Viaggio nel mondo delle librerie python

Danilo Abbasciano

danilo.abbasciano@par-tec.it

Who?

Danilo Abbasciano

Senior Cloud Engineer

More than 15 years in managing large infrastructure, coding in several languages but loves writing in Python.

I try to optimize the world!

https://www.par-tec.it

What?

Today i am going to list 25 python libraries which have been a part of my toolbelt and should be a part of yours as well.

Why?

click

Command Line Interface Creation Kit

https://click.palletsprojects.com

click: example

```
import click

@click.command()
@click.option("--count", default=1, help="Number of greetings.")
@click.option("--name", prompt="Your name",
              help="The person to greet.")

def hello(count, name):
    "Simple program that greets NAME for a total of COUNT times."
    for _ in range(count):
        click.echo("Hello, %s!" % name)

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

click: example

```
$ python hello.py --help
Usage: hello.py [OPTIONS]

  Simple program that greets NAME for a total of COUNT times.

Options:
  --count INTEGER  Number of greetings.
  --name TEXT      The person to greet.
  --help           Show this message and exit.
```

arrow

better dates and times

http://arrow.readthedocs.io/en/latest/

arrow: Why?

Python's standard library have near-complete date, time and timezone functionality but don't work very well from a usability perspective:

- Too many modules: datetime, time, calendar, dateutil, pytz and more
- Too many types: date, time, datetime, tzinfo, timedelta, relativedelta, etc
- Timezones and timestamp conversions are verbose

arrow: Example

```
>>> import arrow
>>> utc = arrow.utcnow()
>>> utc
<Arrow [2022-05-11T21:23:58.970460+00:00]>

>>> utc = utc.shift(hours=-1)
>>> utc
<Arrow [2022-05-11T20:23:58.970460+00:00]>

>>> utc.to('US/Pacific')
<Arrow [2022-05-11T13:23:58.970460-07:00]>

>>> arrow.get('2013-05-11T21:23:58.970460+00:00')
<Arrow [2022-05-11T21:23:58.970460+00:00]>
```

arrow: Example 2

```
>>> local.timestamp
1368303838

>>> local.format()
'2022-05-11 13:23:58 -07:00'

>>> local.format('YYYY-MM-DD HH:mm:ss ZZ')
'2022-05-11 13:23:58 -07:00'

>>> local.humanize()
'an hour ago'

>>> local.humanize(locale='ko_kr')
'1시간 전'
```

colorama

Makes ANSI escape character sequences for producing colored terminal text

https://pypi.org/project/colorama/

colorama: example

```
from colorama import init, Fore, Back, Style
init()

print(Fore.RED + 'some red text')
print(Back.GREEN + 'and with a green background')
print(Style.DIM + 'and in dim text')
print(Style.RESET_ALL)
print('back to normal now')
```

structlog

Makes logging faster, less painful, and more powerful by adding structure to your log entries.

https://www.structlog.org/en/stable/

structlog: usage

```
>>> import structlog

>>> log = structlog.get_logger()

>>> log.msg("greeted", whom="world", more_than_a_str=[1, 2])
2016-09-17 10:13.45 greeted   more_than_a_str=[1, 2] whom='world'
```

structlog: usage

```
>>> import structlog

>>> structlog.configure(
...     processors=[structlog.processors.JSONRenderer()]
... )

>>> structlog.get_logger().msg("hi")
{"event": "hi"}
```

bokeh

is an interactive visualization library that targets modern web browsers for presentation

https://bokeh.pydata.org/en/latest/

bokeh

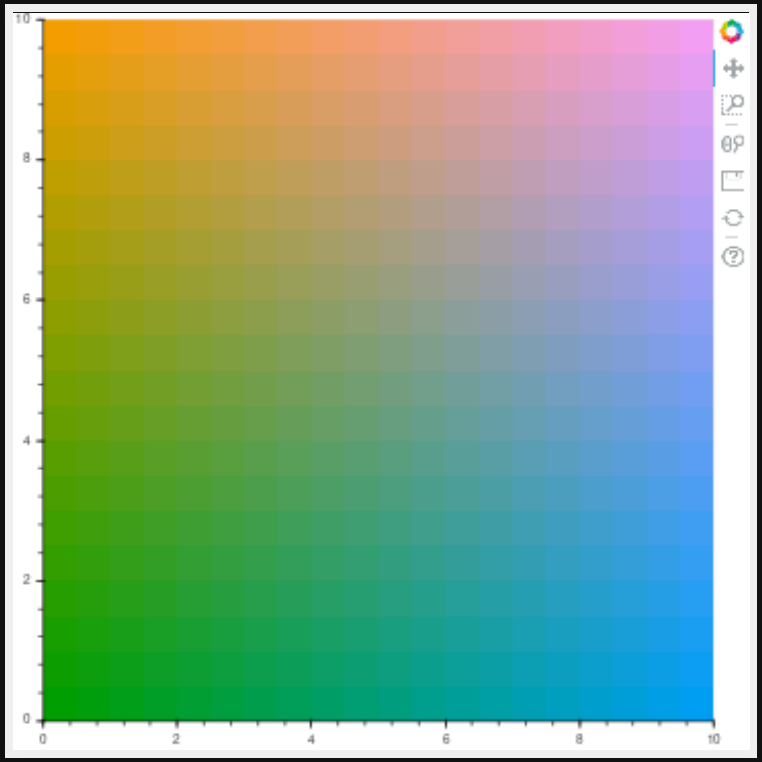
Its goal is to provide elegant, concise construction of versatile graphics, and to extend this capability with high-performance interactivity over very large or streaming datasets.

Bokeh can help anyone who would like to quickly and easily create interactive plots, dashboards, and data applications.

bokeh: example

```
from __future__ import division
import numpy as np
from bokeh.plotting import figure, show, output_file

N = 20
img = np.empty((N,N), dtype=np.uint32)
view = img.view(dtype=np.uint8).reshape((N, N, 4))
for i in range(N):
    for j in range(N):
        view[i, j, 0] = int(i/N*255)
        view[i, j, 1] = 158
        view[i, j, 2] = int(j/N*255)
        view[i, j, 3] = 255

p = figure(x_range=(0,10), y_range=(0,10))
```

boke: output



psutil

Cross-platform lib for retrieving information on process and system monitoring

https://github.com/giampaolo/psutil

psutil

psutil (process and system utilities) is a cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network, sensors).

It is useful mainly for system monitoring, profiling and limiting process resources and management of running processes.

psutil: example

```
>>> import psutil
>>> psutil.cpu_count()
4
>>> for x in range(3):
...     psutil.cpu_percent(interval=1, percpu=True)
...
[4.0, 6.9, 3.7, 9.2]
[7.0, 8.5, 2.4, 2.1]
[1.2, 9.0, 9.9, 7.2]
```

psutil: example

```
>>> psutil.disk_usage('/')
sdiskusage(total=52576092160, used=44855128064, free=5019832320,

>>> psutil.users()
[suser(name='giampaolo', terminal='pts/2', host='localhost', star
 suser(name='piuma', terminal='tty2', host='/dev/tty2', started=1
```

hug

Drastically simplify API development over multiple interfaces

http://www.hug.rest/

hug

Design and develop your API once, then expose it however your clients need to consume it. Be it locally, over HTTP, or through the command line

hug: example

```
import hug

@hug.get(examples='name=Timothy&age=26')
@hug.local()
def happy_birthday(name: hug.types.text, age: hug.types.number, h
    """Says happy birthday to a user"""
    return {'message': 'Happy {0} Birthday {1}!'.format(age, name
```

```
$ hug -f example.py
```

hug: response

```
$ curl 'http://localhost:8000/happy_birthday'
{"errors":
  {
    "name": "Required parameter 'name' not supplied",
     "age": "Required parameter 'age' not supplied"
  }
}
```

```
$ curl 'http://localhost:8000/happy_birthday?name=Danilo&age=39'
{"message": "Happy 39 Birthday Danilo!", "took": 0.0}
```

scrapy

framework for extracting the data you need from websites. In a fast, simple, yet extensible way.

https://scrapy.org/

scrapy: example

```
import scrapy

class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['https://blog.scrapinghub.com']

    def parse(self, response):
        for title in response.css('h2.entry-title'):
            yield {'title': title.css('a ::text').extract_first()}

        for next_page in response.css('div.prev-post > a'):
            yield response.follow(next_page, self.parse)
```

scrapy: selectors

```
>>> from scrapy.selector import Selector
>>> from scrapy.http import HtmlResponse
```

```
>>> body = '<html><body><span>good</span></body></html>'
>>> Selector(text=body).xpath('//span/text()').extract()
[u'good']
```

```
>>> response = HtmlResponse(url='http://example.com', body=body)
>>> Selector(response=response).xpath('//span/text()').extract()
[u'good']
```

```
>>> response.selector.xpath('//span/text()').extract()
[u'good']
```

sh

sh is a full-fledged subprocess replacement that allows you to call any program as if it were a function

https://amoffat.github.io/sh/

sh: example

```
from sh import ifconfig
print(ifconfig("wlan0"))
```

```
try:
    sh.ls("/doesnt/exist")
except sh.ErrorReturnCode_2:
    print("directory doesn't exist")
```

```
sh.wc(sh.ls("-1"), "-l")
```

ntfy

brings notification to your shell

https://ntfy.readthedocs.io/en/latest/

ntfy: example

```
$ ntfy send test

# send a notification when the command `sleep 10` finishes
# this sends the message '"sleep 10" succeeded in 0:10 minutes'
$ ntfy done sleep 10

$ ntfy -b linux send "Linux Desktop Notifications!"
```

pydantic

Data validation and settings management using python type annotations

https://pydantic-docs.helpmanual.io/

pydantic: example

```
from datetime import datetime
from typing import List, Optional
from pydantic import BaseModel

class User(BaseModel):
    id: int
    name = 'John Doe'
    signup_ts: Optional[datetime] = None
    friends: List[int] = []

external_data = {
    'id': '123',
    'signup_ts': '2019-06-01 12:22',
    'friends': [1, 2, '3'],
}
```

pydantic: example

```
from pydantic import ValidationError

try:
    User(signup_ts='broken', friends=[1, 2, 'not number'])
except ValidationError as e:
    print(e.json())
```

pydantic: example

```
[{
    "loc": ["id"],
    "msg": "field required",
    "type": "value_error.missing"
  },
  {
    "loc": ["signup_ts"],
    "msg": "invalid datetime format",
    "type": "value_error.datetime"
  },
  {
    "loc": ["friends", 2],
    "msg": "value is not a valid integer",
    "type": "type_error.integer"
  }
]
```

networkx

for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

https://networkx.github.io/

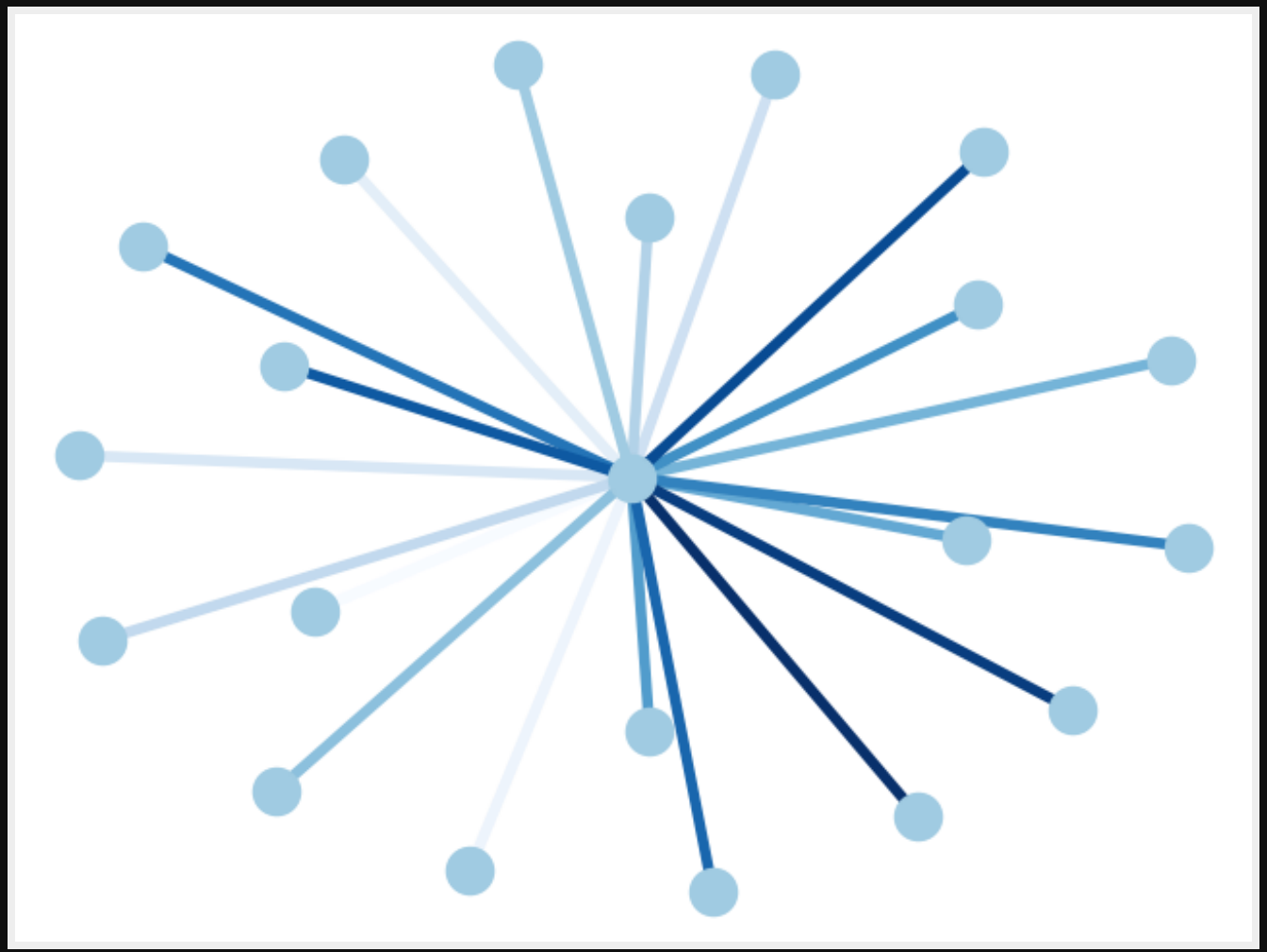
networkx: example

```
>>> import networkx as nx
>>> G = nx.Graph()

>>> G.add_node(1)
>>> G.add_nodes_from([2, 3])

>>> G.add_edge(1, 2)
>>> G.add_edges_from([(1, 2), (1, 3)])
```

networkx: example

```
import matplotlib.pyplot as plt
import networkx as nx

G = nx.star_graph(20)
pos = nx.spring_layout(G)
colors = range(20)
nx.draw(G, pos, node_color='#A0CBE2', edge_color=colors,
        width=4, edge_cmap=plt.cm.Blues, with_labels=False)
plt.show()
```

networkx: example



prompt-toolkit

building powerful interactive command lines and terminal applications

prompt-toolkit: features

- Syntax highlighting
- Multi-line input editing
- Advanced code completion
- Selecting text for copy/paste. (Both Emacs and Vi style)
- Mouse support for cursor positioning and scrolling
- Auto suggestions. (Like fish shell)

prompt-toolkit: autocompletion example

```
from prompt_toolkit import prompt
from prompt_toolkit.completion import WordCompleter

items = ['<html>', '<body>', '<head>', '<title>']
html_completer = WordCompleter(items)

text = prompt('Enter HTML: ', completer=html_completer)
print('You said: %s' % text)
```

prompt-toolkit: autocompletion

asciimatics

help people create simple ASCII animations on any platform

http://asciimatics.readthedocs.io

asciimatics: Why?

- 256 colour terminals
- Cursor positioning
- Keyboard input (without blocking or echoing)
- Mouse input
- Detecting and handling when the console resizes
- Screen scraping

asciimatics: Why?

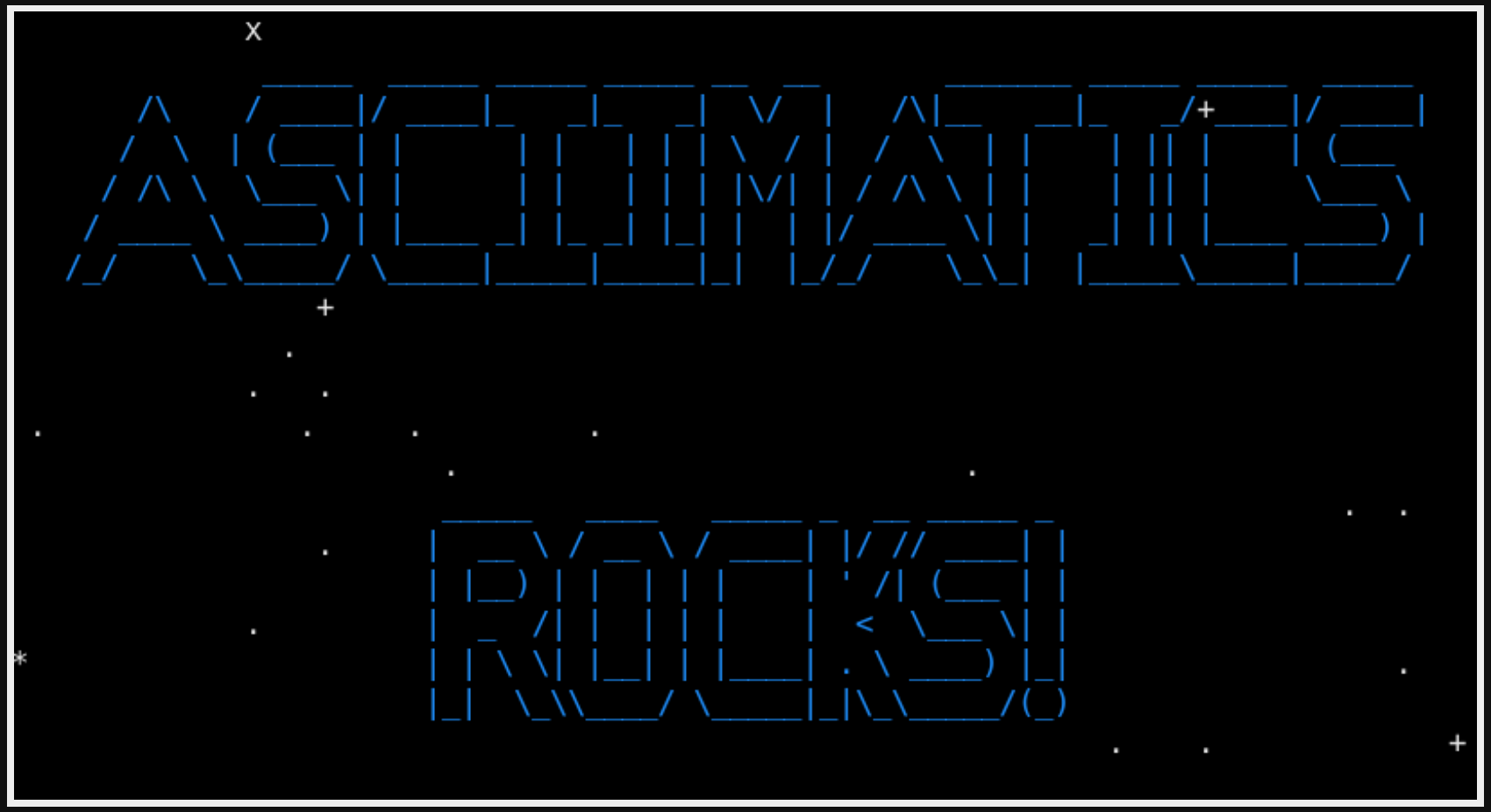
In addition, it provides more complex features including:

- Anti-aliased ASCII line-drawing
- Image to ASCII conversion - including JPEG and GIF formats
- Many animation effects - e.g. sprites, particle systems, banners, etc.
- Various widgets for text UIs - e.g. buttons, text boxes, radio...

asciimatics: example

```
from asciimatics.screen import Screen
from asciimatics.scene import Scene
from asciimatics.effects import Cycle, Stars
from asciimatics.renderers import FigletText

def demo(screen):
    effects = [
        Cycle(
            screen,
            FigletText("ASCIIMATICS", font='big'),
            screen.height // 2 - 8),
        Cycle(
            screen,
            FigletText("ROCKS!", font='big'),
            screen.height // 2 + 3),
        Stars(screen, (screen.width + screen.height) // 2)
```

asciimatics: example



prettytable

represent tabular data in visually appealing ASCII tables

https://pypi.org/project/PrettyTable/

prettytable: example

```
from prettytable import PrettyTable
x = PrettyTable()

x.field_names(["City name", "Area", "Population", "Annual Rainfal
x.add_row(["Adelaide",1295, 1158259, 600.5])
x.add_row(["Brisbane",5905, 1857594, 1146.4])
x.add_row(["Darwin", 112, 120900, 1714.7])
x.add_row(["Hobart", 1357, 205556, 619.5])
x.add_row(["Sydney", 2058, 4336374, 1214.8])
x.add_row(["Melbourne", 1566, 3806092, 646.9])
x.add_row(["Perth", 5386, 1554769, 869.4])
```

prettytable: print(x)

```
+-----------+------+------------+-----------------+
| City name | Area | Population | Annual Rainfall |
+-----------+------+------------+-----------------+
| Adelaide  | 1295 |  1158259   |      600.5      |
| Brisbane  | 5905 |  1857594   |      1146.4     |
| Darwin    | 112  |   120900   |      1714.7     |
| Hobart    | 1357 |   205556   |      619.5      |
| Melbourne | 1566 |  3806092   |      646.9      |
| Perth     | 5386 |  1554769   |      869.4      |
| Sydney    | 2058 |  4336374   |      1214.8     |
+-----------+------+------------+-----------------+
```

TheFuzz

Fuzzy string matching like a boss.

It uses Levenshtein Distance to calculate the differences between sequences in a simple-to-use package.

https://github.com/seatgeek/thefuzz

theFuzz: example

```
>>> from thefuzz import fuzz
>>> fuzz.ratio("fuzzy wuzzy was a bear", "wuzzy fuzzy was a bear")
    91
```

progressbar2

A text progress bar is typically used to display the progress of a long running operation, providing a visual cue that processing is underway

https://progressbar-2.readthedocs.io/

progressbar2: example

```
import time
import progressbar

for i in progressbar.progressbar(range(100)):
    time.sleep(0.02)
```

```
19% (19 of 100) |##                  | Elapsed Time: 0:00:00 ETA: 0:00:01
```

doctest

Searches for pieces of text that look like interactive Python sessions, and then executes those sessions to verify that they work exactly as shown

https://docs.python.org/3/library/doctest.html

doctest: example

```
The ``example`` module
======================

Using ``factorial``
-------------------

This is an example text file in reStructuredText format.
First import ``factorial`` from the ``example`` module:

    >>> from example import factorial

Now use it:

    >>> factorial(6)
    120
```

```
import doctest
doctest.testfile("example.txt")
```

doctest: example

```
File "./example.txt", line 14, in example.txt
Failed example:
    factorial(6)
Expected:
    120
Got:
    720
```

pillow

friendly PIL (Python Imaging Library) fork

https://pillow.readthedocs.io/

pillow: creating thumbnails

```
from PIL import Image

size = (128, 128)

try:
    im =  Image.open("Image.png")
except:
    print "Unable to load image"

im.thumbnail(size)
im.save("image.jpeg")
im.show()
```

requests-futures

Small add-on for the python requests http library.

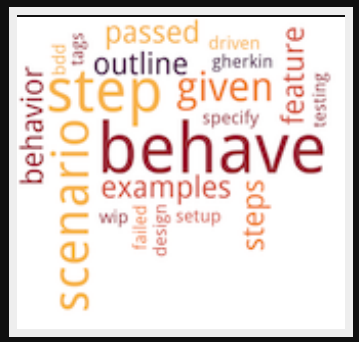
https://github.com/ross/requests-futures

requests-futures: example

```
from concurrent.futures import as_completed
from requests_futures.sessions import FuturesSession

futures=[session.get(f'http://httpbin.org/get?{i}') for i in rang

for future in as_completed(futures):
    resp = future.result()
    print(resp.json()['url'])
```

behave

uses tests written in a natural language style, backed up by Python code



https://behave.readthedocs.io/en/latest/

behave: tutorial.feature

```
Feature: showing off behave

  Scenario: run a simple test
     Given we have behave installed
      When we implement a test
      Then behave will test it for us!
```

behave: tutorial.py

```
from behave import *

@given('we have behave installed')
def step_impl(context):
    pass

@when('we implement a test')
def step_impl(context):
    assert True is not False

@then('behave will test it for us!')
def step_impl(context):
    assert context.failed is False
```

behave: run

```
$ behave
Feature: showing off behave # features/tutorial.feature:1

  Scenario: run a simple test          # features/tutorial.feature:
    Given we have behave installed     # features/steps/tutorial.py
    When we implement a test           # features/steps/tutorial.py
    Then behave will test it for us!   # features/steps/tutorial.py

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
3 steps passed, 0 failed, 0 skipped, 0 undefined
```

sched

implements a general purpose event scheduler

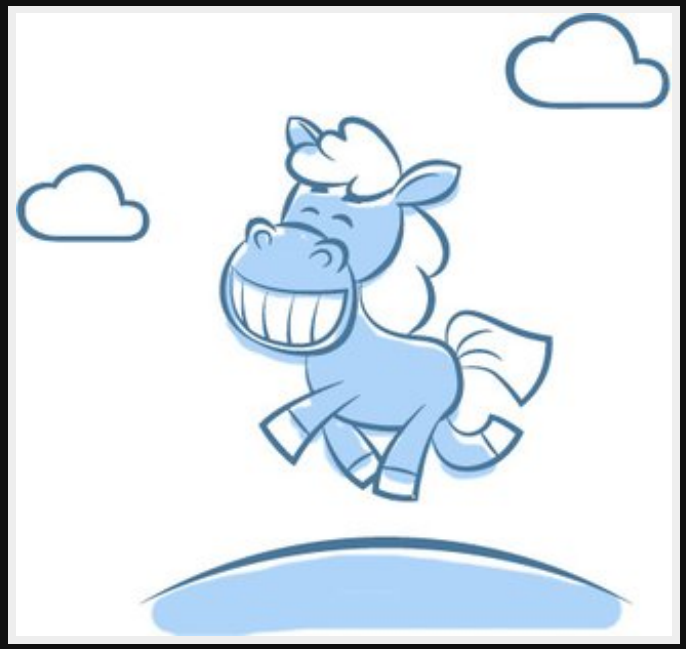
https://docs.python.org/3/library/sched.html

sched: example

```
>>> import sched, time
>>> s = sched.scheduler(time.time, time.sleep)
>>> def print_time(a='default'):
...     print("From print_time", time.time(), a)
...
>>> def print_some_times():
...     print(time.time())
...     s.enter(10, 1, print_time)
...     s.enter(5, 2, print_time, argument=('positional',))
...     s.enter(5, 1, print_time, kwargs={'a': 'keyword'})
...     s.run()
...
>>> print_some_times()
930343690.257
From print_time 930343695.274 positional
From print_time 930343695.275 keyword
```

Pony

Write SQL queries using Python generators & lambdas



https://ponyorm.com/

pony: example

```
select(c for c in Customer if sum(c.orders.price) > 1000)
```

```
SELECT "c"."id"
FROM "customer" "c"
  LEFT JOIN "order" "order-1"
    ON "c"."id" = "order-1"."customer"
GROUP BY "c"."id"
HAVING coalesce(SUM("order-1"."total_price"), 0) > 1000
```

pony: example 2

```
select(c for c in Customer if sum(c.orders.price) > 1000)
```

Here is the same query written using the lambda function:

```
Customer.select(lambda c: sum(c.orders.price) > 1000)
```

Thanks

click arrow colorama structlog bokeh psutil hug scrapy sh ntfy pydantic networkx prompt-toolkit asciimatics prettytable theFuzz progressbar2 doctest pillow requests-futures behave sched pony

https://github.com/piuma/pylibs-slide

danilo.abbasciano@par-tec.it