13 – Python modules

Bálint Aradi

Scientific Programming in Python (2024)

https://atticlectures.net/scipro/python-2024/

Python module

- File containing routines, constants etc. which can be **used by other Python scripts**.
- Modules enable logical structuring and reusability

solvers.py

```
"""Routines for solving a linear system of equations."""
import numpy as np

def gauss_eliminate(aa, bb):
    """...
    """
    print("Linsolve: received aa:", aa)
    print("Linsolve: received bb:", bb)
    xx = np.zeros((len(bb),), dtype=float)
    return xx
```

Function within a module

Doc-string documenting the module

Using a module

- Modules can be imported by the **import** command

```
import solvers
```

- The module content can be accessed by the **dot-notation**

```
xx_gauss = solvers.gauss_eliminate(aa, bb)
```

- At import Python looks up following places for the module:
  - Local directory
  - Directories contained in the **PYTHONPATH** environment variable
  - Package directories of the Python distribution

- The **PYTHONPATH** environment variable can be set for the current BASH shell (or in **~/.bashrc** if it should be always set):

```
export PYTHONPATH="/…/directory1:/…/directory2"
```

Executable Python script

- When a Python script is run / a Python module is imported: all commands in it are executed
- In order to make also Python scripts importable, the commands to be executed should be placed into a function (usually called **main()**)
- Pythons internal **__name__** variable can be used to check, whether the code is executed as a standalone script (or imported as a module)

test_solvers.py

```
def main():
    """Main script functionality."""
    test_1()
    …

def test_1():
   …

if __name__ == '__main__':
    main()
```