HydPy Anwenderdokumentation

# Einführung

[https://hydpy-dev.github.io/hydpy/index.html#](https://hydpy-dev.github.io/hydpy/index.html)

<https://hydpy-dev.github.io/hydpy/examples.html#hydpy.examples.prepare_full_example_1>

<https://hydpy-dev.github.io/hydpy/examples.html#hydpy.examples.prepare_full_example_2>

# Installation

1. Installation Python (nicht die aus dem Softwarecenter) sondern

[\\mt1.fs.bafg.de\Software\Allgemein\SW-Entwicklung\Python\Python\_3.7.2\_64bit](file:///\\mt1.fs.bafg.de\Software\Allgemein\SW-Entwicklung\Python\Python_3.7.2_64bit)

Python Launcher, add to Path und for all users aktivieren

1. In der Windows Umgebungsvariable PYTHONPATH alle Referenzen auf andere Python Versionen entfernen
2. Installation Visual C++ Compiler: z.B. unter

[\\mt1.fs.bafg.de\Software\Allgemein\SW-Entwicklung\Compiler\Visual C++ Build Tools 2015](file:///\\mt1.fs.bafg.de\Software\Allgemein\SW-Entwicklung\Compiler\Visual%20C++%20Build%20Tools%202015)

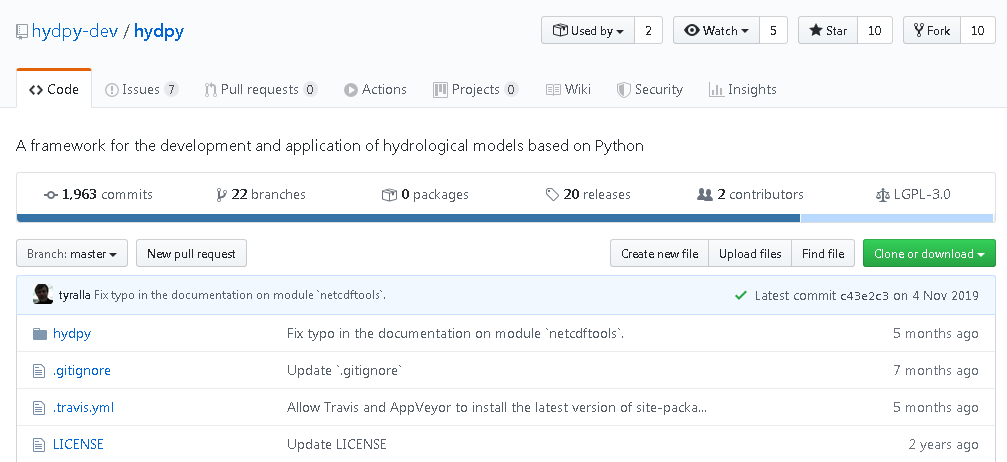
1. Weitere Bibliotheken installieren, bzw. upgraden !In windows Konsole eingeben, nicht in Python prompt!

pip install --upgrade bokeh  
pip install --upgrade coverage  
pip install --upgrade Cython  
pip install --upgrade matplotlib  
pip install --upgrade netcdf4  
pip install --upgrade numpy  
pip install --upgrade pandas  
pip install --upgrade setuptools  
pip install --upgrade typing\_extensions  
pip install --upgrade scipy  
pip install --upgrade wheel  
pip install --upgrade wrapt  
pip install --upgrade xmlschema  
pip install --upgrade NLopt  
pip install --upgrade networkx  
pip install --upgrade sphinx  
pip install --upgrade sphinxcontrib-fulltoc

1. Quell-Code HydPy downloaden a), für Entwickler HydPy forken b):
2. GIT repository clonen

cd d:\workspace  
git config --global http.sslVerify false  
git clone <https://github.com/hydpy-dev/hydpy.git>

1. - GIT repository auf github forken (GitHub account erforderlich)

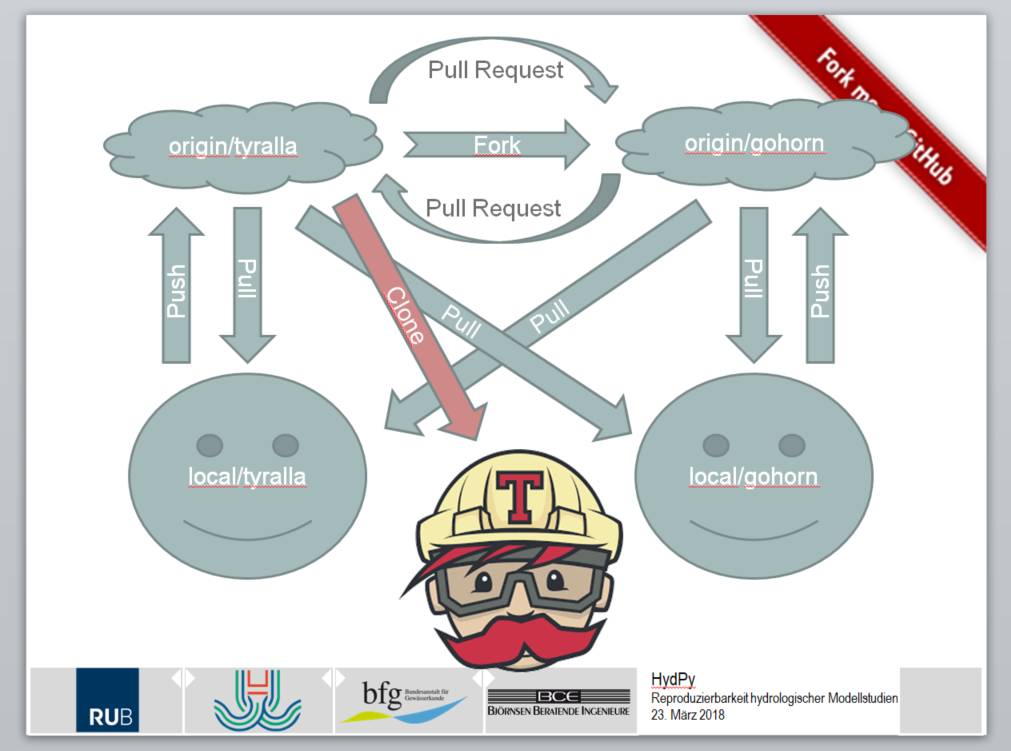


Einloggen GitHub und Fork drücken

* Persönliche HydPy-Version clonen

cd d:\workspace  
git config --global http.sslVerify false  
git clone [https://github.com/<username>/hydpy.git](https://github.com/%3cusername%3e/hydpy.git)

Prinzip der gemeinsamen Entwicklung am Framework:



Installation der hydPy-Version siehe 6)

1. **Installation: Für Entwickler: um sicher zu gehen ob mit der Installation von Python alles geklappt hat und alle Komponenten aktiv sind, von der source b) installieren. Für Andwender pip a) nutzen.**  
   a) Installation mit pip

Installation HydPy: !In windows Konsole eingeben, nicht in Python prompt!

pip install hydpy

Spezielle Version von HydPy installieren (z.B. prerelease)

<https://pypi.org/project/HydPy/#history>

pip install mit versions nummer

pip install HydPy==4.0a11

b) Installation from source !Visual C++ Compiler muss installiert sein (siehe Punkt 2)!

Eine HydPy release herunterladen und entpacken:

<https://github.com/hydpy-dev/hydpy/releases>

Windows cmd Konsole: in das Verzeichnis der release wechseln und ausführen:

cd d:\workspace\hydpy  
python setup.py install

Wenn beim Installieren Probleme auftreten:

1. unable to find vcvarsall.bat <https://hydpy-dev.github.io/hydpy/install.html>
2. andere Fehler: alle alten HydPy Versionen (Ordner mit HydPy im Namen) in <Python-Installation>\Lib\site-packages (z.B. C:\Python36\Lib\site-packages) löschen

danach erneut versuchen zu installieren

Wenn immer noch Probleme auftreten -> Hiiiilllllllfffffffeeeeeee!!!!

1. Test: python öffnen und hydpy importieren

im Windows cmd

python

Im python prompt hydpy importieren

>>import hydpy

1. Entwicklungsumgebung installieren, z.B. PyCharm [\\mt1.fs.bafg.de\Software\Allgemein\SW-Entwicklung\PyCharm](file:///\\mt1.fs.bafg.de\Software\Allgemein\SW-Entwicklung\PyCharm)

am besten im Prompt der Entwicklungsumgebung auch import hydpy testen

1. los gehts...

# Erste Schritte

Installation des mitgelieferten HydPy-Modells im iotesting Verzeichnis und Ausgabe der generierten Ordnerstruktur:

>>> from hydpy.examples import prepare\_full\_example\_1

>>> # functions installs the model iotesting folder of HydPy

>>> # z.B. C:\Python36\lib\site-packages\hydpy\tests\iotesting

>>> prepare\_full\_example\_1()

>>> from hydpy import TestIO

>>> import os

>>> # TestIO \_\_enter\_\_() changes working directory to the iotesting directory

>>> # TestIO \_\_exit\_\_() changes the working directory back to the current working directory

>>> # print installed directories

>>> with TestIO():

>>> print('working dir:', os.getcwd())

>>> print('root:', \*sorted(os.listdir('.')))

>>> for folder in ('control', 'conditions', 'series'):

>>> print(f'LahnH/{folder}:',

>>> \*sorted(os.listdir(f'LahnH/{folder}')))

root: C:\Python36\lib\site-packages\hydpy\tests\iotesting

root: LahnH \_\_init\_\_.py

LahnH/control: default

LahnH/conditions: init\_1996\_01\_01\_00\_00\_00

LahnH/series: input node output temp

Wechsel des working directories in das Verzeichnis iotesting um with TestIO(): zu vermeiden, alternativ das LahnH example project im subpackage data in ein working directory der Wahl:

>>> # change working directory to iotesting

>>> from hydpy.tests import iotesting

>>> os.chdir(os.path.join(hydpy.tests.iotesting.\_\_path\_\_[0]))

Erstelle HydPy-Instanz HydPy('LahnH') des HydPy-Modells, das Modell-Verzeichnis muss sich im Working Directory befinden

>>> from hydpy import HydPy

>>> hp = HydPy('LahnH')

Die HydPy Instanz weiß bisher noch nichts über die Struktur, sondern nur den Projektnamen.

Einladen Netzwerk-Struktur

>>> hp.prepare\_network()

Knoten als Nodes Object in hp.nodes

>>> hp.nodes

Nodes("dill", "lahn\_1", "lahn\_2", "lahn\_3")

>>> type(hp.nodes)

hydpy.core.devicetools.Nodes

Zugriff auf Node Objekte in hp.nodes hp.nodes.<name>

Eingänge in das Node-Objekt .entries, Ausgänge, .exits

>>> hp.nodes.dill.entries

Elements("land\_dill")

>>> hp.nodes.dill.exits

Elements("stream\_dill\_lahn\_2")

Zugriff auf die Zeitreihen auf dem Knoten .sequences

>>> hp.nodes.dill.sequences

sim(0.0)

obs(0.0)

>>> type(hp.nodes.dill.sequences)

hydpy.core.sequencetools.NodeSequences

>>> hp.nodes.dill.sequences.sim

sim(0.0)

Elemente mit Platzhaltern definiert

>>> hp.elements

Elements("land\_dill", "land\_lahn\_1", ...,"stream\_lahn\_1\_lahn\_2",

"stream\_lahn\_2\_lahn\_3")

>>> hp.elements.stream\_dill\_lahn\_2

Element("stream\_dill\_lahn\_2",

inlets="dill",

outlets="lahn\_2",

keywords="river")

Durch die Methode prepare\_models() werden Informationen aus den Kontroll-Dateien gelesen und die Model-Objekte erstellt. Vorher muss der Simulationszeitraum und Zeitschrittweite festgelegt werden, z.B. für Zeitschrittabhängige Variablen.

Diese wird im Attribute timegrids im Module pub (public) festgelegt, in dem globale Konfigurationen definiert werden. Beim Starten von HydPy wird automatische ein object pub der Klasse Pub erstellt, mit dem auf die attribute zugegriffen wird

>>> from hydpy import pub

>>> pub.timegrids = '1996-01-01', '1996-01-05', '1d'

>>> pub.timegrids

Timegrids(Timegrid('1996-01-01 00:00:00',

'1996-01-05 00:00:00',

'1d'))

>>> pub.timegrids.sim[0]

Date('1996-01-01 00:00:00')

>>> pub.timegrids.sim[1]

Date('1996-01-02 00:00:00')

Nach der Definition können die Modelle erstellt werden

>>> hp.prepare\_models()

und auf die einzelnen Modelle zugegriffen werden

>>> model = hp.elements.land\_dill.model

>>> type(model)

<class 'hydpy.models.hland\_v1.Model'>

>>> model.name

'hland\_v1'

auf Parameter der Modelle zugreifen:

>>> model.parameters.control.icmax

icmax(field=1.0, forest=1.5)

>>> model.parameters.derived.rellandarea

rellandarea(1.0)

Inputs, states und fluxes darstellen

>>> model.sequences.inputs.t

t(nan)

>>> model.sequences.states.sm

sm(nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan)

>>> model.sequences.fluxes.qt

qt(nan)

Anfangszustände laden, hydPy nimmt automatisch die Zustände die zum Startzeitpunkt und Zeitschrittweite passen

>>> hp.load\_conditions()

Zustände zeigen

>>> model.sequences.states.sm

sm(185.13164, 181.18755, 199.80432, 196.55888, 212.04018, 209.48859,

222.12115, 220.12671, 230.30756, 228.70779, 236.91943, 235.64427)

Bevor Zeitreihen eingeladen werden können, muss erst der Speicherplatz reserviert werden, die gesamte Zeitreihe anzeigen mit .series

>>> hp.prepare\_allseries()

>>> model.sequences.states.sm.series

InfoArray([[ nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,

nan, nan],

[ nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,

nan, nan],

[ nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,

nan, nan],

[ nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,

nan, nan]])

Input-Zeitreihen einladen

>>> hp.load\_inputseries()

>>> model.sequences.inputs.t.series

InfoArray([-0.29884643, -0.81153886, -2.49384849, -5.96884868])

Simulieren

>>> hp.simulate()

# only first hru

>>> model.sequences.states.sm.series[:,0]

InfoArray([ 182.86802746, 182.54940643, 182.33452483, 182.04960737])

>>> model.sequences.fluxes.qt

qt(0.44312548095870447)

Endzustand als initial states speichern

>>> hp.save\_conditions()

Alle Ergebnisse speichern, da HydPy als default keine Zeitreihen überschreibt -> pub.sequencemanager.generaloverwrite = True setzen. !Achtung! auch die Input-Zeitreihen werden mti den Werten der selektierten Zeitperiode überschrieben.

pub.sequencemanager.generaloverwrite = True

hp.save\_allseries()

Zum Vorbereiten des Modells für die Simulation gibt es die convinience function prepare\_everything()

hp.prepare\_everything()

das die folgende Schritte ausführt:

self.prepare\_network()

self.prepare\_models()

self.load\_conditions()

with hydpy.pub.options.warnmissingobsfile(False):

self.prepare\_nodeseries()

self.prepare\_modelseries()

self.load\_inputseries()

# Framework-Beschreibung

## Klasse Pup

Klasse Pub ist im Module pubtools definiert: hydpy.core.pubtools.Pub

<https://hydpy-dev.github.io/hydpy/pubtools.html>

Dieses Modul steuert globale Projekeinstellungen, beim laden von HydPy wird automatisch ein pup Objeht der Klasse Pup erzeugt, in dem die Einstellungen geändert werden können. Um pup zu importieren:

from hydpy import pub

### Attribute

|  |  |
| --- | --- |
| **Attribute** | **Beschreibung** |
| .projectname | Projekt Name |
| .indexer | Zeitbasierte Indizes, die auf timegrids angewendet werden (z.B. Tag im Jahr)  Modul indextools hydpy.core.indextools.Indexer  <https://hydpy-dev.github.io/hydpy/indextools.html> |
| .sequencemanager | Steuerung Input/Output tasks |
| .networkmanager | Manager of network files  Module filetools hydpy.core.filetools.NetworkManager  <https://hydpy-dev.github.io/hydpy/filetools.html#hydpy.core.filetools.NetworkManager> |
| .controlmanager | Manager of control files  Module filetools hydpy.core.filetools.ControlManager  <https://hydpy-dev.github.io/hydpy/filetools.html#hydpy.core.filetools.ControlManager> |
| .conditionmanager | Manager of condition files  Module filetools hydpy.core.filetools.ConditionManager  [https://hydpy-dev.github.io/hydpy/filetools.html#hydpy.core.filetools.ConditionManager](https://hydpy-dev.github.io/hydpy/filetools.html#hydpy.core.filetools.ControlManager) |
| .selections | Tools um subsets von Node und Element Objekten zu erstellen |
| .timegrids | hydpy.core.timetools.Timegrids Objekt für die Definition Initialisierungs- / Simulationszeitraum  <https://hydpy-dev.github.io/hydpy/timetools.html#hydpy.core.timetools.Timegrids> |
| options | Definition globale Optionen  <https://hydpy-dev.github.io/hydpy/optiontools.html#hydpy.core.optiontools.Options> |

### SequenceManager

Klasse SequenceManager ist im Module filetools definiert: hydpy.core.filetools.SequenceManager

<https://hydpy-dev.github.io/hydpy/filetools.html?highlight=sequencemanager#hydpy.core.filetools.SequenceManager>

Steuerung Input/Output tasks

Im Allgemeinen existiert nur ein SequenceManager für ein HydPy-Project, der in pub gespeichert ist.

Attribute \*filetypes Dateitypen, \*overwrite, \*dirpath, \*aggregation der input, flux, state, node, temp Zeitreihen

#### Methoden

|  |  |
| --- | --- |
| **Methoden** | **Beschreibung** |
| load\_file() | Lade Datei |
| save\_file() | Speicher Datei |
| open\_netcdfreader() | neues netcdf Interface um Zeitreihen zu lesen |
| close\_netcdfreader() | lese Zeitreihen und schließe netcdf Reader |
| open\_netcdfwriter | neues netcdf Interface um Zeitreihen zu schreiben |
| close\_netcdfwriter() | schreibe Zeitreihe und schließe netcdf writer |

#### Beispiele

from hydpy import pub

# Ueberschreibe existierende Ergebnis-Dateien

pub.sequencemanager.generaloverwrite = True

pub.sequencemanager.generalfiletype = 'asc'

### NetworkManager

Klasse NetworkManager ist im Modul filetools definiert: hydpy.core.filetools.NetworkManager

<https://hydpy-dev.github.io/hydpy/filetools.html#hydpy.core.filetools.NetworkManager>

Manager of network files

### Selections

Klasse Selections ist im Module selectiontools definiert: hydpy.core.selectiontools.Selections

[https://hydpy-dev.github.io/hydpy/selectiontools.html#hydpy.core.selectiontools.Selection](https://hydpy-dev.github.io/hydpy/selectiontools.html#hydpy.core.selectiontools.Selections)s

Collection class for Selection objects.

#### Attribute

|  |  |
| --- | --- |
| **Attribute** | **Beschreibung** |
| names | A tuple containing the names of the actual Selection objects. |
| nodes | A set containing the Node objects of all handled Selection objects. |
| elements | A set containing the Element objects of all handled Selection objects. |

Auf die Selection-Elemente kann über .name zugegriffen werden, neue Selectionen zugewiesen werden und mit del Selektionen gelöscht werden.

Membership-Test **in**

Anzahl Selection **len**

Iterator verfügbar

“+”, “-“, “+=”, “-=, “==” Operatoren um Selections-Objekte zu ändern und zu vergleichen

### Selection

Klasse Selection ist im Module selectiontools definiert: hydpy.core.selectiontools.Selection

[https://hydpy-dev.github.io/hydpy/selectiontools.html#hydpy.core.selectiontools.Selection](https://hydpy-dev.github.io/hydpy/selectiontools.html#hydpy.core.selectiontools.Selections)

Tools um subsets von Node und Element Objekten zu erstellen

### Options

Klasse Options ist im Module optiontools definiert: hydpy.core.optiontools.Options

### <https://hydpy-dev.github.io/hydpy/optiontools.html#hydpy.core.optiontools.Options>

Definition Globale Optionen

#### Attribute

|  |  |
| --- | --- |
| **Attribute** | **Beschreibung** |
| autocompile | A True/False flag for enabling/disabling the automatic conversion of pure Python models to computationally more efficient Cython models whenever a existing Cython model may be outdated. Default 1 |
| checkseries | True/False flag for raising an error when trying to load an input time series not spanning the whole initialisation period or containing nan values.. Default 1 |
| dirverbose | A True/False flag for letting the autocompletion textbox include all members of an object or only the most relevant ones. So far, this option affects the behaviour of a few implemented classes only. Default 0 |
| ellipsis | Ellipsis points are used to shorten the string representations of iterable HydPy objects containing many entries. Set a value to define the maximum number of entries before and behind ellipsis points. Set it to zero to avoid any ellipsis points. Set it to -999 to rely on the default values of the respective iterable objects. Default -999 |
| flattennetcdf | A True/False flag relevant when working with NetCDF files that decides whether to handle multidimensional time series as a larger number of 1-dimensional time series (True) or to keep the original shape (False) (see the documentation on module netcdftools for further information). Default 0. |
| forcecompiling | A True/False flag for enabling that each cythonizable model is cythonized when imported. Default 0 |
| isolatenetcdf | A True/False flag relevant when working with NetCDF files that decides whether to handle only the time series of a single sequence type (True) or the time series of multiple sequence types (False) in individual NetCDF files (see the documentation on module netcdftools for further information). Default 0 |
| printprogress | A True/False flag for printing information about the progress of some processes to the standard output. Default 1. |
| printincolor | A True/False flag for printing progress information in colour eventually. Default 1 |
| reprcomments | A True/False flag for including comments into string representations. So far, this option affects the behaviour of a few implemented classes, only. Default 0 |
| reprdigits | Required precision of string representations of floating point numbers, defined as the minimum number of digits to be reproduced by the string representation (see function repr\_). Default -1 |
| skipdoctests | A True/False flag for skipping the automatic execution of documentation tests. Default 0 |
| timeaxisnetcdf | An integer value relevant when working with NetCDF files that determines the axis of the time variable (see the documentation on module netcdftools for further information). Default 1 |
| trimvariables | A True/False flag for enabling/disabling function trim(). Set it to False only for good reasons. Default 1 |
| usecython | TA True/False flag for applying cythonized models if possible, which are much faster than pure Python models. Default 1 |
| usedefaultvalues | A True/False flag for initialising parameters with standard values. Default 0 |
| utcoffset | Offset of your local time from UTC in minutes. Defaults to 60, which corresponds to UTC+01:00. |
| warnmissingcontrolfile | A True/False flag for only raising a warning instead of an exception when a necessary control file is missing. Default 0 |
| warnmissingobsfile | A True/False flag for raising a warning when a requested observation sequence demanded by a node instance is missing. Default 1 |
| warnmissingsimfile | A True/False flag for raising a warning when a requested simulation sequence demanded by a node instance is missing. Default 1 |
| warnsimulationstep | A True/False flag for raising a warning when function simulationstep() called for the first time directly by the user. Default 1 |
| warntrim | A True/False flag for raising a warning when a Variable object trims its value(s) in order no not violate certain boundaries. To cope with the limited precision of floating point numbers only those violations beyond a small tolerance value are reported (see function trim()). Default 1. |

### Beispiele

from hydpy import pub

pub.timegrids = '1996-01-01', '1996-01-05', '1d'

pub.sequencemanager.generaloverwrite = True

pub.sequencemanager.generalfiletype = 'asc'

pub.options.checkseries(False)

# Neues Verzeichnis für Kontrolldateien

pub.controlmanager.currentdir = 'newdir'

## Klasse HydPy

Klasse HydPy ist im Module hydpytools definiert: hydpy.core.hydpytools.HydPy

<https://hydpy-dev.github.io/hydpy/hydpytools.html>

Properties:

nodes(self) -> devicetools.Nodes: Knoten des HydPy-Modells

def elements(self) -> devicetools.Elements:: Elemente des HydPy-Modells

### Methoden

|  |  |
| --- | --- |
| **Befehl** | **Beschreibung** |
| .prepare\_everything | initialisiere ein lauffähiges HydPy-Modell |
| .prepare\_network() | liest Netzwerk-Struktur des HydPy-Modells aus den Kontroll Dateien |
| .prepare\_models() | liest Kontroll-Dateien der Element Objekte, initialisiert die Elemente und liest Parameter |
| .save\_controls() | Schreibe Kontrolldateien in das Verzeichnis pub.controlmanager.currentdir |
| . load\_conditions() | Lade alle relevanten Initial Conditions |
| .save\_conditions() | Schreibt alle aktuellen Initial Conditions pub.conditionmanager.currentdir |
| .trim\_conditions() | Prüfe initial conditions auf boundary violations |
| .reset\_conditions() | Anfangsbedingungen auf die Werte bei Simulationsstart zurück setzen |
| .print\_networkproperties() | Stelle Netzwerkeigenschaften dar |
| .open\_files() | Öffne alle internen Zeitreihen Dateien |
| close\_files() | Schließe alle internen Zeitreihen Dateien |
| update\_devices() | ??? |
| simulate() | Starte Simulation |
| prepare\_allseries(), prepare\_\*series() | Speicherplatz für Zeitreihen simulieren, im RAM prepare\_allseries(True), oder auf Festplatte prepare\_allseries(False) |
| save\_allseries(), save\_\*series() | SpeicherZeitreihen |
| load\_allseries(), load\_\*series() | Lade Zeitreihen |

## Klasse Nodes

Klasse hydpy.core.devicetools.Nodes, Modul devicetools

<https://hydpy-dev.github.io/hydpy/devicetools.html#hydpy.core.devicetools.Nodes>

subclass von hydpy.core.devicetools.Devices

<https://hydpy-dev.github.io/hydpy/devicetools.html#hydpy.core.devicetools.Devices>

Container für Node Objekte, das Objekt ist iterable (d.h. z.B. in for Schleifen verwendbar) übliche Methoden für Container sind definiert: len(), in, del

### Properties/Attribute

|  |  |
| --- | --- |
| **Befehl** | **Beschreibung** |
| names <Devices> | A sorted tuple of the names of the handled devices. |
| devices <Devices> | A tuple of the handled devices sorted by the device names. |
| keywords <Devices> | A set of all keywords of all handled devices. |
| forceiterable <Devices> | true, es wird beim selektieren immern ein nodes Objekt zurück gegeben! false (default) wenn ein einzelner Knoten selektiert wird, wird dieser als Node zurückgegeben. |
| variables | Return a sorted list of the variables of all handled Node objects. |

### Methoden

|  |  |
| --- | --- |
| **Befehl** | **Beschreibung** |
| add\_device <Devices> | Add the given Node or Element object to the actual Nodes or Elements object. |
| remove\_device <Devices> | Remove the given Node or Element object from the actual Nodes or Elements object. |
| search\_keywords<Devices> | Search for all devices handling at least one of the given keywords and return them. |
| open\_files <Devices> | Call method open\_files() or open\_files() of all contained Node or Element objects. |
| close\_files <Devices> | Call method close\_files() or close\_files() of all contained Node or Element objects. |
| copy <Devices> | Return a shallow copy of the actual Nodes or Elements object. |
| Node.extract\_new() <Device> | Gather all “new” Node or Element objects. |
| Node.query\_all() <Device> | Get all Node or Element objects initialised so far |
| Node.clear\_all() <Device> | Clear the registry from all initialised Node or Element objects. |
| .prepare\_allseries() | Call methods prepare\_simseries() and prepare\_obsseries(). |
| . prepare\_simseries() | Call method prepare\_simseries() of all handled Node objects. |
| . prepare\_obsseries() | Call method prepare\_obsseries() of all handled Node objects. |
| .load\_allseries() | Call methods load\_simseries() and load\_obsseries(). |
| .load\_simseries() | Call method load\_ext() of all Sim objects with an activated memoryflag. |
| .load\_obsseries() | Call method load\_ext() of all Obs objects with an activated memoryflag. |
| .save\_allseries() | Call methods save\_simseries() and save\_obsseries(). |
| .save\_simseries() | Call method save\_ext() of all Sim objects with an activated memoryflag. |
| .save\_obsseries() | Call method save\_ext() of all Obs objects with an activated memoryflag. |

### Beispiele

>>> from hydpy import Node, Nodes

>>> # erstelle Nodes Objekt

>>> nodes = Nodes('na',

... Node('nb', variable='W'),

... Node('nc', keywords=('group\_a', 'group\_1')),

... Node('nd', keywords=('group\_a', 'group\_2')),

... Node('ne', keywords=('group\_b', 'group\_1')))

# Zugriff über Namen oder keywords

>>> nodes.na

Node("na", variable="Q")

>>> nodes.group\_2

Node("nd", variable="Q",

keywords=["group\_2", "group\_a"])

>>> nodes.group\_1

Nodes("nc", "ne")

>>> # Zugriff auf nich existierenden Knoten wirft Fehler

>>> nodes.wrong

Traceback (most recent call last):

File "<input>", line 1, in <module>

File "C:\Python36\lib\site-packages\hydpy\core\devicetools.py", line 694, in \_\_getattr\_\_

f'The selected {objecttools.classname(self)} object has '

AttributeError: The selected Nodes object has neither a `wrong` ...

>>> # iterable object erzwingen

>>> Nodes.forceiterable = True

>>> nodes.wrong

Nodes()

>>> # Membership test

>>> 'na' in nodes

True

>>> # Knoten löschen

>>> del nodes.a

>>> # Anzahl Elemente

>>> len(nodes)

4

>>> # for Schleife über Knotenelemente

>>> for node in nodes:

... print(node.name, end=',')

nb,nc,nd,ne,

>>> # variablen

>>> nodes.variables

['Q', 'W']

## Klasse Node

Klasse hydpy.core.devicetools.Node, Modul devicetools   
<https://hydpy-dev.github.io/hydpy/devicetools.html#hydpy.core.devicetools.Node>

Subclass von hydpy.core.devicetools.Device   
<https://hydpy-dev.github.io/hydpy/devicetools.html#hydpy.core.devicetools.Device>

### Properties/Attribute

|  |  |
| --- | --- |
| **Properties/Attribute** | **Beschreibung** |
| name <Device> | Name of the actual Node or Element object. |
| .keywords <Device> | Keywords describing the actual Node or Element object. |
| .sequences | Node objects haben immer zwei sequences sim und obs |
| .entries | Group of Element objects which set the the simulated value of the Node object. |
| .exits | Group of Element objects which query the simulated or observed value of the actual Node object. |
| .variable | The variable handled by the actual Node object. |
| .deploymode | Defines the kind of information a Node object offers. |

### Methoden:

|  |  |
| --- | --- |
| Methode | Beschreibung |
| Node.extract\_new() <Device> | Gather all “new” Node or Element objects. |
| Node.query\_all() <Device> | Get all Node or Element objects initialised so far |
| Node.clear\_all() <Device> | Clear the registry from all initialised Node or Element objects. |
| .get\_double() | Return the Double object appropriate for the given Element input or output group and the actual deploymode. |
| .reset() | Reset the actual value of the simulation sequence to zero. |
| .open\_files() | Call method open\_files() of the Sequences object handled by the actual Node object. |
| .close\_files() | Call method close\_files() of the Sequences object handled by the actual Node object. |
| .prepare\_allseries() | Prepare the series object both of the Sim and the Obs sequence. |
| .prepare\_simseries() | Prepare the series object of the Sim sequence. |
| .prepare\_obsseries() | Prepare the series object of the Obs sequence. |
| .plot\_allseries() | Plot the series data of both the Sim and the Obs sequence object. |
| .plot\_simseries() | Plot the series of the Sim sequence object. |
| .plot\_obsseries() | Plot the series of the Obs sequence object. |

## Klasse Elements

Klasse hydpy.core.devicetools.Elements, Modul devicetools

<https://hydpy-dev.github.io/hydpy/devicetools.html#hydpy.core.devicetools.Elements>

subclass von hydpy.core.devicetools.Devices

<https://hydpy-dev.github.io/hydpy/devicetools.html#hydpy.core.devicetools.Devices>

Container für Element Objekte, das Objekt ist iterable (d.h. z.B. in for Schleifen verwendbar) übliche Methoden für Container sind definiert: len(), in, del

### Properties/Attribute

|  |  |
| --- | --- |
| **Befehl** | **Beschreibung** |
| names <Devices> | A sorted tuple of the names of the handled devices. |
| devices <Devices> | A tuple of the handled devices sorted by the device names. |
| keywords <Devices> | A set of all keywords of all handled devices. |
| forceiterable <Devices> | true, es wird beim selektieren immern ein nodes Objekt zurück gegeben! false (default) wenn ein einzelner Knoten selektiert wird, wird dieser als Node zurückgegeben. |
| conditions | A nested dictionary containing the values of all ConditionSequence objects of all currently handled models. |

### Methoden

|  |  |
| --- | --- |
| **Befehl** | **Beschreibung** |
| add\_device <Devices> | Add the given Node or Element object to the actual Nodes or Elements object. |
| remove\_device <Devices> | Remove the given Node or Element object from the actual Nodes or Elements object. |
| search\_keywords<Devices> | Search for all devices handling at least one of the given keywords and return them. |
| open\_files <Devices> | Call method open\_files() or open\_files() of all contained Node or Element objects. |
| close\_files <Devices> | Call method close\_files() or close\_files() of all contained Node or Element objects. |
| copy <Devices> | Return a shallow copy of the actual Nodes or Elements object. |
| Element.extract\_new() <Device> | Gather all “new” Node or Element objects. |
| Element.query\_all() <Device> | Get all Node or Element objects initialised so far |
| Element.clear\_all() <Device> | Clear the registry from all initialised Node or Element objects. |
| prepare\_models() | Call method prepare\_model() of all handle Element objects. |
| save\_controls() | Save the control parameters of the Model object handled by each Element object and eventually the ones handled by the given Auxfiler object. |
| load\_conditions() | Save the initial conditions of the Model object handled by each Element object. |
| save\_conditions() | Save the calculated conditions of the Model object handled by each Element object. |
| trim\_conditions() | Call method trim\_conditions() of the Sequences object handled (indirectly) by each Element object. |
| reset\_conditions() | Call method reset() of the Sequences object handled (indirectly) by each Element object. |
| prepare\_allseries() | Call method prepare\_allseries() of all handled Element objects. |
| prepare\_inputseries | Call method prepare\_inputseries() of all handled Element objects. |
| prepare\_fluxseries | Call method prepare\_fluxseries() of all handled Element objects. |
| prepare\_stateseries | Call method prepare\_stateseries() of all handled Element objects. |
| load\_allseries() | Call methods load\_inputseries(), load\_fluxseries(), and load\_stateseries(). |
| load\_inputseries() | Call method load\_ext() of all InputSequence objects with an activated memoryflag. |
| load\_fluxseries() | Call method load\_ext() of all FluxSequence objects with an activated memoryflag. |
| load\_stateseries() | Call method load\_ext() of all StateSequence objects with an activated memoryflag. |
| save\_allseries() | Call methods save\_inputseries(), save\_fluxseries(), and save\_stateseries(). |
| save\_inputseries() | Call method save\_ext() of all InputSequence objects with an activated memoryflag. |
| save\_fluxseries() | Call method save\_ext() of all FluxSequence objects with an activated memoryflag. |
| save\_stateseries | Call method save\_ext() of all StateSequence objects with an activated memoryflag. |

## Klasse Element

Klasse hydpy.core.devicetools.Element, Modul devicetools   
https://hydpy-dev.github.io/hydpy/devicetools.html#hydpy.core.devicetools.Element

Subclass von hydpy.core.devicetools.Device   
<https://hydpy-dev.github.io/hydpy/devicetools.html#hydpy.core.devicetools.Device>

Handles a Model object and connects it to other models via Node objects.

### Properties/Attribute

|  |  |
| --- | --- |
| **Properties/Attribute** | **Beschreibung** |
| name <Device> | Name of the actual Node or Element object. |
| .keywords <Device> | Keywords describing the actual Node or Element object. |
| model | The Model object handled by the actual Element object. |
| inlets | Group of Node objects from which the actual Element object queries its “upstream” input values (e.g. inflow). |
| outlets | Group of Node objects to which the actual Element object passes its “downstream” output values (e.g. outflow). |
| receivers | Group of Node objects from which the actual Element object queries its “remote” information values (e.g. discharge at a remote downstream). |
| senders | Group of Node objects to which the actual Element object passes its “remote” information values (e.g. water level in a dam model). |
| variables | A set of all different variable values of the Node objects directly connected to the actual Element object. |

### Methoden:

|  |  |
| --- | --- |
| Methode | Beschreibung |
| Element.extract\_new() <Device> | Gather all “new” Node or Element objects. |
| Element.query\_all() <Device> | Get all Node or Element objects initialised so far |
| Element.clear\_all() <Device> | Clear the registry from all initialised Node or Element objects. |
| prepare\_model() | Load the control file of the actual Element object, initialise its Model object, build the required connections via (an eventually overridden version of) method connect() of class Model, and update its derived parameter values via calling (an eventually overridden version) of method update() of class Parameters. |
| open\_files | Call method open\_files() of the Sequences object handled (indirectly) by the actual Element object. |
| close\_files() | Call method close\_files() of the Sequences object handled (indirectly) by the actual Element object. |
| prepare\_allseries | Prepare the series objects of all input, flux and state sequences of the model handled by this element. |
| prepare\_inputseries | Prepare the series objects of the input sequences of the model handled by this element. |
| prepare\_fluxseries | Prepare the series objects of the flux sequences of the model handled by this element. |
| prepare\_stateseries | Prepare the series objects of the state sequences of the model handled by this element. |
| plot\_inputseries | Plot (the selected) InputSequence series values. |
| plot\_fluxseries | Plot the flux series of the handled model. |
| plot\_stateseries | Plot the state series of the handled model. |
|  |  |
|  |  |
|  |  |

## Klasse Model

Klasse hydpy.core.modeltools.Model, Modul modeltools  
https://hydpy-dev.github.io/hydpy/devicetools.html#hydpy.core.devicetools.Element

Base class for all hydrological models.

### Properties/Attribute

|  |  |
| --- | --- |
| **Properties/Attribute** | **Beschreibung** |
| name | Name of the model type. |
| parameters | All parameters of the actual model. |
| sequences | All sequences of the actual model. |
| idx\_sim | The index of the current simulation time step. |
| masks | All predefined masks of the actual model type contained in a Masks objects. |
|  |  |
|  |  |
|  |  |

### Methoden:

|  |  |
| --- | --- |
| Methode | Beschreibung |
| connect() | Connect all LinkSequence objects of the actual model to the corresponding NodeSequence objects. |
| abstract simulate | Perform a simulation run over a single simulation time step. |
| load\_data() | Call method load\_data() of attribute sequences. |
| save\_data | Call method save\_data() of attribute sequences. |
| update\_inlets() | Call all methods defined as “INLET\_METHODS” in the defined order. |
| update\_outlets() | Call all methods defined as “OUTLET\_METHODS” in the defined order. |
| update\_receivers | Call all methods defined as “RECEIVER\_METHODS” in the defined order. |
| update\_senders | Call all methods defined as “SENDER\_METHODS” in the defined order. |
| new2old() | Call method new2old() of subattribute sequences.states. |
| get\_methods() | Convenience method for iterating through all methods selected by a Model subclass. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Klasse Sequences

# Existierende Modelle

* Kurzbeschreibung existierende Modelle mit Verweis auf Dokumentation

# Anwendung hydrologischer Modelle

## Dateistruktur HydPy-Modelle

Ordner Struktur HydPy-Modelle

<Projektname>

-- conditions: Anfangszustände

---- init\_1996\_01\_01\_00\_00\_00: Anfangszustände für den 1.1.1996, für jedes Element eine Datei

------ land\_dill.py: Anfangszustände für das Element land\_dill

------ ....: Anfangszustände für weitere land-Elemente

------ stream\_dill\_lahn\_2.py: Anfangszustände für stream\_dill\_lahn\_2

------ ...: Anfangszustände für weitere stream-Elemente

----....: weitere Ordner für andere Zeitpunkte

-- control: Parameter-/Konfigurations-Dateien

---- default: Default-Parameter-/Kontroll-Dateien

------ land.py: Allgemeine Parameter für alle land-Modelle

------ land\_dill.py: Parameter für land\_dill

------ ...: weitere Parameter-Dateien für land-Modelle

------ stream\_dill\_lahn\_2.py Parameter-Datei für stream\_dill\_lahn\_2

------ ... weitere Parameter-Dateien für stream-Modelle

---- ....: weitere Konfigurationen

-- network: Netzwerk-Struktur: Definition der Netzwerk-Struktur

---- default: Default- Definition der Netzwerk-Struktur, jede Datei wird als eine Selektion eingeladen

------ headwaters.py Selection headwaters Kopfgebiete (Elemente) und Netzwerk-Knoten

------ nonheadwaters.py Selection nonheadwaters Zwischengebiete (Elemente) und Netzwerk-Knoten

------ streams.py: Selection streams Definition der Stream-Modelle im Netzwerk

---- ....: weitere Netzwerk-Definitionen

--series: Zeitreihen

---- input: Input Zeitreihen

------ land\_dill\_input\_epn.asc: Mittlere Verdunstung

------ land\_dill\_input\_p.asc: Niederschlag

------ land\_dill\_input\_t.asc: Temperatur

------ land\_dill\_input\_tn.asc: Mittlere Temperatur

------...: Zeitreihen für die anderen Land-Gebiete

---- node: Ausgabe-Zeitreihen auf dem Knoten

---- output: Ausgabe-Zeitreihen

---- temp: Temporäre Dateien

## Systemdaten HydPy-Modelle

### Zustandsdateien:

Für jedes Element gibt es eine Zustands-Datei. Die Zustands-Dateien sind ausführbarer Python-Code.

# -\*- coding: utf-8 -\*-

from hydpy.models.hland\_v1 import \*

controlcheck(projectdir="LahnH", controldir="default")

ic(0.9694, 1.4694, 0.97058, 1.47058, 0.97172, 1.47172, 0.97281, 1.47281,

0.97386, 1.47386, 0.97487, 1.47487)

sp(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)

wc(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)

sm(185.13164, 181.18755, 199.80432, 196.55888, 212.04018, 209.48859,

222.12115, 220.12671, 230.30756, 228.70779, 236.91943, 235.64427)

uz(7.25228)

lz(8.70695)

quh(0.0)

Abbildung : Beispiel einer Zustandsdatei für land\_dill

# -\*- coding: utf-8 -\*-

from hydpy.models.hstream\_v1 import \*

controlcheck(projectdir="LahnH", controldir="default")

qjoints(10.85182)

Abbildung : Beispiel einer Zustandsdatei für stream\_dill\_lahn\_2

### Kontroll-Dateien

Für jedes Element gibt es eine Parameter/Kontroll-Datei. Die Kontroll-Dateien sind ausführbarer Python-Code. Es kann auch eine Parameter-Datei mit Parametern die für alle land Gebiete gelten:

# -\*- coding: utf-8 -\*-

from hydpy.models.hland\_v1 import \*

simulationstep("1h")

parameterstep("1d")

pcorr(1.0)

icmax(field=1.0, forest=1.5)

resparea(True)

alpha(1.0)

Abbildung : Beispiel für die Parameter-Datei land

# -\*- coding: utf-8 -\*-

from hydpy.models.hland\_v1 import \*

simulationstep("1h")

parameterstep("1d")

area(692.3)

nmbzones(12)

zonetype(FIELD, FOREST, FIELD, FOREST, FIELD, FOREST, FIELD, FOREST, FIELD,

FOREST, FIELD, FOREST)

zonearea(14.41, 7.06, 70.83, 84.36, 70.97, 198.0, 27.75, 130.0, 27.28,

56.94, 1.09, 3.61)

zonez(2.0, 2.0, 3.0, 3.0, 4.0, 4.0, 5.0, 5.0, 6.0, 6.0, 7.0, 7.0)

zrelp(3.75)

zrelt(3.75)

zrele(3.665)

pcorr(auxfile='land')

pcalt(0.1)

...

Abbildung : Beispiel für die Parameter-/Kontroll-Datei land\_dill

# -\*- coding: utf-8 -\*-

from hydpy.models.hstream\_v1 import \*

simulationstep("1h")

parameterstep("1d")

lag(0.0)

damp(0.0)

Abbildung : Beispiel für die Parameter-/Kontroll-Datei stream\_dill\_lahn\_2

### Netzwerkdateien

Die Netzwerkdateien sind ausführbarer Python-Code. Jede Datei entspricht einer Selection, einer Auswahl von Konten und Elementen. Der Dateiname entspricht dem Selection-Name. Die definierten Selections können überlappend (natürlich konstistent) sein. Zusätzlich wird aus allen Datein eine Selection complete erstellt die alle Elemente und Konten enthält.

# -\*- coding: utf-8 -\*-

from hydpy import Node, Element

Node("dill", variable="Q",

keywords=['gauge'])

Node("lahn\_1", variable="Q",

keywords=['gauge'])

Element("land\_dill",

outlets="dill",

keywords=['catchment'])

Element("land\_lahn\_1",

outlets="lahn\_1",

keywords=['catchment'])

Abbildung : Beispiel für die Netzwerkdatei headwaters

# -\*- coding: utf-8 -\*-

from hydpy import Node, Element

Node("lahn\_2", variable="Q",

keywords=['gauge'])

Node("lahn\_3", variable="Q",

keywords=['gauge'])

Element("land\_lahn\_2",

outlets="lahn\_2",

keywords=['catchment'])

Element("land\_lahn\_3",

outlets="lahn\_3",

keywords=['catchment']) keywords=['gauge'])

Element("land\_dill",

outlets="dill",

keywords=['catchment'])

Element("land\_lahn\_1",

outlets="lahn\_1",

keywords=['catchment'])

Abbildung : Beispiel für die Netzwerkdatei nonheadwaters

# -\*- coding: utf-8 -\*-

from hydpy import Node, Element

Element("stream\_dill\_lahn\_2",

inlets="dill",

outlets="lahn\_2",

keywords=['river'])

Element("stream\_lahn\_1\_lahn\_2",

inlets="lahn\_1",

outlets="lahn\_2",

keywords=['river'])

Element("stream\_lahn\_2\_lahn\_3",

inlets="lahn\_2",

outlets="lahn\_3",

keywords=['river'])

Element("land\_lahn\_2",

outlets="lahn\_2",

keywords=['catchment'])

Element("land\_lahn\_3",

outlets="lahn\_3",

keywords=['catchment']) keywords=['gauge'])

Element("land\_dill",

outlets="dill",

keywords=['catchment'])

Element("land\_lahn\_1",

outlets="lahn\_1",

keywords=['catchment'])

Abbildung : Beispiel für die Netzwerkdatei streams

## Input-Zeitreihen

Die Inputzeitreihen können im ASCII-Format oder im NCDF-Format übergeben werden.

### NumPy .npy

### ASCII .asc

Für jedes land-Element und jede Variable eine eigene Datei, Namenskonvention: land\_<gebietsname>\_input\_<Variable>.asc

Timegrid('1996-01-01 00:00:00+01:00',

'2007-01-01 00:00:00+01:00',

'1d')

0.000000000000000000e+00

0.000000000000000000e+00

0.000000000000000000e+00

0.000000000000000000e+00

0.000000000000000000e+00

1.288461886051925465e+00

6.923126578846573731e-02

0.000000000000000000e+00

9.846148439192260460e-01

3.846109707691931778e-02

0.000000000000000000e+00

0.000000000000000000e+00

0.000000000000000000e+00

0.000000000000000000e+00

0.000000000000000000e+00

...

Abbildung : Beispiel für den Niederschlagsinput für das Gebiet dill

Ausgabedateien, für jedes Element und Variable Ausgabezeitreihe, Unterscheidung zwischen state und flux Variablen. Für Variablen die für HRUs definiert sind, werden auch HRU-Werte in den Gebietsergebnissdateien ausgegeben, Namenskonvention land\_<gebietsnames>\_state\_<variable>.asc, land\_<gebietsnames>\_flux\_<variable>.asc

imegrid('1996-01-01 00:00:00+01:00',

'1996-01-05 00:00:00+01:00',

'1d')

5.964567837649028537e-01 5.964567837649028537e-01 2.964567837649026982e-01 2.964567837649026982e-01

3.401105700559600153e-01 3.401105700559600153e-01 4.011057005595997094e-02 4.011057005595997094e-02

0.000000000000000000e+00 0.000000000000000000e+00 0.000000000000000000e+00 0.000000000000000000e+00

0.000000000000000000e+00 0.000000000000000000e+00 0.000000000000000000e+00 0.000000000000000000e+00

Abbildung : Beispiel eine Ausgabe für die einzelnen HRUs des Elements

### NetCDF .nc

## Anwendungsbeispiele

### Beobachtung an einem Knoten übernehmen

Mit der Knoten-Eigenschaft: property deploymode kann definiert werden ob an einem Knoten

1. newsim (default): die neue Simulation
2. obs: die beobachteten Werte, die Simulation oberhalb ist in der sequence sim verfügbar
3. oldsim: die Simulationswerte einer alten Simulation, die in der sequence sim gespeichert ist, in diesm Fall ist die Simulation oberhalb nicht verfügbar

### Plotten

Um die Ergebnisse einer Simulation an einem Knoten zu plotten, lieftert das hydpy Framework die Funktion plot\_allseries, plot\_obsseries, plot\_simseries für die Knotenobjekte

>>> from hydpy.examples import prepare\_full\_example\_2

>>> hp, \_, \_ = prepare\_full\_example\_2(lastdate='1997-01-01')

>>> hp.simulate()

>>> dill = hp.nodes.dill

>>> dill.sequences.obs.series = dill.sequences.sim.series + 10.0

>>> # interaktiven mode von matplotlib aktivierne

>>> import matplotlib

>>> matplotlib.pyplot.ion()

>>> dill.plot\_allseries()

>>> land = hp.elements.land\_dill

>>> land.plot\_inputseries()

>>> # save to disk

>>> from matplotlib import pyplot

>>> from hydpy.docs import figs

>>> pyplot.savefig(figs.\_\_path\_\_[0] + '/Element\_plot\_inputseries.png')

>>> pyplot.close()

>>> land.plot\_fluxseries(['q0', 'q1'], linewidth=2)

>>> pyplot.savefig(figs.\_\_path\_\_[0] + '/Element\_plot\_fluxseries.png')

>>> pyplot.close()

>>> land.plot\_stateseries(['sp', 'wc'])

>>> pyplot.savefig(figs.\_\_path\_\_[0] + '/Element\_plot\_stateseries1.png')

>>> pyplot.close()

>>> land.plot\_stateseries(['sm'], color='grey')

>>> land.plot\_stateseries(

... ['sm'], average=True, color='black', linewidth=3)

>>> pyplot.savefig(figs.\_\_path\_\_[0] + '/Element\_plot\_stateseries2.png')

>>> pyplot.close()

## Steuerung XML-Config Datei

# Erstellung Hydrologischer Modelle

Fluxes, States, Input Sequences, Parameter sind erst verfügbar, wenn Sie in einer der modeltools.Methoden in \*\_model.py mit CONTROLPARAMETERS = (,DERIVEDPARAMETERS = (, REQUIREDSEQUENCES = ( , UPDATEDSEQUENCES = (, RESULTSEQUENCES = ( definiert wurden. Dies ist vor allem bei der Entwicklung beim Testen von Parametern wichtig, da diese sonst nicht erkannt werden.

## Input-Zeitreihen definieren

Inout-Zeitreihen werden als Klassen definiert, die von der Klasse sequencetools.InputSequence aus dem Modul sequencetools abgeleitet sind:

<https://hydpy-dev.github.io/hydpy/sequencetools.html#hydpy.core.sequencetools.InputSequence>

die eine Subklasse der Klasse ModelSequence ist:

<https://hydpy-dev.github.io/hydpy/sequencetools.html#hydpy.core.sequencetools.ModelSequence>

die eine Subklasse der Klasse IOSequence ist

<https://hydpy-dev.github.io/hydpy/sequencetools.html#hydpy.core.sequencetools.IOSequence>

die eine Subklasse der Klasse Sequence ist:

<https://hydpy-dev.github.io/hydpy/sequencetools.html#hydpy.core.sequencetools.Sequence>

die eine Subklasse der Variable ist

Klasse <https://hydpy-dev.github.io/hydpy/variabletools.html#hydpy.core.variabletools.Variable>

Erstellung Datei gr4jland\_inputs.py im Ordner hydpy.models.gr4jland

NDIM: int Anzahl Dimension

TYPE: Type

# -\*- coding: utf-8 -\*-

# pylint: disable=missing-docstring

# pylint: enable=missing-docstring

from hydpy.core import sequencetools

class P(sequencetools.InputSequence):

"""Precipitation [mm]."""

NDIM, NUMERIC = 0, False

class E(sequencetools.InputSequence):

"""Potential Evapotranspiration (PE) [mm]."""

NDIM, NUMERIC = 0, False>>> pyplot.savefig(figs.\_\_path\_\_[0] + '/Element\_plot\_stateseries2.png')

## Parameter definieren

## Quell-Code In C kompilieren

In der \_\_init\_\_.py des model packages, z.B. hydpy.models.hland wird das Objekt cythonizer definiert:

cythonizer = Cythonizer()

cythonizer.finalise()

Im Normalfall wird bei ersten Aufruf des Packages der Quell-Code kompiliert (Option autocompile = True). Um die Kompilierung zu testen kann manuell auf das Objekt zugegriffen werden:

>>> from hydpy.models.hland import \*

>>> cythonizer.cythonize()

## Doc-Tests ausführen

In der \_\_init\_\_.py des model packages, z.B. hydpy.models.hland wird das Objekt tester definiert:

tester = Tester()

Auf dieses Objekt kann zugegriffen werden um die doctests auszuführen.

>>> from hydpy.models.hland import \*

>>> tester.perform\_tests()

## Dokumentation mit Sphinx

Python packages sphinx, sphinxcontrib-fulltoc erforderlich (mit pip installieren).

Um ein neues Modell zur Dokumentation hinzuzufügen, in der Datei hydpy/docs/rst/modelcollection.rst ein neues Modell (z.B. HydPy-XXX unterhalb von HydPy-L) hinzufügen und eine neue rst Datei hydpy/docs/rst/HydPy-XXX analog zu HydPy-L oder HydPy-H erstellen.

Vor der Dokumentation unbedingt aus der source installieren, da nach der Installation alle relevanten Ressourcen aus D:\Programme\Python37\Lib\site-packages in das HydPy source Verzeichnis zurück kopiert werden (vor allem Dateien aus D:\Programme\Python37\Lib\site-packages\hydpy\cythons\autogen). Sollte aus der source installieren nicht funktioniert haben, die Dateien aus D:\Programme\Python37\Lib\site-packages\hydpy\cythons\autogen manuell in den entsprechenden Ordner von der source kopieren.

Die HTML-Dokumentation von HydPy wird automatisch durch SPHINX erzeugt. Erstellung Dokumentation:

Windows-Kommandozeile

1. Im Verzeichnis hydpy/docs „python prepare.py“ ausführen

Meldung coverage.html could not be moved, kann ignoriert werden, da sie nicht mehr existiert, da inzwischen das gesamte Framework mit Tests abgedeckt ist

1. Im Verzeichnis hydpy/docs/auto „python apidoc.py“ ausführen

Im Zweifel in hydpy.core.optiontools autocompile auf False setzen, usecython auf False setzen, damit die Quelldateien nicht kompiliert werden (falls es Probleme mit Cython gibt).

# Tutorials

# Fragen

1. Log Sequence INIT value setzen, am besten 0.

class **Q9**(sequencetools.LogSequence):

NDIM, NUMERIC, SPAN = 1, False, (0., None)

INIT = 0.

def **\_\_call\_\_**(*self*, \*args):

try:

sequencetools.LogSequence.\_\_call\_\_(*self*, \*args)

*self*.values[-1] = 0.

except BaseException as exc:

sequencetools.LogSequence.\_\_call\_\_(

*self*, numpy.sum(args)/(*self*.shape[0]-1))

*self*.values[-1] = 0.

warnings.warn(

*f'Due to the following problem, log sequence '*

*f'{objecttools.elementphrase(self)} handling model '*

*f'`{self.subseqs.seqs.model}` could be initialised '*

*f'with a averaged value only: {exc}'*)

1. Warum werden derived parameters nicht automatisch upgedated wenn sich der parameter ändert?