

- Haven't imported `os` module so Python won't understand



# Let's practice!

INTERMEDIATE PYTHON FOR DEVELOPERS

# Packages

INTERMEDIATE PYTHON FOR DEVELOPERS



**George Boorman**

Curriculum Manager, DataCamp

# Modules are Python files

- Module = Python file
- Anyone can create a Python file!



<sup>1</sup> Image source: <https://unsplash.com/@jstrippa>

# Packages

- A collection of modules = **Package**
  - Might also hear it called a library
- Packages are publicly available and free
- First need to be downloaded from PyPI
- Then can be imported and used like modules



<sup>1</sup> <https://pypi.org/>

# Installing a package

- Terminal / Command Prompt
  - Allows us to run commands to perform tasks

```
python3 -m pip install <package_name>
```

- `python3` - Used to execute Python code from the terminal
- `pip` - Preferred Installer Program

# Installing a package

```
python3 -m pip install pandas
```



# Importing with an alias

```
# Import pandas
import pandas
```

- Need to write `pandas` before every function

```
# Import pandas using an alias
import pandas as pd
```

# Creating a DataFrame

```
# Sales dictionary
sales = {"user_id": ["KM37", "PR19", "YU88"],
         "order_value": [197.75, 208.21, 134.99]}

# Convert to a pandas DataFrame
sales_df = pd.DataFrame(sales)

sales_df
```

	user_id	order_value
0	KM37	197.75
1	PR19	208.21
2	YU88	134.99



# Reading in a CSV file

```
# Reading in a CSV file in our current directory
sales_df = pd.read_csv("sales.csv")

# Checking the data type
type(sales_df)
```

```
pandas.core.frame.DataFrame
```

# Previewing the file

```
# DataFrame method to preview the first five rows  
sales_df.head()
```

```
   user_id  order_value  
0    KM37      197.75  
1    PR19      208.21  
2    YU88      134.99  
3    NT43      153.54  
4    IW06      379.47
```

- See [DataCamp](#) for pandas courses!

# Functions versus methods

- Function = code to perform a task
- Method = a function that is specific to a data type

# Functions versus methods

```
# This is a built-in function  
sum([1, 2, 3, 4, 5])
```

```
15
```

```
# This is a pandas function  
sales_df = pd.DataFrame(sales)
```

- `.head()` won't work with other data types:
  - e.g., lists, dictionaries

```
# This is a method  
# It is specific to a DataFrame data type  
sales_df.head()
```

	user_id	order_value
0	KM37	197.75
1	PR19	208.21
2	YU88	134.99
3	NT43	153.54
4	IW06	379.47

# Let's practice!

INTERMEDIATE PYTHON FOR DEVELOPERS