Preparatory Course Informatics for Life Scientists

An Introduction to Python 4: Modules and Packages

Philipp Thiel

September 13, 2022

Modules

- So far we have written programs with only a few lines of code
- We often call that ’scripting’ and the resulting Python file a ’script’
- In fact, every Python file is a so-called **module**

- Larger projects are usually distributed over many Python files
- Such projects with many files are called **packages**
- Setting up packages instead of a single file has many advantages
  → Such projects easier to handle
  → Interesting parts can be easily reused in other projects
  → Sharing the work with others is easy with packages

- Indeed, Python is by design extensible by **importing** packages
- Official repositories exist that provide a huge variety of packages
- Creation and installation of packages is standardized and easy

Modules

plusminus.py

```python
# Module plusminus
# Our reusable module for basic arithmetics

# Addition
def plus(s1, s2):
    return s1 + s2

# Subtraction
def minus(s1, s2):
    return s1 - s2
```

Modules

plusminus.py

```python
# Module plusminus
# Our reusable module for basic arithmetics

# Addition
def plus(s1, s2):
    return s1 + s2

# Subtraction
def minus(s1, s2):
    return s1 - s2
```

myscript.py

```python
import plusminus
import plusminus as pm

print( plusminus.plus(1,2) )
print( pm.minus(1,2) )
```

Modules

- The **import** statement allows to load (import) external modules
- The variant with **as** allows to give the imported module another name
- Python searches in predefined locations for modules to be imported
- The first search location is the working directory

Modules

- The **import** statement allows to load (import) external modules
- The variant with **as** allows to give the imported module another name
- Python searches in predefined locations for modules to be imported
- The first search location is the working directory

- It is also possible to load individual elements of an external module
- The corresponding statement uses the keywords **from**

myscript.py

```python
from plusminus import plus
from plusminus import minus as m

print( plus(1,2) )
print( m(1,2) )
```

Modules

- Every Python module can be used as a stand-alone script
- This can be very handy ...
  → testing a module
  → stand-alone usage

plusminus.py

```python
# Module plusminus
# Our reusable module for basic arithmetics

# Addition
def plus(s1, s2):
    return s1 + s2

# Subtraction
def minus(s1, s2):
    return s1 - s2

if plus(12,13) == 25 and minus(12,13) == -1:
    print("Module is working correctly")
```

Modules

- When used as an imported module ’script-part’ should be excluded

plusminus.py

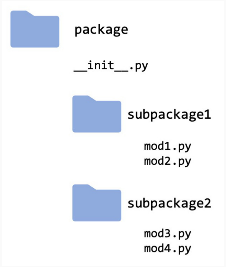
```python
# Module plusminus
# Our reusable module for basic arithmetics

# Addition
def plus(s1, s2):
    return s1 + s2

# Subtraction
def minus(s1, s2):
    return s1 - s2

# A module that is executed as a script can be identified
# The special variable __name__ is set to '__main__' in this case
if (__name__ == '__main__'):
    if plus(12,13) == 25 and minus(12,13) == -1:
        print("Module is working correctly")
```

Packages

- Large projects can have many modules (single Python files)
- Modules can be combined in a **package**
- Packages can further be structured into **subpackages**

- Packages and subpackages are just folders and subfolders
- Modules in packages can be imported using **dot** notation

- Packages can contain a module called ’**__init__.py**
- This module is loaded upon import and can be used to initialize



Packages

- Assume we created a package 'basecalc' with 'plusminus' module

myscript.py

```python
import basecalc.plusminus
print( basecalc.plusminus.plus(1,2) )
```

myscript.py

```python
import basecalc.plusminus as bcpm
print( bcpm.plus(1,2) )
```

myscript.py

```python
from basecalc.plusminus import plus
print( plus(1,2) )
```

myscript.py

```python
from basecalc.plusminus import *
print( minus(1,2) )
```

Python Standard Library

- Python comes with a many ready to use packages and modules
- These form the so-called **Python standard library**
- Thus, these modules do not have to be installed separately

- A few examples:

math	mathematical functions
gzip	guess what ;)
os	dealing with the operating system
urllib	URL handling
datetime	functionality for date and time handling
csv	reading and writing of CSV files

Diving Deeper ...

Modules and Packages

→ https://realpython.com/python-modules-packages/

→ https://www.w3schools.com/python/python_modules.asp

Python Standard Library

→ https://docs.python.org/3/library/

The Python Package Index

→ https://pypi.org/

Practice Time ... 005

1. p001: Create a package called ’basecalc’ with two modules: ’plusminus.py’ and ’multdiv.py’
2. p002: Implement plusminus.py as given in the example in the slides
3. p003: Implement multdiv.py to add new functionality for multiplication and division
4. p004: Create a subpackage ’linalg’ with a module ’euclidean.py’
5. p005: Add functionality to calculate the euclidean distance in 3D to ’euclidean.py’
   - Hint 1: think about how you could represent points in 3D
   - Hint 2: if you stumble over the square root: remember there’s a standard library - Hint 3: first implement the function call in ‘myscript.py‘

License and Contributors

Main Author
Philipp Thiel

Additional Contributors
n.a.