

PyQGIS the comfortable way

Tricks to efficiently work with Python and QGIS

Marco Bernasocchi



@mbernasocchi
marco@opengis.ch

OPENGIS.ch

MOBILE · QGIS · POSTGIS



@mbernasocchi?

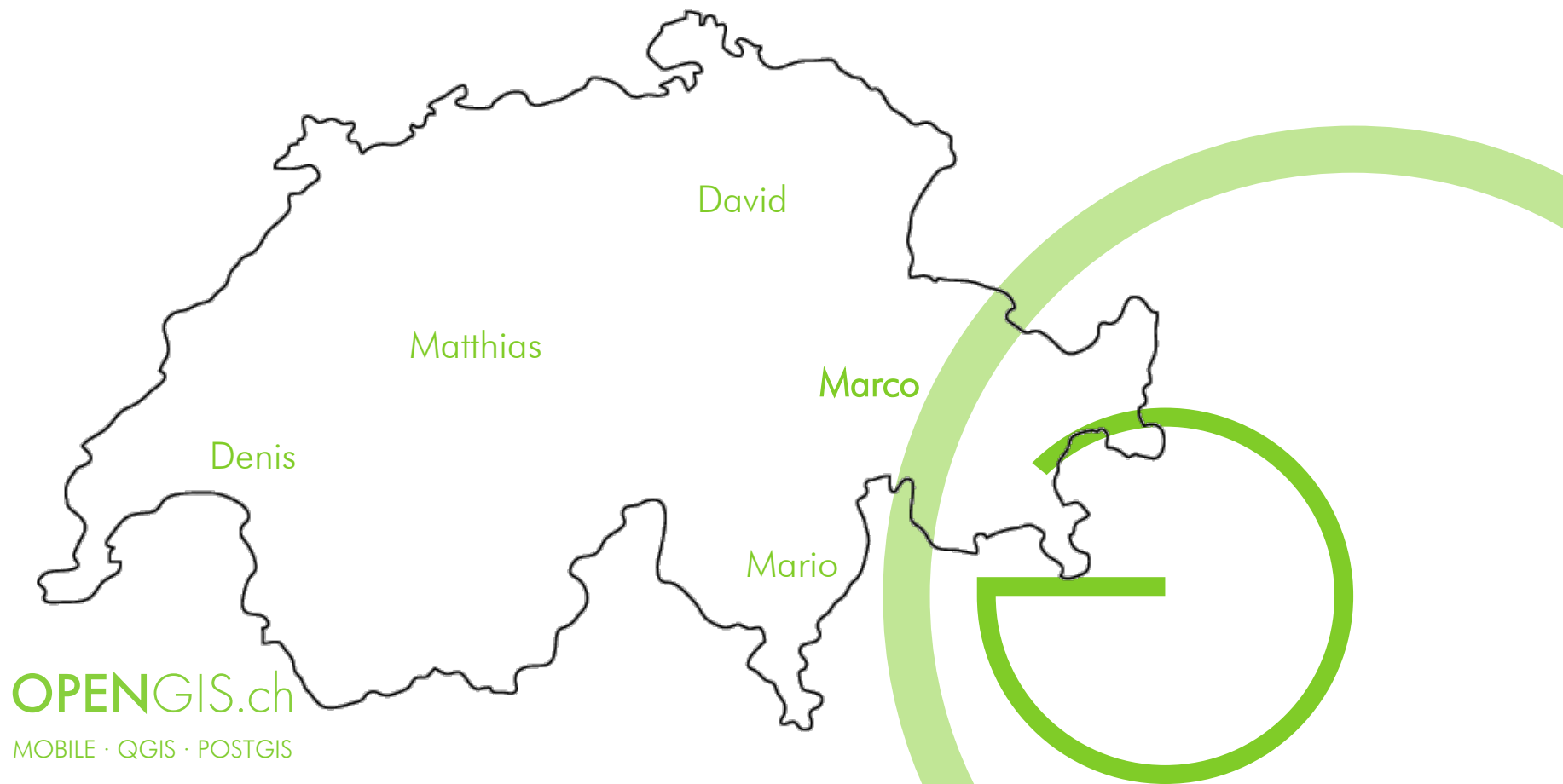
- QGIS Co-Chair
- QGIS on Android dad
- OPENGIS.ch director

OPENGIS.ch

MOBILE · QGIS · POSTGIS

OPENGIS.ch LLC

Opensource geospatial experts at your doorsteps



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

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

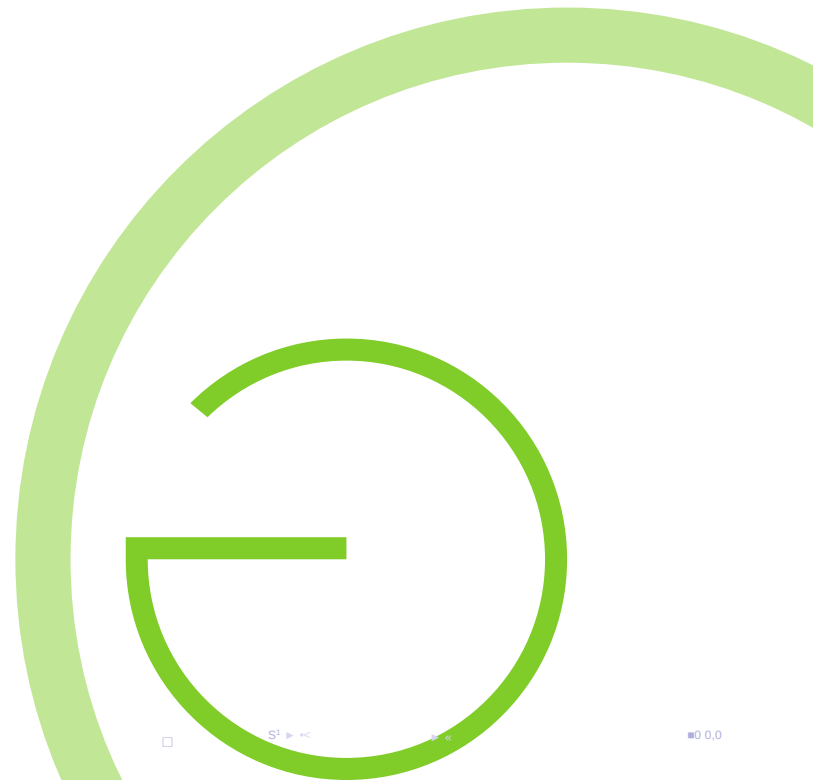


What is a Decorator

- "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>

A simpler explanation

- Decorators help to write code that is easier to write and read. It helps to avoid repeating "boilerplate code".
- It's syntactic sugar.

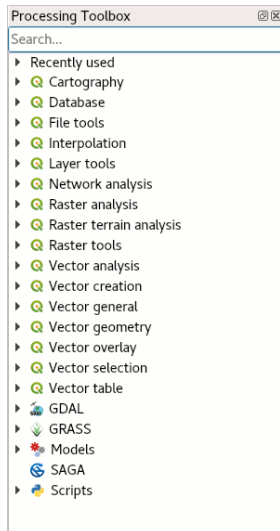


Expression functions

```
1 @qgsfunction(args='auto', group='Custom')
2 def sum(value1, value2, feature, parent):
3     """
4     Calculates the sum of the two parameters value1
5     and value2.
6     <h2>Example usage:</h2>
7     <ul>
8         <li>my_sum(5, 8) -> 13</li>
9         <li>my_sum("field1", "field2") -> 42</li>
10    </ul>
11    """
12    return value1 + value2
```

Processing

- ▶ Modular data processing pipelines
- ▶ Less effort to create the GUI
- ▶ But: A lot of boilerplate code
 - ▶ Processing provider
 - ▶ Methods for input and output definition
 - ▶ Methods for help
 - ▶ Method for the algorithm itself



Processing Algorithm

```
1 class GeoCoding(Algorithm):
2     INPUT: 'INPUT'
3     OUTPUT: 'OUTPUT'
4     COLUMN_PREFIX: 'COLUMN_PREFIX'
5
6     def name(self):
7         return 'geocoding'
8
9     def initAlgorithm(self, config=None):
10         self.addParameter(
11             QgsProcessingParameterFeatureSource(
12                 self.INPUT,
13                 self.tr('Address Layer')
14             )
15         )
16     def displayName(self):
17     def group(self):
18     def shortHelpString(self):
19     ...
```

processing.alg decorator

```
1 @alg(name="geocode", label=alg.tr("GeoCode"))
2 @alg.input(type=alg.SOURCE, name="INPUT", label="
    Adress layer")
3 @alg.input(type=alg.SINK, name="OUTPUT", label="Output
    layer")
4 def geocode(instance, parameters, context, feedback,
    inputs):
5     """
6     Geocode locations. Addresses in, points out.
7     May produce multiple points for an address if
8     ambiguous.
9     """
10    source = instance.parameterAsSource(parameters, "
        INPUT", context)
11    (sink, dest\_id) = instance.parameterAsSink(
        parameters, "OUTPUT", context, source.fields()
        , QgsWkbTypes.Point,
        QgsCoordinateReferenceSystem(4326))
12    GeoCoder.resolve(source, sink)
13
14    return {"OUTPUT": dest\_id}
```

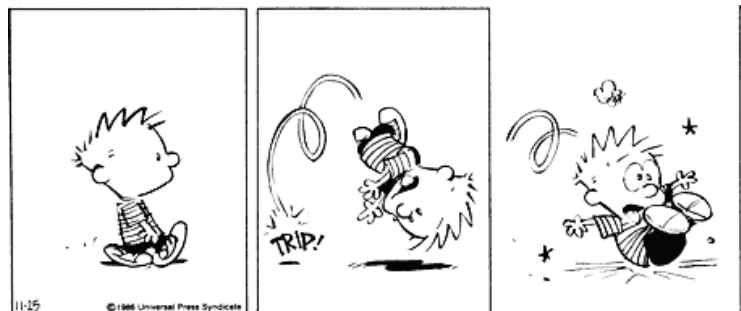
processing.alg decorator

```
1 @alg(name="geocode", label=alg.tr("GeoCode"))
```

Validity checks

```
1 @check.register(type=QgsAbstractValidityCheck.  
    TypeLayoutCheck)  
2 def layout_map_crs_choice_check(context, feedback):  
3     layout = context.layout  
4     results = []  
5     for i in layout.items():  
6         if isinstance(i, QgsLayoutItemMap) and i.crs()  
            .authid() == 'EPSG:3857':  
7             res = QgsValidityCheckResult()  
8             res.type = QgsValidityCheckResult.Warning  
9             res.title='Map projection is misleading'  
10            res.detailedDescription='The projection  
                for the map item {} is set to <i>Web  
                Mercator (EPSG:3857)</i> which  
                misrepresents areas and shapes.  
                Consider using an appropriate local  
                projection instead.'.format(i.  
                    displayName())  
11            results.append(res)  
12  
13     return results
```


When things go wrong



Converting a geometry to a point

```
1 QgsGeometry.fromWkt('POINT(3 4)').asPoint()
```

Converting a geometry to a point

```
1 QgsGeometry.fromWkt('POINT(3 4)').asPoint()  
<QgsPointXY: POINT(3 4)>
```

Converting a geometry to a point

```
1 QgsGeometry.fromWkt('LINESTRING((3 4), (7 8))').  
  asPoint()
```

Converting a geometry to a point

```
1 QgsGeometry.fromWkt('LINESTRING((3 4), (7 8))').  
  asPoint()
```

```
<QgsPointXY: POINT(0 0)>
```

Converting a geometry to a point

```
1 QgsGeometry.fromWkt('LINESTRING((3 4), (7 8))').  
  asPoint()
```

```
<QgsPointXY: POINT(0 0)>
```

... only until QGIS 3.4

Converting a geometry to a point

```
1 QgsGeometry.fromWkt('LINESTRING((3 4), (7 8))').  
  asPoint()
```

Converting a geometry to a point

```
1 QgsGeometry.fromWkt('LINESTRING((3 4), (7 8))').  
  asPoint()
```

```
Traceback (most recent call last):
```

```
  File "plugins/somewhere/plugin.py",
```

```
      line 90, in broken_method
```

```
TypeError: LineString geometry cannot be converted  
      to a point. Only Point types are permitted.
```


When things go wrong

```
1 mp = QgsMultiPoint()  
2 mp.addGeometry(QgsPoint(1,1))  
3 mp.addGeometry(QgsPoint(2,2))
```

When things go wrong

```
1 mp = QgsMultiPoint()  
2 mp.addGeometry(QgsPoint(1,1))  
3 mp.addGeometry(QgsPoint(2,2))
```

```
1 mp.geometryN(3)
```

When things go wrong

```
1 mp = QgsMultiPoint()  
2 mp.addGeometry(QgsPoint(1,1))  
3 mp.addGeometry(QgsPoint(2,2))
```

```
1 mp.geometryN(3)
```

```
IndexError: 3
```

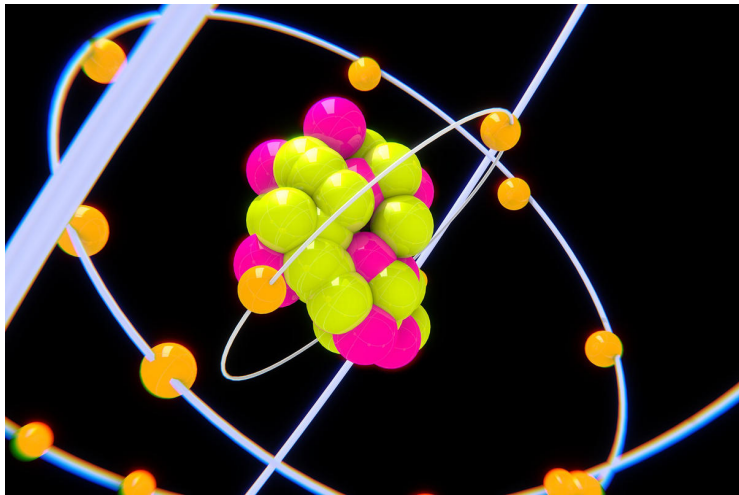
When things go wrong

```
1 try:
2     mp.geometryN(3)
3 except IndexError as e:
4     show_error(self.tr('The geometry is too short.
        Input data linestrings need to have at least 4
        vertices.'))
```

When things go wrong



Atomic Operations



Atomic operations using with

```
1 # Fix population from absolute to relative
2 layer.startEditing()
3 for feat in layer.getFeatures():
4     feat['population'] = feat['population'] / feat['
        area']
5     layer.updateFeature(feat)
6 layer.commitChanges()
7 layer.stopEditing()
```

Atomic operations using with

```
1 # Fix population from absolute to relative
2 layer.startEditing()
3 for feat in layer.getFeatures():
4     feat['population'] = feat['population'] / feat['
        area']
5     layer.updateFeature(feat)
6 layer.commitChanges()
7 layer.stopEditing()
```

ZeroDivisionError: division by zero

Atomic Operations



Atomic operations using "with"

- ▶ Only part of the features are modified
- ▶ The layer may or may not be in edit state any more

Atomic operations using "with"

Let's introduce "with"

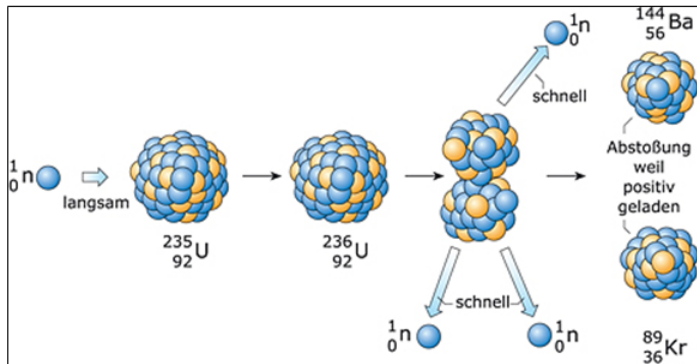
Atomic operations using "with"

```
1 # Fix population from absolute to relative
2 with edit(layer):
3     for feat in layer.getFeatures():
4         feat['population'] = feat['population'] / feat
5         feat['area']
6         layer.updateFeature(feat)
7 # Changes are committed automatically if no error
8 occurred
```

Atomic operations using "with"

```
1 # Fix population from absolute to relative
2 with edit(layer):
3     for feat in layer.getFeatures():
4         feat['population'] = feat['population'] / feat
5         feat['area']
6         layer.updateFeature(feat)
7 # Changes are committed automatically if no error
8     occurred
9
10 # Or if an error occurs, no changes are applied at all
11 ZeroDivisionError: division by zero
```

Atomic Operations



Working with geometries

Iterating vertices

```
1 line = QgsGeometry.fromWkt('LINESTRING(1 1, 2 2)')
2 for vertex in line.vertices():
3     print(vertex)
```


Iterating vertices

```
1 line = QgsGeometry.fromWkt('LINESTRING(1 1, 2 2)')
2 for vertex in line.vertices():
3     print(vertex)
```

```
<QgsPoint: Point (1 1)>
```

```
<QgsPoint: Point (2 2)>
```

Iterating parts

```
1 multipoint = QgsGeometry.fromWkt('MULTIPOINT((1 1), (2  
    2), (3 3))')  
2 for point in multipoint.parts():  
3     print(point)
```

Iterating parts

```
1 multipoint = QgsGeometry.fromWkt('MULTIPOINT((1 1), (2  
    2), (3 3))')  
2 for point in multipoint.parts():  
3     print(point)
```

```
<QgsPoint: Point (1 1)>  
<QgsPoint: Point (2 2)>  
<QgsPoint: Point (3 3)>
```

Representing objects

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

Representing objects

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

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

Representing objects, Since QGIS 3.2

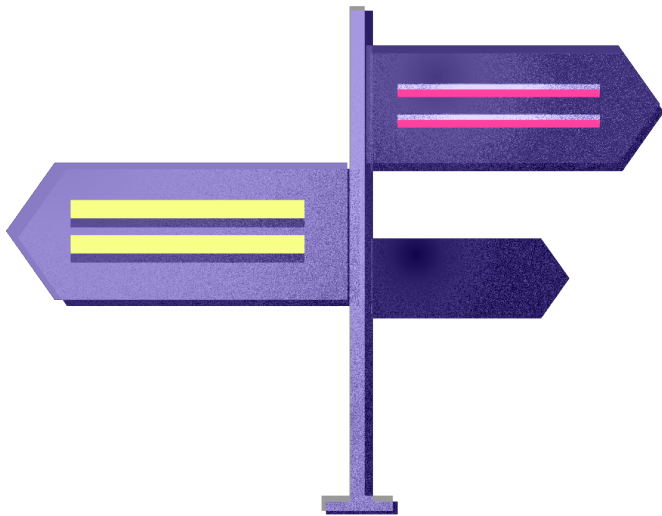
```
1 QgsPoint(2635450,1244252)
```

Representing objects, Since QGIS 3.2

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

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

Outlook



Exception handling

- ▶ Exceptions are good
 - ▶ Exceptions help to fix problems
 - ▶ Exceptions help in case of data corruption
- ▶ More exceptions
- ▶ E.g. instead of return values

Easier initialization

- ▶ A lot of boilerplate code is required to get started with a standalone application
- ▶ Goal: reduce that

More pythonic constructs

- ▶ More decorators
- ▶ More iterators

Nice API

- ▶ But that is not Python specific

Start Coding

- ▶ Let's get to work



Links

- madmanwoo.gitlab.io/foss4g-python-workshop
- docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/
- qgis.org/pyqgis/



Shameless Plug



<https://opengis.ch/qgis-on-the-road>



QGIS needs you!

QGIS needs more contributors



- Users, Testers
- Contributors of case studies
- Documentors and Translators
- Hosts of meetings and hackfests
- Developers
- Sysadmins for infrastructure
- Funders, Donors, Sponsors



We strive for a friendly, welcoming and diverse worldwide community!

Visit <https://www.qgis.org/> for more information or follow @QGIS on Twitter

OPENGIS.ch

MOBILE · QGIS · POSTGIS



Thanks! Questions?

A photograph showing four men in a brewery setting. They are gathered around a large, horizontal copper barrel, which is tilted. The man on the far left is wearing a bright green jacket and a dark cap. The man next to him is wearing a brown sweater. The man in the center is wearing a dark hoodie. The man on the far right is wearing a grey sweater and glasses. They are all looking down at the barrel, which is being poured into a metal container. The background features a stone wall on the left and copper pipes and equipment on the right. A sign hanging in the background reads "WENGERT Brenner- und Bauteile".

Get in touch
@OPENGISch
info@opengis.ch