family$_{birthdays}$

Versión 1.0.0

25 de julio de 2020

Contents:

La librería **family_birthdays** es un packet que permite ilustrar el funcionamiento de una código python en la nube, particularmente en jupyter notebooks.

CAPÍTULO 1

Introduccion

1.1 ¿Porqué?

Sólo para demostrar el funcionamiento del framework GeneralSimulationLibrary.

1.2 ¿Qué hace?

Calcula los cumpleaños más cercanos y las edades, a partir de un diccionario proporcionado con las fechas de nacimiento.

CAPÍTULO 2

Installation

El repositorio se encuentra en https://github.com/sebastiandres/family_birthdays.

La implementación fue desarrollada en Python 3.

2.1 Install from repository

La librería es solo de ejemplo, no está en pipit. Puedes instalarla directamente desde github:

```
pip install git+https://github.com/sebastiandres/family_birthdays.git
```

No es una librería muy avanzada, fue creada con propósito ilustrativo y para mostrar cómo se puede ejecutar código python en jupyter notebooks en la nube.

CAPÍTULO 3

Examples

3.1 Ejemplo en mybinder

Acá hay un ejemplo ejecutable usando MyBinder. No requiere una cuenta.

3.2 Ejemplo en Google colab

Acá hay un ejemplo en Google Colab. Requiere tener una cuenta de gmail (google).

3.3 Ejemplo en python

Para ejecutar el código necesitas haber instalado la librería. To run it, you need to install the library.

```
from family_birthdays import SimulationInterface

birthdates = {"Captain America":"04-07-1918",
                "Ironman":"29-05-1970",
                "Spiderman":"10-08-2001",
                "Batman":"17-04-1915",
                "Hulk":"18-12-1969",
}

SI = SimulationInterface()
SI.new({"birthdates":birthdates})
SI.simulate()
```

CAPÍTULO 4

How to use it

Ok, I'm in. What should I do?

4.1 Considerations

Through this explanation, let's asume you have the username *my-username*, that you will rename the project name from *family_birthdays* to *MyProject* and that you will rename the class *SimulationInterface* to *MyInterface*.

4.2 Getting the code and versioning

You should start by getting a copy of the repo to play around: https://github.com/sebastiandres/family_birthdays.

There are several ways to do this.

The first option: if you have a github account, you can fork the project to your github account. You can then clone **your** project to your local environment to start making changes:

```
git clone https://github.com/PUT-YOUR-USERNAME-HERE/family_birthdays.git
```

The second option is to download the zip file of the repository from the repository github's webpage. This allows to start with no previous commits. But now you need to add the project to your github account.

In both cases, you'll have full control to version the project and push your changes. You should end up with a project stored at *https://github.com/my-username/MyProject*

Test it by making a small change of the README.md, making a commit and pushing it. If you don't know how to do any of those, look at a tutorial like https://try.github.io/.

It may look as a overkill to do **git-versioning** and **library versioning** for your small project. **It is not**. Learn the tools and you will save an HUGE amount of time and frustration. Smash those bugs and document the code increments.

4.3 Personalize the Simulation Code

The provided code already takes into account a **simple interface** and the creation of a **simulation seed**. Henceforth, code complexity is been hiden from the end user and a «simulation seed» can store all the information (inputs, system and libraries configuration, options and outputs) to guarantte reproducibility.

To personalize the code, you need to replace the mentions to **family_birthdays** and **SimulationInterface** with the choosen names, in this example, MyProject and MyInterface. You should edit the files at *family_birthdays/family_birthdays/* (in your project shoul be at *MyProject/family_birthdays/*), in particular *simulation_interface.py* and *__init__.py*. You can add more files if needed.

The code version is centralized and stored at *MyProject/family_birthdays/version.py*. Follow some updating rule.

4.4 Personalize the Documentation

The documentation is stored at *family_birthdays/docs/source/* (in your project shoul be at *MyProject/docs/source*) You should personalize all the rst files. The files should give you some pointers on how to use, but you can consult the rst specification

To rebuilt the documentation, execute the following at the path *family_birthdays/docs/* (or *MyProject/docs/* at your project):

```
make clean
make html
```

You can check how the documentation turn out at *family_birthdays/docs/build/* (or *MyProject/docs/build* at your project), the main file being *index.html*.

You can now go to read-the-docs and import your library documentation to make it public.

4.5 Distribution

The project is a **pip-installable library**. This has been taken care in the project structure and the file *setup.py*. You should edit *setup.py* and own it: make the changes of library name, version, author, packages, repository url, licence and description.

The project can be distributed from day one from the github repository.

If you have a stable version, that you would like to distribute through pypi. There's a nice pypi tutorial you can follow.

You need to have installed twine (*pip intall twine*), and to have accounts at pypi and test.pypi. Note that that they required different accounts.

First, at the main folder, test the distribution at testpypi:

```
python setup.py sdist bdist_wheel
python -m twine upload --repository testpypi dist/*
```

There's a Makefile, so you can also just do *make test.pypi* at *family_birthdays/* (in your project shoul be at *MyProject/*)

You can check how everything looks at https://test.pypi.org/. If everything is looking good, upload it to (real) pypi:

```
python setup.py sdist bdist_wheel
python -m twine upload --repository pypi dist/*
```

Using the Makefile, you can just do *make pypi* at *family_birthdays/* (in your project shoul be at *MyProject/*)

CAPÍTULO 5

SimulationInterface class

Esta es la documentación de la librería.

class simulation_interface.**SimulationInterface**

Bases: object

family_birthdays is a package computes the family birthdays The framework can be personalized and extended for a specific simulation need. Link: https://family_birthdays.readthedocs.io/

download(*filename*)

Utility to download file, using colab

load(*filename*)

Loads a simulation from a simulation file generated with the *save* method to restore the simulation.

Parámetros **filename** (*string*) – Name for the simulation file.

new(*inputs*, *plot_options=None*)

Associates inputs and plot options to the simulation.

Parámetros

- **inputs** (*dict*) – The inputs that will be used in the simulation. This can be completely personalized.
- **plot_options** (*dict, optional*) – The plot options, defaults to None

save(*filename*)

Saves the current state of the simulation, with all the provided information. The created file can be used with the *load* method to restore the simulation.

Parámetros **filename** (*string*) – Name for the simulation file.

simulate()

Conditionally imports the numpy library.

status()

Prints out the detected configuration: environment, python and library versions.

Índice