



Libraries

Scientific Programming with Python

Nicola Chiapolini



This work is licensed under the *Creative Commons Attribution-ShareAlike 3.0 License*.



More tools





Argparse

Easy parsing of commandline options using argparse.

```
import argparse

parser = argparse.ArgumentParser(description="Process some integers.")
parser.add_argument("integers", metavar="N", type=int, nargs="+",
                    help="an integer for the accumulator")
parser.add_argument("--sum", dest="accumulate", action="store_const",
                    const=sum, default=max,
                    help="sum the integers (default: find the max)")

args = parser.parse_args()
print(args.accumulate(args.integers))

$ script.py --sum 1 2 3 4
10
```



Click

Makes writing CLI (Command Line Interfaces) even easier.

```
#!/usr/bin/env python3
""" Process some integers. """
import click
@click.command()
@click.argument("integers", type=int, nargs=-1, required=True)
@click.option("--sum/--max", "-s/-m", "sum_", default=False, help="use sum or max(default: max)")
def process(sum_, integers):
    accumulator = max
    if sum_:
        accumulator = sum
    print(accumulator(integers))

if __name__ == "__main__":
    # pylint: disable=no-value-for-parameter
    process()
```

```
# $ ./script.py --sum 1 2 3 4
# 10
```



Configparser

Load configuration from files.

```
import configparser

conf = configparser.ConfigParser()
conf.read(["config.ini"])

accumulator = max
if conf["setup"].getboolean("sum", False):
    accumulator = sum

integers = [
    int(v) for v in (conf["values"]["ints"]).split(",")
]

print(accumulator(integers))
```

```
[setup]
sum = True
```

```
[values]
ints = 1,2,3,4
```



Webscraping

Requests

`requests` perform web-requests, both GET and POST (and more) to interact with anything reachable over the internet.

`BeautifulSoup` parses XML/HTML documents.

```
import requests
from bs4 import BeautifulSoup

# Re-use the connection to the server
session = requests.Session()
# Get the webpage
response = session.get(url)
# Fail early if unexpected response
response.raise_for_status()
# Read it into a datastructure that is easy to query
soup = BeautifulSoup(response.text, "lxml")
links = [a['href'] for a in soup.select("a.internal")]
```



Webscraping

Selenium

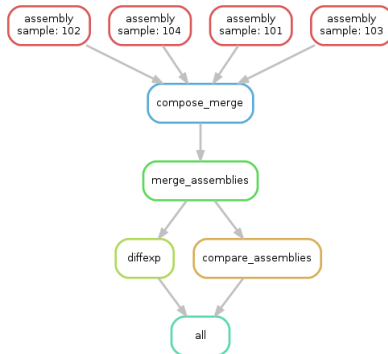
Demo



Snakemake

Automate your analysis flow using snakemake.

```
rule targets:  
    input:  
        "plots/dataset1.pdf",  
        "plots/dataset2.pdf"  
  
rule plot:  
    input:  
        "raw/{dataset}.csv"  
    output:  
        "plots/{dataset}.pdf"  
    shell:  
        "somecommand {input} {output}"
```





Subprocess

Sometimes you need to run external commands, for which no Python module exists. This can be done with the `subprocess` module.

It has recently (Python 3.7) been simplified a lot:

```
import subprocess

result = subprocess.run(["du", "-h", "."], capture_output=True)
print(result.stdout)
print(result.stderr)
# ...

result2 = subprocess.run(["cat"], capture_output=True, input=b"test")
print(result2.stdout)
# b'test'
```



Frameworks

Some fields have even created their own toolkits:

- ▶ Computational biology: <https://biopython.org/>
- ▶ Astronomy: <http://www.astropy.org/>
- ▶ High-energy particle physics: <https://github.com/scikit-hep>