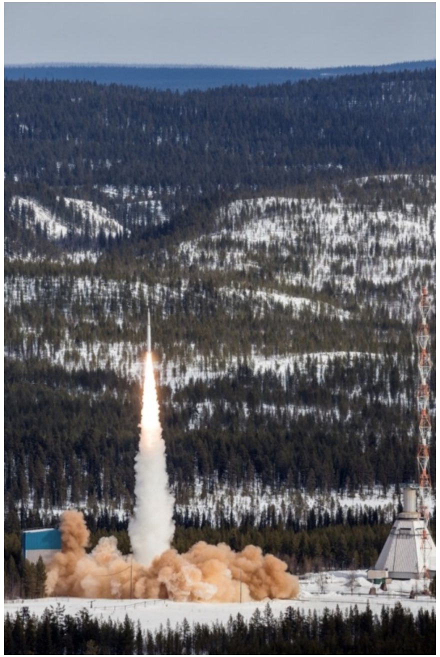
OREKIT IN PYTHON

ACCESS THE PYTHON SCIENTIFIC ECOSYSTEM

Petrus Hyvönen

2017-11-27

SSC ACTIVITIES



Science Services

Satellite Management Services

Image: ESA, Juergen Mai

Engineering Services

INITIAL REASON OF PYTHON WRAPPED OREKIT

- SSC is providing ground network services for customers
- Need tools to analyze ground network performance
- Was using STK but wanted something more scriptable (and free)
- Was using Matlab but had only a few licenses, needed to be on corporate network and map plotting wasn’t fantastic

- Started to look seriously at Python
  - A general purpose language
  - “Interactive computing”
  - Great map plotting libraries
  - Great community
  - Lots happening in scientific computing in Python

- But no real astrodynamics library…

WRAPPING JAVA FOR PYTHON

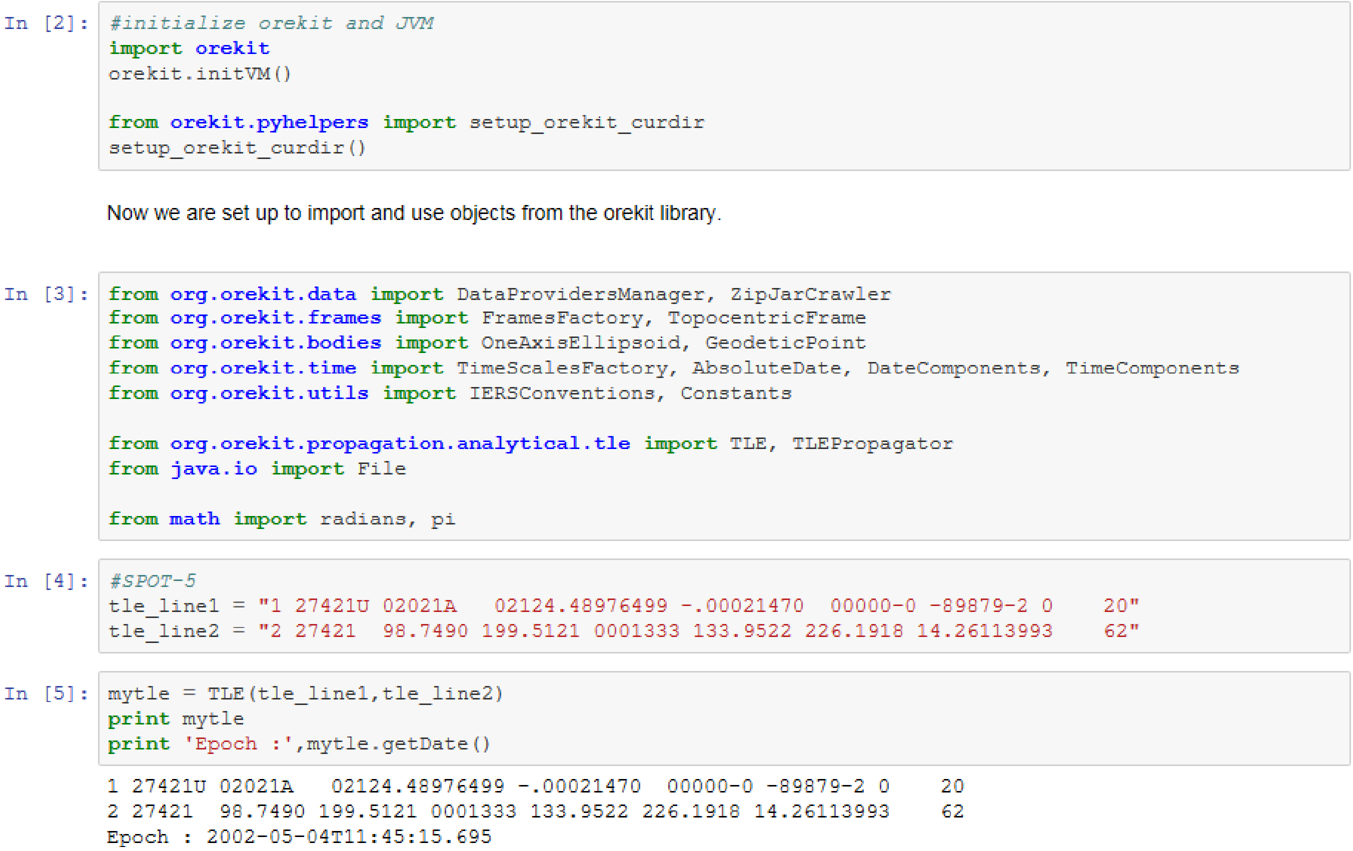
NO GIVEN SOLUTION TODAY

- There are and was several tools available, in various state and featuers
- Dynamic wrapping or “compiled wrapping”

- JCC
  - JCC is part of the Apache pylucene library, a text search library
  - Generates C++ code that wraps a Java library via Java Native Interfaces (JNI)
  - Generates C++ wrappers that is then available in Python
  - Pythonic wrapping, looks and used almost fully like a python library
  - JCC is since this year available for python 3 and python 3
  - Can be tricky to get all steps in the compile to work
  - Mainly one key developer
  - Apache 2.0 license

SOME CODE EXAMPLE

CREATING A TLE OBJECT



In [2]:
```
#initialize orekit and JVM
import orekit
orekit.initVM()

from orekit.pyhelpers import setup_orekit_curdir
setup_orekit_curdir()
```

Now we are set up to import and use objects from the orekit library.

In [3]:
```
from org.orekit.data import DataProvidersManager, ZipJarCrawler
from org.orekit.frames import FramesFactory, TopocentricFrame
from org.orekit.bodies import OneAxisEllipsoid, GeodeticPoint
from org.orekit.time import TimeScalesFactory, AbsoluteDate, DateComponents, TimeComponents
from org.orekit.utils import IERSConventions, Constants

from org.orekit.propagation.analytical.tle import TLE, TLEPropagator
from java.io import File

from math import radians, pi
```

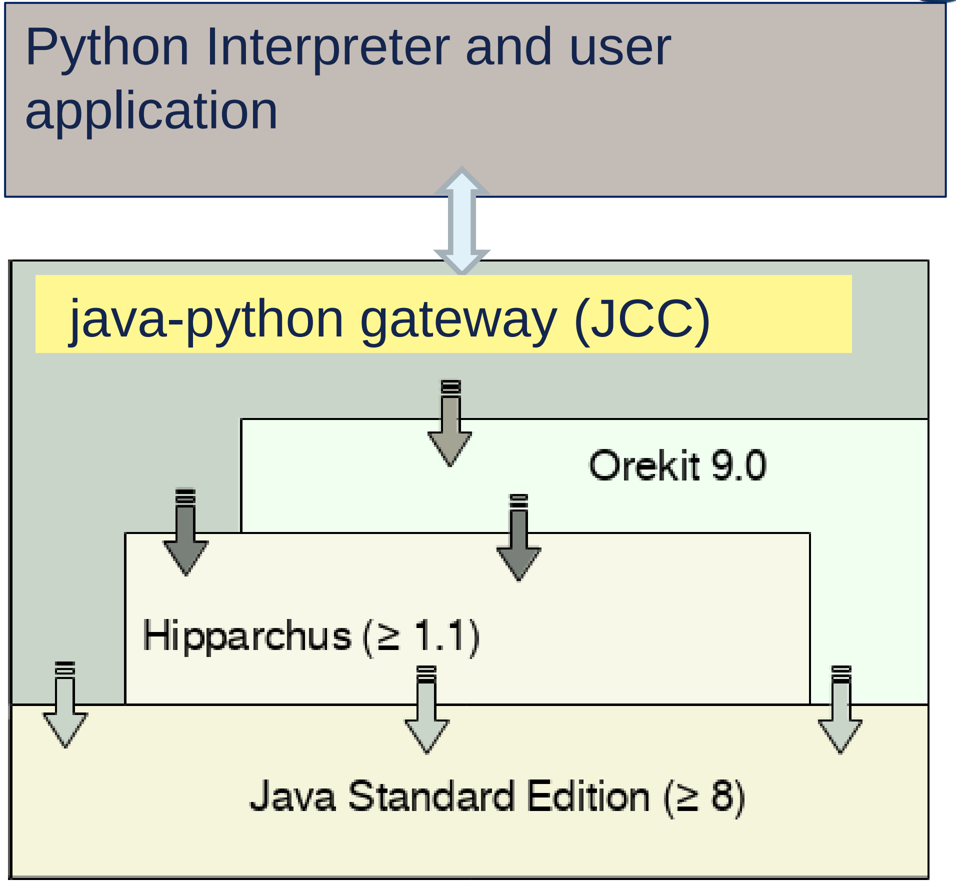
In [4]:
```
#SPOT-5
tle_line1 = "1 27421U 02021A   02124.48976499 -.00021470  00000-0 -89879-2 0    20"
tle_line2 = "2 27421  98.7490 199.5121 0001333 133.9522 226.1918 14.26113993    62"
```

In [5]:
```
mytle = TLE(tle_line1,tle_line2)
print mytle
print 'Epoch :',mytle.getDate()
```

```
1 27421U 02021A   02124.48976499 -.00021470  00000-0 -89879-2 0    20
2 27421  98.7490 199.5121 0001333 133.9522 226.1918 14.26113993    62
Epoch : 2002-05-04T11:45:15.695
```

ARCHITECTURE

- Orekit and Hipparchus libraries explicitly wrapped.
- Classes needed for methods or class initialization are wrapped as well (can be java.io.* classes for example)
- Orekit is started in Python with the command orekit.initVM() that starts up the JRE and enables the interaction



Python Interpreter and user application

java-python gateway (JCC)

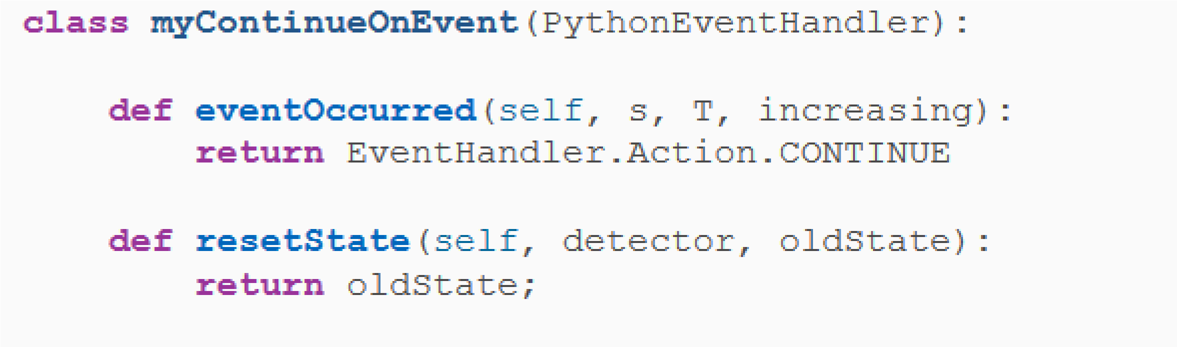
Orekit 9.0

Hipparchus (≥ 1.1)

Java Standard Edition (≥ 8)

SUBCLASSING JAVA CLASSES IN PYTHON

- Subclassing of java classes in python is possible but some adjustments in java are needed to the classes that are to be subclassed.
- Specific PythonClassname classes created for a few classes that could be usable with subclassing
- A domain org.orekit.python is used for these classes today



```
class myContinueOnEvent(PythonEventHandler):

    def eventOccurred(self, s, T, increasing):
        return EventHandler.Action.CONTINUE

    def resetState(self, detector, oldState):
        return oldState;
```

Currently implemented classes for subclassing in python:

- *PythonAbstractDetector.java*
- *PythonEventDetector.java*
- *PythonEventHandler.java*
- *PythonOrekitFixedStepHandler.java*
- *PythonUnivariateFunction.java*

CASTING

- Casting is done through the .cast_ method of the Python class that is the desired class:

```
sun = CelestialBodyFactory.getSun()     # Here we get it as an CelestialBody
sun = PVCoordinatesProvider.cast_(sun)  # But we want the PVCoord interface
```

ROADMAP PYTHON OREKIT WRAPPER

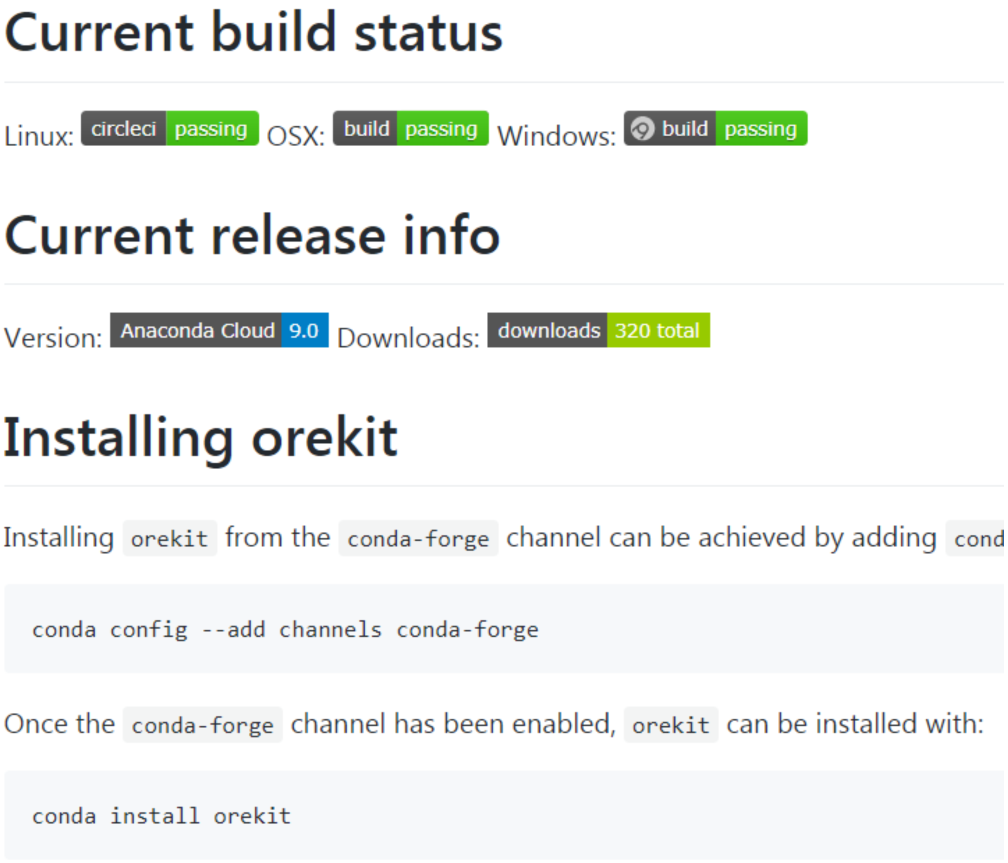
- Plan is to keep the Python Orekit Wrapper as close as possible to the Java API
- Follow release schedule of Java version with minor updates for Python stuff
- Focus on the automated built packages for Anaconda Python Distribution

“Add-ons”:

- Review the PythonClassname strategy, which classes to include and if it is better to modify original java classes in Python branch
- Would be nice with javadoc text as Python help text

INSTALLATION AND BUILDING

- Building the Orekit module for Python should in *principle* be straightforward
- Practically lots of issues has been experienced over the years. Use pre-built packages!
- Started to use Anaconda Python distribution with the conda package manager
  - Cross platform package and dependency manager focusing on Python for scientific and data intensive computing
- Since this year automated Orekit builds for win, Linux and osx for python 2.7, 3.4, 3.5 and 3.6
- Package available through the conda-forge community channel



Current build status

Linux: circleci passing OSX: build passing Windows: build passing

Current release info

Version: Anaconda Cloud 9.0 Downloads: downloads 320 total

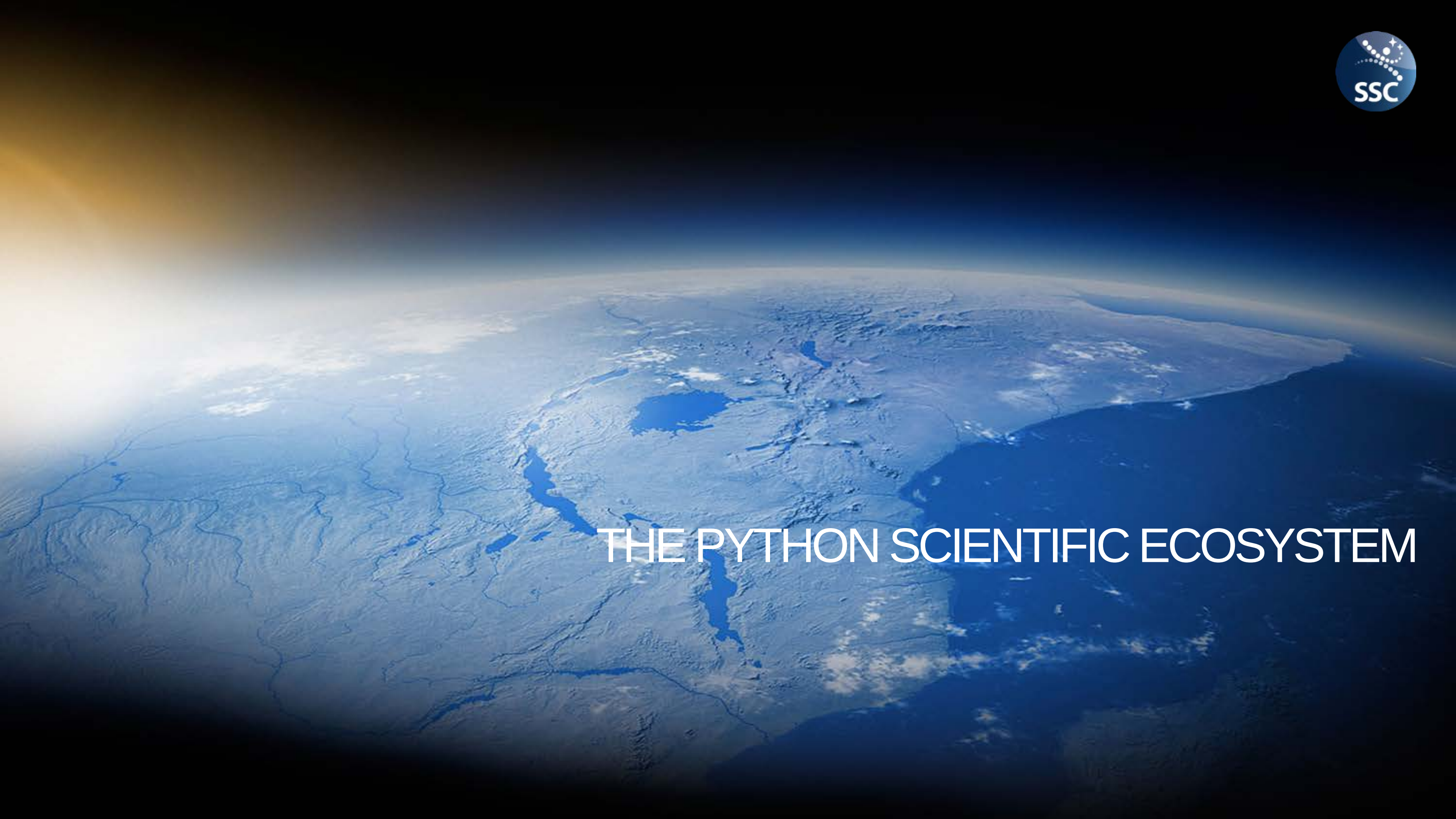
Installing orekit

Installing `orekit` from the `conda-forge` channel can be achieved by adding `cond`

```
conda config --add channels conda-forge
```

Once the `conda-forge` channel has been enabled, `orekit` can be installed with:

```
conda install orekit
```



THE PYTHON SCIENTIFIC ECOSYSTEM

THE PYTHON SCIENTIFIC ECOSYSTEM

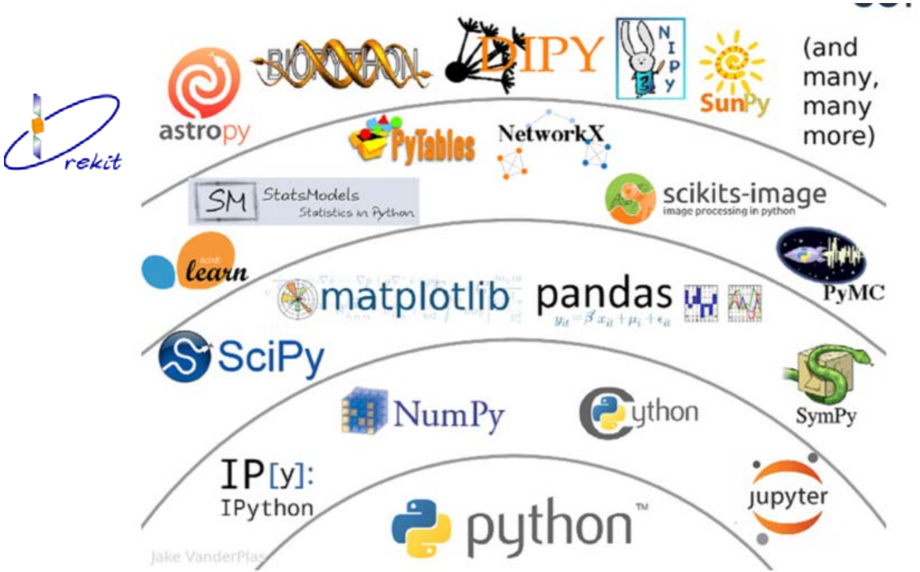


Illustration:Jake VanderPlas

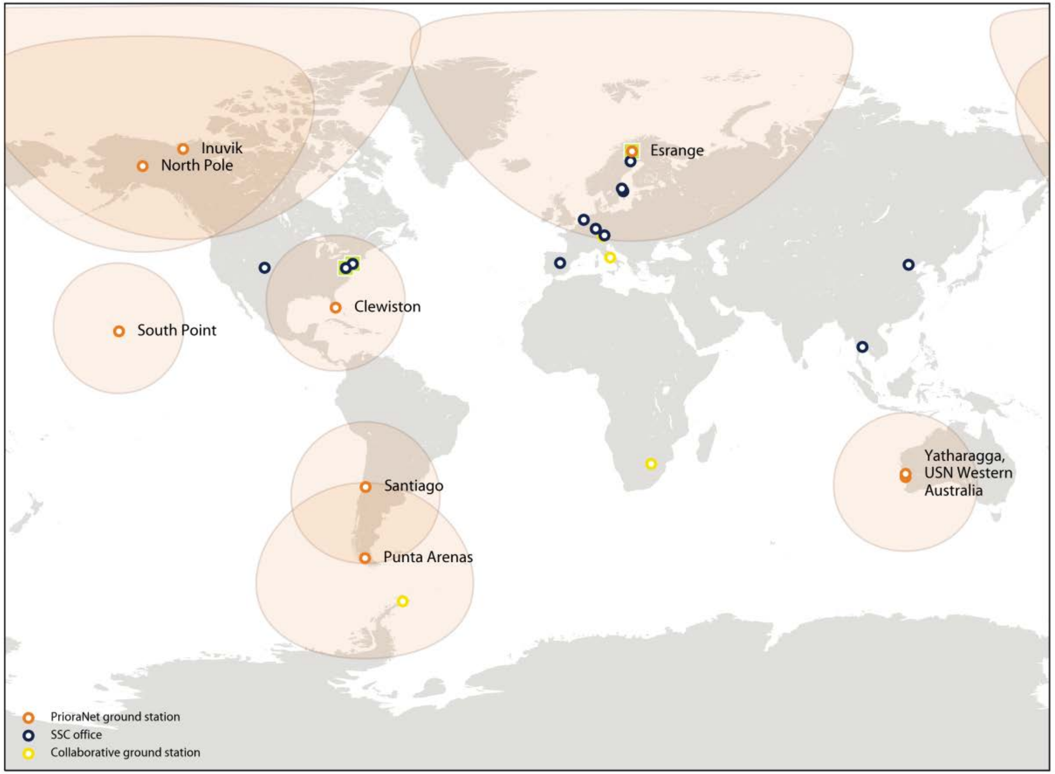
JUPYTER NOTEBOOK

- Web application that integrates live code, results, visualizations and rich documentation in same view
- ”Document based”
- Last exection results part of file!
- Browser interface familiar to large number of users
- Direct interaction and easy modification
- We use it as frontend for a set of analysis routines

MATPLOTLIB + BASEMAP / CARTOPY

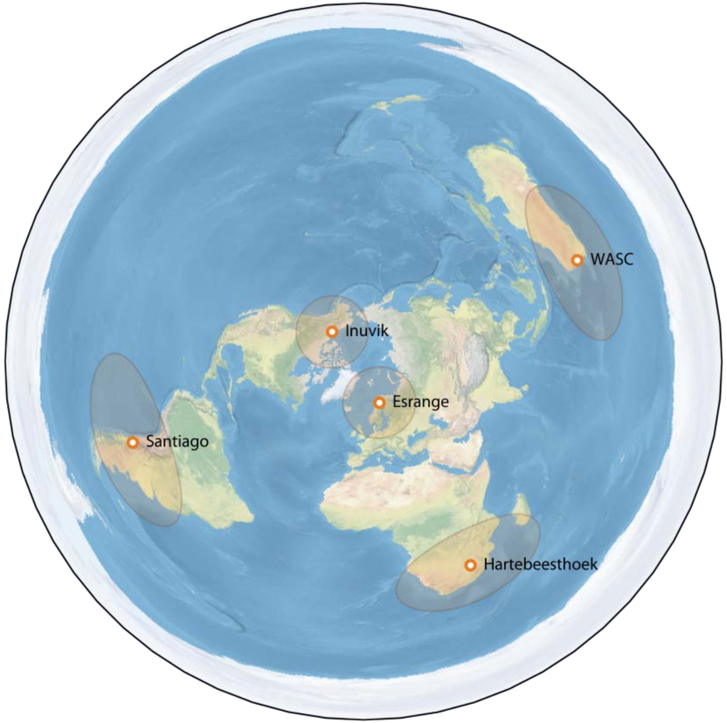
- “Starndard” package for 2-D plots
- Quck plot modes
- Advanced control for publication quality plots
- Outputs both bitmap and vector graphics
- Inline output in jupyter notebooks
- Cartopy and basemap are add-ons for advanced map generation
  - Automatic transformation between projections
  - Shapefile support and accesses different map bitmaps / vector maps online

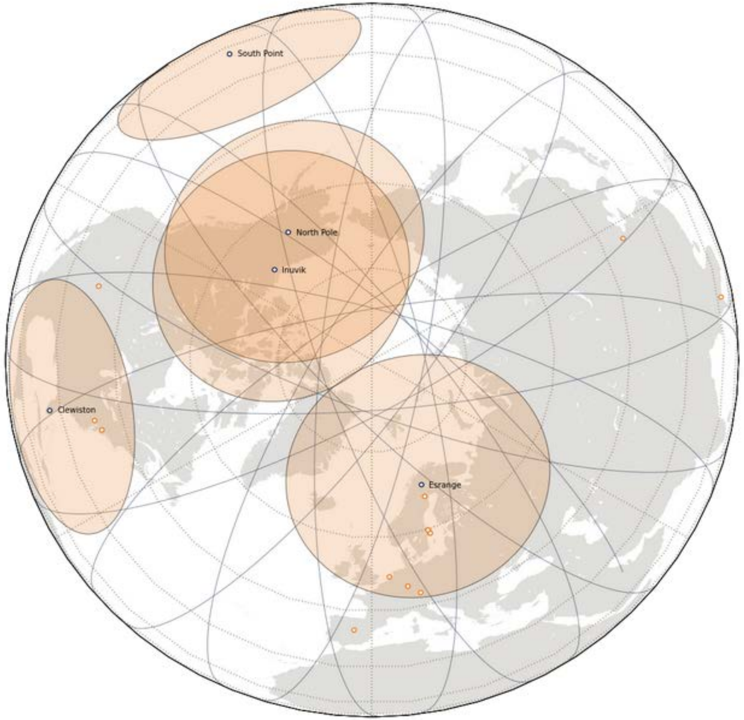
EXAMPLE MATPLOTLIB



EXAMPLE MATPLOTLIB

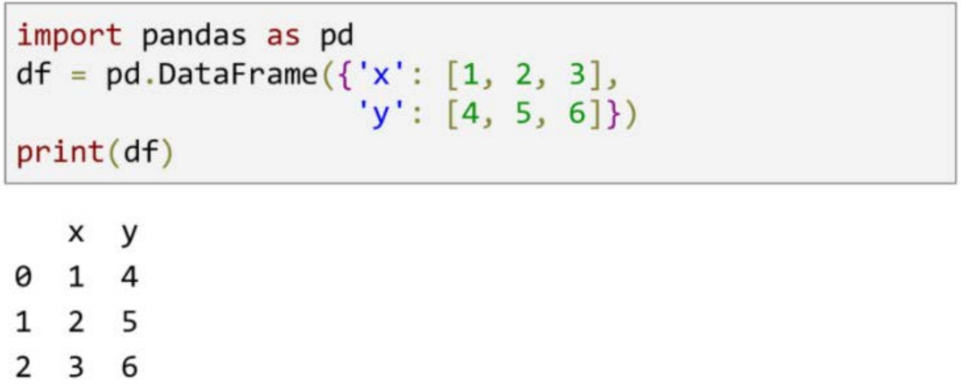
DIFFERENT PROJECTIONS





PANDAS

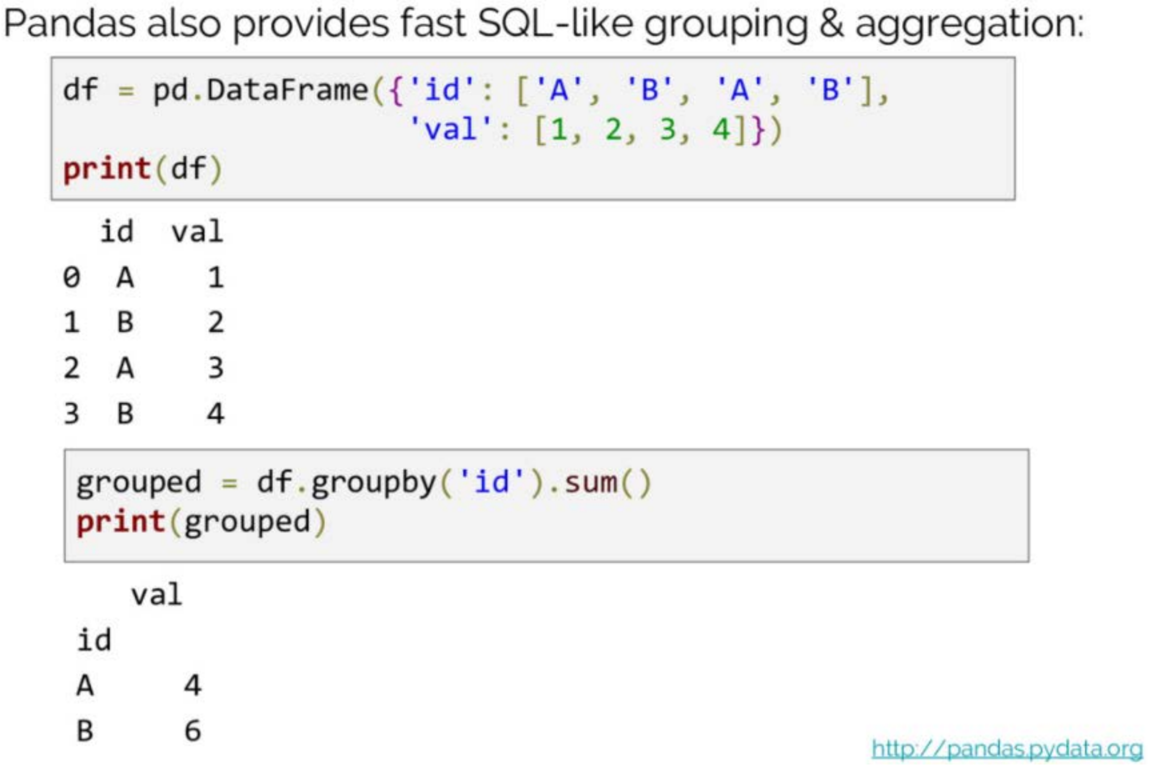
- Labeled arrays and dataframes based on NumPy arrays
- Easy to read / write different formats and sources (csv, excel, web tables, databases,…)
- Integrates well with the other Python ecosystem
- Handles missing data, mixed types and dates well
- Database type of joins, filters etc.



```
import pandas as pd
df = pd.DataFrame({'x': [1, 2, 3],
                   'y': [4, 5, 6]})
print(df)
```

```
   x  y
0  1  4
1  2  5
2  3  6
```

MORE PANDAS



Pandas also provides fast SQL-like grouping & aggregation:

```
df = pd.DataFrame({'id': ['A', 'B', 'A', 'B'],
                   'val': [1, 2, 3, 4]})
print(df)
```

```
  id  val
0  A    1
1  B    2
2  A    3
3  B    4
```

```
grouped = df.groupby('id').sum()
print(grouped)
```

```
    val
id
A     4
B     6
```

http://pandas.pydata.org

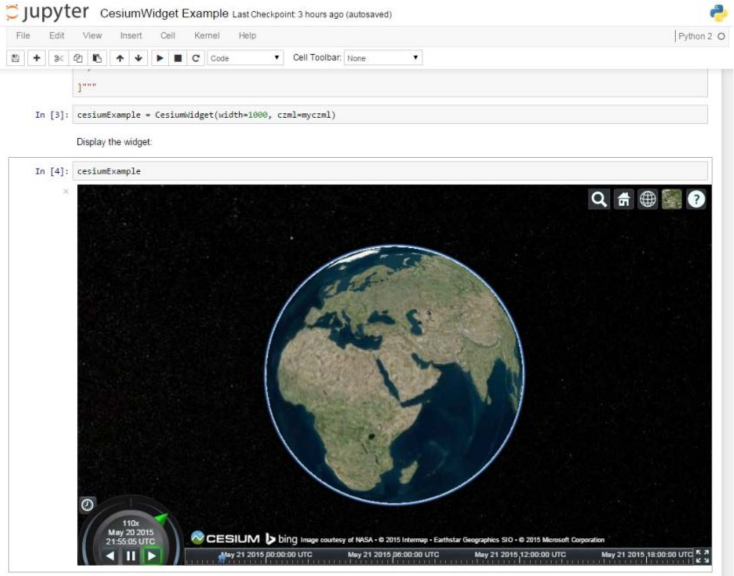
INTERESTING CURRENT DEVELOPMENTS

Lots of development and investments in Python as a scientific computing platform

Example of interesting developments:

- JupyterLab (Multi-user notebook)
- Bokeh (Interactive visualizatoin)
- Numba (JIT compilation)
- Dask (Distributed computing)

Open source project volunteers needed for CesiumWidget, cesiumjs in Jupyter notebook (javascript) ☺



```
]"""
In [3]: cesiumExample = CesiumWidget(width=1000, czml=myczml)
```

Display the widget:

```
In [4]: cesiumExample
```

Example of visualization of javascript widget (cesiumjs) in Jupyter Notebook

USEFUL LINKS

Installation:

- Anaconda Python Distribution: http://docs.continuum.io/anaconda/install.html
- Instruction and source of the Orekit package for anaconda: https://github.com/conda-forge/orekit-feedstock

Development:

- Orekit Python Wrapper Main site: https://www.orekit.org/forge/projects/orekit-python-wrapper
- Github place for the Java code of Orekit with Python additions: https://github.com/petrushy/Orekit

WE HELP EARTH
BENEFIT FROM SPACE

www.sscspace.com