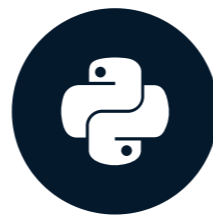


Built-in functions

INTERMEDIATE PYTHON FOR DEVELOPERS



Jasmin Ludolf

Senior Data Science Content Developer

What we'll cover

- **Chapter 1:**
 - Built-in functions, modules, and packages
- **Chapter 2 :**
 - Custom functions
- **Chapter 3:**
 - Error-handling

Introduction to Python for Developers

Functions we know

```
# Printing  
print("Password is accepted!")
```

```
Password is accepted!
```

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

```
builtin_function_or_method
```

```
# Password attempts  
for attempt in range(1, 4):  
    print("Attempt", attempt)
```

```
Attempt 1  
Attempt 2  
Attempt 3
```

Built-in functions

- Help build features with less code
- Performance monitoring dashboard ☐



max() and min()

```
# List of preparation times (minutes)
preparation_times = [19.23, 15.67, 48.57, 23.45, 12.06, 34.56, 45.67]

# Find the longest preparation time
print(max(preparation_times))
```

48.57

```
# Find the shortest preparation time
print(min(preparation_times))
```

12.06

sum() and round()

```
# Calculate the total preparation time  
print(sum(preparation_times))
```

199.21

```
# Store total time  
total_time = sum(preparation_times)  
  
# Round to one decimal place  
print(round(total_time, 1))
```

199.2

len()

- Counts the number of elements

```
# Count the number of orders  
print(len(preparation_times))
```

```
7
```

```
# Calculate average preparation time  
print(sum(preparation_times) / len(preparation_times))
```

```
28.4585714
```

len()

- Counts the number of characters, including spaces

```
# Length of a string  
print(len("Burger Hub"))
```

```
10
```

sorted()

```
# Sort a list in ascending order  
print(sorted(preparation_times))
```

```
[12.06, 15.67, 19.23, 23.45, 34.56,  
45.67, 48.57]
```

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

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

Benefits of functions

- Perform complex tasks with less code

```
# Find total preparation time  
print(sum(preparation_times))
```

```
199.21
```

Benefits of functions

```
# Find total preparation time
# Create a variable to increment
time_count = 0

# Loop through preparation times
for time in preparation_times:
    # Add each time to time_count
    time_count += time
    print(time_count)
```

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

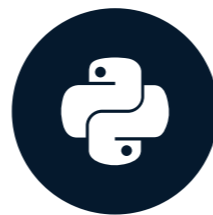
```
19.23
34.9
83.47
106.92
118.98
153.54
199.21
```

Let's practice!

INTERMEDIATE PYTHON FOR DEVELOPERS

Modules

INTERMEDIATE PYTHON FOR DEVELOPERS



Jasmin Ludolf

Senior Data Science Content Developer

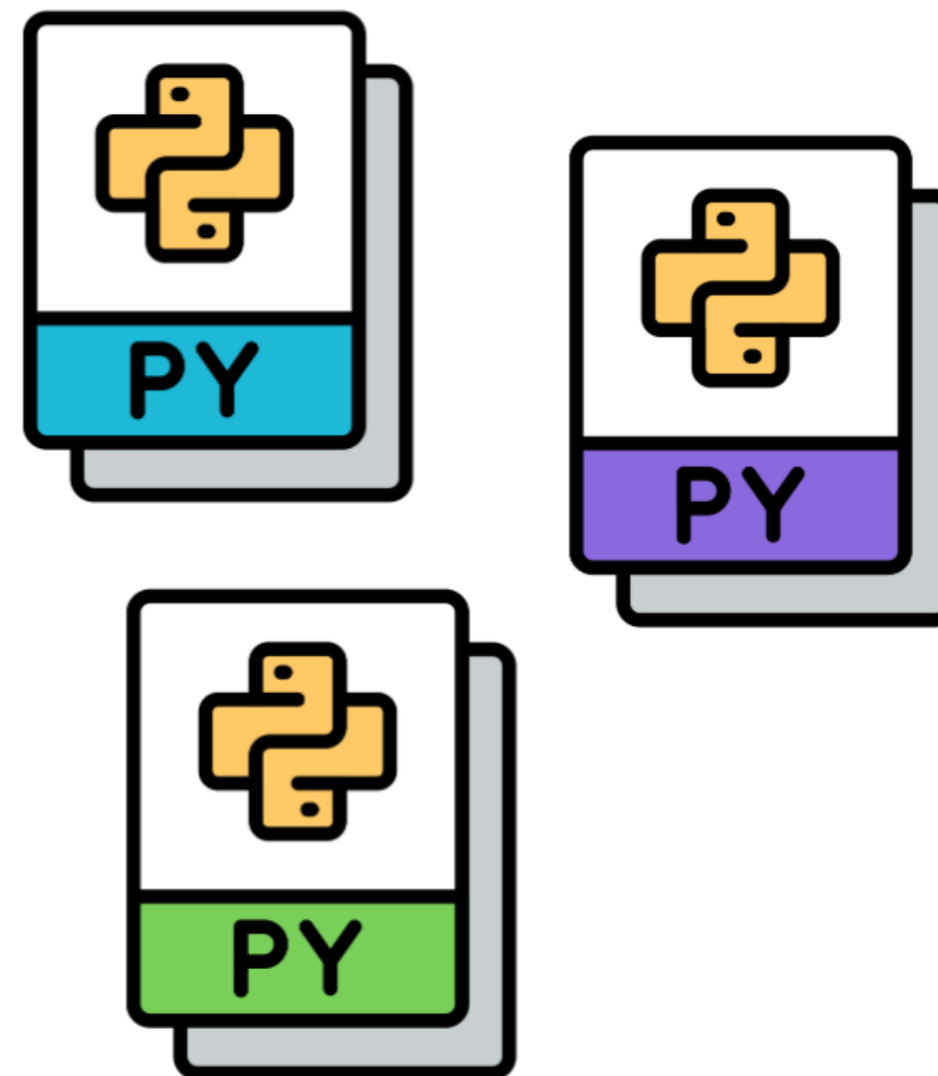
What are modules?

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



Python modules

- There are around 200 built-in modules
 - [Documentation](#)
- `os` :
 - Used for interacting with our operating system
 - Check the current directory
 - List available files
 - Access environment variables
- `string` :
 - Simplifies text processing tasks



Importing a module

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

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

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

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

Finding a module's functions

- Check the [documentation](#)

```
# Call help()
# Warning - will return a very large output!
print(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/3.12/library/os.html>

...

Getting the current working directory

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

```
/home/courses/intermediate_python_for_developers
```

- Useful if we need to reference the directory later

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

Changing directory

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

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

```
/home/courses
```

```
# Confirm work_dir has not changed  
print(work_dir)
```

```
/home/courses/intermediate_python_for_developers
```

Module attributes

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

```
# Get the local environment  
print(os.environ)
```

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

String module

```
import string
```

```
print(string.ascii_lowercase)
```

```
abcdefghijklmnopqrstuvwxyz
```

- Check if a string contains letters, numbers, or specific characters
- Handy for validating user input ☐

```
print(string.digits)
```

```
0123456789
```

```
print(string.punctuation)
```

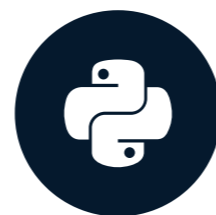
```
!"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~
```

Let's practice!

INTERMEDIATE PYTHON FOR DEVELOPERS

Packages

INTERMEDIATE PYTHON FOR DEVELOPERS



Jasmin Ludolf

Senior Data Science Content Developer

Modules are Python files

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



Packages

- A collection of modules = **Package**
 - Also called a **library**
- Publicly available and free
- Downloaded from PyPI
- Then can be imported and used like modules



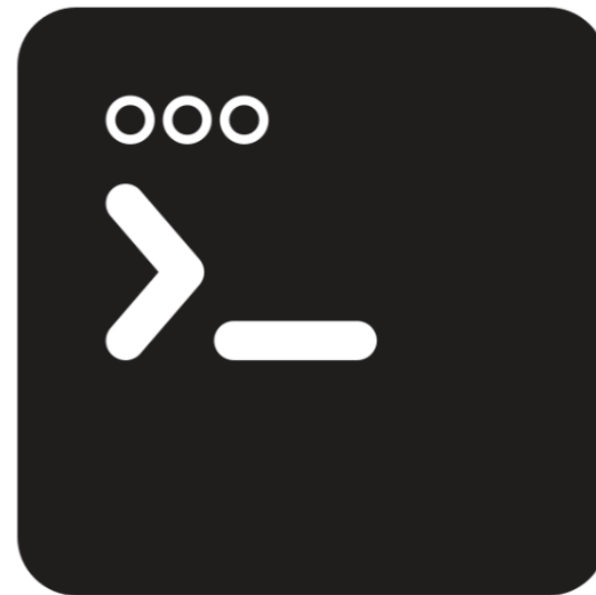
¹ <https://pypi.org/>

Installing a package

- Terminal / Command Prompt

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

- `python3` - executes Python code from the terminal
- `pip` - preferred installer



Installing a package

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



- Package for data manipulation and analysis

Importing with an alias

```
# Import pandas  
import pandas
```

- Use an alias to shorten the code

```
# 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)

print(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
print(type(sales_df))
```

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

Previewing the file

```
# DataFrame method to preview the first five rows  
print(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

Checking the file info

```
# Checking the file info  
print(sales_df.info())
```

```
RangeIndex: 3 entries, 0 to 2  
Data columns (total 2 columns):  
#   Column      Non-Null Count  Dtype  
<hr />  -----  -  
0    user_id      3 non-null     object  
1    order_value   3 non-null     float64  
dtypes: float64(1), object(1)  
memory usage: 180.0+ bytes
```

Functions versus methods

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

15

- Function = code to perform a task

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

- `.head()` only works with pandas DataFrames

```
# This is a method
print(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

- Method = a function that is specific to a data type

Let's practice!

INTERMEDIATE PYTHON FOR DEVELOPERS