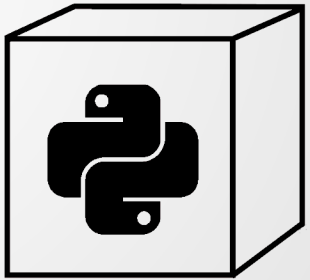
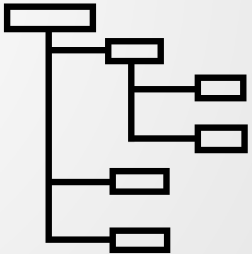
Empaquetado en Python



Cómo estructuro mi repo



Cómo estructuro mi repo

Ejemplo de estructura:

```
README.rst
LICENSE.txt

setup.py
MANIFEST.in

requirements.txt

simpleai/__init__.py
simpleai/foo.py

docs/conf.py
docs/index.rst
docs/foo.rst

tests/test_foo.py
```

Cómo estructuro mi repo

Sobre dónde poner las cosas:

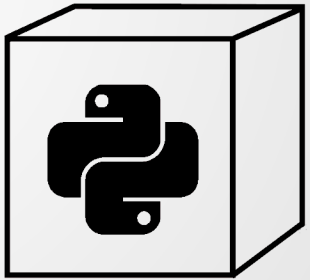
- Nombre del directorio del paquete no ambiguo (**no** "src", "code", etc).
- El README es un **overview**, no la doc. La doc va en `docs`.
- Tests y docs fuera del paquete.
- Y por esto no es tan buena idea usar doctests, mejor unittests.

Cómo estructuro mi repo

Sobre el versionado:

- Separar branches de releases públicos y de desarrollo (buena idea usar algo como **git-flow**).
- Usar **tags** para marcar los releases! Se encuentra fácil código de cada versión, y github las ofrece en la sección de descargas.

Cómo armo el paquete



Cómo armo el paquete

- Hay varias herramientas: setuptools, distutils, distutils2, packaging, distribute.
- Packaging == distutils2 (fiu! una menos),
- Distribute y setuptools anunciaron que se mergean.
- Packaging va a estar por default en python 3.
- Distutils viene por default en python 2.x.
- Por ahora elegimos **distutils**. A futuro puede convenir **packaging**.

Cómo armo el paquete

Peeeero, hay veces que distutils no se lleva bien con endpoints de paquetes hechos con setuptools, así que alguna vez puede que tengamos que pasar a setuptools (distutils no estará ya arreglado? habría que ver)

Cómo armo el paquete

Creamos un `setup.py` en la raíz del repo, algo así:

```
from distutils.core import setup

setup(
    name='simpleai',
    version='0.7.7',
    description=u'An implementation of AI algorithms based on aima-python',
    long_description=open('README.rst').read(),
    author = u'Juan Pedro Fisanotti',
    author_email = 'fisadev@gmail.com',
    url='http://github.com/simpleai-team/simpleai',
    packages=['simpleai', 'simpleai.search', 'simpleai.machine_learning'],
    package_data={'simpleai.search': ['web_viewer_resources/*.*']},
    license='LICENSE.txt',
    classifiers = [
        'Intended Audience :: Developers',
        'License :: OSI Approved :: MIT License',
        'Natural Language :: English',
```

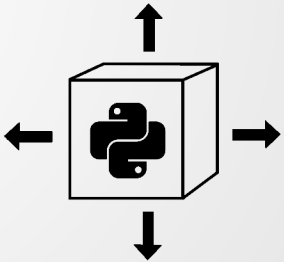
```
        'Operating System :: OS Independent',
        'Programming Language :: Python',
        'Topic :: Scientific/Engineering :: Artificial Intelligence',
    ],
)
```

Cómo armo el paquete

- Distutils es bastante inteligente para darse cuenta de qué cosas incluir.
- Pero a veces falla, o por ahí queremos incluir cosas extras.
- Podemos hacerlo explícito con un `MANIFEST.in` en la raíz del repo, algo así:

```
include README.rst
include LICENSE.txt
recursive-include simpleai *.py
recursive-include simpleai/search/web_viewer_resources *.*
```

Cómo distribuyo el paquete



Cómo distribuyo el paquete

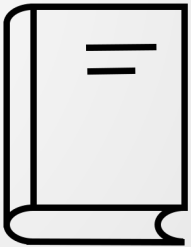
- La gente ya puede instalar con pip apuntando a la url del repo.
- Es mejor subir el paquete a **PyPI** y que la gente pueda instalar con `pip install my_lib`.
- Solo la primera vez, creamos un usuario en http://pypi.python.org y registramos el paquete:

```
python setup.py register
```

- Cada vez que queremos subir una versión nueva:

```
python setup.py sdist upload
```

Cómo armo la doc



Cómo armo la doc

- Lo más común es utilizar **sphinx** y escribir la doc en **ReST** dentro del repo.
- Instalamos sphinx:

```
sudo pip install sphinx
```

- Entramos al directorio `docs` e inicializamos sphinx (respondiendo lo que pida):

```
sphinx-quickstart
```

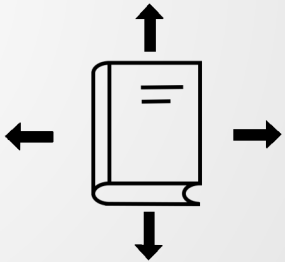
- Nos quedan un montón de archivos en el directorio `docs`. Agregamos todo al repo, **menos** `_build`.

Cómo armo la doc

- Completamos los archivos `.rst` con la doc en sí.
- Hay herramientas para que parsee nuestra lib y genere doc automatizada de funciones, clases, etc.
- Podemos probar compilar la doc a html para leerla localmente con algo como esto:

```
make html
google-chrome _build/html/index.html
```

Cómo publico la doc



Cómo publico la doc

- Creamos usuario en http://readthedocs.org
- Registramos el proyecto.
- Configuramos url del repo, e indicando que la doc está en el directorio `docs`.
- Si el repo está en **GitHub**, configuramos el service hook de readthedocs para que se actualice cada vez que pushemos versiones nuevas (si no se puede hacer a mano).

Cómo ser feliz y tener una vida llena de sentido después de haber publicado un paquete como corresponde

Cómo ser feliz y tener una vida llena de sentido después de haber publicado un paquete como corresponde

Listo! Repasamos:

- Repo bien estructurado ✓
- Paquete de python armado y publicado ✓
- Doc escrita con rst y que se publica sola ✓

Cómo ser feliz y tener una vida llena de sentido después de haber publicado un paquete como corresponde

Cómo trabajamos a diario?:

- codeamos y pusheamos sobre los branches de **desarrollo**.

Cómo ser feliz y tener una vida llena de sentido después de haber publicado un paquete como corresponde

Cómo releaseamos versiones estables?:

- actualizamos números de versión en `setup.py` y `docs/conf.py`
- mergeamos al branch de stable
- tageamos
- pusheamos
- subimos release a pypi (`python setup.py sdist upload`)

Cómo ser feliz y tener una vida llena de sentido después de haber publicado un paquete como corresponde

Y después una cerveza o una coca, dependiendo del sujeto.