PhreeqPy Documentation

Release 0.4.0

Mike Müller

http://www.hydrocomputing.com

Jan 23, 2021

CONTENTS

CHAPTER

ONE

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. Please subscribe to get your questions about PhreeqPy answered.

Download latest version of this documentation as PDF

License: BSD

1.2 Installation

Pythons (tested): Python 2.6, 2.7, 3.3, 3.4, 3.5, 3.6, 3.8, 3.7, 3.9 PyPy

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

Platforms: Unix/Posix and Windows

PyPI package name: phreeqpy

Installation with *pip*:

```
pip install -U phreeqpy
```

You need an 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. There is no 64-bit Windows distributed with PhreeqPy. Please install the **`Windows COM 64-bit`_** module of IPhreeqc and use *phreeqpy.iphreeqc.phreeqc_com as phreeqc_mod* instead of *import phreeqpy.iphreeqc.phreeqc_dll as phreeqc_mod* as shown in the example. If you use Anaconda or Miniconda make sure to:

conda install pywin32

before you install PhreeqPy.

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 -xzvf 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 libray when making an instance of `phreeqc_dll.IPhreeqc`

```
phreeqc = 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-pacges/PhreeqPy-0.2.0-py2.7.egg/
↪phreeqpy/iphreeqc/libiphreeqc.so.0.0.0
```

1.3 Benchmark Test Comparing PhreeqPy to External Processes, COM and C++

This publication demonstrates how PhreeqPy can be used for reactive transport modeling.

Müller M., Parkhurst D.L., Charlton S.R. (2011) Programming PHREEQC Calculations with C++ and Python - A Comparative Study , In: Maxwell R., Poeter E., Hill M., Zheng C. (2011) MODFLOW and More 2011 - Integrated Hydrological Modeling, Proceedings, pp. 632 - 636.

CHAPTER

TWO

CLASS IPHREEQC

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

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

Access PHREEQC-DLL via ctypes.

This is exchangeable with the COM interface.

class phreeqpy.iphreeqc.phreeqc_dll.**IPhreeqc**(*dll_path=None*)

Wrapper for the IPhreeqc DLL.

Connect to DLL and create IPhreeqc.

The optional *dll_path* takes a path to the IPhreeqc shared library. If not provided it tries to select an appropriate library. Make sure you have the right library for your operating system. You may download one from here: ftp://brrftp.cr.usgs.gov/pub/charlton/iphreeqc/

See the PhreeqPy documentation for help on compiling a IPhreeqc shared library.

accumulate_line(*line*)

Put line in input buffer.

add_error(*phc_error_msg*)

Add an error message to Phreeqc.

add_warning(*phc_warn_msg*)

Add an warning message to Phreeqc.

clear_accumlated_lines()

Clear the input buffer.

property column_count

Get number of columns in selected output.

property component_count

Return the number of components.

create_iphreeqc()

Create a IPhreeqc object.

destroy_iphreeqc()

Delete the current instance of IPhreeqc.

get_component(*index*)

Get one component.

get_component_list()

Return all component names.

get_error_string()
Retrieves the error messages.

get_selected_output_array()
Get all values from selected output.

get_selected_output_column(*col*)
Get all values for one column from selected output.

get_selected_output_row(*row*)
Get all values for one from selected output.

get_selected_output_value(*row*, *col*)
Get one value from selected output at given row and column.

load_database(*database_name*)
Load a database with given file_name.

load_database_string(*input_string*)
Load a datbase from a string.

static raise_ipq_error(*error_code*)
There was an error, raise an exception.

raise_string_error(*errors*)
Raise an exception with message from IPhreeqc error.

property row_count
Get number of rows in selected output.

run_string(*cmd_string*)
Run PHREEQC input from string.

set_output_file_off()
Turn on writing to selected output file.

set_output_file_on()
Turn on writing to selected output file.

set_selected_output_file_off()
Turn on writing to selected output file.

set_selected_output_file_on()
Turn on writing to selected output file.

exception phreeqpy.iphreeqc.phreeqc_dll.**PhreeqcException**
Error in Phreeqc call.

class phreeqpy.iphreeqc.phreeqc_dll.**VAR**
Struct with data type and data values.

See Var.h in PHREEQC source.

class phreeqpy.iphreeqc.phreeqc_dll.**VARUNION**
Union with types.

See Var.h in PHREEQC source.

CHAPTER

THREE

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
```

(continues on next page)

(continued from previous page)

```
        Cl              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)
```

(continues on next page)

(continued from previous page)

```
    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("\t-cb      %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.
    """
```

(continues on next page)

(continued from previous page)

```
    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):
        """Convinience 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')
```

(continues on next page)

(continued from previous page)

```
    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)
```

CHAPTER

FOUR

CHANGELOG HISTORY

4.1 Changes between versions 0.2 and 0.3

- Added *set_output_file_off()`and `set_output_file_on()* that call IPhreeqc's *SetOutputFileOn*

4.2 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.

CHAPTER

FIVE

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

CHAPTER

SIX

DATENSCHUTZERKLÄRUNG

Diese Seite beschreibt die Erhebung, Verarbeitung und Nutzung Ihrer personenbezogenen Daten beim Besuch dieser Website. Ihre Daten werden im Rahmen der gesetzlichen Vorschriften geschützt. Sie erfahren hier, welche Daten während Ihres Besuchs auf der Website erfasst und wie diese genutzt werden.

Über diese Website erheben wir keine personenbezogenen Daten.

6.1 1. Datenerhebung und -protokollierung

Wir speichern von Ihrem Besuch Log-Dateien mit diesem Inhalt:

- pseudonymisierte IP-Adresse (ohne Personenbezug, siehe Erklärung unten)
- Datum und Uhrzeit des Zugriffs
- Art des Zugriffes (z.B. GET oder POST)
- besuchten Seiten innerhalb unseres Angebots
- Protokoll (z.B. HTTP 1.1)
- übertragene Datenmenge
- Meldung über erfolgreichen Abruf
- anfragende Domain (z.B. google.com)
- Webbrowser und Betriebssystem

Bei uns werden nur pseudonymisierte IP-Adressen von Besuchern der Website gespeichert. Auf Webserver-Ebene erfolgt dies dadurch, dass im Logfile standardmäßig stat der tatsächlichen IP-Adresse des Besuchers z.B. 123.123.123.123 eine IP-Adresse 123.123.123.XXX gespeichert wird, wobei XXX ein Zufallswert zwischen 1 und 254 ist. Die Herstellung eines Personenbezuges ist nicht mehr möglich.

6.2 2. Verwendung von Cookies

Wir verwenden keine Cookies.

CHAPTER

SEVEN

DATA PROTECTION STATEMENT

English Version for your convenience. The German version below is the legally binding one.

This page describes the collection, processing, and usage of your personal data while visiting this website. Your data is protected within the legal regulations. This page explains what data are collected while visiting this website and how they are used.

We do not collect personal via this website.

7.1 1. Data Collection and Data Logging

We store log files about your visit with this content:

- pseudonymised IP (no person identity can be establish, see explain below)
- date and time of access
- type of access (e.g. GET or POST)
- visited page on website
- protocol (e.g. HTTP 1.1)
- amount of transmitted data
- referring domain (e.g. google.com)
- type of webbrowser

We do not save hostname or Ip address of your request. The collected data is not personal and don't allow to find out who is visiting our website.

We only store pseudonymised IP addresses of visitors to the website. At the web server level, this happens by default by storing an IP address 123.123.123.XXX in the log file instead of the visitor's actual IP address, for example, 123.123.123.123. The "XXX" is a random value between 1 and 254, so it is no longer possible to establish the true identity of the visitor.

7.2 2. Use of Cookies

We do not use cookies.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

S

V