pyswmm Documentation

Release 0.5.2

Bryant E. McDonnell (EmNet LLC)

Oct 09, 2019

Contents

CHAPTER 1

Overview

PySWMM is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and function of complex networks.

With PySWMM you can load and manipulate USEPA Stormwater Management Models. With the development of PySWMM, control algorithms can now be developed exclusively in Python which allows the use of functions and objects as well as storing and tracking hydraulic trends for control actions.

1.1 Who uses PySWMM?

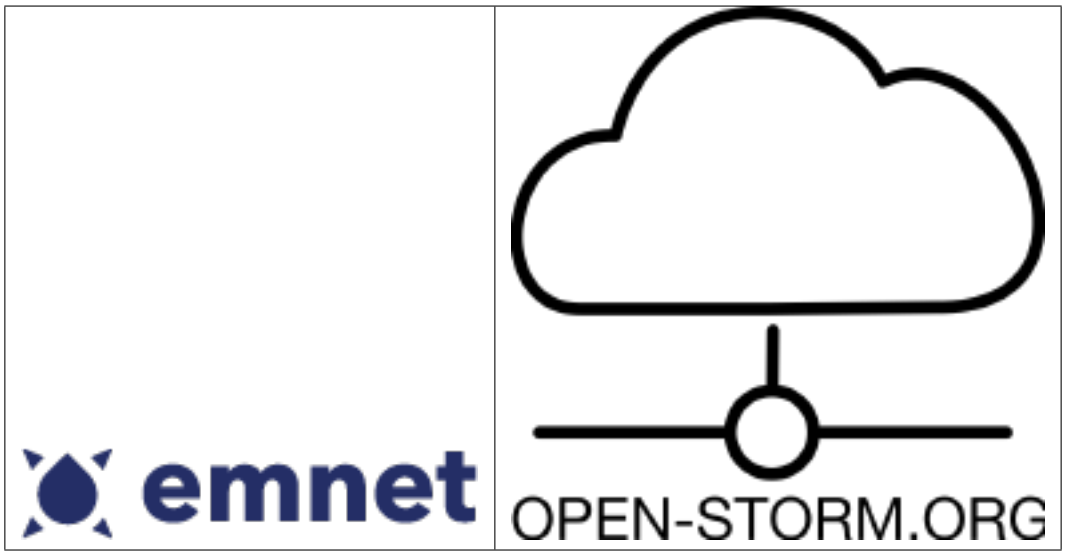
PySWMM is used by engineers, modelers, and researchers who want to streamline stormwater modeling optimization, controls, and post-processing results.

1.2 Goals

PySWMM is intended to provide

- tools for the study of the structure and dynamics within USEPA SWMM5,
- a standard programming interface and graph implementation that is suitable for many applications,
- a rapid development environment for collaborative, multidisciplinary projects,
- an interface to USEPA SWMM5,
- development and implementation of control logic outside of native EPA-SWMM Controls,
- methods for users to establish their own node inflows,
- a coding interface to binary output files,
- new modeling possibilities for the SWMM5 Community.

1.3 Powered By



1.4 Free software

PySWMM is free software; you can redistribute it and/or modify it under the terms of the *BSD License*. We welcome contributions from the community. Information on PySWMM development is found at the PySWMM Github Page https://github.com/OpenWaterAnalytics/pyswmm

1.4.1 What Next

- *A Brief Tour*
- *Installing*
- *Reference*
- Examples

CHAPTER 2

Download

2.1 Software

Source and binary releases: http://pypi.python.org/pypi/pyswmm/

Github (latest development): https://github.com/OpenWaterAnalytics/pyswmm

CHAPTER 3

Authors

3.1 Maintainers

- Bryant E. McDonnell
- Katherine Ratliff
- Jennifer Wu

3.2 Authors with Copyrighted Contributions

- Bryant E. McDonnell
- Katherine Ratliff
- Jennifer Wu
- Gonzalo Peña-Castellanos
- Stephen Roberts
- Abhiram Mullapudi
- Jiada Li

3.3 Authors with Contributions in the Public Domain

- Michael Tryby

CHAPTER 4

Installing

4.1 Installing with pip

Try to install it with

```
pip install pyswmm
```

and an attempt will be made to find and install an appropriate version that matches your operating system and Python version.

You can also get pyswmm from the Python Package Index manually at http://pypi.python.org/pypi/pyswmm To use pip, you need to have setuptools installed.

4.2 Requirements

4.2.1 Python

To use PySWMM you need Python 3.6 or greater.

CHAPTER 5

Tutorial

Start here to begin working with pyswmm.

The pyswmm package allows seamless interaction with the USEPA-SWMM5 data model. Parameter getters/setters and results getters have been exposed, allowing the user to see results while a simulation is running as well as update link settings.

5.1 Loading a Model

There are three options to load a model. If there is no desire to interact with the simulation then the simplest way to run the model is the following:

```
>>> from pyswmm import Simulation
>>>
>>> sim = Simulation('./testmodel.inp')
>>> sim.execute()
```

The following method allows the user to read in a model and manually step through the simulation and get/set parameters and results. This scenario is the cleanest solution using a with statement. It automatically cleans up after the simulation is complete.

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('./testmodel.inp') as sim:
...     for step in sim:
...             pass
```

One feature that pyswmm adds to the modeling world is the simulation stride function `step_advance`. Assuming a user has developed all of their control rules in in a Python script, to reduce simulation time a user can specify how often Python controls should be evaluated.

For example, let's assume `testmodel.inp` has a 30 second routing step (using the dynamic wave solver, this step could vary significantly). If complex control scenarios are developed, evaluating rules could add significant time to the simulation.

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('testmodel.inp') as sim:
...     sim.step_advance(300)
...     for step in sim:
...             print(sim.current_time)
...             # or here! sim.step_advance(newvalue)

2015-11-01 14:05:00
2015-11-01 14:10:00
2015-11-01 14:15:00
2015-11-01 14:20:00
```

5.2 Nodes

For interacting with nodes a *pyswmm.nodes.Nodes* object must be initialized. See the following example. Once the Nodes object is initialized, you can then initialize a *pyswmm.nodes.Node*

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('./testmodel.inp') as sim:
...     node_object = Nodes(sim)
...
...     #J1 node instantiation
...     J1 = node_object["J1"]
...     print(J1.invert_elevation)
...     print(J1.is_junction())
...
...     #Step through a simulation
...     for step in sim:
...             print(J1.total_inflow)
...
```

5.3 Links

For interacting with links a *pyswmm.links.Links* object must be initialized. See the following example. Once the Links object is initialized, you can then initialize a *pyswmm.links.Link*

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('./testmodel.inp') as sim:
...     link_object = Links(sim)
...
...     #C1:C2 link instantiation
...     c1c2 = link_object["C1:C2"]
...     print(c1c2.flow_limit)
...     print(c1c2.is_conduit())
...
...     #Step through a simulation
...     for step in sim:
...             print(c1c2.flow)
...             if c1c2.flow > 10.0:
```

(continues on next page)

(continued from previous page)

```
...                     c1c2.target_setting = 0.5
...
```

5.4 Subcatchments

For interacting with subcatchments a *pyswmm.subcatchments.Subcatchments* object must be initialized. See the following example. Once the `Subcatchments` object is initialized, you can then initialize a *pyswmm.subcatchments.Subcatchment*

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('./testmodel.inp') as sim:
...     subcatch_object = Subcatchments(sim)
...
...     #SC1 subcatchment instantiation
...     SC1 = subcatch_object["S1"]
...     print(SC1.area)
...
...     #Step through a simulation
...     for step in sim:
...             print(SC1.runoff)
...
```

In the example above we introduce the option to change a link's settings.

5.5 PySWMM Controls

The pyswmm package exposes new possibility in interfacing with models. All control rules can now be removed from USEPA SWMM5 and brought into Python. Now that this functionality exists, open-source Python packages can now be used in conjunction with pyswmm to bring even more complex control routines.

The following example illustrates the use of functions for comparing two depths.

```
>>> from pyswmm import Simulation, Links, Nodes
>>>
>>> def TestDepth(node, node2):
>>>     if node > node2:
>>>             return True
>>>     else:
>>>             return False
>>>
>>> with Simulation('./testmodel.inp') as sim:
...     link_object = Links(sim)
...
...     #C1:C2 link instantiation
...     c1c2 = link_object["C1:C2"]
...
...     node_object = Nodes(sim)
...     #J1 node instantiation
...     J1 = node_object["J1"]
...     #J2 node instantiation
...     J2 = node_object["J2"]
```

(continues on next page)

(continued from previous page)

```
...
...     #Step through a simulation
...     for step in sim:
...             if TestDepth(J1.depth, J2.depth):
...                     c1c2.target_setting = 0.5
...
```

If an EPA-SWMM5 Model has existing control actions within, any control rules developed using pyswmm will have the highest priority. All pyswmm control actions are evaluated at the end of each simulation step, after EPA-SWMM native controls have been evaluated. If control actions are reported, any control action updated by pyswmm will be output to the *.rpt file.

5.6 Generate Node Inflows

Among the newest features pyswmm brings to SWMM5 modeling is the ability to set a nodes inflow. This can enable the user to model different behavior such as runoff or seasonality.

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('/testmodel.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...             j1.generated_inflow(9)
```

5.7 Lid Controls

For interacting with lid controls a *pyswmm.lidcontrols.LidControls* object must be initialized. See the following example. Once the `LidControls` object is initialized, you can then initialize a *pyswmm.lidcontrols.LidControl*. Once the `LidControl` object is initialized, you can then interact with the parameters defined in each layers within an Lid Control: `Surface`, `Soil`, `Storage`, `Pavement`, `Drain`, `DrainMat`.

The layers parameters that can be accessed using PySWMM are listed in the table below.

```
>>> from pyswmm import Simulation, LidControls
>>>
>>> with Simulation('/testmodel.inp') as sim:
...     rain_barrel = LidControls(sim)["rain_barrel"]
...     print(rain_barrel.drain.coefficient)
...     rain_barrel.drain.coefficient = 0.60
...     print(rain_barrel.drain.coefficient)
```

All LidControl parameters can be accessed before and during model simulations. All LidControl parameters can be set before model simulation. Only some LidControl parameters can be set during model simulation.

5.8 Lid Groups

For interacting with group of lids defined on a subcatchment *pyswmm.lidgroups.LidGroups* object must be initialized. See the following example. Once the `LidGroups` object is initialized, you can then initialize a *pyswmm.lidgroups.LidGroup*. Once the `LidGroup` object is initialized, you can then interact with the lid units defined on the subcatchment. You can iterate through the list of lid units using the LidGroup object.

```
>>> from pyswmm import Simulation, LidGroups
>>>
>>> with Simulation('/testmodel.inp') as sim:
...     lid_on_sub = LidGroups(sim)["subcatch_id"]
...     for lid in lid_on_sub:
...             print(lid)
...     print(lid_on_sub[0])
...     for step in sim:
...             print(lid_on_sub.old_drain_flow)
```

5.9 Lid Units

For interacting with group of lids defined on a subcatchment *pyswmm.lidgroups.LidGroups* object must be initialized. See the example above. Once the `LidGroups` object is initialized, you can then initialize a *pyswmm.lidgroups.LidGroup*. Once the `LidGroup` object is initialized, you can then interact with the lid units defined on the subcatchment. You can iterate through the list of lid units using the LidGroup object.

```
>>> from pyswmm import Simulation, LidGroups
>>>
>>> with Simulation('/testmodel.inp') as sim:
...     lid_on_sub = LidGroups(sim)["subcatch_id"]
...     for lid in lid_on_sub:
...             print(lid)
...     print(lid_on_sub[0])
...     for step in sim:
...             print(lid_on_sub.WaterBalance.inflow)
...             print(lid_on_sub.WaterBalance.evaporation)
```

All LidUnits parameters can be accessed before and during model simulations. All LidUnits parameters can be set before model simulation. Only some LidUnits parameters can be set during model simulation.

What Next

Now that you have an idea of what the PySWMM package provides, you should investigate the parts of the package most useful for you.

Reference Section provides details on PySWMM.

CHAPTER 6

Reference

6.1 Introduction

6.1.1 PySWMM Basics

After starting Python, import the pyswmm module with

```
>>> from pyswmm import Simulation
```

To save repetition, in the documentation we assume that PySWMM has been imported this way.

If importing pyswmm fails, it means that Python cannot find the installed module. Check your installation and your PYTHONPATH.

The following simulation classes are available:

Simulation This class initializes a simulation object from a SWMM input file (*.inp).

Initialize a SWMM Model with

```
>>> sim = Simulation(r"./example.inp")
```

Once a model is initialized, there are several options available to run a simulation as well as edit the simulation.

6.2 simulation module

Base class for a SWMM Simulation.

class pyswmm.simulation.**Simulation**(*inputfile*, *reportfile=None*, *outputfile=None*, *swmm_lib_path=None*)

Bases: object

Base class for a SWMM Simulation.

The model object provides several options to run a simulation. User can specified SWMM library path. Uses default lib if not provided.

Initialize the Simulation class.

Parameters

- **inpfile** (*str*) – Name of SWMM input file (default '')
- **rptfile** (*str*) – Report file to generate (default None)
- **binfile** (*str*) – Optional binary output file (default None)
- **swmm_lib_path** (*str*) – User-specified SWMM library path (default None).

Examples:

Intialize a simulation and iterate through a simulation. This approach requires some clean up.

```
>>> from pyswmm import Simulation
>>>
>>> sim = Simulation('tests/data/TestModel1_weirSetting.inp')
>>> for step in sim:
...     pass
>>>
>>> sim.report()
>>> sim.close()
```

Intialize using with statement. This automatically cleans up after a simulation

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     for step in sim:
...         pass
...     sim.report()
```

Initialize the simulation and execute. This style does not allow the user to interact with the simulation. However, this approach tends to be the fastest.

```
>>> from pyswmm import Simulation
>>>
>>> sim = Simulation('tests/data/TestModel1_weirSetting.inp')
>>> sim.execute()
```

add_after_close(*callback*)
Add callback function/method/object to execute after the simulation is closed. Needs to be callable.

Parameters **callback** (*func*) – Callable Object

(See self.add_before_start() for more details)

add_after_end(*callback*)
Add callback function/method/object to execute after the simulation ends. Needs to be callable.

Parameters **callback** (*func*) – Callable Object

(See self.add_before_start() for more details)

add_after_step(*callback*)
Add callback function/method/object to execute after a simlation step. Needs to be callable.

Parameters **callback** (*func*) – Callable Object

(See self.add_before_start() for more details)

add_before_end(*callback*)
Add callback function/method/object to execute after the simulation ends. Needs to be callable.

Parameters **callback** (*func*) – Callable Object

(See self.add_before_start() for more details)

add_before_start(*callback*)
Add callback function/method/object to execute before the simlation starts. Needs to be callable.

Parameters **callback** (*func*) – Callable Object

```
>>> from pyswmm import Simulation
>>>
>>> def test_callback():
...     print("CALLBACK - Executed")
>>>
>>> with Simulation('./TestModel1_weirSetting.inp') as sim:
...
...     sim.before_start(test_callback) #<- pass function handle.
...     print("Waiting to Start")
...     for ind, step in enumerate(sim):
...         print("Step {}".format(ind))
...     print("Complete!")
... print("Closed")
>>>
>>> "Waiting to Start"
>>> "CALLBACK - Executed"
>>> "Step 0"
>>> "Step 1"
>>> ...
>>> "Complete!"
>>> "Closed"
```

add_before_step(*callback*)
Add callback function/method/object to execute before a simlation step. Needs to be callable.

Parameters **callback** (*func*) – Callable Object

(See self.add_before_start() for more details)

after_close()
Get After Close Callback.

Returns Callbacks

after_end()
Get After End Callback.

Returns Callbacks

after_step()
Get After Step Callback.

Returns Callbacks

before_end()
Get Before End Callback.

Returns Callbacks

before_start()
Get Before Start Callback.

Returns Callbacks

before_step()
Get Before Step Callback.

Returns Callbacks

close()
Intialize a simulation and iterate through a simulation. This approach requires some clean up.

Examples:

```
>>> from pyswmm import Simulation
>>>
>>> sim = Simulation('./TestModel1_weirSetting.inp')
>>> for step in sim:
...     pass
>>>
>>> sim.report()
>>> sim.close()
```

current_time
Get Simulation Current Time.

Returns Current Simulation Time

Return type Datetime

Examples:

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     for step in sim:
...         print(sim.current_time)
...     sim.report()
>>>
>>> 2015-11-01 14:00:30
>>> 2015-11-01 14:01:00
>>> 2015-11-01 14:01:30
>>> 2015-11-01 14:02:00
```

end_time
Get/set Simulation end time.

Examples:

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     print sim.end_time
...     sim.end_time = datetime(2016,5,10,15,15,1)
>>>
>>> datetime.datetime(2016,5,10,15,15,1)
```

engine_version
Retrieves the SWMM Engine Version.

Returns Engine Version

Return type StrictVersion

Examples:

```
>>> sim = PYSWMM(r'\test.inp')
>>> sim.engine_version
StrictVersion("5.1.13")
```

execute()
Open an input file, run SWMM, then close the file.

Examples:

```
>>> sim = PYSWMM(r'\test.inp')
>>> sim.execute()
```

flow_routing_error
Retrieves the Flow Routing Mass Balance Error.

Returns Flow Routing Mass Balance Error

Return type float

Examples:

```
>>> sim = PYSWMM(r'\test.inp')
>>> sim.execute()
>>> sim.flow_routing_error
0.01
```

flow_units
Get Simulation Units (CFS, GPM, MGD, CMS, LPS, MLD).

Returns Flow Unit

Return type str

Examples:

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     print sim.flow_units
>>>
>>> CFS
```

initial_conditions(*init_conditions*)
Intial Conditions for Hydraulics and Hydrology can be set from within the api by setting a function to the initial_conditions property.

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('./TestModel1_weirSetting.inp') as sim:
...     nodeJ1 = Nodes(sim)["J1"]
...
...     def init_conditions():
...         nodeJ1.initial_depth = 4
...
...     sim.initial_conditions(init_conditions)
...
```

(continues on next page)

(continued from previous page)

```
...     for step in sim:
...         pass
...     sim.report()
```

next()
Next

percent_complete
Get Simulation Percent Complete.

Returns Current Percent Complete

Return type Datetime

Examples:

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     for step in sim:
...         print(sim.percent_complete)
...     sim.report()
>>>
>>> 0.01
>>> 0.25
>>> 0.50
>>> 0.75
```

quality_error
Retrieves the Quality Routing Mass Balance Error.

Returns Quality Routing Mass Balance Error

Return type float

Examples:

```
>>> sim = PYSWMM(r'\test.inp')
>>> sim.execute()
>>> sim.quality_error
0.01
```

report()
Writes to report file after simulation.

Examples:

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     for step in sim:
...         pass
...     sim.report()
```

report_start
Get/set Simulation report start time.

Examples:

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     print sim.report_start
...     sim.report_start = datetime(2015,5,10,15,15,1)
>>>
>>> datetime.datetime(2015,5,10,15,15,1)
```

runoff_error
Retrieves the Runoff Mass Balance Error.

Returns Runoff Mass Balance Error

Return type float

Examples:

```
>>> sim = PYSWMM(r'\test.inp')
>>> sim.execute()
>>> sim.runoff_error
0.01
```

start()
Start Simulation

start_time
Get/set Simulation start time.

Examples:

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     print sim.start_time
...     sim.start_time = datetime(2015,5,10,15,15,1)
>>>
>>> datetime.datetime(2015,5,10,15,15,1)
```

step_advance(*advance_seconds*)
Advances the model by X number of seconds instead of intervening at every routing step. This does not change the routing step for the simulation; only lets python take back control after each advance period.

Parameters **advance_seconds** (*int*) – Seconds to Advance simulation

Examples:

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     sim.step_advance(300)
...     for step in sim:
...         print(step.current_time)
...         # or here! sim.step_advance(newvalue)
...     sim.report()
>>>
>>> 2015-11-01 14:00:30
>>> 2015-11-01 14:01:00
>>> 2015-11-01 14:01:30
>>> 2015-11-01 14:02:00
```

system_units
Get system units (US, SI).

Returns System Unit

Return type str

Examples:

```
>>> from pyswmm import Simulation
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     print sim.system_units
>>>
>>> US
```

6.3 nodes module

Nodes module for the pythonic interface to SWMM5.

class pyswmm.nodes.**Nodes**(*model*)
Bases: object

Node Iterator Methods.

Parameters **model** (*object*) – Open Model Instance

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     for node in Nodes(sim):
...         print node
...         print node.nodeid
...
>>> <swmm5.Node object at 0x031B0350>
>>> J1
>>> <swmm5.Node object at 0x030693D0>
>>> J2
>>> <swmm5.Node object at 0x031B0350>
>>> J3
>>> <swmm5.Node object at 0x030693D0>
>>> J0
```

Iterating over Nodes Object

```
>>> nodes = Nodes(sim)
>>> for node in nodes:
...     print node.nodeid
>>> J0
>>> J1
>>> J2
>>> J3
```

Testing Existence

```
>>> nodes = Nodes(sim)
>>> "J1" in nodes
>>> True
```

Initializing a node Object

```
>>> nodes = Nodes(sim)
>>> j1 = nodes['J1']
>>> print(j1.invert_elevation)
>>> 12
>>>
>>> j1.invert_elevation = 200
>>> print(j1.invert_elevation)
>>> 200
```

next()

class pyswmm.nodes.**Node**(*model*, *nodeid*)

Bases: object

Node Methods.

Parameters

- **model** (*object*) – Open Model Instance
- **nodeid** (*str*) – Node ID

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.invert_el
...     for step in simulation:
...         print j1.depth
... 0.0
```

depth

Get Node Results for Depth.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.depth
>>> 0
>>> 0.5
>>> 0.51
>>> 0.52
>>> 0.49
```

flooding
Get Node Results for Flooding Rate.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.flooding
>>> 0
>>> 0
>>> 0.01
>>> 0
>>> 0
```

full_depth
Get node full depth (Physical Depth of manhole).

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.full_depth
>>> 10
```

Setting the value

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.full_depth
...     j1.full_depth = 50
...     print j1.full_depth
>>> 10
>>> 50
```

generated_inflow(*inflowrate*)
Generate and Set a Node Inflow Rate.

The value is held constant in the model until it is redefined. Generated inflows work like any SWMM inflow. This does not introduce any continuity errors since all flows is counted as an inflow.

Parameters **inflowrate** (*float*) – Inflow Rate

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         j1.generated_inflow(9)
>>>
```

head

Get Node Results for Head.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.head
>>> 10
>>> 10.5
>>> 10.51
>>> 10.52
>>> 10.49
```

initial_depth

Get/set node initial depth.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.initial_depth
>>> 0
```

Setting the value

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.initial_depth
...     j1.initial_depth = 1
...     print j1.initial_depth
>>> 0
>>> 1
```

invert_elevation
Get/set node invert elevation.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.invert_elevation
>>> 0.1
```

Setting the value

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.invert_elevation
...     j1.invert_elevation = 0.2
...     print j1.invert_elevation
>>> 0.1
>>> 0.2
```

is_divider()
Check if node is a Divider Type.

Returns is divider

Return type bool

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.is_divider()
>>> True
```

is_junction()
Check if node is a Junction Type.

Returns is junction

Return type bool

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.is_junction()
>>> True
```

is_outfall()
Check if node is a Outfall Type.

Returns is outfall

Return type bool

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.is_outfall()
>>> True
```

is_storage()

Check if node is a Storage Type.

Returns is storage

Return type bool

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.is_storage()
>>> True
```

lateral_inflow

Get Node Results for lateral Inflow rate.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.lateral_inflow
>>> 0
>>> 0.25
>>> 0.25
>>> 0.3
>>> 0.4
```

losses

Get Node Results for Losses Rate (Evap and Exfiltration).

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.losses
>>> 0
>>> 0.01
>>> 0.01
>>> 0.01
>>> 0.01
```

nodeid

Get Node ID.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.nodeid
>>> J1
```

ponding_area

Get/set node ponding area.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.ponding_area
>>> 0
```

Setting the value

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.ponding_area
...     j1.ponding_area = 50
...     print j1.ponding_area
>>> 0
>>> 50
```

statistics

Node Statistics. The stats returned are rolling/cumulative. Indeces are as follows:

average_depth
max_depth
max_depth_date
max_report_depth
flooding_volume
flooding_duration
surcharge_duration
courant_crit_duration
lateral_inflow_vol
peak_lateral_inflowrate
peak_total_inflow
peak_flooding_rate
max_ponded_volume
max_inflow_date
max_flooding_date

Returns Group of Stats

Return type dict

surcharge_depth

Get/set node surcharge depth.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.surcharge_depth
>>> 10
```

Setting the value

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     print j1.surcharge_depth
...     j1.surcharge_depth = 50
...     print j1.surcharge_depth
>>> 10
>>> 50
```

total_inflow

Get Node Results for Total Inflow Rate.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.total_inflow
>>> 0
>>> 1.2
>>> 1.5
>>> 1.9
>>> 1.2
```

total_outflow

Get Node Results for Total Outflow Rate.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.total_outflow
>>> 0
>>> 1.2
>>> 1.5
>>> 1.9
>>> 1.2
```

volume

Get Node Results for Volume.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         print j1.volume
>>> 0
>>> 1.2
>>> 1.5
>>> 1.9
>>> 1.2
```

class pyswmm.nodes.**Outfall**

Bases: *pyswmm.nodes.Node*

Outfall Object: Subclass of Node Object.

cumulative_inflow
Get Cumulative Outfall Loading.

If Simulation is not running this method will raise a warning and return 0.

Returns Cumulative Volume

Return type float

outfall_stage(*stage*)
Generate and Set an Outfall Stage (head).

The value is held constant in the model until it is redefined. Using the function overrides the mechanism within SWMM that would internerally set the outfall stage. This does not introduce any continuity errors since all flows is counted as an inflow.

Parameters **stage** (*float*) – Outfall Stage (Head)

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     j1 = Nodes(sim)["J1"]
...     for step in sim:
...         j1.outfall_stage(9)
>>>
```

outfall_statistics
Outfall Stats. The stats returned are rolling/cumulative. Indeces are as follows:

average_flowrate
peak_flowrate
pollutant_loading
total_periods

Returns Group of Stats

Return type list

class pyswmm.nodes.**Storage**
Bases: *pyswmm.nodes.Node*

Storage Object: Subclass of Node Object.

storage_statistics
Storage Stats. The stats returned are rolling/cumulative. Indeces are as follows:

initial_volume
average_volume
max_volume
peak_flowrate
evap_loss
exfil_loss
max_vol_date

Returns Group of Stats

Return type list

6.4 links module

Links module for the pythonic interface to SWMM5.

class pyswmm.links.**Links**(*model*)

Bases: object

Link Iterator Methods.

Parameters **model** (*object*) – Open Model Instance

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     for link in Links(sim):
...         print link
...         print link.linkid
...
>>> <swmm5.Link object at 0x031B0350>
>>> C1
>>> <swmm5.Link object at 0x030693D0>
>>> C2
>>> <swmm5.Link object at 0x031B0350>
>>> C3
>>> <swmm5.Link object at 0x030693D0>
>>> C0
```

Iterating or Links Object

```
>>> links = Links(sim)
>>> for link in links:
...     print link.linkid
>>> C1:C2
>>> C2
>>> C3
```

Testing Existence

```
>>> links = Links(sim)
>>> "C1:C2" in links
>>> True
```

Initializing a link Object

```
>>> links = Links(sim)
>>> c1c2 = links['C1:C2']
>>> c1c2.flow_limit = 12
>>> c1c2.flow_limit
>>> 12
```

next()

class pyswmm.links.**Link**(*model*, *linkid*)

Bases: object

Link Methods.

Parameters

- **model** (*object*) – Open Model Instance
- **linkid** (*str*) – Link ID

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.flow
...     for step in simulation:
...         print c1c2.flow
... 0.0
```

average_head_loss

Get/set Average Conduit Loss.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.average_head_loss
>>> 0
```

Setting the value

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.average_head_loss
...     c1c2.average_head_loss = 0.2
...     print c1c2.average_head_loss
>>> 0
>>> 0.2
```

connections

Get link upstream and downstream node IDs.

Returns ("UpstreamNodeID","DownstreamNodeID")

Return type tuple

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.connections
>>> ("C1","C2")
```

current_setting
Get Link current setting.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.current_setting
>>> 0
>>> 1
>>> 0
>>> 0.5
>>> 1
```

depth
Get Link Results for Depth.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.depth
>>> 0
>>> 1.2
>>> 1.5
>>> 1.9
>>> 1.2
```

ds_xsection_area
Get Link Results for Downstream X-section Flow Area.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
```

(continues on next page)

(continued from previous page)

```
...         print c1c2.ds_xsection_area
>>> 0
>>> 1.2
>>> 1.5
>>> 1.9
>>> 1.2
```

flow

Get Link Results for Flow.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.flow
>>> 0
>>> 1.2
>>> 1.5
>>> 1.9
>>> 1.2
```

flow_limit

Get/set link flow limit.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.flow_limit
>>> 0
```

Setting the Value

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.flow_limit
...     c1c2.flow_limit = 0.2
...     print c1c2.flow_limit
>>> 0
>>> 0.2
```

froude

Get Link Results for Froude.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.froude
>>> 0
>>> 1.2
>>> 1.5
>>> 1.9
>>> 1.2
```

initial_flow

Get/set Link Initial Flow.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.initial_flow
>>> 0
```

Setting the Value

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.initial_flow
...     c1c2.initial_flow = 0.2
...     print c1c2.initial_flow
>>> 0.1
>>> 0.2
```

inlet_head_loss

Get/set Inlet Head Loss.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
```

(continues on next page)

(continued from previous page)

```
...     print c1c2.inlet_head_loss
>>> 0
```

Setting the Value

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.inlet_head_loss
...     c1c2.inlet_head_loss = 0.2
...     print c1c2.inlet_head_loss
>>> 0
>>> 0.2
```

inlet_node

Get link inlet node ID.

Returns Inlet node ID

Return type str

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.inlet_node
>>> C1
```

inlet_offset

Get/set Upstream Offset Depth.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.inlet_offset
>>> 0.1
```

Setting the value

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.inlet_offset
...     c1c2.inlet_offset = 0.2
...     print c1c2.inlet_offset
>>> 0.1
>>> 0.2
```

is_conduit()
Check if link is a Conduit Type.

Returns is conduit

Return type bool

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.is_conduit()
>>> True
```

is_orifice()
Check if link is a Orifice Type.

Returns is orifie

Return type bool

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.is_orifice()
>>> False
```

is_outlet()
Check if link is a Outlet Type.

Returns is outlet

Return type bool

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.is_outlet()
>>> False
```

is_pump()
Check if link is a Pump Type.

Returns is pump

Return type bool

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
```

(continues on next page)

(continued from previous page)

```
...     print c1c2.is_pump()
>>> False
```

is_weir()
Check if link is a Weir Type.

Returns is weir

Return type bool

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.is_weir()
>>> False
```

linkid
Get Link ID.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.linkid
>>> "C1"
```

outlet_head_loss
Get/set Outlet Head Loss.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.outlet_head_loss
>>> 0
```

Setting the Value

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.outlet_head_loss
...     c1c2.outlet_head_loss = 0.2
```

(continues on next page)

(continued from previous page)

```
...     print c1c2.outlet_head_loss
>>> 0
>>> 0.2
```

outlet_node

Get link outlet node ID.

Returns Outlet node ID

Return type str

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.outlet_node
>>> C2
```

outlet_offset

Get/set Downstream Offset Depth.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.outlet_offset
>>> 0.1
```

Setting the value

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.outlet_offset
...     c1c2.outlet_offset = 0.2
...     print c1c2.outlet_offset
>>> 0.1
>>> 0.2
```

seepage_rate

Get/set Conduit Seepage Loss.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
```

(continues on next page)

(continued from previous page)

```
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.seepage_rate
>>> 0
```

Setting the Value

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     print c1c2.seepage_rate
...     c1c2.seepagerate = 0.2
...     print c1c2.seepage_rate
>>> 0
>>> 0.2
```

target_setting

Get/set Link Target Setting.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.target_setting
>>> 0
>>> 0
>>> 1
>>> 0.5
>>> 1
```

Setting the Value

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.target_setting
...         if c1c2.flow > 3:
...             c1c2.target_setting = 0.1
>>> 0
>>> 0
>>> 0.1
>>> 0.1
>>> 0.1
```

ups_xsection_area

Get Link Results for Upstream X-section Flow Area.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.ups_xsection_area
>>> 0
>>> 1.2
>>> 1.5
>>> 1.9
>>> 1.2
```

volume

Get Link Results for Volume.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Links
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     c1c2 = Links(sim)["C1:C2"]
...     for step in sim:
...         print c1c2.volume
>>> 0
>>> 1.2
>>> 1.5
>>> 1.9
>>> 1.2
```

class pyswmm.links.**Conduit**

Bases: *pyswmm.links.Link*

Conduit Object: Subclass of Link Object.

conduit_statistics

Conduit Flow Stats. The stats returned are rolling/cumulative. Indeces are as follows:

peak_flow
peak_flow_date
peak_velocity
peak_depth
time_normal_flow
time_inlet_control
time_surcharged
time_full_upstream
time_full_downstream
time_full_flow
time_capacity_limited
time_in_flow_class
time_courant_crit
flow_turns
flow_turn_sign

Time in Flow Class: (Fraction of Total Time)

0	1	2	3	4	5	6
Dry	Up Dry	Down Dry	Sub Crit	Sup Crit	Up Crit	Down Crit

Returns Group of Stats

Return type dict

class pyswmm.links.**Pump**

Bases: *pyswmm.links.Link*

Pump Object: Subclass of Link Object.

pump_statistics

Pump Stats. The stats returned are rolling/cumulative. Indeces are as follows:

percent_utilized
min_flowrate
average_flowrate
max_flowrate
total_volume
energy_consumed
off_curve_low
off_curve_high
number_startups
total_periods

Returns Group of Stats

Return type dict

6.5 lidcontrols module

class pyswmm.lidcontrols.**LidControl**(*sim*, *model*, *lidcontrolid*)

Bases: object

can_overflow
Get lid control surface layer option for immediate outflow of excess water

Returns Parameter Value

Return type char

class pyswmm.lidcontrols.**LidControls**(*model*)
Bases: object

Lid Control Iterator Methods.

Parameters **model** (*object*) – Open Model Instance

next()

6.6 lidgroups module

class pyswmm.lidgroups.**LidGroup**(*model*, *subcatchmentid*)
Bases: object

flow_to_pervious
Get lid group total flow sent to pervious area

Returns Parameter Value

Return type double

new_drain_flow
Get lid group total drain flow in current period

Returns Parameter Value

Return type double

next()

old_drain_flow
Get lid group total drain flow in pervious period

Returns Parameter Value

Return type double

pervious_area
Get lid group amount of pervious area

Returns Parameter Value

Return type double

class pyswmm.lidgroups.**LidGroups**(*model*)
Bases: object

LidGroups Iterator Methods.

Parameters **model** (*object*) – Open Model Instance

next()

class pyswmm.lidgroups.**LidUnit**(*model*, *subcatchmentid*, *lidid*)
Bases: object

Lid Unit Methods.

Parameters

- **model** (*object*) – Open Model Instance
- **subcatchmentid** (*str*) – Subcatchment ID
- **lidid** (*str*) – Lid unit ID

Lid Unit Parameters

Parameter	Getter	Setter Before Sim	Setter During Sim
subcatchment	enabled	disabled	disabled
lid_control	enabled	disabled	disabled
unit_area	enabled	enabled	disabled
full_width	enabled	enabled	disabled
initial_saturation	enabled	enabled	disabled
from_impervious	enabled	enabled	disabled
from_pervious	enabled	enabled	disabled
index	enabled	enabled	disabled
number	enabled	enabled	disabled
to_pervious	enabled	enabled	disabled
drain_subcatchment	enabled	enabled	enabled
drain_node	enabled	enabled	enabled

drain_node
Get lid drain to node index Negative if no recieving node

Returns Parameter Value

Return type int

drain_subcatchment
Get lid drain to subcatchment index Negative if no recieving subcatchment

Returns Parameter Value

Return type int

dry_time
Get lid time since last rainfall (sec)

Returns Parameter Value

Return type double

evaporation
Get lid current evaporation rate

Returns Parameter Value

Return type double

from_impervious
Get lid fraction of impervious area runoff treated

Returns Parameter Value

Return type double

from_pervious
Get lid fraction of pervious area runoff treated

Returns Parameter Value

Return type double

full_width
Get lid unit full top width.

Returns Parameter Value

Return type double

index
Get lid control index

Returns Parameter Value

Return type int

initial_saturation
Get lid initial saturation of soil and storage layers.

Returns Parameter Value

Return type double

lid_control

native_infiltration
Get lid native infilration rate limit

Returns Parameter Value

Return type double

new_drain_flow
Get lid current drain flow

Returns Parameter Value

Return type double

number
Get lid number of replicate units

Returns Parameter Value

Return type int

old_drain_flow
Get lid pervious drain flow

Returns Parameter Value

Return type double

subcatchment

to_pervious

Get lid to pervious area (1 if outflow sent to pervious area) (0 if not)

Returns Parameter Value

Return type int

unit_area
Get lid unit area.

Returns Parameter Value

Return type double

6.7 lidlayers module

class pyswmm.lidlayers.**Drain**(*model*, *lidcontrol*)
Bases: object

Layer	Parameter	Setter Before Sim	Setter During Sim
Drain	coefficient	enabled	enabled
Drain	exponent	enabled	enabled
Drain	offset	enabled	enabled
Drain	delay	enabled	enabled
Drain	open_head	enabled	enabled
Drain	close_head	enabled	enabled

close_head
Get lid control drain layer head when drain closes (ft)

Returns Parameter Value

Return type double

coefficient
Get lid control drain layer underdrain flow coefficient

Returns Parameter Value

Return type double

delay
Get lid control drain layer rain barrel drain delay time (sec)

Returns Parameter Value

Return type double

exponent
Get lid control drain layer underdrain head exponent

Returns Parameter Value

Return type double

offset
Get lid control drain layer offset height of underdrain

Returns Parameter Value

Return type double

open_head
Get lid control drain layer head when drain opens (ft)

Returns Parameter Value

Return type double

class pyswmm.lidlayers.**DrainMat**(*model*, *lidcontrol*)
Bases: object

Layer	Parameter	Setter Before Sim	Setter During Sim
DrainMat	thickness	enabled	disabled
DrainMat	void_fraction	enabled	disabled
DrainMat	roughness	enabled	enabled
DrainMat	alpha	enabled	disabled

alpha
Get lid control drainmat layer slope/roughness term in Manning equation

Returns Parameter Value

Return type double

roughness
Get lid control drainmat layer Mannings n for green roof drainage mats

Returns Parameter Value

Return type double

thickness
Get lid control drainmat layer thickness

Returns Parameter Value

Return type double

void_fraction
Get lid control drainmat layer void volume / total volume

Returns Parameter Value

Return type double

class pyswmm.lidlayers.**Pavement**(*model*, *lidcontrol*)
Bases: object

Layer	Parameter	Setter Before Sim	Setter During Sim
Pavement	thickness	enabled	disabled
Pavement	void_fraction	enabled	disabled
Pavement	impervious_fraction	enabled	disabled
Pavement	k_saturated	enabled	disabled
Pavement	clog_factor	enabled	enabled
Pavement	regeneration	enabled	disabled
Pavement	regeneration_degree	enabled	disabled

clog_factor
Get lid control pavement layer clogging factor

Returns Parameter Value

Return type double

impervious_fraction
Get lid control pavement layer impervious area fraction

Returns Parameter Value

Return type double

k_saturated
Get lid control pavement layer permeability

Returns Parameter Value

Return type double

regeneration
Get lid control pavement layer clogging regeneration interval (days)

Returns Parameter Value

Return type double

regeneration_degree
Get lid control pavement layer clogging regeneration degree

Returns Parameter Value

Return type double

thickness
Get lid control pavement layer thickness

Returns Parameter Value

Return type double

void_fraction
Get lid control pavement layer void volume / total volume

Returns Parameter Value

Return type double

class pyswmm.lidlayers.**Soil**(*model*, *lidcontrol*)
Bases: object

Layer	Parameter	Setter Before Sim	Setter During Sim
Soil	thickness	enabled	disabled
Soil	porosity	enabled	disabled
Soil	field_capacity	enabled	disabled
Soil	wilting_point	enabled	disabled
Soil	k_saturated	enabled	disabled
Soil	k_slope	enabled	disabled
Soil	suction_head	enabled	disabled

field_capacity
Get lid control soil layer field capacity

Returns Parameter Value

Return type double

k_saturated
Get lid control soil layer saturated hydraulic conductivity

Returns Parameter Value

Return type double

k_slope
Get lid control soil layer slope of log(k) v. moisture content curve

Returns Parameter Value

Return type double

porosity
Get lid control soil layer void volume / total volume

Returns Parameter Value

Return type double

suction_head
Get lid control soil layer suction head at wetting front

Returns Parameter Value

Return type double

thickness
Get lid control soil layer thickness

Returns Parameter Value

Return type double

wilting_point
Get lid control soil layer wilting point

Returns Parameter Value

Return type double

class pyswmm.lidlayers.**Storage**(*model*, *lidcontrol*)
Bases: object

Layer	Parameter	Setter Before Sim	Setter During Sim
Storage	thickness	enabled	disabled
Storage	void_fraction	enabled	disabled
Storage	k_saturated	enabled	disabled
Storage	clog_factor	enabled	enabled

clog_factor
Get lid control storage layer clogging factor

Returns Parameter Value

Return type double

k_saturated
Get lid control storage layer saturated hydraulic conductivity

Returns Parameter Value

Return type double

thickness
Get lid control storage layer thickness

Returns Parameter Value

Return type double

void_fraction
Get lid control storage layer void volume / total volume

Returns Parameter Value

Return type double

class pyswmm.lidlayers.**Surface**(*model*, *lidcontrol*)
Bases: object

Layer	Parameter	Setter Before Sim	Setter During Sim
Surface	thickness	enabled	disabled
Surface	void_fraction	enabled	disabled
Surface	roughness	enabled	enabled
Surface	slope	enabled	disabled
Surface	side_slope	enabled	disabled
Surface	alpha	enabled	disabled

alpha
Get lid control surface layer swale side slope (run/rise)

Returns Parameter Value

Return type double

roughness
Get lid control surface layer surface Mannings n

Returns Parameter Value

Return type double

side_slope
Get lid control surface layer swale side slope (run/rise)

Returns Parameter Value

Return type double

slope
Get lid control surface layer land surface slope (fraction)

Returns Parameter Value

Return type double

thickness
Get lid control surface layer thickness

Returns Parameter Value

Return type double

void_fraction
Get lid control surface layer available fraction of storage volume

Returns Parameter Value

Return type double

6.8 lidunits module

class pyswmm.lidunits.**Pavement**(*model*, *lidunit*)
Bases: object

depth
Get lid depth of water in poroous pavement layer

Returns Parameter Value

Return type double

evaporation
Get lid evaporation from pavement layer

Returns Parameter Value

Return type double

flux_rate
Get lid flux rate from pavement layer

Returns Parameter Value

Return type double

percolation
Get lid percolation from pavement layer

Returns Parameter Value

Return type double

class pyswmm.lidunits.**Soil**(*model*, *lidunit*)
Bases: object

evaporation
Get lid evaporation from soil layer

Returns Parameter Value

Return type double

flux_rate
Get lid flux rate from soil layer

Returns Parameter Value

Return type double

moisture
Get lid moisture content of biocell soil layer

Returns Parameter Value

Return type double

percolation
Get lid percolation from soil layer

Returns Parameter Value

Return type double

class pyswmm.lidunits.**Storage**(*model*, *lidunit*)
Bases: object

depth
Get lid depth of water in storage layer

Returns Parameter Value

Return type double

drain
Get lid drain rate from storage layer

Returns Parameter Value

Return type double

evaporation
Get lid evaporation rate from storage layer

Returns Parameter Value

Return type double

exfiltration
Get lid exfiltration rate from storage layer

Returns Parameter Value

Return type double

flux_rate
Get lid flux rate from storage layer

Returns Parameter Value

Return type double

inflow
Get lid inflow rate to storage rate

Returns Parameter Value

Return type double

class pyswmm.lidunits.**Surface**(*model*, *lidunit*)
Bases: object

depth
Get lid depth of ponded water on surface layer

Returns Parameter Value

Return type double

evaporation
Get lid evaporation rate from surface layer

Returns Parameter Value

Return type double

flux_rate
Get lid flux rate from surface layer

Returns Parameter Value

Return type double

infiltration
Get lid infiltration rate from surface layer

Returns Parameter Value

Return type double

inflow
Get lid precip. + runon to LID unit

Returns Parameter Value

Return type double

outflow
Get lid outflow from surface layer

Returns Parameter Value

Return type double

class pyswmm.lidunits.**WaterBalance**(*model*, *lidunit*)
Bases: object

drain_flow
Get lid water balance total underdrain flow

Returns Parameter Value

Return type double

evaporation
Get lid water balance total evaporation

Returns Parameter Value

Return type double

final_volume
Get lid water balance final stored volume

Returns Parameter Value

Return type double

infiltration
Get lid water balance total infiltration

Returns Parameter Value

Return type double

inflow
Get lid water balance total inflow

Returns Parameter Value

Return type double

initial_volume
Get lid water balance initial stored volume

Returns Parameter Value

Return type double

surface_flow
Get lid water balance total surface runoff

Returns Parameter Value

Return type double

6.9 raingages module

Raingages module for the pythonic interface to SWMM5.

class pyswmm.raingages.**RainGage**(*model*, *raingageid*)

Bases: object

Raingage Methods.

Parameters

- **model** (*object*) – Open Model Instance
- **raingageid** (*str*) – Raingage ID

Examples:

```
>>> from pyswmm import Simulation, Raingages
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     rg1 = Raingages(sim)["Gage1"]
...     print(rg1.raingageid)
...     for step in simulation:
...         print(rg1.total_precip)
... Gage1
... 0
... 10
```

rainfall

Get raingage total rainfall rate (like in/hr or mm/hr).

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, RainGages
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     rg1 = RainGages(sim)["Gage1"]
...     print(rg1.rainfall)
>>> 1.0
```

raingageid

Get Rain Gage ID.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, RainGages
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     rg = RainGage(sim)["Gage1"]
...     print(rg.raingageid)
>>> Gage1
```

snowfall

Get raingage total snowfall rate (like in/hr or mm/hr).

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, RainGages
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     rg1 = RainGages(sim)["Gage1"]
...     print(rg1.snowfall)
>>> 0.0
```

total_precip

Get/set raingage total precipitation rate (like in/hr or mm/hr).

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, RainGages
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     rg1 = RainGages(sim)["Gage1"]
...     print(rg1.total_precip)
>>> 1.0
```

Setting the value

```
>>> from pyswmm import Simulation, RainGages
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     rg1 = RainGages(sim)["Gage1"]
...     print(rg1.total_precip)
...     rg1.total_precip = 0.2
...     print(rg1.total_precip)
>>> 1.0
>>> 0.2
```

class pyswmm.raingages.**RainGages**(*model*)

Bases: object

Rain Gages Iterator Methods.

Parameters **model** (*object*) – Open Model Instance

Examples:

```
>>> from pyswmm import Simulation, Nodes
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     for raingage in RainGages(sim):
...         print(raingage)
...         print(raingage.raingageid)
...
>>> <swmm5.RainGage object at 0x031B0350>
>>> Gage1
>>> <swmm5.RainGage object at 0x030693D0>
>>> Gage4
```

(continues on next page)

(continued from previous page)

```
>>> <swmm5.RainGage object at 0x031B0350>
>>> Gage3
>>> <swmm5.RainGage object at 0x030693D0>
>>> Gage10
```

Iterating over Nodes Object

```
>>> raingages = RainGages(sim)
>>> for raingage in raingages:
...     print(raingage.raingageid)
>>> Gage1
>>> Gage4
>>> Gage3
>>> Gage10
```

Testing Existence

```
>>> raingages = RainGages(sim)
>>> "Gage1" in raingages
>>> True
```

Initializing a node Object

```
>>> raingages = RainGages(sim)
>>> gage1 = raingages['Gage1']
>>> print(gage1.total_precip)
>>> 0.04
>>>
>>> gage1.total_precip = 1
>>> print(gage1.total_precip)
>>> 1
```

next()

6.10 subcatchments module

Subcatchments module for the pythonic interface to SWMM5.

class pyswmm.subcatchments.**Subcatchments**(*model*)

Bases: object

Subcatchment Iterator Methods.

Parameters **model** (*object*) – Open Model Instance

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     for subcatchment in Subcatchments(sim):
...         print subcatchment
...         print subcatchment.subcatchmentid
...
>>> <swmm5.Subcatchment object at 0x031B0350>
>>> S1
```

(continues on next page)

(continued from previous page)

```
>>> <swmm5.Subcatchment object at 0x030693D0>
>>> S2
>>> <swmm5.Subcatchment object at 0x031B0350>
>>> S3
>>> <swmm5.Subcatchment object at 0x030693D0>
>>> S4
```

Iterating over Subcatchments Object

```
>>> subcatchments = Subcatchments(sim)
>>> for subcatchment in subcatchments:
...     print subcatchment.subcatchmentid
>>> S0
>>> S1
>>> S2
>>> S3
```

Testing Existence

```
>>> subcatchments = Subcatchments(sim)
>>> "S1" in subcatchments
>>> True
```

Initializing a subcatchment Object

```
>>> subcatchments = Subcatchments(sim)
>>> s1 = subcatchments['S1']
>>> print(s1.area)
>>> 12
>>>
>>> s1.area = 200
>>> print(s1.area)
>>> 200
```

next()

class pyswmm.subcatchments.**Subcatchment**(*model*, *subcatchmentid*)

Bases: object

Subcatchment Methods.

Parameters

- **model** (*object*) – Open Model Instance
- **subcatchmentid** (*str*) – Subcatchment ID

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.rainfall
...     for step in sim:
...         print s1.rainfall
... 0.04
```

area

Get/set subcatchment area.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.area
>>> 10
```

Setting the value

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.area
...     s1.area = 50
...     print s1.area
>>> 10
>>> 50
```

buildup

Get Pollutant Results for Surface Buildup on a Subcatchment.

If Simulation is not running this method will raise a warning and return 0.

Returns Group of Subcatchment Surface Buildup Values.

Return type dict

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/buildup-test.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print(s1.buildup)
>>> {'test-pollutant': 8.0}
>>> {'test-pollutant': 8.0}
>>> {'test-pollutant': 7.998}
>>> {'test-pollutant': 7.991}
>>> {'test-pollutant': 7.981}
```

concPonded

Get Pollutant Results for Concentration of Pollutant in Ponded Water on a Subcatchment.

If Simulation is not running this method will raise a warning and return 0.

Returns Group of Subcatchment Ponded Water Quality Values.

Return type dict

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/buildup-test.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print(s1.concPonded)
>>> {'test-pollut1': 0.0, 'test-pollut2': 0.0}
>>> {'test-pollut1': 0.0, 'test-pollut2': 0.0}
>>> {'test-pollut1': 0.0, 'test-pollut2': 0.0}
```

connection

Get Subcatchment Outlet Connection.

This function return the type of loading surface and the ID. The two load to objects are nodes and other subcatchments.

Node	2
Subcatchment	1

Returns (Loading Surface Type, ID)

Return type tuple

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.connection
>>> (2, 'J2')
```

curb_length

Get/set subcatchment curb length.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.curb_length
>>> 0
```

Setting the value

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.curb_length
...     s1.curb_length = 100
...     print s1.curb_length
```

(continues on next page)

(continued from previous page)

```
>>> 0
>>> 100
```

evaporation_loss

Get Subcatchment Results for evaporation loss.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print s1.evaporation_loss
>>> 0.01
>>> 0.01
>>> 0.01
>>> 0.01
>>> 0.01
```

infiltration_loss

Get Subcatchment Results for Infiltration Loss.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print s1.infiltration_loss
>>> 0
>>> 0.01
>>> 0.01
>>> 0.01
>>> 0.01
```

percent_impervious

Get/set subcatchment percent impervious.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
```

(continues on next page)

(continued from previous page)

```
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.percent_impervious
>>> 10
```

Setting the value

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.percent_impervious
...     s1.percent_impervious = 50
...     print s1.percent_impervious
>>> 10
>>> 50
```

rainfall

Get Subcatchment Results for Rainfall.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print s1.rainfall
>>> 0
>>> 1.2
>>> 1.5
>>> 1.9
>>> 1.2
```

runoff

Get Subcatchment Results for Run Off Rate.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print s1.runoff
>>> 0
>>> 0
```

(continues on next page)

(continued from previous page)

```
>>> 0.01
>>> 0
>>> 0
```

runon

Get Subcatchment Results for Run On.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print s1.runon
>>> 0
>>> 1.2
>>> 1.5
>>> 1.9
>>> 1.2
```

slope

Get/set subcatchment slope.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.slope
>>> 0.01
```

Setting the value

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.slope
...     s1.slope = 0.02
...     print s1.slope
>>> 0.1
>>> 0.2
```

snow_depth

Get Subcatchment Results for Snow Depth.

If Simulation is not running this method will raise a warning and return 0.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     for step in sim:
...         print s1.snow_depth
>>> 0
>>> 0.5
>>> 0.51
>>> 0.52
>>> 0.49
```

statistics

Subcatchment Flow Stats. The stats returned are rolling/cumulative. Indeces are as follows:

precipitation
runon
evaporation
infiltration
runoff
peak_runoff_rate

Returns Group of Stats

Return type dict

subcatchmentid

Get Subcatchment ID.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.subcatchmentid
>>> S1
```

width

Get/set subcatchment width.

Returns Parameter Value

Return type float

Examples:

```
>>> from pyswmm import Simulation, Subcatchments
>>>
```

(continues on next page)

(continued from previous page)

```
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.width
>>> 100.0
```

Setting the value

```
>>> from pyswmm import Simulation, Subcatchments
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     s1 = Subcatchments(sim)["S1"]
...     print s1.width
...     s1.width = 30
...     print s1.width
>>> 100
>>> 30
```

6.11 system module

System module for the pythonic interface to SWMM5.

class pyswmm.system.**SystemStats**(*model*)

Bases: object

System-Wide Flow and Runoff Routing Accumulation Volume Methods.

Parameters **model** (*object*) – Open Model Instance

Examples:

```
>>> from pyswmm import Simulation, SystemStats
>>>
>>> with Simulation('tests/data/TestModel1_weirSetting.inp') as sim:
...     system_routing = SystemStats(sim)
...
...     for step in sim:
...         print system_routing.routing_stats
...         print system_routing.runoff_stats
```

routing_stats

Get rolling/cumulative routing stats. Follow Data are returned:

dry_weather_inflow
wet_weather_inflow
groundwater_inflow
II_inflow
external_inflow
flooding
outflow
evaporation_loss
seepage_loss
reacted
initial_storage
final_storage
routing_error

Returns Statistics

Return type dict

runoff_stats
Get rolling/cumulative runoff stats. Follow Data are returned:

rainfall
evaporation
infiltration
runoff
drains
runon
init_storage
final_storage
init_snow_cover
final_snow_cover
snow_removed
routing_error

Returns Statistics

Return type dict

6.12 lib module

SWMM5 compiled libraries. This module provides the user with some options for selecting the SWMM5 engine.

lib.**use**(*arg*)
Set the SWMM5 DLL.

This method allows the user to define the engine they would like to use for the simulation. It is important to understand that previous verisons of EPA-SWMM5 do not have the expanded toolkit functionality. Therefore, only basic functionality for running a simulation is available.

To use this, the user should copy and rename their SWMM5 DLL into the `site-packages/pyswmm/lib/windows` directory. The example below outlines the steps. This should be done before Simulation is imported.

Examples:

```
>>> import pyswmm
>>> pyswmm.lib.use("swmm5")
>>>
>>> from pyswmm import Simulation
```

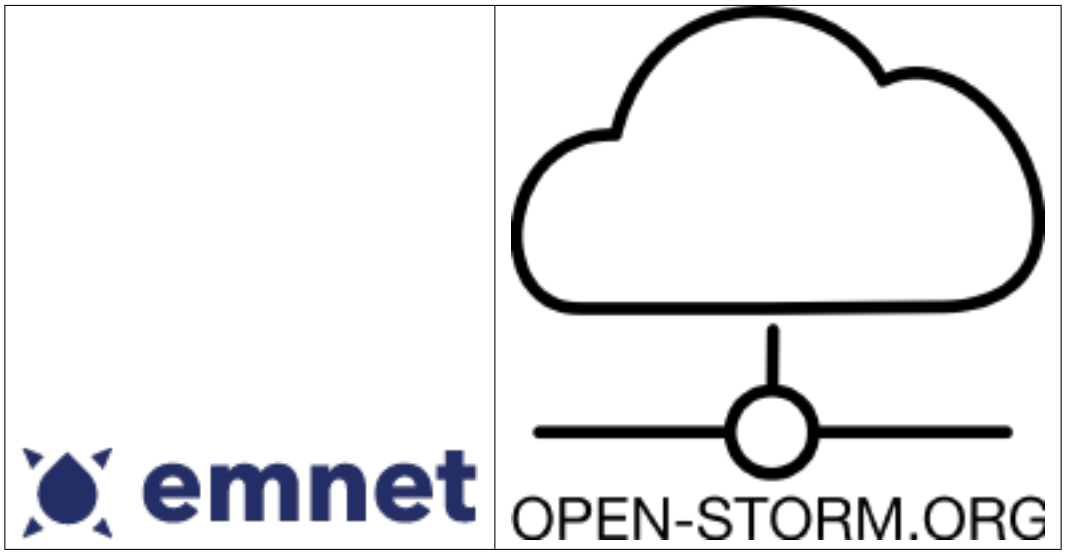
6.13 License

PySWMM is distributed with the BSD-2 license.

```
Copyright (C) 2014-, See AUTHORS
Bryant E. McDonnell <bemcdonnell@gmail.com>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

  * Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.

  * Redistributions in binary form must reproduce the above
    copyright notice, this list of conditions and the following
    disclaimer in the documentation and/or other materials provided
    with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

CHAPTER 7

Powered By



CHAPTER 8

Indices and tables

- genindex
- modindex
- search
- Glossary

Python Module Index

l

p

Index

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W