Algorithmica Research AB

Using Quantlab in Python

A primer

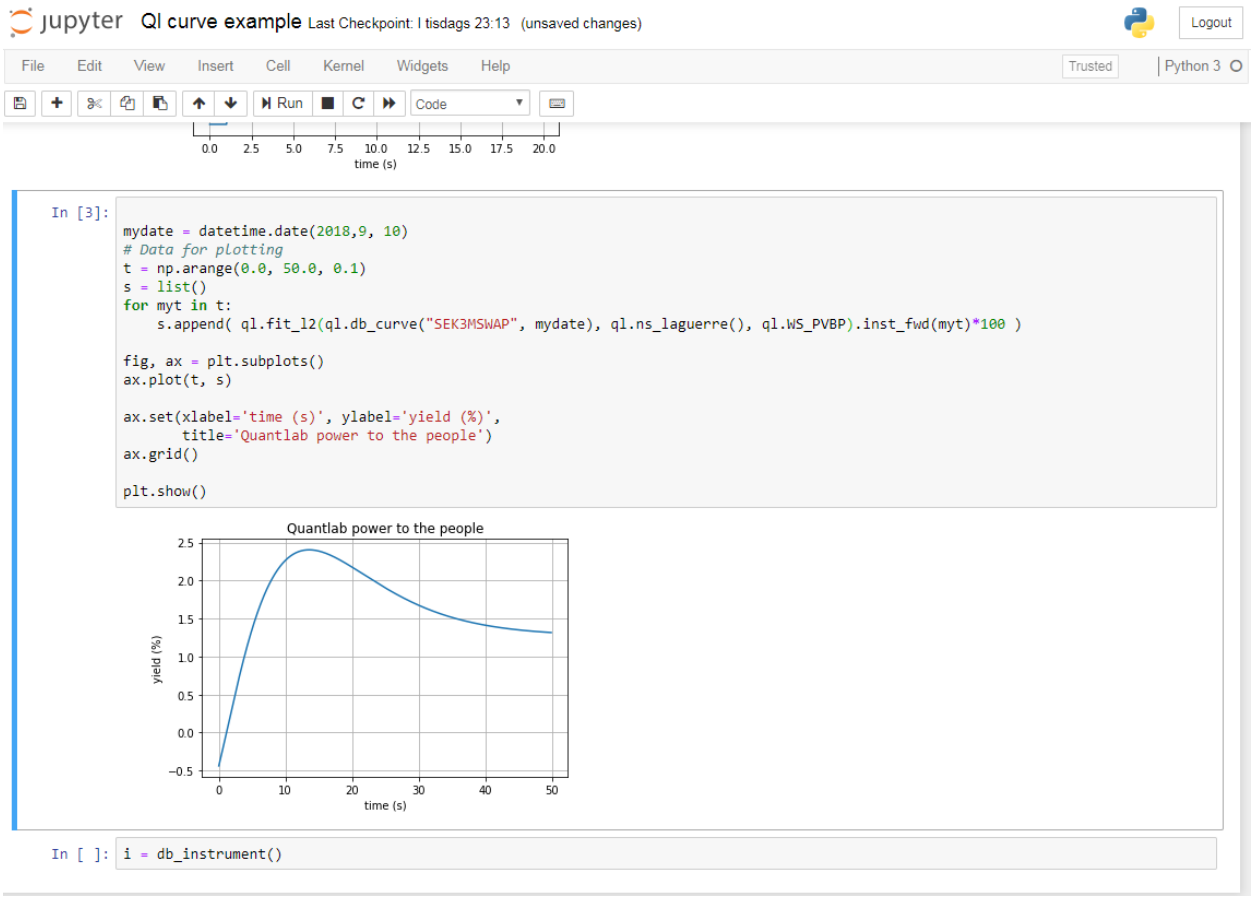
[Using Quantlab ver 3.1.1606 or later]

Last update: 2018-09-14

Algorithmica Research AB

Using the Quantlab Python interface

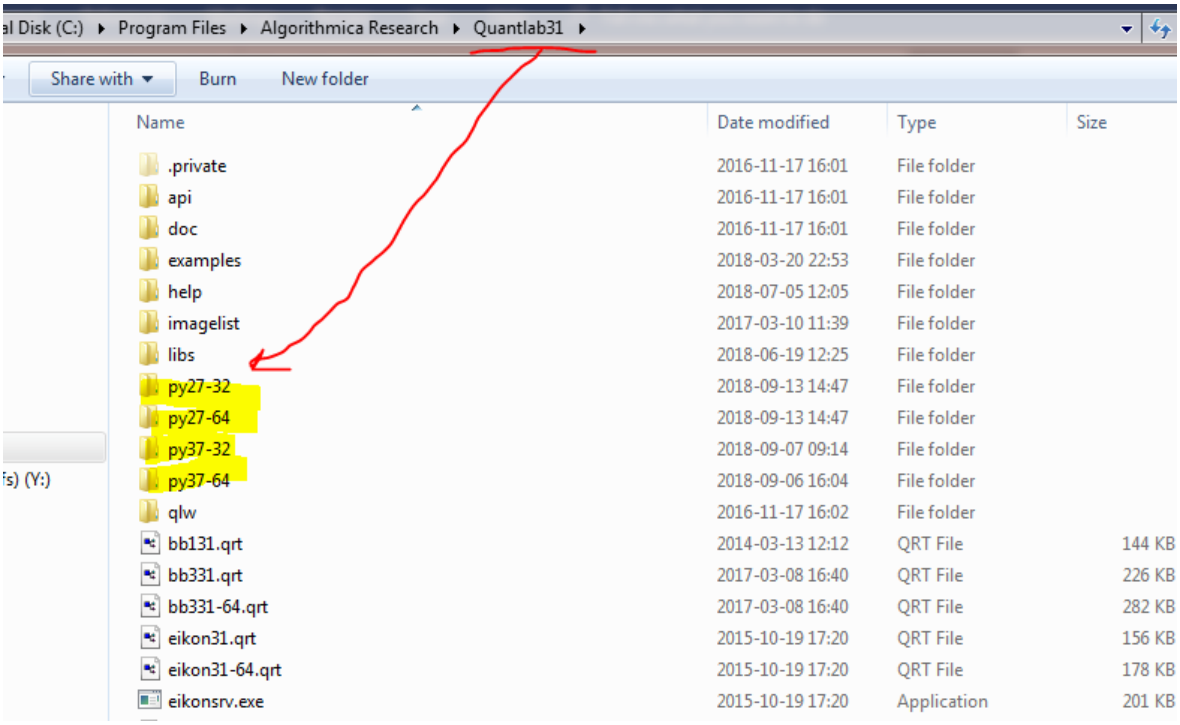
All functions available in Quantlab may be called from any Python 2.7 and/or Python 3.7 environment. This includes not only the built-in library of financial and mathematical functions, but also user-written functions and classes implemented in Qlang (and saved as a library file). Internally compiled c/c++ binarys implemented as a Dynamic Link Library (DLLs) will also be exported to Python.



Using Jupyter Notebook

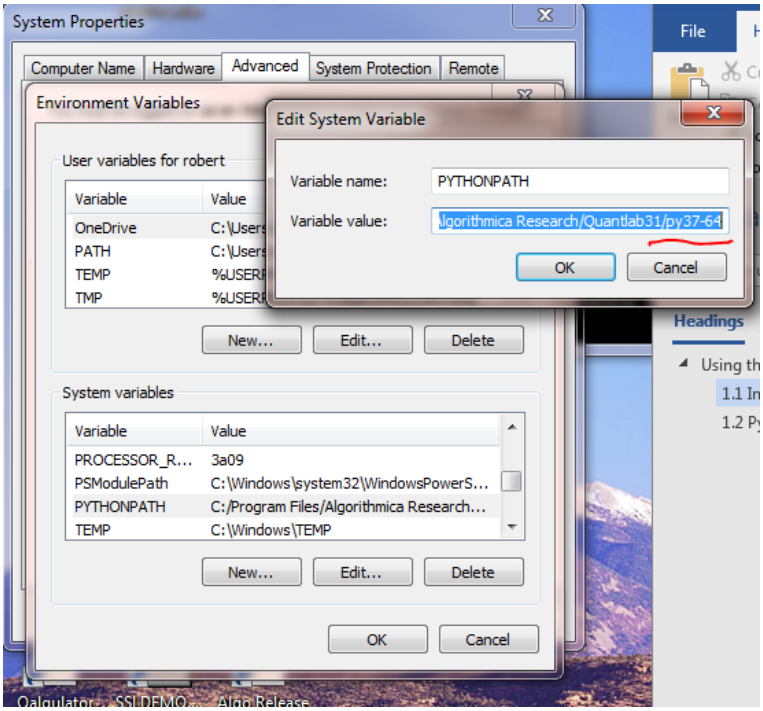
1.1 Installing the Python Quantlab API

Before getting started you need to ensure that you have the proper Quantlab python library in your local Quantlab setup. The relevant version of python support should be placed directly under the Quantlab root. Both 32-bit and 64-bit Quantlab is currently supported.

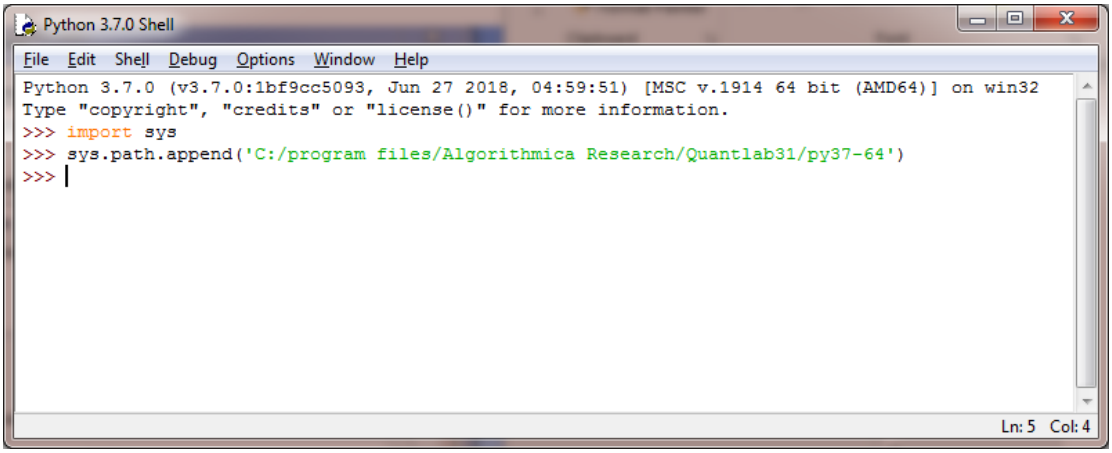


There are two options to load the Quantlab application into Python.

1) To permanently add a system variable PYTHONPATH to the systems environment variables. It should point to the corresponding Python version folder using either 32-bit or 64-bit Quantlab. This means that any time python is started it will recognize the possibility to load the Quantlab library.



2) To dynamically add the Python path using the sys library. Then the Python code will work for any user having a correct Quantlab and Python file-setup regardless of system environment.




```
import sys
sys.path.append("c:/program files/algorithmica research/quantlab31/py37-64")
```

To use Quantlab functions in Python it is enough to use the “import ql” command to have Quantlab available. The session will load all library functions set in the ini file of the current Quantlab. If the instr31.dll is available and a valid ODBC source is pointing to an instrument and curve database, the instrument definitions will be loaded in the same way as starting a Quantlab session.

Remember that also real-time connections to Bloomberg and Reuters will be available to the Python interface.

NOTE: Loading of the Quantlab function set including your internal ql/qll library files is automatically done every time the “import ql” command is run in Python. There is no need to export and register tlb files as with the COM connection.

1.2 Quantlab to Python – handling of function overloading

Note that Python (as well as COM) **does not allow** for overloaded functions and classes to be exported. If any function or class constructor have overloaded variants in Quantlab, none of them will be loaded. To get around this, it is possible to publish a user-defined name to the Python/COM interface. Python and COM use the same user-defined code extension.

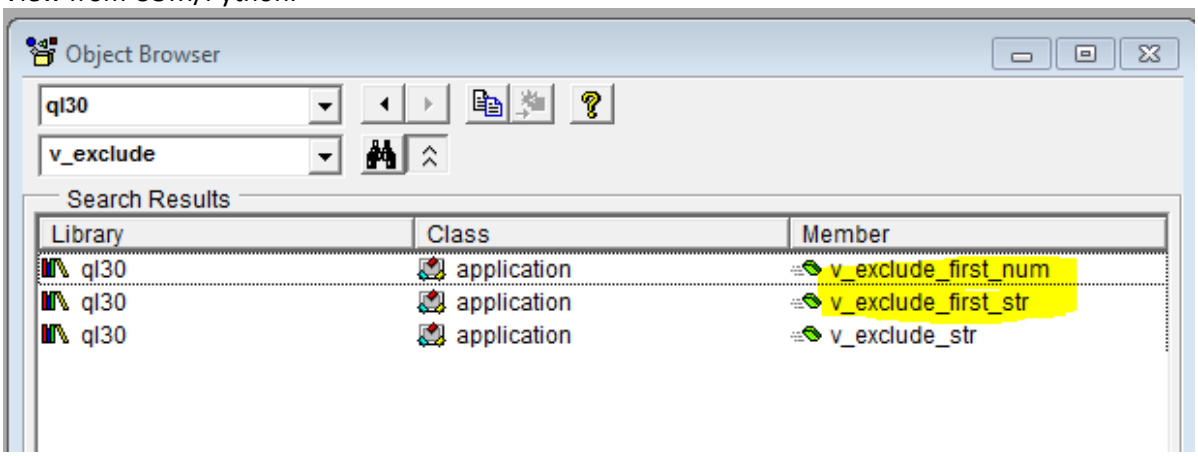
By using the key-word: option (com_name: <string>) directly after the function name, an alternate name of the function will be published to the Python/COM interface.

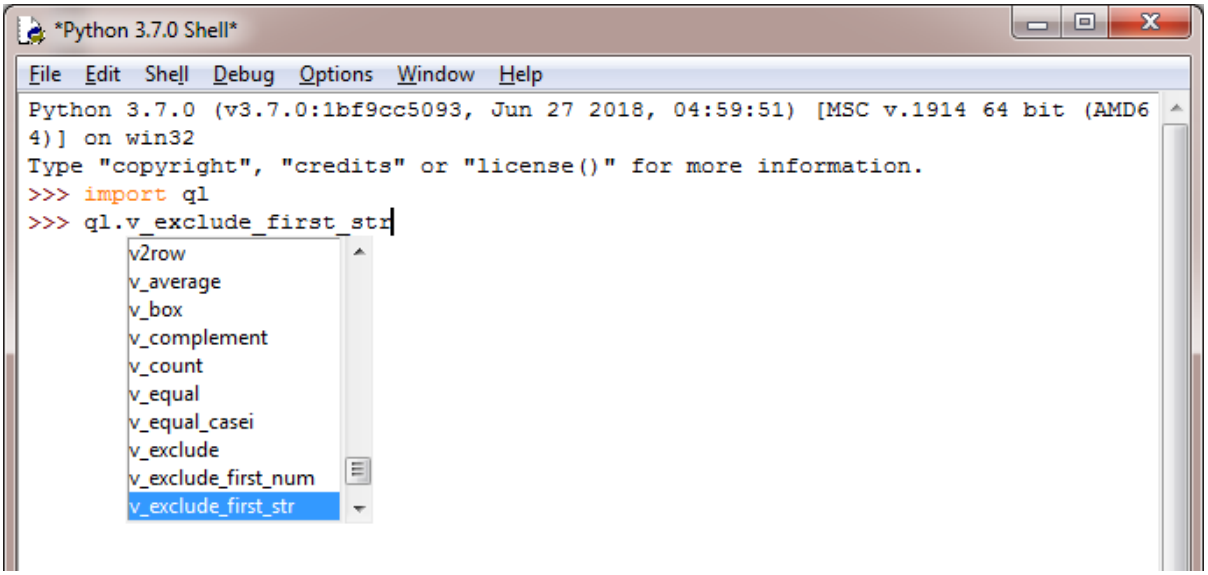
Example:

libs/extras/Python_playground.ql

```
vector(number) v_exclude_first(vector(number) x)
    option(com_name:'v_exclude_first_num')
{
    return x[1:];
}
vector(string) v_exclude_first(vector(string) x)
    option(com_name:'v_exclude_first_str')
{
    return x[1:];
}
```

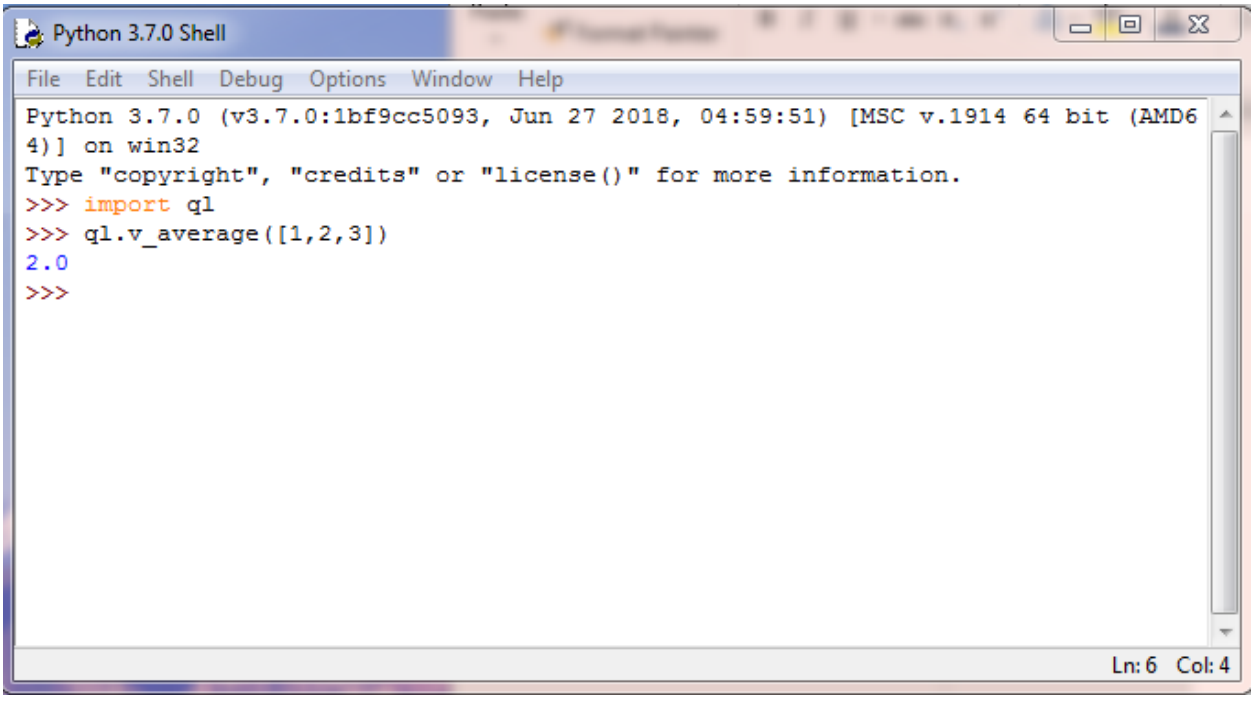
View from COM/Python:



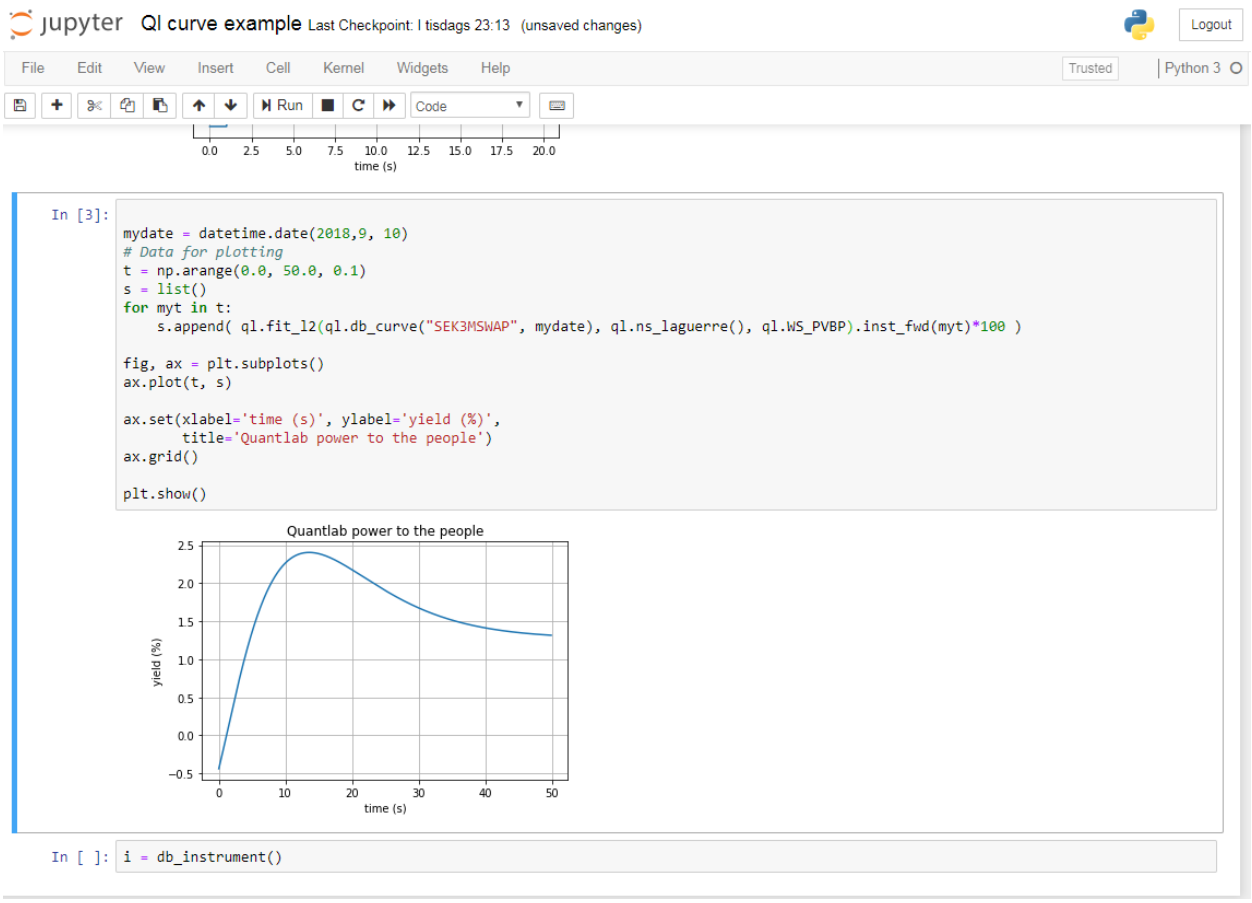


1.3 Python examples

There are many ways to write and run Python code. The Quantlab Python API works equally well from both the IDLE command shell as well as from a Jupyter notebook.



Using the Python Shell

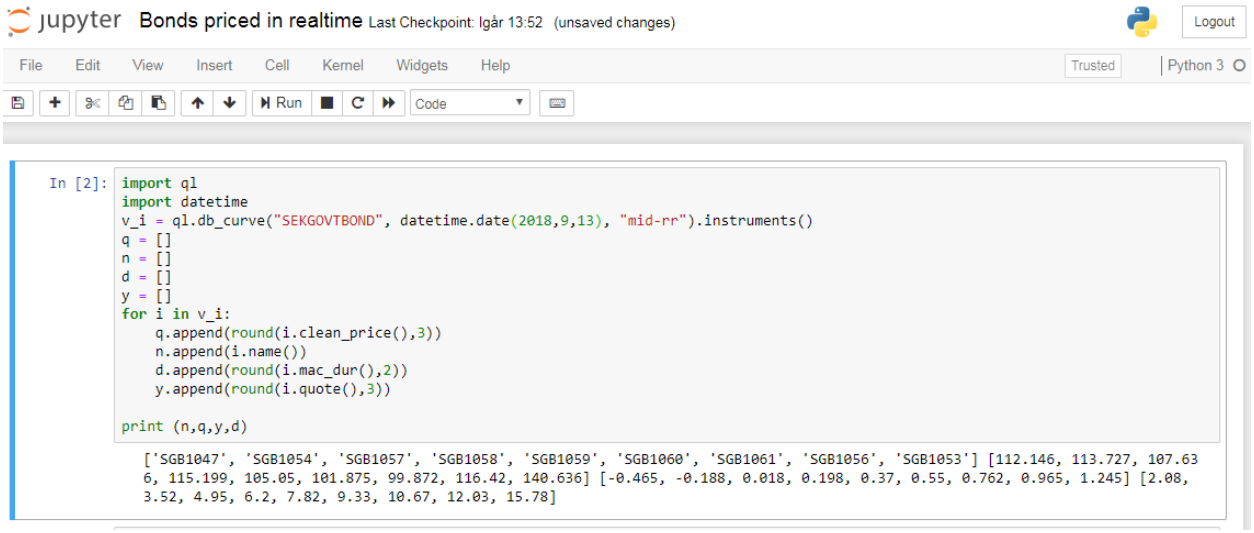


Using Jupyter Notebook

A few things to note when using Qlang / Quantlab from Python:

- As mentioned above, no overloading in Python
- No vector and matrix function expansion in Python. Must loop in Python.
- Function variables passed by reference are not allowed in Python, as they are in Qlang and C++.
- Date type in Python has no overloaded operator +/- which means that “today() – 1” does not work.

A code example calling a set of bonds and extracting some static and pricing data:




```
import ql
import datetime
v_i = ql.db_curve("SEKGOVTBOND", datetime.date(2018,9,14), "mid-rr").instruments()
q = []
n = []
d = []
y = []
for i in v_i:
    q.append(round(i.clean_price(),3))
    n.append(i.name())
    d.append(round(i.mac_dur(),2))
    y.append(round(i.quote(),3))
print (n,q,y,d)
```

A code example calling the database for printing historical price quotes:

```
In [3]: start = datetime.datetime(2018,1,1)
        end = datetime.datetime.today()
        daterange = [start + datetime.timedelta(days=x) for x in range(0, (end-start).days)]
        q = []
        for d in daterange:
            tmp = ql.db_instrument("SHB A:XSTO:SEK", d).quote_adj(True,True)
            if tmp != None:
                q.append(round(tmp,2))

        print(q)
```

```
[104.93, 103.72, 105.68, 106.47, 107.66, 109.52, 110.46, 109.71, 108.47, 107.68, 107.54, 107.26, 108.1, 107.64, 109.36, 109.8,
109.27, 107.17, 108.01, 108.08, 106.7, 106.98, 108.08, 107.54, 106.4, 103.48, 104.28, 102.92, 101.43, 102.27, 103.55, 106.38,
107.36, 108.36, 108.33, 108.73, 107.64, 107.78, 107.33, 106.35, 106.61, 107.08, 103.81, 104.53, 105.4, 106.89, 108.45, 108.24,
108.52, 108.47, 108.61, 107.57, 107.29, 105.44, 104.21, 103.2, 99.63, 100.62, 99.68, 100.28, 101.38, 104.22, 101.45, 101.62, 1
02.57, 100.72, 100.45, 100.07, 98.69, 98.4, 98.2, 99.06, 100.12, 99.37, 99.97, 99.85, 100.75, 99.18, 95.67, 97.83, 99.2, 98.0
4, 98.54, 98.32, 98.75, 98.6, 99.32, 100.47, 100.05, 99.84, 99.96, 98.28, 99.7, 99.64, 98.8, 98.8, 98.09, 98.47, 97.82, 98.23,
96.92, 97.32, 96.64, 98.81, 97.52, 96.61, 97.34, 97.22, 99.18, 99.56, 96.91, 98.78, 96.92, 96.15, 97.31, 97.52, 96.99, 97.28,
97.45, 98.56, 98.65, 99.69, 98.54, 98.53, 98.17, 98.45, 98.93, 99.4, 99.94, 99.15, 99.05, 98.7, 100.25, 101.25, 98.42, 99.04,
102.55, 105.1, 106.6, 106.53, 107.75, 107.62, 107.55, 108.53, 108.18, 108.22, 108.47, 107.82, 108.35, 108.22, 109.8, 108.7, 10
8.6, 107.78, 109.03, 110.28, 110.55, 111.03, 110.8, 110.93, 111.2, 111.1, 112.2, 111.18, 111.97, 111.72, 110.88, 110.82, 110.
2, 109.72, 109.97, 108.22, 108.47, 107.4, 106.55]
```

```
start = datetime.datetime(2018,1,1)
end = datetime.datetime.today()
daterange = [start + datetime.timedelta(days=x) for x in range(0, (end-start).days)]
q = []
for d in daterange:
    tmp = ql.db_instrument("SHB A:XSTO:SEK", d).quote()
    if tmp != None:
        q.append(round(tmp,2))

print(q)
```