给Python代码打包

2019.7.18

JCBioinformatics-2019-Python

教程主页: https://nanguage.github.io/JCBioinformatics-Python/

Contents

- 为什么要打包
- 创建一个 Python package
- 接口
- 文档化
- 单元测试
- 版本控制、上传到PyPI
- 其它

为什么要打包?

- 提升可复用性
- 更好的组织代码
  - 模块化，减少重复
  - 提高可读性，可维护性
- 方便与人分享
- 方便迁移

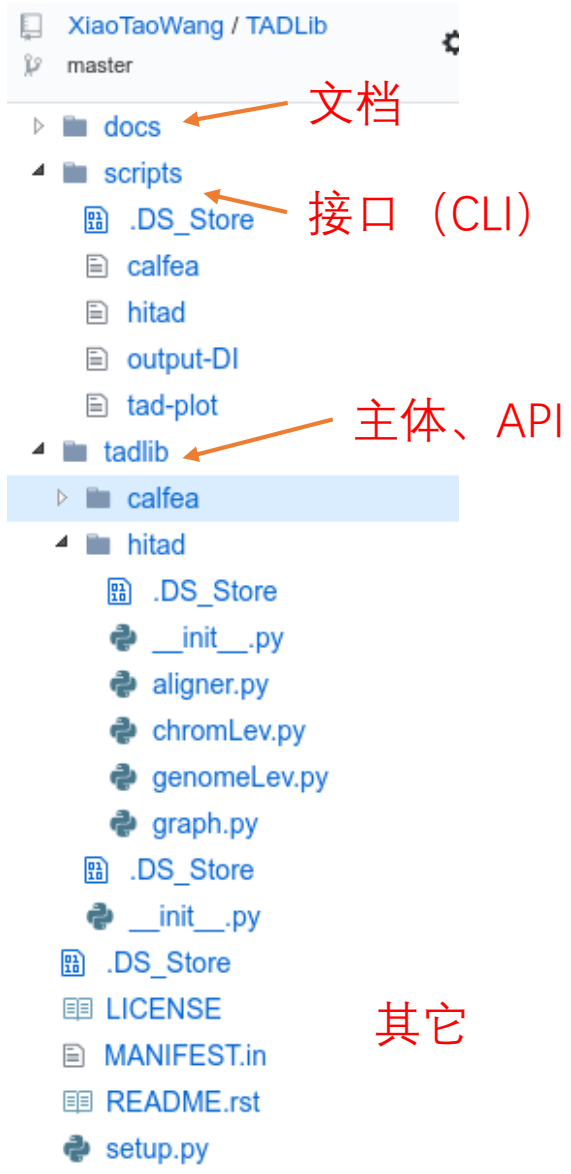
创建一个Python package

创建一个简单的项目作为示范:

比如 …

创建基本的主体、接口、文档等等。。

一个完整 Python 工程的结构



接口

三种接口类型:

- API (Application Programming Interface)
- CLI (Command Line Interface)
- GUI (Graphical User Interface)

API

API可以理解为定义良好（用途明确、文档清晰），暴露给其它程序员的特定编程语言下的接口。

比如说：

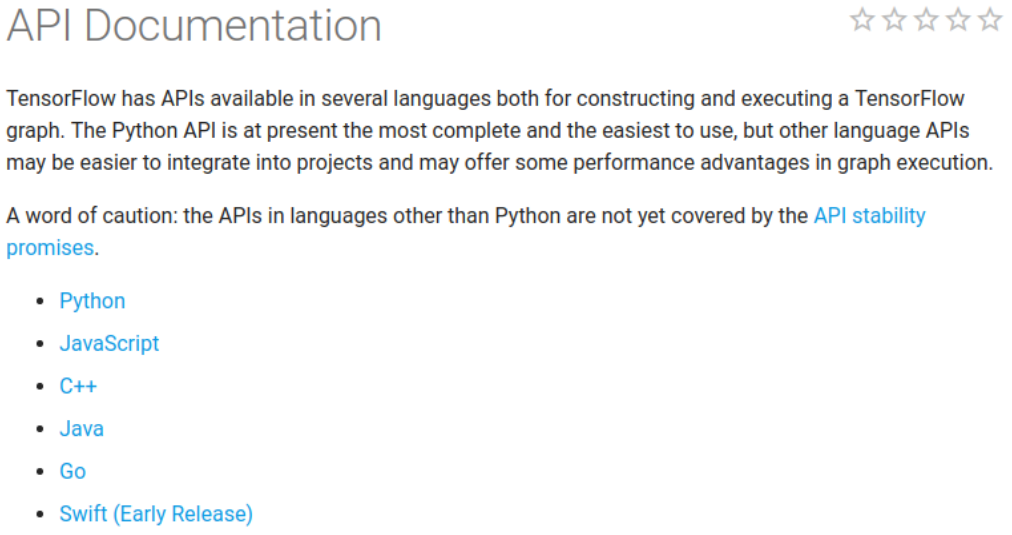
TensorFlow API、Numpy API …

Pros:

- 灵活，可编程，方便控制。
- 很好地结合语言特性，清晰，优雅。

Cons:

- 依赖于特定的编程语言




```
In [8]: np.random.randint(10, size=(5, 5))
Out[8]:
array([[7, 2, 8, 5, 5],
       [5, 3, 1, 7, 6],
       [7, 3, 0, 1, 0],
       [5, 1, 6, 3, 1],
       [2, 0, 3, 7, 1]])
```

```
Docstring:
randint(low, high=None, size=None, dtype='l')

Return random integers from `low` (inclusive) to `high` (exclusive).

Return random integers from the "discrete uniform" distribution of
the specified dtype in the "half-open" interval [`low`, `high`). If
`high` is None (the default), then results are from [0, `low`).

Parameters
----------
low : int
    Lowest (signed) integer to be drawn from the distribution (unless
    ``high=None``, in which case this parameter is one above the
    *highest* such integer).
high : int, optional
    If provided, one above the largest (signed) integer to be drawn
    from the distribution (see above for behavior if ``high=None``).
size : int or tuple of ints, optional
    Output shape.  If the given shape is, e.g., ``(m, n, k)``, then
    ``m * n * k`` samples are drawn.  Default is None, in which case a
    single value is returned.
dtype : dtype, optional
    Desired dtype of the result. All dtypes are determined by their
    name, i.e., 'int64', 'int', etc, so byteorder is not available
    and a specific precision may have different C types depending
    on the platform. The default value is 'np.int'.

    .. versionadded:: 1.11.0
```

CLI

CLI 可以理解为 一种特殊的 API，即 shell 语言的 API。Unix 下最常用的shell为Bash、Zsh，Windows下为CMD、PowerShell。

Python 下创建 CLI 的几种选择:

- argparse （内置）
- click
- docopt
- fire

Pros:

- 灵活
- 使用Shell是操作系统（特别是Unix）下的一种共识。不依赖于特定的语言。

Cons:

- Shell语言一般作为编程语言来说，非常烂！缺乏基本的设施、语法奇怪。

```
@@ -348,7 +348,7 @@ case "$DISTRO" in
-  rm -rf /usr /lib/nvidia-current/xorg/xorg
+  rm -rf /usr/lib/nvidia-current/xorg/xorg
```

GUI

Python GUI 编程：

Tkinter (内置)

Kivy

Usage example

See how easy it is to create a simple *Hello World* application that shows an actionable button:

```
from kivy.app import App
from kivy.uix.button import Button

class TestApp(App):
    def build(self):
        return Button(text='Hello World')

TestApp().run()
```

Result

Jupyter GUI:

ipywidgets

Pros:

- 直观、形象、无需编程知识。

Cons:

- 不灵活，难以程序调用。

文档化

- 代码注释
- Docstring: Numpy style
- Language for doc: Markdown, reStructuredText
- Document to webpage: Sphinx
- GitHub wiki

单元测试

- 单元测试是代码模块行为正确性的证明
- 帮助明确模块函数的功能，TDD (Test Drive Development)

doctest

pytest

版本控制

- Git 作为基本的版本控制工具
- 关于版本号：使用语义化规范的软件版本编号规范 SemVer
- 发布版本 GitHub release、PyPI

上传 Python package 到 PyPI:

参考 https://medium.com/@joel.barmettler/how-to-upload-your-python-package-to-pypi-65edc5fe9c56