# pymake

*Release 1.2.7*

**Joseph D. Hughes**

**Mar 28, 2023**

# CONTENTS

This is a python package for compiling MODFLOW-based and other Fortran, C, and C++ programs. The package determines the build order using a directed acyclic graph and then compiles the source files using GNU compilers (`gcc`, `g++`, `gfortran`) or Intel compilers (`ifort`, `icc`).

pymake can be run from the command line or it can be called from within python. By default, pymake sets the optimization level, Fortran flags, C/C++ flags, and linker flags that are consistent with those used to compile MODFLOW-based programs released by the USGS.

pymake includes example scripts for building MODFLOW 6, MODFLOW-2005, MODFLOW-NWT, MODFLOW-USG, MODFLOW-LGR, MODFLOW-2000, MODPATH 6, MODPATH 7, GSFLOW, VS2DT, MT3DMS, MT3D-USGS, SEAWAT, and SUTRA. Example scripts for creating the utility programs CRT, Triangle, and GRIDGEN are also included. The scripts download the distribution file from the USGS (and other organizations) and compile the source into a binary executable.

The main documentation for the site is organized into the following sections:

CHAPTER

ONE

# GETTING STARTED

Pymake can be used directly from the command line and within a python script. Examples of both use cases are given below.

## 1.1 Command Line Usage

When pymake is installed, a `mfpymake` (or `mfpymake.exe` for Windows) program is installed. `mfpymake` can be used to compile MODFLOW 6 in the bin/ subdirectory directly from the command line using the Intel Fortran compiler from a subdirectory at the same level as the `src` subdirectory by specifying:

```
mfpymake src/ mf6 -mc --subdirs -fc ifort --appdir bin
```

MODFLOW 6 would be To see help for running from command line, use the following statement.

```
mfpymake -h
```

The help message identifies required positional arguments and optional arguments that can be provided to overide default values.

```
usage: mfpymake [-h] [-fc {ifort,mpiifort,gfortran,ftn,none}]
                [-cc {gcc,clang,clang++,icc,icl,mpiicc,g++,cl,none}]
                [-ar {ia32,ia32_intel64,intel64}] [-mc] [-dbl] [-dbg]
                [-e] [-dr] [-sd] [-ff FFLAGS] [-cf CFLAGS]
                [-sl {-lc,-lm}] [-mf] [-md] [-cs COMMONSRC]
                [-ef EXTRAFILES] [-exf EXCLUDEFILES] [-so]
                [-ad APPDIR] [-v] [--keep] [--zip ZIP] [--inplace]
                [--networkx] [--mb] [-mbd]
                srcdir target

This is the pymake program for compiling fortran, c, and c++ source
files, such as the source files that come with MODFLOW. The program
works by building a directed acyclic graph of the module dependencies
and then compiling the source files in the proper order.

positional arguments:
  srcdir                Path source directory.
  target                Name of target to create. (can include path)

optional arguments:
  -h, --help            show this help message and exit
```

(continues on next page)

(continued from previous page)

```
-fc {ifort,mpiifort,gfortran,ftn,none}
                      Fortran compiler to use. (default is
                      gfortran)
-cc {gcc,clang,clang++,icc,icl,mpiicc,g++,cl,none}
                      C/C++ compiler to use. (default is gcc)
-ar {ia32,ia32_intel64,intel64}, --arch {ia32,ia32_intel64,intel64}
                      Architecture to use for Intel and Microsoft
                      compilers on Windows. (default is intel64)
-mc, --makeclean      Clean temporary object, module, and source
                      files when done. (default is False)
-dbl, --double        Force double precision. (default is False)
-dbg, --debug         Create debug version. (default is False)
-e, --expedite        Only compile out of date source files. Clean
                      must not have been used on previous build.
                      (default is False)
-dr, --dryrun         Do not actually compile. Files will be
                      deleted, if --makeclean is used. Does not
                      work yet for ifort. (default is False)
-sd, --subdirs        Include source files in srcdir
                      subdirectories. (default is None)
-ff FFLAGS, --fflags FFLAGS
                      Additional Fortran compiler flags. Fortran
                      compiler flags should be enclosed in quotes
                      and start with a blank space or separated
                      from the name (-ff or --fflags) with a equal
                      sign (-ff='-O3'). (default is None)
-cf CFLAGS, --cflags CFLAGS
                      Additional C/C++ compiler flags. C/C++
                      compiler flags should be enclosed in quotes
                      and start with a blank space or separated
                      from the name (-cf or --cflags) with a equal
                      sign (-cf='-O3'). (default is None)
-sl {-lc,-lm}, --syslibs {-lc,-lm}
                      Linker system libraries. Linker libraries
                      should be enclosed in quotes and start with a
                      blank space or separated from the name (-sl
                      or --syslibs) with a equal sign
                      (-sl='-libgcc'). (default is None)
-mf, --makefile       Create a GNU make makefile. (default is
                      False)
-md, --makefile-dir   GNU make makefile directory. (default is '.')
-cs COMMONSRC, --commonsrc COMMONSRC
                      Additional directory with common source
                      files. (default is None)
-ef EXTRAFILES, --extrafiles EXTRAFILES
                      List of extra source files to include in the
                      compilation. extrafiles can be either a list
                      of files or the name of a text file that
                      contains a list of files. (default is None)
-exf EXCLUDEFILES, --excludefiles EXCLUDEFILES
                      List of extra source files to exclude from
                      the compilation. excludefiles can be either a
```

(continues on next page)

(continued from previous page)

```
                        list of files or the name of a text file that
                        contains a list of files. (default is None)
  -so, --sharedobject   Create shared object or dll on Windows.
                        (default is False)
  -ad APPDIR, --appdir APPDIR
                        Target path that overides path defined target
                        path (default is None)
  -v, --verbose         Verbose output to terminal. (default is
                        False)
  --keep                Keep existing executable. (default is False)
  --zip ZIP             Zip built executable. (default is False)
  --inplace             Source files in srcdir are used directly.
                        (default is False)
  --networkx            Use networkx package to build Directed
                        Acyclic Graph use to determine the order
                        source files are compiled in. (default is
                        False)
  --mb, --meson-build   Use meson to build executable. (default is
                        False)
  -mbd, --mesonbuild-dir
                        meson directory. (default is '.')

Note that the source directory should not contain any bad
or duplicate source files as all source files in the source
directory, the common source file directory (srcdir2), and
the extra files (extrafiles) will be built and linked.
Files can be excluded by using the excludefiles command
line switch.

Examples:

Compile MODFLOW 6 from the root directory containing the
source files in subdirectories in the src/ subdirectory:

$ mfpymake src/ mf6 --subdirs

Compile MODFLOW 6 in the bin subdirectory using the Intel
Fortran compiler from the root directory containing the source
files in subdirectories in the the src/ subdirectory:

$ mfpymake src/ mf6 --subdirs -fc ifort --appdir bin
```

Note that command line arguments for Fortran flags, C/C++ flags, and syslib libraries should be enclosed in quotes and start with a space prior to the first value (`-ff ' -O3'`) or use an equal sign separating the command line argument and the values (`-ff='-O3'`). The command line argument to use an `-O3` optimization level when compiling MODFLOW 6 in the bin/ subdirectory with the Intel Fortran compiler would be:

```
mfpymake ../src/ mf6 -mc --subdirs -fc ifort -ff='-O3' --appdir bin
```

## 1.2 From Python

### 1.2.1 Script to compile MODFLOW 6

When using the pymake object (`Pymake()`) only the positional arguments (`srcdir`, `target`) need to be specified in the script.

```
import pymake

pm = pymake.Pymake()
pm.srcdir = '../src'
pm.target = 'mf6'
pm.include_subdirs = True
pm.build()
```

It is suggested that optional variables required for successful compiling and linking be manually specified in the script to mininimize the potential for unsuccessful builds. For MODFLOW 6, subdirectories in the `src` subdirectory need to be included and '`pm.include_subdirs = True`' has been specified in the script. Custom optimization levels and compiler flags could be specified to get consistent builds.

Non-default values for the optional arguments can specified as command line arguments. For example, MODFLOW 6 could be compiled using Intel compilers instead of the default GNU compilers with the script listed above by specifying:

```
python mymf6script.py -fc ifort -cc icc
```

## 1.3 Notes

If gfortran is used to compile MODFLOW-based codes, the `openspec.f` and `FILESPEC.inc` (MT3DMS) files will automatically be changed to the following so that binary files are created properly using standard Fortran:

```
c -- created by pymake.py
CHARACTER*20 ACCESS,FORM,ACTION(2)
DATA ACCESS/'STREAM'/
DATA FORM/'UNFORMATTED'/
DATA (ACTION(I),I=1,2)/'READ','READWRITE'/
c -- end of include file
```

Use of `STREAM` access does not delete an existing unformatted file before opening it for writing. As a result, data from previous runs may exist in the file if the model is run for a shorter period of time. This does not apply to MODFLOW 6 simulations.

CHAPTER

TWO

# PYMAKE INSTALLATION

To install a stable version from PyPI:

```
pip install mfpymake
```

To install pymake directly from the git repository type:

```
pip install https://github.com/modflowpy/pymake/zipball/master
```

To update your version of pymake with the latest from the git repository type:

```
pip install https://github.com/modflowpy/pymake/zipball/master --upgrade
```

CHAPTER

THREE

# BUILDING APPLICATIONS

When pymake is installed, a `make-program` (or `make-program.exe` for Windows) program is installed, which is usually installed to a directory in the PATH (depending on the Python setup). From a console a list of command line arguments and options can be determined by executing:

```
$ make-program --help
usage: make-program [-h] [--release_precision]
                    [-fc {ifort,mpiifort,gfortran,ftn,none}]
                    [-cc {gcc,clang,clang++,icc,icx,icl,mpiicc,g++,cl,none}]
                    [-dr] [-ff FFLAGS] [-cf CFLAGS] [-ad APPDIR] [-v] [--keep]
                    [--zip ZIP]
                    targets

Download and build USGS MODFLOW and related programs.

positional arguments:
  targets               Program(s) to build. Options: crt, gridgen, gsflow,
                        libmf6, mf2000, mf2005, mf6, mflgr, mfnwt, mfusg, mp6,
                        mp7, mt3dms, mt3dusgs, prms, sutra, swtv4, triangle,
                        vs2dt, zbud6, zonbud3, zonbudusg, :. Specifying the
                        target to be ':' will build all of the programs.

optional arguments:
  -h, --help            show this help message and exit
  --release_precision   If release_precision is False, then the release
                        precision version will be compiled along with a double
                        precision version of the program for programs where
                        the standard_switch and double_switch in
                        usgsprograms.txt is True. default is True.
  -fc {ifort,mpiifort,gfortran,ftn,none}
                        Fortran compiler to use. (default is gfortran)
  -cc {gcc,clang,clang++,icc,icx,icl,mpiicc,g++,cl,none}
                        C/C++ compiler to use. (default is gcc)
  -dr, --dryrun         Do not actually compile. Files will be deleted, if
                        --makeclean is used. Does not work yet for ifort.
                        (default is False)
  -ff FFLAGS, --fflags FFLAGS
                        Additional Fortran compiler flags. Fortran compiler
                        flags should be enclosed in quotes and start with a
                        blank space or separated from the name (-ff or
                        --fflags) with a equal sign (-ff='-O3'). (default is
```

(continues on next page)

(continued from previous page)

```
                        None)
  -cf CFLAGS, --cflags CFLAGS
                        Additional C/C++ compiler flags. C/C++ compiler flags
                        should be enclosed in quotes and start with a blank
                        space or separated from the name (-cf or --cflags)
                        with a equal sign (-cf='-O3'). (default is None)
  -ad APPDIR, --appdir APPDIR
                        Target path that overides path defined target path
                        (default is None)
  -v, --verbose         Verbose output to terminal. (default is False)
  --keep                Keep existing executable. (default is False)
  --zip ZIP             Zip built executable. (default is False)

Examples:

  Download and compile MODFLOW 6 in the current directory:
    $ make-program mf6

  Download and compile triangle in the ./temp subdirectory:
    $ make-program triangle --appdir temp

  Download and compile all programs in the ./temp subdirectory:
    $ make-program : --appdir temp
```

`make-program` can be used to build MODFLOW 6, MODFLOW-2005, MODFLOW-NWT, MODFLOW-USG, MODFLOW-LGR, MODFLOW-2000, MODPATH 6, MODPATH 7, GSFLOW, VS2DT, MT3DMS, MT3D-USGS, SEAWAT, GSFLOW, PRMS, and SUTRA. Utility programs CRT, Triangle, and GRIDGEN can also be built. `make-program` downloads the distribution file from the USGS (requires internet connection), unzips the distribution file, sets the pymake settings required to build the program, and compiles the source files to build the program. MT3DMS will be downloaded from the University of Alabama and Triangle will be downloaded from netlib.org. Optional command line arguments can be used to customize the build (`-fc`, `-cc`, `--fflags`, etc.). For example, MODFLOW 6 could be built using intel compilers and an `O3` optimation level by specifying:

```
make-program mf6 -fc=ifort --fflags='-O3'
```

CHAPTER

# FOUR

# API-DOCS

This section contains the Documentation of the Application Programming Interface (API) of pymake. The information in this section is automatically created from the documentation strings in original Python code. In the left-hand menu you will find the different categories of the API documentation.

## 4.1 pymake.pymake module

`Pymake()` class to make a binary executable for a FORTRAN, C, or C++ program, such as MODFLOW 6.

An example of how to build MODFLOW-2005 from source files in the official release downloaded from the USGS using Intel compilers is:

```
import pymake

# create an instance of the Pymake object
pm = pymake.Pymake(verbose=True)

# reset select pymake settings
pm.target = "mf2005"
pm.appdir = "../bin"
pm.fc = "ifort"
pm.cc = "icc"
pm.fflags = "-O3 -fbacktrace"
pm.cflags = "-O3"

# download the target
pm.download_target(pm.target, download_path="temp")

# build the target
pm.build()

# clean up downloaded files
pm.finalize()
```

All other settings not specified in the script would be based on command line arguments or default values. The same `Pymake()` object could be used to compile MODFLOW 6 by appending the following code to the previous code block:

```
# reset the target
pm.target = "mf6"
```

(continues on next page)

(continued from previous page)

```
# download the target
pm.download_target(pm.target, download_path="temp")

# build the target
pm.build()

# clean up downloaded files
pm.finalize()
```

The Intel compilers and fortran flags defined previously would be used when MODFLOW 6 was built.

**class Pymake**(*name='pymake'*, *verbose=None*)

Bases: `object`

Pymake class for interacting with pymake functionality. This is essentially a wrapper for all of the pymake functions needed to download and build a target.

**argv_reset_settings**(*args*)

Reset settings using command line arguments

**Parameters**

**args** (*Namespace object*) – reset self.variables using command line arguments

**build**(*target=None*, *srcdir=None*, *modify_exe_name=False*)

Build the target

**Parameters**

- **target** (*str*) – target name. If target is None self.target is used. (default is None)
- **srcdir** (*str*) – path to directory with source files. (default is None)
- **modify_exe_name** (*bool*) – boolean that determines if the target name can be modified to include precision (dbl) and debugging (d) indicators.

**compress_targets**()

Compress targets in build_targets list.

**download_setup**(*target*, *url=None*, *download_path='.'*, *verify=True*, *timeout=30*)

Setup download

**Parameters**

- **target** (*str*) – target name
- **url** (*str*) – url of asset
- **download_path** (*str*) – path where the asset will be saved
- **verify** (*bool*) – boolean defining ssl verification
- **timeout** (*int*) – download timeout in seconds (default is 30)

**download_target**(*target*, *url=None*, *download_path='.'*, *verify=True*, *timeout=30*)

Setup and download url

**Parameters**

- **target** (*str*) – target name
- **url** (*str*) – url of asset

- **download_path** (*str*) – path where the asset will be saved
- **verify** (*bool*) – boolean defining ssl verification
- **timeout** (*int*) – download timeout in seconds (default is 30)

**Returns**
**success** – boolean flag indicating download success

**Return type**
bool

**download_url**()

Download files from the url

**Returns**
**success** – boolean flag indicating download success

**Return type**
bool

**finalize**()

Finalize Pymake class

**reset**(*target*)

Reset PyMake object variables for a target

**Parameters**
**target** (*str*) – target name

**set_build_target_bool**(*target=None*)

Evaluate if the executable exists and if so and the command line argument –keep is specified then the executable is not built.

**Parameters**
**target** (*str*) – target name. If target is None self.target will be used. (default is None)

**Returns**
**build** – boolean indicating if the executable should be built

**Return type**
bool

**update_build_targets**()

Add target to build_targets list if it is not in the list

**update_target**(*target*, *modify_target=False*)

Update target name with executable extension on Windows and based on pymake settings.

**Parameters**

- **target** (*str*) – target name
- **modify_target** (*bool*) – boolean indicating if the target name can be modified based on pymake double and debug settings (default is False)

**Returns**
**target** – updated target name

**Return type**
str

## 4.2 pymake.pymake_build_apps module

Function to build MODFLOW-based models and other utility software based on targets defined in the usgsprograms database (usgsprograms.txt). The usgsprograms database can be queried using functions in the usgsprograms module. An example of using `pymake.build_apps()` to build MODFLOW 6 is:

```
import pymake
pymake.build_apps(["mf6",])
```

which will download the latest MODFLOW 6 software release, compile the code, and delete the downloaded files after successfully building the application. Multiple applications can be built by adding additional targets to the tuple in `pymake.build_apps()`. For example, MODFLOW 6 and MODFLOW-2005 could be built by specifying:

```
import pymake
pymake.build_apps(["mf6","mf2005"]))
```

Applications are built in the order they are listed in the list. All valid USGS applications are built if no list is passed to `pymake.build_apps()`.

**build_apps**(*targets=None*, *pymake_object=None*, *download_dir=None*, *appdir=None*, *verbose=None*, *release_precision=True*, *meson=False*, *mesondir='.'*, *clean=True*)

Build all of the current targets or a subset of targets.

**Parameters**

- **targets** (*str or list of str*) – targets to build. If targets is None, all current targets will be build. Default is None
- **pymake_object** (Pymake()) – Pymake object created outside of build_apps
- **download_dir** (*str*) – download directory path
- **appdir** (*str*) – target path
- **release_precision** (*bool*) – boolean indicating if only the release precision version should be build. If release_precision is False, then the release precision version will be compiled along with a double precision version of the program for programs where the standard_switch and double_switch in usgsprograms.txt is True. default is True.
- **meson** (*bool*) – boolean indicating that the executable should be built using the meson build system. (default is False)
- **mesondir** (*str*) – Main meson.build file path
- **clean** (*bool*) – boolean determining of final download should be removed

**Returns**

**returncode** – integer value indicating successful completion (0) or failure (>0)

**Return type**

int

## 4.3 pymake.pymake_base module

Main pymake function, `pymake.main()`, that is called when pymake is run from the command line. `pymake.main()` can also be called directly from a script in combination with `pymake.parser()`.

```
import pymake
args = pymake.parser()
pymake.main(
    args.srcdir,
    args.target,
    fc=args.fc,
    cc=args.cc,
    makeclean=args.makeclean,
    expedite=args.expedite,
    dryrun=args.dryrun,
    double=args.double,
    debug=args.debug,
    include_subdirs=args.subdirs,
    fflags=args.fflags,
    cflags=args.cflags,
    arch=args.arch,
    syslibs=args.syslibs,
    makefile=args.makefile,
    srcdir2=args.commonsrc,
    extrafiles=args.extrafiles,
    excludefiles=args.excludefiles,
    sharedobject=args.sharedobject,
    appdir=args.appdir,
    verbose=args.verbose,
    inplace=args.inplace,
)
```

The script could be run from the command line using:

```
python myscript.py ../src myapp -fc=ifort -cc=icc
```

**get_temporary_directories**(*appdir=None*, *target=None*)

Get paths to temporary object, module, and source files

**Parameters**

- **appdir** (*str*) – path for executable
- **target** (*str*) – target name to be appended to the temporary directories. Default is None

**Returns**

- **obj_temp** (*str*) – path to temporary object files
- **mod_temp** (*str*) – path to temporary module files
- **src_temp** (*str*) – path to temporary source files

**main**(*srcdir=None*, *target=None*, *fc='gfortran'*, *cc='gcc'*, *makeclean=True*, *expedite=False*, *dryrun=False*, *double=False*, *debug=False*, *include_subdirs=False*, *fflags=None*, *cflags=None*, *syslibs=None*, *arch='intel64'*, *makefile=False*, *makefiledir='.'*, *srcdir2=None*, *extrafiles=None*, *excludefiles=None*, *sharedobject=False*, *appdir=None*, *verbose=False*, *inplace=False*, *networkx=False*, *meson=False*, *mesondir='.'*)

Main pymake function.

**Parameters**

- **srcdir** (*str*) – path for directory containing source files
- **target** (*str*) – executable name or path for executable to create
- **fc** (*str*) – fortran compiler
- **cc** (*str*) – c or cpp compiler
- **makeclean** (*bool*) – boolean indicating if intermediate files should be cleaned up after successful build
- **expedite** (*bool*) – boolean indicating if only out of date source files will be compiled. Clean must not have been used on previous build.
- **dryrun** (*bool*) – boolean indicating if source files should be compiled. Files will be deleted, if makeclean is True.
- **double** (*bool*) – boolean indicating a compiler switch will be used to create an executable with double precision real variables.
- **debug** (*bool*) – boolean indicating is a debug executable will be built
- **include_subdirs** (*bool*) – boolean indicating source files in srcdir subdirectories should be included in the build
- **fflags** (*list*) – user provided list of fortran compiler flags
- **cflags** (*list*) – user provided list of c or cpp compiler flags
- **syslibs** (*list*) – user provided syslibs
- **arch** (*str*) – Architecture to use for Intel Compilers on Windows (default is intel64)
- **makefile** (*bool*) – boolean indicating if a GNU make makefile should be created
- **makefiledir** (*str*) – GNU make makefile path
- **srcdir2** (*str*) – additional directory with common source files.
- **extrafiles** (*str*) – path for extrafiles file that contains paths to additional source files to include
- **excludefiles** (*str*) – path for excludefiles file that contains filename of source files to exclude from the build
- **sharedobject** (*bool*) – boolean indicating a shared object will be built
- **appdir** (*str*) – path for executable
- **verbose** (*bool*) – boolean indicating if output will be printed to the terminal
- **inplace** (*bool*) – boolean indicating that the source files in srcdir, srcdir2, and defined in extrafiles will be used directly. If inplace is False, source files will be copied to a directory named srcdir_temp. (default is False)
- **networkx** (*bool*) – boolean indicating that the NetworkX python package will be used to create the Directed Acyclic Graph (DAG) used to determine the order source files are compiled in. The NetworkX package tends to result in a unique DAG more often than the standard algorithm used in pymake. (default is False)
- **meson** (*bool*) – boolean indicating that the executable should be built using the meson build system. (default is False)

- **mesondir** (*str*) – Main meson.build file path

**Returns**
**returncode** – return code

**Return type**
int

## 4.4 pymake.pymake_parser module

Parser used to process command line arguments when running pymake directly from the command line or in a script. The standard argparse module is used to parse command line arguments. Available command line arguments are programmatically developed by a protected dictionary. The parser can be accessed using:

```
import pymake
args = pymake.parser()
```

**parser**(*examples=None*)

Construct the parser and return argument values.

**Parameters**
**examples** (*str*) –

**Returns**
**args** – Namespace with command line arguments

**Return type**
Namespace object

## 4.5 pymake.plot package

### 4.5.1 Submodules

#### pymake.plot.dependency_graphs module

Dependency graphs for applications can be created using:

```
import os
import pymake

srcpth = os.path.join("..", "src")
deppth = "dependencies"
if not os.path.exists(deppth):
    os.makedirs(deppth)

pymake.visualize.make_plots(srcpth, deppth, include_subdir=True)
```

**make_plots**(*srcdir*, *outdir*, *include_subdir=False*, *level=3*, *extension='.png'*, *verbose=False*, *networkx=False*)

Create plots of module dependencies.

**Parameters**

- **srcdir** (*str*) – path for source files

- **outdir** (*str*) – path for output images
- **include_subdir** (*bool*) – boolean indicating is subdirectories in the source file directory should be included
- **level** (*int*) – dependency level (1 is the minimum)
- **extension** (*str*) – output extension (default is .png)
- **verbose** (*bool*) – boolean indicating if output will be printed to the terminal
- **networkx** (*bool*) – boolean indicating that the NetworkX python package will be used to create the Directed Acyclic Graph (DAG) used to determine the order source files are compiled in. The NetworkX package tends to result in a unique DAG more often than the standard algorithm used in pymake. (default is False)

**to_pydot**(*dag*, *filename='mygraph.png'*)

Create a png file of a Directed Acyclic Graph

**Parameters**

- **dag** (*object*) – directed acyclic graph
- **filename** (*str*) – path of the graph png

### 4.5.2 Module contents

Functions to plot source code dependencies determined using a directed acyclic graph (DAG). pydotplus is used to plot the DAG.

## 4.6 pymake.utils package

### 4.6.1 Submodules

#### pymake.utils.download module

Utility functions to:

1. download and unzip software releases from the USGS and other organizations (triangle, MT3DMS).
2. download the latest MODFLOW-based applications and utilities for MacOS, Linux, and Windows from https://github.com/MODFLOW-USGS/executables
3. determine the latest version (GitHub tag) of a GitHub repository and a dictionary containing the file name and the link to a asset on contained in a github repository
4. compress all files in a list, files in a list of directories

**download_and_unzip**(*url*, *pth='./'*, *delete_zip=True*, *verify=True*, *timeout=30*, *max_requests=10*, *chunk_size=2048000*, *verbose=False*)

Download and unzip a zip file from a url.

**Parameters**

- **url** (*str*) – url address for the zip file
- **pth** (*str*) – path where the zip file will be saved (default is the current path)

- **delete_zip** (*bool*) – boolean indicating if the zip file should be deleted after it is unzipped (default is True)
- **verify** (*bool*) – boolean indicating if the url request should be verified
- **timeout** (*int*) – url request time out length (default is 30 seconds)
- **max_requests** (*int*) – number of url download request attempts (default is 10)
- **chunk_size** (*int*) – maximum url download request chunk size (default is 2048000 bytes)
- **verbose** (*bool*) – boolean indicating if output will be printed to the terminal

**get_repo_assets**(*github_repo=None*, *version=None*, *error_return=False*, *verify=True*)

Return a dictionary containing the file name and the link to the asset contained in a github repository.

**Parameters**

- **github_repo** (*str*) – Repository name, such as MODFLOW-USGS/modflow6. If github_repo is None set to 'MODFLOW-USGS/executables'
- **version** (*str*) – github repository release tag
- **error_return** (*bool*) – boolean indicating if None will be returned if there are GitHub API issues
- **verify** (*bool*) – boolean indicating if the url request should be verified

**Returns**

**result_dict** – dictionary of file names and links

**Return type**

dict

**getmfexes**(*pth='.'*, *version=None*, *platform=None*, *exes=None*, *verbose=False*, *verify=True*)

Get the latest MODFLOW binary executables from a github site (https://github.com/MODFLOW-USGS/executables) for the specified operating system and put them in the specified path.

**Parameters**

- **pth** (*str*) – Location to put the executables (default is current working directory)
- **version** (*str*) – Version of the MODFLOW-USGS/executables release to use. If version is None the github repo will be queried for the version number.
- **platform** (*str*) – Platform that will run the executables. Valid values include mac, linux, win32 and win64. If platform is None, then routine will download the latest asset from the github repository.
- **exes** (*str or list of strings*) – executable or list of executables to retain
- **verbose** (*bool*) – boolean indicating if output will be printed to the terminal
- **verify** (*bool*) – boolean indicating if the url request should be verified

**getmfnightly**(*pth='.'*, *platform=None*, *exes=None*, *verbose=False*, *verify=True*)

Get the latest MODFLOW 6 binary nightly-build executables from github (https://github.com/MODFLOW-USGS/modflow6-nightly-build/) for the specified operating system and put them in the specified path.

**Parameters**

- **pth** (*str*) – Location to put the executables (default is current working directory)

- **platform** (*str*) – Platform that will run the executables. Valid values include mac, linux, win32 and win64. If platform is None, then routine will download the latest asset from the github repository.
- **exes** (*str or list of strings*) – executable or list of executables to retain
- **verbose** (*bool*) – boolean indicating if output will be printed to the terminal
- **verify** (*bool*) – boolean indicating if the url request should be verified

**class pymakeZipFile**(*file*, *mode='r'*, *compression=0*, *allowZip64=True*, *compresslevel=None*)

Bases: ZipFile

ZipFile file attributes are not being preserved. This class preserves file attributes as described on StackOverflow at https://stackoverflow.com/questions/39296101/python-zipfile-removes-execute-permissions-from-binaries

**static compressall**(*path*, *file_pths=None*, *dir_pths=None*, *patterns=None*)

Compress selected files or files in selected directories.

**Parameters**

- **path** (*str*) – output zip file path
- **file_pths** (*str or list of str*) – file paths to include in the output zip file (default is None)
- **dir_pths** (*str or list of str*) – directory paths to include in the output zip file (default is None)
- **patterns** (*str or list of str*) – file patterns to include in the output zip file (default is None)

**Returns**
**success** – boolean indicating if the output zip file was created

**Return type**
bool

**extract**(*member*, *path=None*, *pwd=None*)

**Parameters**

- **member** (*str*) – individual file to extract. If member does not exist, all files are extracted.
- **path** (*str*) – directory path to extract file in a zip file (default is None, which results in files being extracted in the current directory)
- **pwd** (*str*) – zip file password (default is None)

**Returns**
**ret_val** – return value indicating status of file extraction

**Return type**
int

**extractall**(*path=None*, *members=None*, *pwd=None*)

Extract all files in the zipfile.

**Parameters**

- **path** (*str*) – directory path to extract files in a zip file (default is None, which results in files being extracted in the current directory)
- **members** (*str*) – individual files to extract (default is None, which extracts all members)

- **pwd** (*str*) – zip file password (default is None)

**repo_latest_version**(*github_repo=None*, *verify=True*)

Return a string of the latest version number (tag) contained in a github repository release.

**Parameters**

**github_repo** (*str*) – Repository name, such as MODFLOW-USGS/modflow6. If github_repo is None set to 'MODFLOW-USGS/executables'

**Returns**

**version** – string with the latest version/tag number

**Return type**

str

**zip_all**(*path*, *file_pths=None*, *dir_pths=None*, *patterns=None*)

Compress all files in the user-provided list of file paths and directory paths that match the provided file patterns.

**Parameters**

- **path** (*str*) – path of the zip file that will be created
- **file_pths** (*str or list*) – file path or list of file paths to be compressed
- **dir_pths** (*str or list*) – directory path or list of directory paths to search for files that will be compressed
- **patterns** (*str or list*) – file pattern or list of file patterns s to match to when creating a list of files that will be compressed

### pymake.utils.usgsprograms module

Utility functions to extract information for a target from the USGS application database. Available functionality includes:

1. Get a list of available targets
2. Get data for a specific target
3. Get a dictionary with the data for all targets
4. Get the current version of a target
5. Get a list indicating if single and double precsion versions of the target application should be built
6. Functions to load, update, and export a USGS-style "code.json" json file containing information in the USGS application database

A table listing the available pymake targets is included below:

Table 1: Available pymake targets

| target | version | current | url | dirname | srcdir | standard_switch | double_switch | shared_object |
|---|---|---|---|---|---|---|---|---|
| mf6 | 6.4.1 | True | https://github.com/MODFLOW-USGS/modflow6/releases/download/6.4.1/mf6.4.1_linux.zip | mf6.4.1_linux | src | True | False | False |
| zbud6 | 6.4.1 | True | https://github.com/MODFLOW-USGS/modflow6/releases/download/6.4.1/mf6.4.1_linux.zip | mf6.4.1_linux | utils/zonebudget/src | True | False | False |
| libmf6 | 6.4.1 | True | https://github.com/MODFLOW-USGS/modflow6/releases/download/6.4.1/mf6.4.1_linux.zip | mf6.4.1_linux | srcbmi | True | False | True |
| mp7 | 7.2.001 | True | https://water.usgs.gov/water-resources/software/MODPATH/modpath_7_2_001.zip | modpath_7_2_001 | source | True | False | False |
| mt3dms | 5.3.0 | True | https://hydro.geo.ua.edu/mt3d/mt3dms_530.exe | mt3dms5.3.0 | src/true-binary | True | False | False |
| mt3dusgs | 1.1.0 | True | https://water.usgs.gov/water-resources/software/MT3D-USGS/mt3dusgs1.1.0.zip | mt3dusgs1.1.0 | src | True | False | False |
| vs2dt | 3.3 | True | https://water.usgs.gov/water-resources/software/VS2DI/vs2dt3_3.zip | vs2dt3_3 | include | True | False | False |
| triangle | 1.6 | True | https://www.netlib.org/voronoi/triangle.zip | triangle1.6 | src | True | False | False |
| gridgen | 1.0.02 | True | https://water.usgs.gov/water-resources/software/GRIDGEN/gridgen.1.0.02.zip | gridgen.1.0.02 | src | True | False | False |
| crt | 1.3.1 | True | https://water.usgs.gov/ogw/CRT/CRT_1.3.1.zip | CRT_1.3.1 | SOURCE | True | False | False |
| sutra | 3.0 | True | https://water.usgs.gov/water-resources/software/sutra/SUTRA_3_0_0.zip | Sutra-Suite | SUTRA_3_0/source | True | False | False |
| mf2000 | 1.19.01 | True | https://water.usgs.gov/nrp/gwsoftware/ | mf2k.1_19 | src | True | False | False |

class dotdict

Bases: dict

dot.notation access to dictionary attributes.

class usgs_program_data

Bases: object

USGS program database class.

static export_json(*fpth='code.json'*, *prog_data=None*, *current=False*, *update=True*, *write_markdown=False*, *verbose=False*)

Export USGS program data as a json file.

**Parameters**

- **fpth** (*str*) – Path for the json file to be created. Default is "code.json"
- **prog_data** (*dict*) – User-specified program database. If prog_data is None, it will be created from the USGS program database
- **current** (*bool*) – If False, all USGS program targets are listed. If True, only USGS program targets that are defined as current are listed. Default is False.
- **update** (*bool*) – If True, existing targets in the user-specified program database with values in the USGS program database. If False, existing targets in the user-specified program database will not be updated. Default is True.
- **write_markdown** (*bool*) – If True, write markdown file that includes the target name, version, and the last-modified date of the download asset (url). Default is False.
- **verbose** (*bool*) – boolean for verbose output to terminal

static get_keys(*current=False*)

Get target keys from the USGS program database.

**Parameters**
**current** (*bool*) – If False, all USGS program targets are listed. If True, only USGS program targets that are defined as current are listed. Default is False.

**Returns**
**keys** – list of USGS program targets

**Return type**
list

static get_precision(*key*)

Get the dictionary for a specified target.

**Parameters**
**key** (*str*) – Target USGS program

**Returns**
**precision** – List

**Return type**
list

static get_program_dict()

Get the complete USGS program database.

**Returns**
**program_dict** – Dictionary with USGS program attributes for all targets

**Return type**
dict

`static get_target(key)`

Get the dictionary for a specified target.

**Parameters**
**key** (*str*) – Target USGS program that may have a path and an extension

**Returns**
**program_dict** – Dictionary with USGS program attributes for the specified key

**Return type**
dict

`static get_version(key)`

Get the current version of the specified target.

**Parameters**
**key** (*str*) – Target USGS program

**Returns**
**version** – current version of the specified target

**Return type**
str

`static list_json(fpth='code.json')`

List an existing code json file.

**Parameters**
**fpth** (*str*) – Path for the json file to be listed. Default is "code.json"

`static list_targets(current=False)`

Print a list of the available USGS program targets.

**Parameters**
**current** (*bool*) – If False, all USGS program targets are listed. If True, only USGS program targets that are defined as current are listed. Default is False.

`static load_json(fpth='code.json')`

Load an existing code json file. Basic error checking is done to make sure the file contains the correct keys.

**Parameters**
**fpth** (*str*) – Path for the json file to be created. Default is "code.json"

**Returns**
**json_dict** – Valid USGS program database

**Return type**
dict

`static update_json(fpth='code.json', temp_dict=None)`

UPDATE an existing code json file.

**Parameters**

- **fpth** (*str*) – Path for the json file to be listed. Default is "code.json"
- **temp_dict** (*dict*) – Dictionary with USGS program data for a target

### 4.6.2 Module contents

Utility functions to 1) download and uncompress software releases containing source code or executables, 2) zip files, 3) query a USGS software and utility program database for information on current software releases.

## 4.7 pymake package

### 4.7.1 Subpackages

#### pymake.cmds package

#### Submodules

#### pymake.cmds.build module

Download and build USGS MODFLOW and related programs.

This script originates from pymake: https://github.com/modflowpy/pymake It requires Python 3.6 or later, and has no dependencies.

**main**() → None

Command line interface

**Return type**
None

#### pymake.cmds.createjson module

Create pymake code.json file.

This script originates from pymake: https://github.com/modflowpy/pymake It requires Python 3.6 or later, and has no dependencies.

**main**() → None

Command line interface

**Return type**
None

#### pymake.cmds.mfpymakecli module

Download and build USGS MODFLOW and related programs.

This script originates from pymake: https://github.com/modflowpy/pymake It requires Python 3.6 or later, and has no dependencies.

**main**() → None

mfpymake command line interface

**Return type**
None

#### Module contents

### 4.7.2 Submodules

#### pymake.config module

### 4.7.3 Module contents

pymake is a python package for compiling MODFLOW-based and other Fortran, C, and C++ programs. The package determines the build order using a directed acyclic graph and then compiles the source files using GNU compilers (`gcc`, `g++`, `gfortran`) or Intel compilers (`ifort`, `icc`).

**class Pymake**(*name='pymake'*, *verbose=None*)

Bases: `object`

Pymake class for interacting with pymake functionality. This is essentially a wrapper for all of the pymake functions needed to download and build a target.

**argv_reset_settings**(*args*)

Reset settings using command line arguments

**Parameters**
**args** (*Namespace object*) – reset self.variables using command line arguments

**build**(*target=None*, *srcdir=None*, *modify_exe_name=False*)

Build the target

**Parameters**

- **target** (*str*) – target name. If target is None self.target is used. (default is None)
- **srcdir** (*str*) – path to directory with source files. (default is None)
- **modify_exe_name** (*bool*) – boolean that determines if the target name can be modified to include precision (dbl) and debugging (d) indicators.

**compress_targets**()

Compress targets in build_targets list.

**download_setup**(*target*, *url=None*, *download_path='.'*, *verify=True*, *timeout=30*)

Setup download

**Parameters**

- **target** (*str*) – target name
- **url** (*str*) – url of asset
- **download_path** (*str*) – path where the asset will be saved
- **verify** (*bool*) – boolean defining ssl verification
- **timeout** (*int*) – download timeout in seconds (default is 30)

**download_target**(*target*, *url=None*, *download_path='.'*, *verify=True*, *timeout=30*)

Setup and download url

**Parameters**

- **target** (*str*) – target name
- **url** (*str*) – url of asset

- **download_path** (*str*) – path where the asset will be saved
- **verify** (*bool*) – boolean defining ssl verification
- **timeout** (*int*) – download timeout in seconds (default is 30)

**Returns**
**success** – boolean flag indicating download success

**Return type**
bool

**download_url**()

Download files from the url

**Returns**
**success** – boolean flag indicating download success

**Return type**
bool

**finalize**()

Finalize Pymake class

**reset**(*target*)

Reset PyMake object variables for a target

**Parameters**
**target** (*str*) – target name

**set_build_target_bool**(*target=None*)

Evaluate if the executable exists and if so and the command line argument –keep is specified then the executable is not built.

**Parameters**
**target** (*str*) – target name. If target is None self.target will be used. (default is None)

**Returns**
**build** – boolean indicating if the executable should be built

**Return type**
bool

**update_build_targets**()

Add target to build_targets list if it is not in the list

**update_target**(*target*, *modify_target=False*)

Update target name with executable extension on Windows and based on pymake settings.

**Parameters**

- **target** (*str*) – target name
- **modify_target** (*bool*) – boolean indicating if the target name can be modified based on pymake double and debug settings (default is False)

**Returns**
**target** – updated target name

**Return type**
str

build_apps(*targets=None*, *pymake_object=None*, *download_dir=None*, *appdir=None*, *verbose=None*, *release_precision=True*, *meson=False*, *mesondir='.'*, *clean=True*)

Build all of the current targets or a subset of targets.

**Parameters**

- **targets** (*str or list of str*) – targets to build. If targets is None, all current targets will be build. Default is None
- **pymake_object** (Pymake()) – Pymake object created outside of build_apps
- **download_dir** (*str*) – download directory path
- **appdir** (*str*) – target path
- **release_precision** (*bool*) – boolean indicating if only the release precision version should be build. If release_precision is False, then the release precision version will be compiled along with a double precision version of the program for programs where the standard_switch and double_switch in usgsprograms.txt is True. default is True.
- **meson** (*bool*) – boolean indicating that the executable should be built using the meson build system. (default is False)
- **mesondir** (*str*) – Main meson.build file path
- **clean** (*bool*) – boolean determining of final download should be removed

**Returns**

**returncode** – integer value indicating successful completion (0) or failure (>0)

**Return type**

int

download_and_unzip(*url*, *pth='./'*, *delete_zip=True*, *verify=True*, *timeout=30*, *max_requests=10*, *chunk_size=2048000*, *verbose=False*)

Download and unzip a zip file from a url.

**Parameters**

- **url** (*str*) – url address for the zip file
- **pth** (*str*) – path where the zip file will be saved (default is the current path)
- **delete_zip** (*bool*) – boolean indicating if the zip file should be deleted after it is unzipped (default is True)
- **verify** (*bool*) – boolean indicating if the url request should be verified
- **timeout** (*int*) – url request time out length (default is 30 seconds)
- **max_requests** (*int*) – number of url download request attempts (default is 10)
- **chunk_size** (*int*) – maximum url download request chunk size (default is 2048000 bytes)
- **verbose** (*bool*) – boolean indicating if output will be printed to the terminal

get_repo_assets(*github_repo=None*, *version=None*, *error_return=False*, *verify=True*)

Return a dictionary containing the file name and the link to the asset contained in a github repository.

**Parameters**

- **github_repo** (*str*) – Repository name, such as MODFLOW-USGS/modflow6. If github_repo is None set to 'MODFLOW-USGS/executables'
- **version** (*str*) – github repository release tag

- **error_return** (*bool*) – boolean indicating if None will be returned if there are GitHub API issues
- **verify** (*bool*) – boolean indicating if the url request should be verified

**Returns**
**result_dict** – dictionary of file names and links

**Return type**
dict

**getmfexes**(*pth='.'*, *version=None*, *platform=None*, *exes=None*, *verbose=False*, *verify=True*)

Get the latest MODFLOW binary executables from a github site (https://github.com/MODFLOW-USGS/executables) for the specified operating system and put them in the specified path.

**Parameters**

- **pth** (*str*) – Location to put the executables (default is current working directory)
- **version** (*str*) – Version of the MODFLOW-USGS/executables release to use. If version is None the github repo will be queried for the version number.
- **platform** (*str*) – Platform that will run the executables. Valid values include mac, linux, win32 and win64. If platform is None, then routine will download the latest asset from the github repository.
- **exes** (*str or list of strings*) – executable or list of executables to retain
- **verbose** (*bool*) – boolean indicating if output will be printed to the terminal
- **verify** (*bool*) – boolean indicating if the url request should be verified

**main**(*srcdir=None*, *target=None*, *fc='gfortran'*, *cc='gcc'*, *makeclean=True*, *expedite=False*, *dryrun=False*, *double=False*, *debug=False*, *include_subdirs=False*, *fflags=None*, *cflags=None*, *syslibs=None*, *arch='intel64'*, *makefile=False*, *makefiledir='.'*, *srcdir2=None*, *extrafiles=None*, *excludefiles=None*, *sharedobject=False*, *appdir=None*, *verbose=False*, *inplace=False*, *networkx=False*, *meson=False*, *mesondir='.'*)

Main pymake function.

**Parameters**

- **srcdir** (*str*) – path for directory containing source files
- **target** (*str*) – executable name or path for executable to create
- **fc** (*str*) – fortran compiler
- **cc** (*str*) – c or cpp compiler
- **makeclean** (*bool*) – boolean indicating if intermediate files should be cleaned up after successful build
- **expedite** (*bool*) – boolean indicating if only out of date source files will be compiled. Clean must not have been used on previous build.
- **dryrun** (*bool*) – boolean indicating if source files should be compiled. Files will be deleted, if makeclean is True.
- **double** (*bool*) – boolean indicating a compiler switch will be used to create an executable with double precision real variables.
- **debug** (*bool*) – boolean indicating is a debug executable will be built
- **include_subdirs** (*bool*) – boolean indicating source files in srcdir subdirectories should be included in the build

- **fflags** (*list*) – user provided list of fortran compiler flags
- **cflags** (*list*) – user provided list of c or cpp compiler flags
- **syslibs** (*list*) – user provided syslibs
- **arch** (*str*) – Architecture to use for Intel Compilers on Windows (default is intel64)
- **makefile** (*bool*) – boolean indicating if a GNU make makefile should be created
- **makefiledir** (*str*) – GNU make makefile path
- **srcdir2** (*str*) – additional directory with common source files.
- **extrafiles** (*str*) – path for extrafiles file that contains paths to additional source files to include
- **excludefiles** (*str*) – path for excludefiles file that contains filename of source files to exclude from the build
- **sharedobject** (*bool*) – boolean indicating a shared object will be built
- **appdir** (*str*) – path for executable
- **verbose** (*bool*) – boolean indicating if output will be printed to the terminal
- **inplace** (*bool*) – boolean indicating that the source files in srcdir, srcdir2, and defined in extrafiles will be used directly. If inplace is False, source files will be copied to a directory named srcdir_temp. (default is False)
- **networkx** (*bool*) – boolean indicating that the NetworkX python package will be used to create the Directed Acyclic Graph (DAG) used to determine the order source files are compiled in. The NetworkX package tends to result in a unique DAG more often than the standard algorithm used in pymake. (default is False)
- **meson** (*bool*) – boolean indicating that the executable should be built using the meson build system. (default is False)
- **mesondir** (*str*) – Main meson.build file path

**Returns**
**returncode** – return code

**Return type**
int

**make_plots**(*srcdir*, *outdir*, *include_subdir=False*, *level=3*, *extension='.png'*, *verbose=False*, *networkx=False*)

Create plots of module dependencies.

**Parameters**

- **srcdir** (*str*) – path for source files
- **outdir** (*str*) – path for output images
- **include_subdir** (*bool*) – boolean indicating is subdirectories in the source file directory should be included
- **level** (*int*) – dependency level (1 is the minimum)
- **extension** (*str*) – output extension (default is .png)
- **verbose** (*bool*) – boolean indicating if output will be printed to the terminal

- **networkx** (*bool*) – boolean indicating that the NetworkX python package will be used to create the Directed Acyclic Graph (DAG) used to determine the order source files are compiled in. The NetworkX package tends to result in a unique DAG more often than the standard algorithm used in pymake. (default is False)

**parser**(*examples=None*)

Construct the parser and return argument values.

**Parameters**
**examples** (*str*) –

**Returns**
**args** – Namespace with command line arguments

**Return type**
Namespace object

**repo_latest_version**(*github_repo=None*, *verify=True*)

Return a string of the latest version number (tag) contained in a github repository release.

**Parameters**
**github_repo** (*str*) – Repository name, such as MODFLOW-USGS/modflow6. If github_repo is None set to 'MODFLOW-USGS/executables'

**Returns**
**version** – string with the latest version/tag number

**Return type**
str

**to_pydot**(*dag*, *filename='mygraph.png'*)

Create a png file of a Directed Acyclic Graph

**Parameters**

- **dag** (*object*) – directed acyclic graph
- **filename** (*str*) – path of the graph png

**class usgs_program_data**

Bases: `object`

USGS program database class.

**static export_json**(*fpth='code.json'*, *prog_data=None*, *current=False*, *update=True*, *write_markdown=False*, *verbose=False*)

Export USGS program data as a json file.

**Parameters**

- **fpth** (*str*) – Path for the json file to be created. Default is "code.json"
- **prog_data** (*dict*) – User-specified program database. If prog_data is None, it will be created from the USGS program database
- **current** (*bool*) – If False, all USGS program targets are listed. If True, only USGS program targets that are defined as current are listed. Default is False.
- **update** (*bool*) – If True, existing targets in the user-specified program database with values in the USGS program database. If False, existing targets in the user-specified program database will not be updated. Default is True.

- **write_markdown** (*bool*) – If True, write markdown file that includes the target name, version, and the last-modified date of the download asset (url). Default is False.
- **verbose** (*bool*) – boolean for verbose output to terminal

**static get_keys**(*current=False*)

Get target keys from the USGS program database.

**Parameters**
**current** (*bool*) – If False, all USGS program targets are listed. If True, only USGS program targets that are defined as current are listed. Default is False.

**Returns**
**keys** – list of USGS program targets

**Return type**
list

**static get_precision**(*key*)

Get the dictionary for a specified target.

**Parameters**
**key** (*str*) – Target USGS program

**Returns**
**precision** – List

**Return type**
list

**static get_program_dict**()

Get the complete USGS program database.

**Returns**
**program_dict** – Dictionary with USGS program attributes for all targets

**Return type**
dict

**static get_target**(*key*)

Get the dictionary for a specified target.

**Parameters**
**key** (*str*) – Target USGS program that may have a path and an extension

**Returns**
**program_dict** – Dictionary with USGS program attributes for the specified key

**Return type**
dict

**static get_version**(*key*)

Get the current version of the specified target.

**Parameters**
**key** (*str*) – Target USGS program

**Returns**
**version** – current version of the specified target

**Return type**
str

static list_json(*fpth='code.json'*)

List an existing code json file.

**Parameters**
**fpth** (*str*) – Path for the json file to be listed. Default is "code.json"

static list_targets(*current=False*)

Print a list of the available USGS program targets.

**Parameters**
**current** (*bool*) – If False, all USGS program targets are listed. If True, only USGS program targets that are defined as current are listed. Default is False.

static load_json(*fpth='code.json'*)

Load an existing code json file. Basic error checking is done to make sure the file contains the correct keys.

**Parameters**
**fpth** (*str*) – Path for the json file to be created. Default is "code.json"

**Returns**
**json_dict** – Valid USGS program database

**Return type**
dict

static update_json(*fpth='code.json'*, *temp_dict=None*)

UPDATE an existing code json file.

**Parameters**

- **fpth** (*str*) – Path for the json file to be listed. Default is "code.json"
- **temp_dict** (*dict*) – Dictionary with USGS program data for a target

zip_all(*path*, *file_pths=None*, *dir_pths=None*, *patterns=None*)

Compress all files in the user-provided list of file paths and directory paths that match the provided file patterns.

**Parameters**

- **path** (*str*) – path of the zip file that will be created
- **file_pths** (*str or list*) – file path or list of file paths to be compressed
- **dir_pths** (*str or list*) – directory path or list of directory paths to search for files that will be compressed
- **patterns** (*str or list*) – file pattern or list of file patterns s to match to when creating a list of files that will be compressed

CHAPTER

FIVE

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## p

# INDEX

## P



## R



## S



## T



## U



## Z