PyQGIS the comfortable way

Tricks to efficiently work with Python and QGIS

Matthias Kuhn

Olten, 13.6.2019

OPENGIS.ch

ANDROID · [Q]GIS · WEB

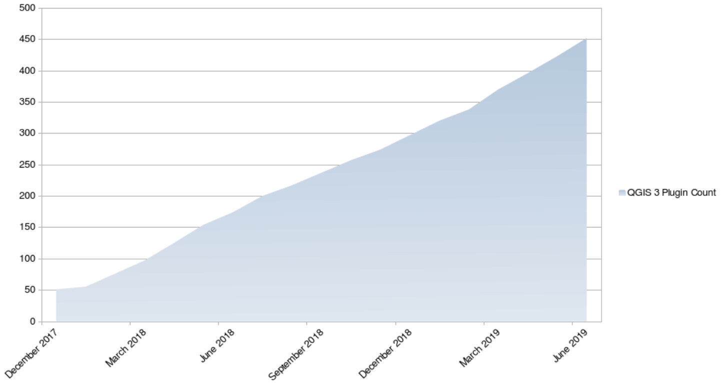
Matthias Kuhn

- QGIS Core Developer
- Co-Founder and CTO of OPENGIS.ch Ltd
- Skier and Mountaineer

Version 0.9 'Ganymede' (2007)

- **Python bindings - This is the major focus of this release it is now possible to create plugins using python. It is also possible to create GIS enabled applications written in python that use the QGIS libraries.**
- Removed automake build system - QGIS now needs CMake for compilation.
- Many new GRASS tools added (with thanks to http://faunalia.it/)
- Map Composer updates
- Crash fix for 2.5D shapefiles
- The QGIS libraries have been refactored and better organised.
- Improvements to the GeoReferencer

Plugin ecosystem



Optimizing PyQGIS

- Various collections of "common pyqgis helper functions" have been written to "make things easier".

See: `http://osgeo-org.1560.x6.nabble.com/`
`QGIS-Developer-Common-PyQGIS-functions-for-QGIS-3-td539564`
`html`

Common PyQGIS functions for QGIS 3

“Wouldn’t it be possible to provide such a collection of common pyqgis functions not only from private persons/projects but from the QGIS-project itself so users could add common functions? I think the chances would be higher that such a "official" collection would be used in the long run and constantly extended."

— Thomas Baumann, QGIS Developer Mailing List

The goal

API first	Make flexible and easy to use APIs. Benefits Python and C++.
Pythonic	Implement “Pythonic" constructs. Leverage modern Python language features.

Decorators

What is a Decorator

“A decorator is the name used for a software design pattern. Decorators dynamically alter the functionality of a function, method, or class without having to directly use subclasses or change the source code of the function being decorated."

— https://wiki.python.org/moin/PythonDecorators

What is a Decorator

“A decorator is the name used for a software design pattern. Decorators dynamically alter the functionality of a function, method, or class without having to directly use subclasses or change the source code of the function being decorated."

— https://wiki.python.org/moin/PythonDecorators

A simpler explanation

Decorators help to write code that is easier to write and read. It's so called "syntactic sugar".

Example

Decorators help to write code that is easier to write and read. It's so called "syntactic sugar".

Atomic operations using with

```python
# Fix population from absolute to relative
layer.startEditing()
for feat in layer.getFeatures():
    feat['population'] = feat['population'] / feat['area']
    layer.updateFeature(feat)
layer.commitChanges()
```

Atomic operations using with

```python
# Fix population from absolute to relative
layer.startEditing()
for feat in layer.getFeatures():
    feat['population'] = feat['population'] / feat['area']
    layer.updateFeature(feat)
layer.commitChanges()
```

```
ZeroDivisionError: division by zero
```

Representing objects, Since QGIS 3.2

```
QgsPoint(2635450,1244252)
```

Representing objects, Since QGIS 3.2

```
QgsPoint(2635450,1244252)
```

```
<QgsPoint: Point (2635450 1244252)>
```

Representing objects

```
QgsPoint(2635450,1244252)
```

Representing objects

```
QgsPoint(2635450,1244252)
```

```
<qgis._core.QgsPoint object at 0x7fcd2b428ee8>
```

Representing objects, Since QGIS 3.2

```
QgsPoint(2635450,1244252)
```

Representing objects, Since QGIS 3.2

```
QgsPoint(2635450,1244252)
```

```
<QgsPoint: Point (2635450 1244252)>
```