
PhreeqPy Documentation

Release 0.2

Mike Müller

<http://www.hydrocomputing.com>

January 21, 2013

CONTENTS

1	PhreeqPy - Python Tools for PHREEQC	1
1.1	Introduction	1
1.2	Installation	1
2	Class IPhreeqc	3
3	Examples for PhreeqPy	5
4	Changelog history	11
4.1	Changes between versions 0.1 and 0.2	11
5	Contact	13
6	Indices and tables	15

PHREEQPY - PYTHON TOOLS FOR PHREEQC

1.1 Introduction

PhreeqPy is Open Source software and provides Python tools to work with [PHREEQC](#).

Currently it provides access to the new [IPhreeqc](#) interface without the need to run a COM server and therefore also works on non-Windows systems. IPhreeqc is described in more detail in this [publication](#).

Please let us know what you do with PhreeqPy or if things do not work as expected. There is a [mailing list](#) for [PhreeqPy](#). Just sent us you email address to subscribe with the header *subscribe to phreeqpy list*.

Download latest version of this documentation as PDF

License: BSD

1.2 Installation

Pythons (tested): Python 2.6, 2.7, 3.3, PyPy

Pythons (untested but should work since using ctypes): IronPython, Jython 2.7

Platforms: Unix/Posix and Windows

PyPI package name: [phreeqpy](#)

1.) Installation options:

```
pip install -U phreeqpy # or
easy_install -U phreeqpy
```

2.) Download:

```
wget http://www.phreeqpy.com/download/phreeqpy0.2.zip
tar -xzf phreeqpy0.2.zip
cd phreeqpy0.2
sudo python setup install
```

You need a a IPhreeqc shared libray for your operating system. PhreeqPy comes with shared libraries for 32-bit Windows, 32-bit Linux and 64-bit Mac OS X. More shared libraries for differnet platforms will come with following releases.

You may download an appropriate library from here: <ftp://brrftp.cr.usgs.gov/pub/charlton/iphreeqc/>

For example for Linux:

```
wget ftp://brrftp.cr.usgs.gov/pub/charlton/iphreeqc/iphreeqc-2.18.4-6386.tar.gz
tar -xvzf iphreeqc-2.18.4-6386.tar.gz
cd iphreeqc-2.18.4-6386
./configure
make
make check
sudo make install
```

Then either use the full path to the shared library when making an instance of `iphreeqc_dll.IPhreeqc`

```
iphreeqc = phreeqpy.iphreeqc.phreeqc_dll.IPhreeqc('/full/path/to/libiphreeqc.so')
```

or copy the shared object into `phreeqpy/iphreeqc` replacing the existing one. For example:

```
sudo cp /usr/local/lib/libiphreeqc.so /path/to/site-packages/PhreeqPy-0.2.0-py2.7.egg/phreeqpy/iphreeqc
```

CLASS IPHREEQC

This is the main class to work with the IPhreeqc interface.

You can *add your own shared library for IPhreeqc*.

EXAMPLES FOR PHREEQPY

This is an example for the use PhreeqPy for one-dimensional advection. It is the example 11 from the PHREEQC users manual.

```
"""Advection with DLL or COM server.

Using MODFIY we update the concentration on every
time step. We shift by one cell per step.
"""

from __future__ import print_function

import sys

# Simple Python 3 compatibility adjustment.
if sys.version_info[0] == 2:
    range = xrange

import os
import timeit

MODE = 'dll' # 'dll' or 'com'

if MODE == 'com':
    import phreeqpy.iphreeqc.phreeqc_com as phreeqc_mod
elif MODE == 'dll':
    import phreeqpy.iphreeqc.phreeqc_dll as phreeqc_mod
else:
    raise Exception('Mode "%s" is not defined use "com" or "dll".' % MODE)

def make_initial_conditions():
    """
    Specify initial conditions data blocks.

    Uniform initial conditions are assumed.
    """
    initial_conditions = """
TITLE Example 11.--Transport and ion exchange.
SOLUTION 0  CaCl2
    units          mmol/kgw
    temp            25.0
    pH              7.0      charge
    pe              12.5      O2 (g)    -0.68
    Ca              0.6
```

```

        C1                1.2
    SOLUTION 1  Initial solution for column
        units            mmol/kgw
        temp              25.0
        pH                7.0      charge
        pe                12.5      O2(g)    -0.68
        Na                1.0
        K                 0.2
        N(5)              1.2
    END
    EXCHANGE 1
        equilibrate 1
        X                0.0011
    END
    """
    return initial_conditions

def make_selected_output(components):
    """
    Build SELECTED_OUTPUT data block
    """
    headings = "-headings      cb      H      O      "
    for i in range(len(components)):
        headings += components[i] + "\t"
    selected_output = """
    SELECTED_OUTPUT
        -reset false
    USER_PUNCH
    """
    selected_output += headings + "\n"
    #
    # charge balance, H, and O
    #
    code = '10 w = TOT("water")\n'
    code += '20 PUNCH CHARGE_BALANCE, TOTMOLE("H"), TOTMOLE("O")\n'
    #
    # All other elements
    #
    lino = 30
    for component in components:
        code += '%d PUNCH w*TOT("%s")\n' % (lino, component)
        lino += 10
    selected_output += code
    return selected_output

def initialize(cells, first=False):
    """
    Initialize IPhreeqc module
    """
    phreeqc = phreeqc_mod.IPhreeqc()
    phreeqc.load_database(r"phreeqc.dat")
    initial_conditions = make_initial_conditions()
    phreeqc.run_string(initial_conditions)
    components = phreeqc.get_component_list()
    selected_output = make_selected_output(components)
    phreeqc.run_string(selected_output)
```

```

phc_string = "RUN_CELLS; -cells 0-1\n"
phreeqc.run_string(phc_string)
conc = get_selected_output(phreeqc)
inflow = {}
initial = {}
for name in conc:
    if first:
        inflow[name] = conc[name][0]
    else:
        inflow[name] = conc[name][1]
        initial[name] = conc[name][1]
task = initial_conditions + "\n"
task += "COPY solution 1 %d-%d\n" % (cells[0], cells[1])
task += "COPY exchange 1 %d-%d\n" % (cells[0], cells[1])
task += "END\n"
task += "RUN_CELLS; -cells %d-%d\n" % (cells[0], cells[1])
task += selected_output
phreeqc.run_string(task)
conc = get_selected_output(phreeqc)
for name in conc:
    value = [initial[name]] * len(conc[name])
    conc[name] = value
return phreeqc, inflow, conc

def advect_step(phreeqc, inflow, conc, cells):
    """Advect by shifting concentrations from previous time step.
    """
    all_names = conc.keys()
    names = [name for name in all_names if name not in ('cb', 'H', 'O')]
    for name in conc:
        # shift one cell
        conc[name][1:] = conc[name][:-1]
        conc[name][0] = inflow[name]
    modify = []
    for index, cell in enumerate(range(cells[0], cells[1] + 1)):
        modify.append("SOLUTION_MODIFY %d" % cell)
        modify.append("\tcb      %e" % conc['cb'][index])
        modify.append("\t-total_h %f" % conc['H'][index])
        modify.append("\t-total_o %f" % conc['O'][index])
        modify.append("\t-totals")
        for name in names:
            modify.append("\t\t%s\t%f" % (name, conc[name][index]))
    modify.append("RUN_CELLS; -cells %d-%d\n" % (cells[0], cells[1]))
    cmd = '\n'.join(modify)
    phreeqc.run_string(cmd)
    conc = get_selected_output(phreeqc)
    return conc

def get_selected_output(phreeqc):
    """Return calculation result as dict.

    Header entries are the keys and the columns
    are the values as lists of numbers.
    """
    output = phreeqc.get_selected_output_array()
    header = output[0]

```

```
conc = {}
for head in header:
    conc[head] = []
for row in output[1:]:
    for col, head in enumerate(header):
        conc[head].append(row[col])
return conc

def run(ncells, shifts, specie_names):
    """Do one run in one process.
    """
    cells = (1, ncells)
    phreeqc, inflow, conc = initialize(cells, first=True)
    outflow = {}
    for name in specie_names:
        outflow[name] = []
    for _counter in range(shifts):
        # advect
        conc = advect_step(phreeqc, inflow, conc, cells)
        for name in specie_names:
            outflow[name].append(conc[name][-1])
    return outflow

def write_outflow(file_name, outflow):
    """Write the outflow values to a file.
    """
    fobj = open(file_name, 'w')
    header = outflow.keys()
    for head in header:
        fobj.write('%20s' % head)
    fobj.write('\n')
    for lineno in range(len(outflow[head])):
        for head in header:
            fobj.write('%20.17f' % outflow[head][lineno])
    fobj.write('\n')

def main(ncells, shifts):
    """Run different versions with and without multiprocessing
    """

    def measure_time(func, *args, **kwargs):
        """Convenience function to measure run times.
        """
        start = timeit.default_timer()
        result = func(*args, **kwargs)
        return result, timeit.default_timer() - start

    print('Dimensions')
    print('=====')
    print('number of cells: ', ncells)
    print('number of shifts ', shifts)
    specie_names = ('Ca', 'Cl', 'K', 'N', 'Na')
    outflow, run_time = measure_time(run, ncells, shifts, specie_names)
    if not os.path.exists('data'):
        os.mkdir('data')
```

```
write_outflow('data/out.txt', outflow)
print('run time:', run_time)
print("Finished simulation\n")

if __name__ == '__main__':
    main(ncells=40, shifts=120)
```


CHANGELOG HISTORY

4.1 Changes between versions 0.1 and 0.2

- Added more IPhreeqc functions.
- Added support for Mac OS X.
- Added error handling turning IPhreeqc errors into Python exceptions.
- Added Python 3 compatibility. Tested with Python 3.3.
- Added documentation in addition to example on website.

CONTACT

hydrocomputing GmbH & Co. KG
Zur Schule 20
04158 Leipzig
Germany

Tel: +49 341 525 599 54
Fax: +49 341 520 4495
mail: info [at] hydrocomputing [dot] com
www: <http://www.hydrocomputing.com>

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*