08c-Importing-objects

September 16, 2022

1 Importing objects

- There are many different ways to import objects into a namespace, but the most common ones are just two: `import module_name` and `from module_name import function_name`.
- The form `import module_name` finds the module `module_name` and defines a name for it in the local namespace where the import statement is executed.
- The form `from module_name import identifier` finds `module_name` and searches for an attribute (or a submodule) and stores a reference to identifier in the local namespace.
- Both forms have the option to change the name of the imported object using the `as` clause, like: `from mymodule import myfunc as better_named_func`
- When you have a structure of files starting in the root of your project, you can use the dot notation to get to the object you want to import into your current namespace, be it a package, a module, a class, a function, or anything else.

Examples

[1]:
```
import math
math.cos(0)
```

[1]:
```
1.0
```

[2]:
```
import numpy as np
np.zeros((4,5))
```

[2]:
```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

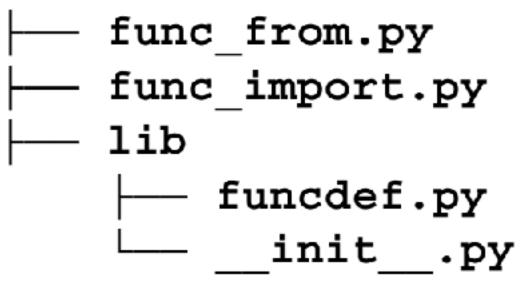
[3]:
```
from sys import path
path
```

[3]:
```
['C:\\Users\\pjsca\\GIT_Respos\\MEEC_Machine_learning\\1-python',
 'C:\\Program Files\\WindowsApps\\PythonSoftwareFoundation.Python.3.9_3.9.3568.0
_x64__qbz5n2kfra8p0\\python39.zip',
 'C:\\Program Files\\WindowsApps\\PythonSoftwareFoundation.Python.3.9_3.9.3568.0
_x64__qbz5n2kfra8p0\\DLLs',
 'C:\\Program Files\\WindowsApps\\PythonSoftwareFoundation.Python.3.9_3.9.3568.0
_x64__qbz5n2kfra8p0\\lib',
 'C:\\Users\\pjsca\\AppData\\Local\\Microsoft\\WindowsApps\\PythonSoftwareFounda
tion.Python.3.9_qbz5n2kfra8p0',
 'c:\\users\\pjsca\\git_respos\\meec_machine_learning\\venv',
 '',
 'c:\\users\\pjsca\\git_respos\\meec_machine_learning\\venv\\lib\\site-
packages',
 'c:\\users\\pjsca\\git_respos\\meec_machine_learning\\venv\\lib\\site-
packages\\win32',
 'c:\\users\\pjsca\\git_respos\\meec_machine_learning\\venv\\lib\\site-
packages\\win32\\lib',
 'c:\\users\\pjsca\\git_respos\\meec_machine_learning\\venv\\lib\\site-
packages\\Pythonwin']
```

[4]:
```python
from os import listdir as ls
ls()
```

[4]:
```
['.ipynb_checkpoints',
 '01-intro.ipynb',
 '02-bool-type.ipynb',
 '02-int-type.ipynb',
 '02-real-and-complex-numbers.ipynb',
 '02-string.ipynb',
 '03-structures.ipynb',
 '04-if-then-else.ipynb',
 '05-logic.ipynb',
 '06-for-loop.ipynb',
 '07-while-loop.ipynb',
 '08a-functions.ipynb',
 '08b-documentation.ipynb',
 '08c-Importing-objects.ipynb',
 '08d-Saving-time-and-memory.ipynb',
 '09-OOP.ipynb',
 '10-exceptions.ipynb',
 '11-Multiprocessing.ipynb',
 'exercises',
 'images',
 'lib',
 'pdfs',
 'readme.md',
 'urls.py']
```

- For some time, in order to tell Python that a folder is a package, we need to put a `__init__.py` module in it.
- `__init__.py` can have code in it as you would with any other module.
- As of Python 3.3, its presence is no longer required to make a folder be interpreted as a Python package



```
├── func_from.py
├── func_import.py
├── lib
    ├── funcdef.py
    └── __init__.py
```

Consider the following structure

where you can see the contents of the files here - lib/**init**.py - `lib/funcdef.py`

then, for instance, it is possible to do the following.

[5]:
```
from lib.funcdef import square
square(2)
```

[5]:
```
4
```

Or

[6]:
```
import lib
lib.funcdef.cube(2)
```

[6]:
```
8
```

See https://docs.python.org/3/tutorial/modules.html for the full documentation

[ ]:
```

```