

Python Basics (2/2)

Computing Methods for Experimental Physics and Data Analysis

L. Baldini

luca.baldini@pi.infn.it

Università and INFN-Pisa

Compiled on September 16, 2024



What is the Python standard library?

- ▷ Three levels hierarchy:
 - ▷ The Python core language (all you get at the interpreter startup)
 - ▷ The Python standard library (e.g., `math`)
 - ▷ An enormous number of third-party packages (e.g., `numpy`)
- ▷ The standard library is included in every Python distribution
 - ▷ And it is (slowly) evolving with time
- ▷ With third-party packages you are on your own
 - ▷ Although Anaconda solves many of the issues
 - ▷ And if you are using GNU-Linux your package manager is probably taking care of everything for you
- ▷ (Well—and of course there are your own modules, too...)
- ▷ Anything that is out of the core is loaded in memory via an `import` statement



Digression: the import system

Basics and best practices

```
1  from math import *
2  [...]
3  # Terrible: where the hell is sqrt coming from?
4  x = sqrt(2.)
5
6  from math import sqrt
7  [...]
8  # Better: if you haven't redefined sqrt this is from the math library
9  x = sqrt(2.)
10
11 import math
12 [...]
13 # Best: five more characters, but at least is clear where sqrt is coming from
14 x = math.sqrt(2.)
```

- ▷ The `$PYTHONPATH` environmental variable is your friend to control where you want to import modules from
 - ▷ You will need to tweak it when you start writing your own packages
- ▷ You will need suitable `__init__.py` files to navigate directories

More on the import system

- ▷ The import system is fairly flexible
 - ▷ Take advantage of it but don't abuse it
- ▷ This is ok...

<https://github.com/lucabaldini/cmepda/tree/master/slides/latex/snippets/import1.py>

```
1  # This is ok, and vastly recognized by the community
2  import numpy as np
3  from matplotlib import pyplot as plt
4
5  x = np.linspace(0., 10., 100)
6  y = x**2.
7  plt.plot(x, y)
```

- ▷ ...and this is a catastrophe!

<https://github.com/lucabaldini/cmepda/tree/master/slides/latex/snippets/import2.py>

```
1  from math import *
2  import logging as log
3
4  # ... 1000 lines of code in the middle
5
6  x = log(2.)
7
8  [Output]
9  Traceback (most recent call last):
10     File "snippets/import2.py", line 6, in <module>
11         x = log(2.)
12  TypeError: 'module' object is not callable
```



Overview of the standard library

time, datetime and calendar

- ▷ Collections of facilities related to date and time
 - ▷ Measure the execution time of your scripts
 - ▷ Convert from time to date and vice-versa
- ▷ This is all but trivial!
 - ▷ Ever heard of UNIX time? And UTC? And time zones?



Overview of the standard library

math

```
1 Python 3.7.4 (default, Jul  9 2019, 16:32:37)
2 [GCC 9.1.1 20190503 (Red Hat 9.1.1-1)] on linux
3 Type "help", "copyright", "credits" or "license" for more information.
4 >>> import math
5 >>> dir(math)
6 ['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__',
7  'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign',
8  'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs',
9  'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf',
10 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10',
11 'loglp', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin',
12 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
13 >>>
```

▷ If you work a lot with arrays you will end up using mostly `numpy`



Overview of the standard library

random

```
1 Python 3.7.4 (default, Jul  9 2019, 16:32:37)
2 [GCC 9.1.1 20190503 (Red Hat 9.1.1-1)] on linux
3 Type "help", "copyright", "credits" or "license" for more information.
4 >>> import random
5 >>> print(dir(random))
6 ['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST',
7 'SystemRandom', 'TWOPI', '_BuiltinMethodType', '_MethodType', '_Sequence',
8 '_Set', '__all__', '__builtins__', '__cached__', '__doc__', '__file__',
9 '__loader__', '__name__', '__package__', '__spec__', '_acos', '_bisect', '_ceil',
10 '_cos', '_e', '_exp', '_inst', '_itertools', '_log', '_os', '_pi', '_random',
11 '_sha512', '_sin', '_sqrt', '_test', '_test_generator', '_urandom', '_warn',
12 '_betavariate', '_choice', '_choices', '_expovariate', '_gammavariate', '_gauss',
13 '_getrandbits', '_getstate', '_lognormvariate', '_normalvariate', '_paretovariate',
14 '_randint', '_random', '_randrange', '_sample', '_seed', '_setstate', '_shuffle',
15 '_triangular', '_uniform', '_vonmisesvariate', '_weibullvariate']
16 >>>
```

▷ Likewise: if you work a lot with arrays you will end up using mostly
numpy



Overview of the standard library

`os`, `os.path`, `glob` and `shutil`

- ▷ Miscellaneous operating system interfaces
 - ▷ Access filesystem (access, create and copy files and directories)
 - ▷ List directory content
 - ▷ Environmental variables
 - ▷ Absolute and relative paths
 - ▷ Exec OS commands
- ▷ All of this in a cross-platform fashion
- ▷ New: take a look at `pathlib`!



Overview of the standard library

argparse

- ▷ Parser for command-line options—this is an important one!
- ▷ Ever found yourself modifying the source code and running your program with different parameters?
 - ▷ This is a terribly bad practice!
 - ▷ And git will complain about modified files :-)
- ▷ Keep the argparse documentation under your pillow!



Overview of the standard library

logging

- ▷ Ever found yourself inserting debug `print()` statements in the code when needed?
 - ▷ This is another terrible bad practice!
 - ▷ And git will complain about modified files :-)
- ▷ Imagine if there was a thing that:
 - ▷ allowed to label messages with different levels of severity (e.g., debug, info, warning, error)
 - ▷ dynamically set a global filter on the severity level (e.g., do not print debug messages)
- ▷ This thing exists and is called `logging`
- ▷ **Always prefer `logging` over `print`**
- ▷ **New: take a look at `loguru`!**



Typical layout of a Python package

Say you have a project called sample

```
1  README.rst
2  LICENSE
3  setup.py
4  requirements.txt
5  sample/__init__.py
6  sample/core.py
7  sample/helpers.py
8  docs/conf.py
9  docs/index.rst
10 tests/test_basic.py
11 tests/test_advanced.py
```

- ▷ Here is how the repository layout might look like:
 - ▷ README.rst
 - ▷ LICENSE (when in doubt use GPL v3)
 - ▷ requirements.txt (dependencies, for pip)
 - ▷ sample (actual python code, note it's the same name as the project)
 - ▷ docs (documentation)
 - ▷ tests (unit tests)
- ▷ We shall talk a lot about installation, documentation and unit tests in the second part of the course (advanced Python)



References

- ▷ <https://docs.python.org/3/library/>
- ▷ <https://pypi.org/>
- ▷ <https://docs.python.org/3/reference/import.html>
- ▷ <https://docs.python-guide.org/>
- ▷ <https://docs.quantifiedcode.com/python-anti-patterns/>