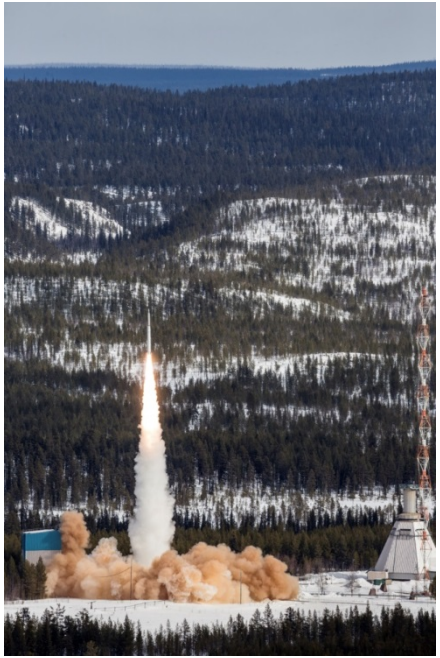
WE HELP
EARTH BENEFIT
FROM SPACE

OREKIT IN PYTHON

Petrus Hyvönen, for Orekit Day 2019

2019-05-23

SSC ACTIVITIES
WHO WE ARE



Science Services

Satellite Management Services

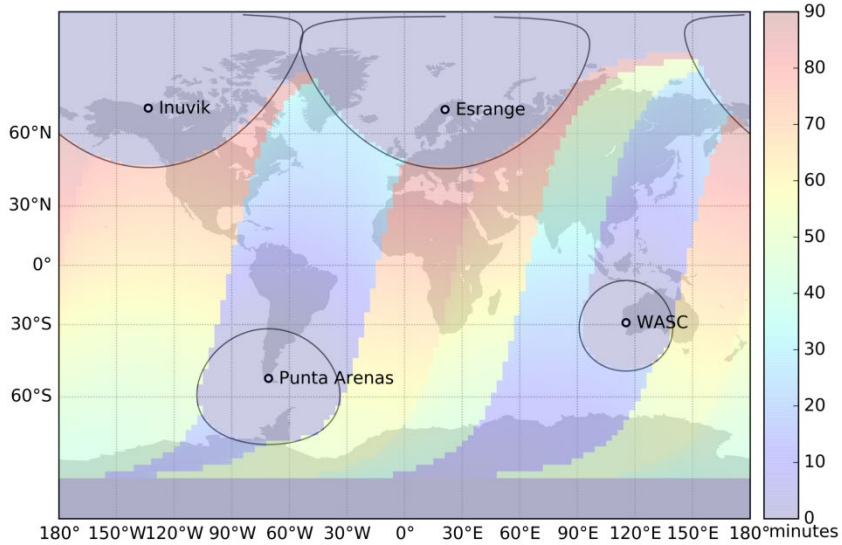
Engineering Services

INITIAL REASON OF PYTHON WRAPPED OREKIT

- SSC is providing ground network services for its customers
- Need tools to analyze ground network performance
- Was using some commercial tool but wanted something more scriptable, free and open
- Was using Matlab also but had only a few licenses, needed to be on corporate network and map plotting was quite poor

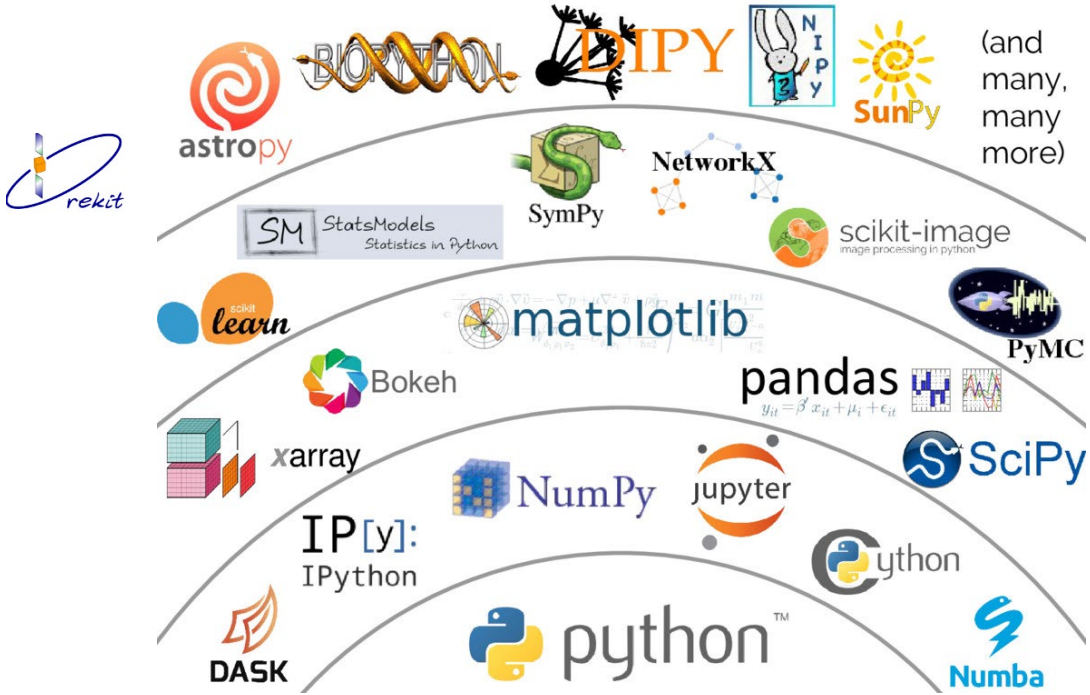
- Started to look seriously at Python
  - A general purpose language
  - Supporting “Exploratory computing” ala Matlab

- But no real astrodynamics library in python… but in java… So we made a wrapper to combine these worlds.



Example of network analysis performed in Python – time from image acquisition to groundstation.

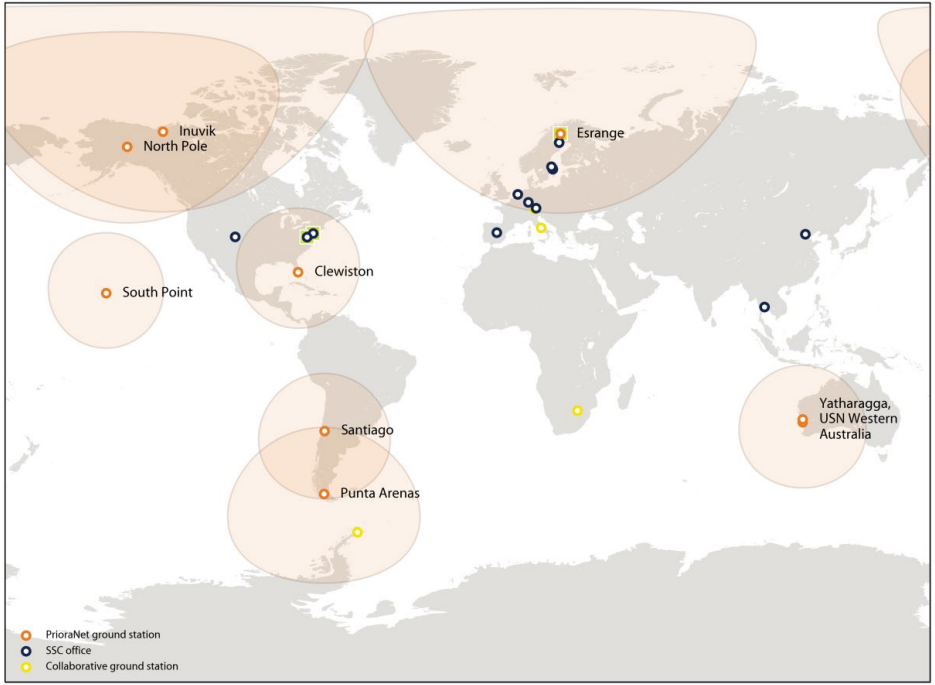
ONE VIEW OF PYTHON SCIENTIFIC ECOSYSTEM



Original Illustration:Jake VanderPlas

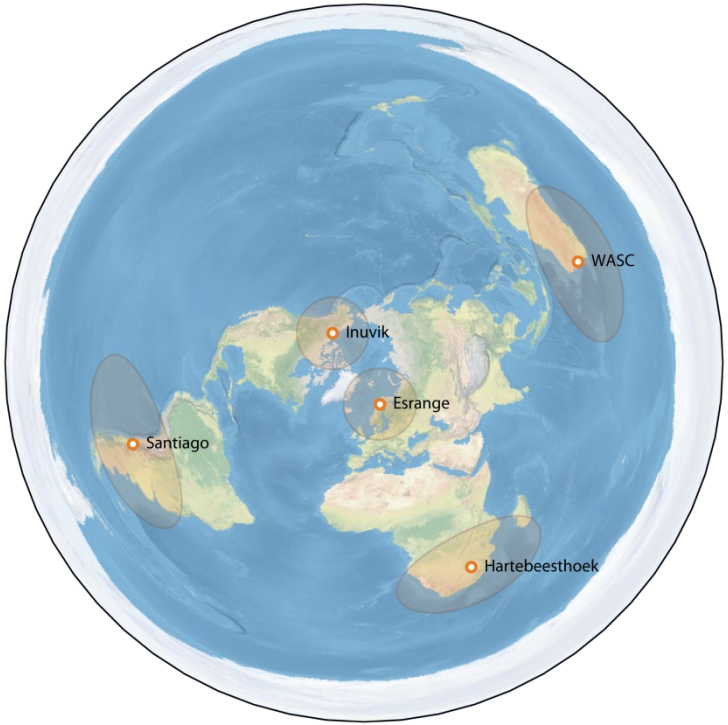
MATPLOTLIB + BASEMAP / CARTOPY

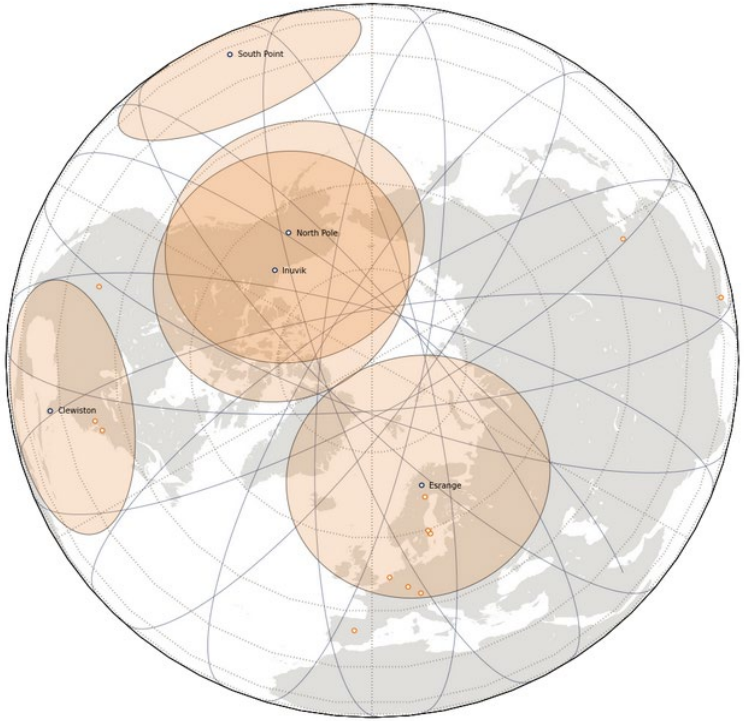
- “Starndard” package for 2-D plots
- Quck plot modes
- Advanced control for publication quality plots
- Outputs both bitmap and vector graphics
- Inline output in jupyter notebooks
- Cartopy and basemap are add-ons for advanced map generation
  - Automatic transformation between projections
  - Shapefile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



EXAMPLE MATPLOTLIB

DIFFERENT PROJECTIONS





PANDAS

“HIGH-PERFORMANCE, EASY-TO-USE DATA STRUCTURES AND DATA ANALYSIS TOOL”

- 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.
- Lots of tools to efficiently visualize dataframes

```
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
```

JUPYTER NOTEBOOK & HUB

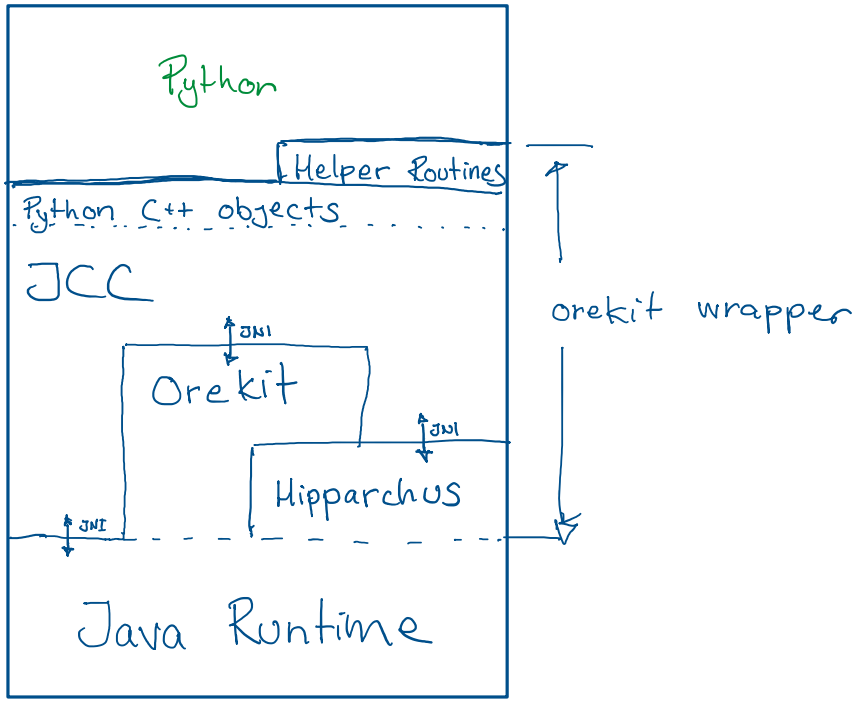
- Web application that integrates live code, results, visualizations and rich documentation in the same view
- ”Document based”
  - Last exection results part of file!
- Browser interface appeals to large number of users
- Exploratory computing
- SSC use it as frontend for a set of analysis routines
- Large set of visualization tools in development connecting javascript libraries with jupyter widgets


ARCHITECTURE OF THE PYTHON
OREKIT WRAPPER

ARCHITECTURE

OREKIT PYTHON WRAPPER

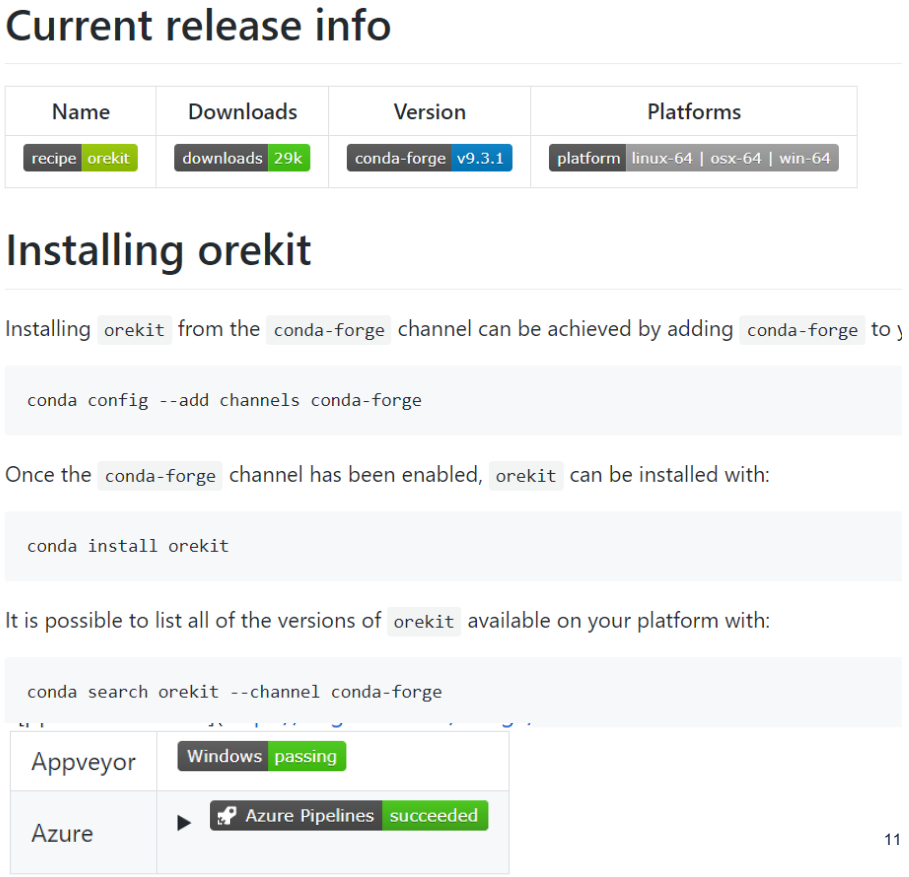
- A gateway layer is created using JCC
  - Analyzes the java library (jar)
  - Generates C++ classes that wraps each Java class using JNI
  - Python interfaces are generated for C++ classes
- Orekit and Hipparchus libraries explicitly wrapped
- Classes needed for methods or class initialization are wrapped as well (can be java.io.* classes for example)
- Python helper functions are included in the pyhelpers module
- Orekit and the JVM is started in Python with the command orekit.initVM()



ANACONDA AND CONDA-FORGE

A GREAT CONTRIBUTION TO SCIENTIFIC PYTHON

- Last years a distribution of Python with packet manager for data science and other usage has become very popular, Anaconda.
- Free and most parts is open source.
- Make installation of a turn-key system with standard set of packages very easy
- A community maintained, automated build system for user maintained packages ”conda-forge”
- Orekit and JCC are part of this, automatically built for a number of platforms, configurations and versions
- Test suite included in the automated build, ensuring high level of confidence that it will run at users installations
- Up to now 29.000 downloads…



Current release info

Name	Downloads	Version	Platforms
recipe orekit	downloads 29k	conda-forge v9.3.1	platform linux-64 | osx-64 | win-64

Installing orekit

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

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

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

```
conda install orekit
```

It is possible to list all of the versions of `orekit` available on your platform with:

```
conda search orekit --channel conda-forge
```

Appveyor	Windows passing
Azure	▶ Azure Pipelines succeeded

ARCHITECTUAL DIFFERENCES JAVA AND PYTHON

WITH SOME WORKAROUNDS

- At a basic Orekit usage level it seems to be almost identical usage
- Most code can be translated almost mechanically

But,

- Typing not as hard in python
  - Sometimes explicit casting is needed from python side
- For subclassing Java classes in Python, specific classes are needed
  - Called ”PythonClassName” in the wrapper
- Python does not have overloaded methods / constructors
  - For subclassing java classes in Python a naming scheme has been applied separating the overloaded Orekit classes

CASTING

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

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

SUBCLASSING JAVA CLASSES IN PYTHON

ONE OF THE QUIRKS

- 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 interfaces and selected classes
- *Consider however all PythonClasses that don’t have a test case as experimental*
- 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;
```

SUBCLASSING AND OVERLOADED METHODS

- Some classes and interfaces in Orekit uses overloaded methods (methods with same name, different parameters)
- This is not supported in Python in this form

- For classes that are to be subclassed in Python, a workaround for overloaded methods was needed
- Current solution:
  - The most ”obvious” usage (!) will have same method name as the java method
  - Other methods (often Field varieties) will have an extension to the method name consisting of the First letter(s) of the input parameters

Example:

```
PythonExtendedPVCoordinatesProvider.getPVCoordinates(AbsoluteDate date, Frame frame)
PythonExtendedPVCoordinatesProvider.getPVCoordinates_FF(FieldAbsoluteDate<T> date, Frame frame)
```

SOME THOUGHTS OF PERFORMANCE

OREKIT PYTHON WRAPPER

- Benchmark of performance would be interesting, but not done
- Python – Java gateway takes overhead
- Reduced possibility for JIT optimization (both on java and python side)
- Effect depends alot on how it is used
- Orekit ”internal” performance not affected
- Perform propagation and event detection in the java side
- Avoid frequent callbacks to python (event detectors etc)

DOCUMENTATION AND EXAMPLES

FOR OREKIT PYTHON WRAPPER

- Wrapper documentation at gitlab wiki pages, quite thin
- Java API documentation is the main documentation for functionality, no need to translate / change
- Some test cases translated to Python and some new test cases for specific features. These are the most advanced examples of usage.
- More translation of test cases welcome as contribution, good way to learn and contribute!
- The new forum has become a good place for questions and support

SUPPORT EXPERIENCE

SPECIFICALLY FOR THE WRAPPER

- Environment variable JCC_JDK needs to be set. Done automatically by anaconda, but the environment needs to be activated. This is by far the most common issue.
- Orekit-data.zip file needed. Tempting to include current version of this in the package but this moves focus away from the importance of this file..
- The orekit forum is a positive experience – increased the interaction

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 in between
- Focus on the automated built packages in conda-forge and ease of use

“Add-ons”:

- Better docstrings. The best would be to have full Javadoc, but as a minimum the call parameter types. (JCC)
- More test cases, translation of the java test suite. This is also important as a documentation of how to use the wrapper.
- Transition github build repos to gitlab
- More examples and tutorials as Jupyter notebooks!

DEMO

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://gitlab.orekit.org/orekit-labs/python-wrapper

- Main documentation:
  https://gitlab.orekit.org/orekit-labs/python-wrapper/wikis/home

WE HELP EARTH
BENEFIT FROM SPACE

www.sscspace.com