ACIT 2515

Python modules and packages

Built-in packages

- You have already used packages in Python
- `import random`
- `import math`
- `import string`
- `from collections import Counter`

Today we are going to learn how to build our own

Important concepts

- Python code is written in files
- Each file is a Python module
- You can `import` other files, and/or `import` only specific symbols of a module
- Let's start with a very simple module: `my_print.py`

```
# Contents of my_print.py

MY_MESSAGE = "Hello!"

def my_print_func(text):
    print(MY_MESSAGE)
    print(text)
```

Import a module from another module

- We can import this file (module) using `import`
- Let's write another module that uses `my_print`:

```
# Contents of main.py
import my_print

def main():
    my_print.my_print_func("Example.")
    print(my_print.MY_MESSAGE)
```

Important

- The entire module `my_print` is imported
- You can access functions or variables using the `.` notation
- Everything in Python is an object!
- You can access "properties" of the module like you access "behaviours" for data types (with the `.`)
- When the module is imported, the code it contains **IS EXECUTED**!
- Make sure you use `if __name__ == "__main"` to prevent side effects.

Import only specific variables / functions

```
from my_print import MY_MESSAGE

print(MY_MESSAGE)
```

Python packages

Packages are collections of modules

Several files in a folder

A Python package is a folder with a `__init__.py` file in it

Note: packages can work without an explicit `__init__.py`, but you should always have one.

Remember: *explicit is better than implicit*

About packages

- Python packages can contain modules, and other packages

```
├── constants.py
├── display             # This is the PACKAGE display
│   ├── __init__.py
│   └── show_map.py     # It contains the MODULE show_map
├── logic               # This is the PACKAGE logic
│   ├── __init__.py
│   ├── computer        # It contains a PACKAGE computer
│   │   ├── __init__.py
│   │   └── aimbot.py   # Which contains a MODULE aimbot
│   │                   # This module has a FUNCTION shoot() in its code
│   ├── game.py         # This module belongs to the PACKAGE computer
│   └── win.py
└── main.py
```

Importing modules and packages

- You can import the whole package like you did with modules: `import logic`
- And then, use `.` to access submodules / packages

```
import logic

logic.computer.aimbot.shoot()       # In the PACKAGE logic
                                    # Look for the PACKAGE computer
                                    # And find the MODULE aimbot
                                    # Run the FUNCTION shoot
logic.game.start()                  # In the PACKAGE logic
                                    # Look for the MODULE game
                                    # Run the FUNCTION start()
```

Importing subpackages

```
import logic.game

logic.game.start()                # OK
logic.computer.aimbot.shoot()     # DOES NOT WORK (we only have logic.game)
```

Importing and renaming

- Can be convenient for long / complicated names

```
import logic.computer.aimbot as bot

bot.shoot()
```

Absolute and relative imports

- By default, Python will look for modules and packages:
  - in the current folder
  - in the folders of your virtual environment / Python installation (more on this later)
- It is sometimes desirable to tell Python to look for modules and packages in paths **relative** to the current module's path
- In that case, you must import the module / package by adding a `.` at the beginning

Relative imports are tricky. Use them only if you know what you are doing, or in `__init__.py` files.

Using `__init__.py` to allow easier access to subpackages

- When Python encounters an `import` statement and the symbol imported is a package, the `__init__.py` file will automatically be run.
- This file can import functions/variables from submodules and packages to allow for easier imports.

```
├── logic
│   ├── __init__.py
│   ├── constants
│   │   ├── __init__.py
│   │   ├── player.py       # Contains NUMBER_PLAYERS = 10
│   │   └── bot.py          # Contains AIMBOT_PRECISION = 1.0
│   └── game.py
└── main.py
```

logic/__init__.py

```
from .constants.player import NUMBER_PLAYERS
from .constants.bot import AIMBOT_PRECISION
```

main.py

```
from logic import NUMBER_PLAYERS

# Or even
from logic import NUMBER_PLAYERS, AIMBOT_PRECISION
```

Packages and paths

- When running a Python program, the working directory is **the directory where you ran the Python command**
- This can have an effect when testing / developing your programs.

```
├── my_package
│   ├── __init__.py
│   └── my_module.py
├── file.txt
└── main.py
```

```python
# Contents of my_package/my_module.py
open("file.txt", "r")
```

- `python my_package/my_module.py` : WORKS
- `python -m my_package.my_module` : WORKS (preferred way)
- `cd my_package` , and then `python my_module.py` : DOES NOT WORK (there is no "main.txt" file in the my_package folder)