Python Documentation

Release 0.1.0

Joon Ro

March 28, 2014

Contents

Contents:

CHAPTER 1

Environment

1.1 Mayavi

Mayavi to use Qt rather than wx.

On Windows PowerShell:

```
[Environment]::SetEnvironmentVariable("ETS_TOOLKIT", "qt4", "User")
```

On GNU/Linux:

```
export ETS_TOOLKIT=qt4
```

CHAPTER 2

Cython

2.1 Passing a `cdef` function to another `cdef` function as an argument

http://stackoverflow.com/questions/14124049/is-there-any-type-for-function-in-cython

```
# Define a new type for a function-type that accepts an integer and
# a string, returning an integer
ctypedef int (*f_type)(int, str)

# function to be passed; the same type as f_type
cdef int do_stuff(int a, str b):
    return 0

# note the f_type type in the argument
cdef void foo(f_type func):
    print func(0, "bar")
```

CHAPTER 3

date

Python's strftime directives

3.1 Get week number

```
import datetime
today = datetime.date.today()
print(today.strftime("%U"))
```

CHAPTER 4

docopt

http://docopt.org

```
Naval Fate.

Usage:
  naval_fate ship new <name>...
  naval_fate ship <name> move <x> <y> [--speed=<kn>]
  naval_fate ship shoot <x> <y>
  naval_fate mine (set|remove) <x> <y> [--moored|--drifting]
  naval_fate -h | --help
  naval_fate --version

Options:
  -h --help     Show this screen.
  --version     Show version.
  --speed=<kn>  Speed in knots [default: 10].
  --moored      Moored (anchored) mine.
  --drifting    Drifting mine.
```

4.1 Positional argument

```
Usage: my_program <host> <port>
```

Words starting with <, ending with > or upper-case words are interpreted as positional arguments.

4.2 Options

```
-o --option
```

Words starting with one or two dashes (with exception of -, -- by themselves) are interpreted as short (one-letter) or long options, respectively.

4.3 Command

All other words that do not follow the above conventions of `--options` or `<arguments>` are interpreted as (sub)commands.

CHAPTER 5

matplotlib

..highlight:: python

5.1 `mpltools`: Tools for Matplotlib

http://tonysyu.github.io/mpltools/

`mpltools` provides tools for Matplotlib that make it easier to adjust the style, choose colors, make specialized plots, etc.

5.2 dates

```
# date formatting
years = mdates.YearLocator()   # every year
yearsFmt = mdates.DateFormatter('%Y', interval=1)

months = mdates.MonthLocator()  # every month
monthsFmt = mdates.DateFormatter('%m', interval=1)

weeks = mdates.WeekdayLocator(byweekday=SU)  # every weekday
weeksFmt = mdates.DateFormatter('%Y-%m-%d', interval=1)

# format the ticks
ax.xaxis.set_major_locator(months)
ax.xaxis.set_major_formatter(monthsFmt)
ax.xaxis.set_minor_locator(weeks)
```

5.3 LaTeX

```
import matplotlib.pyplot as plt
from matplotlib import rc

rc('text', usetex=True)

plt.sunplot(111)

plt.text(0.05, 0.90, r'\underline{\alpha}: ', fontsize=12)
```

5.4 Subplots

```
shape = (1, 1)
figsize = (10, 6)
fig, axs = plt.subplots(*shape,
                        sharex=False, sharey=False,
                        figsize=figsize,
                        squeeze=False)

ax = axs[0, 0]
ax.plot(xdata, ydata, label='label', color='blue', linestyle='-')
ax.set_title('subplot title')
ax.set_xlabel('xlabel')
ax.set_ylabel('ylabel')
ax.legend(loc='lower right')

fig.suptitle('figure title')
fig.tight_layout()  # narrow margin
```

5.5 Save to PDF

```
from matplotlib.backends.backend_pdf import PdfPages
pdffile = '{}.pdf'.format('path')
pp = PdfPages(pdffile)
# pp.savefig can be called multiple times to save to multiple pages
pp.savefig()
pp.close()
```

CHAPTER 6

Numpy

6.1 Find indices of duplicates

```
#-----------------------------------------------------------------------
def duplicate(arr):
    '''
    return indices of duplicated entries of a ndarray
    '''

    ix_dup = ones((arr.shape[0], ), dtype=bool)
    ix_dup[unique(arr, True)[1]] = False

    ix_return = []

    for ix in where(ix_dup)[0]:
        ix_return.append(tuple(where(arr==arr[ix])[0]))

    return(ix_return)
```

CHAPTER 7

Profiling

7.1 Get the profile stats with `cProfile`

```
import cProfile
prof = cProfile.Profile()
prof.runcall(func, *args, **kwargs)
```

Print out the stats:

```
prof.print_stats()
```

Dump the profile stats for GUI consumption:

```
prof.dump_stats('stats.prof')
```

7.1.1 `RunSnakeRun` to visualize the profile stats

Install RunSnakeRun for GUI view:

```
$ pip install RunSnakeRun
```

It needs wxPython:

```
$ conda install wxpython
```

Invoke `RunSnakeRun`:

```
$ runsnake stats.prof
```

- It is useful to look at `Local` column. This gives the time spent on a function itself, excluding the time spent calling other functions during the function call. Hence it gives the *net* time spent.
- On the other hand, `Cum` shows the cumulative time spent on that function, including the time spent for other function calls made during the function call.

7.2 `KCachegrind` with `pyprof2calltree`

Install `KCachegrind` and `pyprof2calltree`:

```
$ zypper in kcachegrind
$ sudo pip install pyprof2calltree
```

Invoke `KCachegrind` through `pyprof2calltree`:

```
$ pyprof2calltree -i stats.prof -k
```

CHAPTER 8

PyTables

8.1 Automatically close already open hdf5 files before opening

```
for hdf5 in ['hdf5_read', 'hdf5_write', 'hdf5_append']:
    if hdf5 in globals():
        globals()[hdf5].close()
```

8.2 Generate a `hdf5` file from a recarray

```
hdf5 = tables.openFile('panel.hdf5', 'w')

filters = tables.Filters(complib='blosc', complevel=5)
table = hdf5.createTable('/', 'panel', panel, filters=filters)
```

8.3 Compress and Compacting PyTables files

http://www.pytables.org/docs/manual-1.4/apc.html#ptrepackDescr

PyTables includes a handy utility called ptrepack which can be very useful not only to compact fragmented files, but also to adjust some internal parameters in order to use better buffer and chunk sizes for optimum I/O speed.

```
$ ptrepack --complevel=5 --complib=blosc source.hdf5 target.hdf5
```

8.4 Modifying value in-place

http://permalink.gmane.org/gmane.comp.python.pytables.user/1703

In order to modify the value,

1. Use `cols` accessor:

```
table.cols.number[1] = 3
```

2. Use `modifyRows()` method:

```
row = table[1]
row['number'] = 3
table.modifyRows(1, rows=[row])
```

CHAPTER 9

Unicode

9.1 Remove unicode characters from a string

```
def removeNonAscii(s):
    return "".join(i for i in s if ord(i) < 128)
```

Also, You can filter all characters from the string that are not printable using `string.printable`

CHAPTER 10

Web Crawling

10.1 Mechanize

http://stockrt.github.io/p/emulating-a-browser-in-python-with-mechanize/

```
import mechanize
import cookielib

# Browser
br = mechanize.Browser()

# Cookie Jar
cj = cookielib.LWPCookieJar()
br.set_cookiejar(cj)

# Browser options
br.set_handle_equiv(True)
br.set_handle_gzip(True)
br.set_handle_redirect(True)
br.set_handle_referer(True)
br.set_handle_robots(False)

# Follows refresh 0 but not hangs on refresh > 0
br.set_handle_refresh(mechanize._http.HTTPRefreshProcessor(), max_time=1)

# Want debugging messages?
#br.set_debug_http(True)
#br.set_debug_redirects(True)
#br.set_debug_responses(True)

# User-Agent
useragent = [('User-agent',
              ("Mozilla/5.0 (Windows NT 6.1; rv:7.0.1) Gecko/20100101 "
               "Firefox/7.0.1"))]
br.addheaders = useragent
```

CHAPTER 11

Resources

- The Hitchhiker's Guide to Python!
- Python Testing Frameworks

CHAPTER 12

Indices and tables

- *genindex*
- *modindex*
- *search*