Davies Documentation

Release 0.1.2

Myotisoft

Jun 12, 2017

Contents

1 Davies **3**

2 Installation **5**

3 Name **7**

4 License **9**

5 API Documentation **11**
- 5.1 davies.compass . . . 11
- 5.2 davies.compass.plt . . . 14
- 5.3 davies.pockettopo . . . 14

6 Indices and Tables **17**

Python Module Index **19**

Contents:

CHAPTER 1

Davies

Davies is a Python package for manipulating cave survey data.

It is currently in a early phase of development, and its interfaces may change as it matures.

Current support includes:

- Reading of Compass Project (.MAK) and Data (.DAT) source files, as well as compiled Plot (.PLT) files.
- Writing Compass Data (.DAT) files.
- Reading PocketTopo exported .TXT survey files.
- That's it! No loop closure algorithms, no visualization or editing tools (though our *examples* directory contains scripts with which to build tools of this sort)

Browse the Davies API documentation.

Example usage with Compass survey data:

```
from davies import compass

# Parse a .DAT file
datfile = compass.DatFile.read('MYCAVE.DAT')

print len(datfile)  # number of surveys in .DAT
>> 17

print datfile.length  # total surveyed footage including splays, etc.
>> 5332.2

print datfile.included_length  # total surveyed footage after discarding excluded
↪shots
>> 5280.0

survey = datfile['BS']  # grab a survey by its survey designation

print survey.name
>> A
```

```
print survey.date
>> 2006-09-30

print survey.length  # surveyed footage including splays, etc.
>> 5332.2

print survey.included_length  # surveyed footage after discarding excluded shots
>> 5280.0

print survey.shots[0]
>> {'FROM': 'A1', 'TO': 'A2', 'LENGTH': 16.8, 'BEARING': 158.0, 'INC': -30.0, 'LEFT':␣
↪12.0, 'RIGHT': 15.0, 'UP': 15.0, 'DOWN': 20.0}

print survey.shots[0].azm  # azimuth after averaging front and backsights, magnetic␣
↪declination
155.2
```

This example shows who has surveyed the most footage in your Compass project:

```
from davies import compass

cavers = {}

for datfilename in sys.argv[1:]:
    for survey in compass.DatFile.read(datfilename):
        for member in survey.team:
            cavers[member] = cavers.get(member, 0.0) + survey.length

for name in sorted(cavers, key=cavers.get, reverse=True):
    print "%s:\t%0.1f" % (name, cavers[name])
```

CHAPTER 2

Installation

This software requires **Python 2.7**. It will *NOT* work with older Python releases, though it *may* work with Python 3.3+.

Releases are available for installation from the Python Package Index, see installation instructions or simply run the following command on Mac OS X or most Linux distributions:

```
$> pip install davies
```

If you've downloaded a source distribution or checked out from the git repository, install locally with:

```
$> python setup.py install
```

Development happens on GitHub.

CHAPTER 3

Name

The name “Davies” is a tribute to William E. Davies, who pioneered the systematic cave survey of West Virginia and authored *Caverns of West Virginia* in 1949. Bill Davies later did the statewide cave survey for the state of Maryland, served the roles of President and Vice-President of the National Speleological Society, and published the definitive US-wide karst map, *Engineering Aspects of Karst*. Davies still serves as an inspiration today to the cave mappers of West Virginia, of the United States, and the World over.

CHAPTER 4

License

Davies is Free / Open Source software licensed under the MIT License, and is copyright (C) 2013 - 2016 Myotisoft LLC.

CHAPTER 5

API Documentation

davies.compass

davies.compass: Module for parsing and working with Compass source files

class davies.compass.**Project**(*name=None*, *filename=None*)

Representation of a Compass .MAK Project file. A Project is a container for *DatFile* objects.

See: Compass Project File Format

Variables

- **name** – (string)
- **filename** – (string)
- **base_location** – (*UTMLocation*)
- **linked_files** – (list of *DatFile*)
- **fixed_stations** – (map of *DatFile* -> station -> *UTMLocation*)

add_linked_file(*datfile*)

Add a *DatFile* to linked_files.

add_linked_station(*datfile*, *station*, *location=None*)

Add a linked or fixed station

static read(*fname*)

Read a .MAK file and produce a *Project*

set_base_location(*location*)

Configure the project's base location

write(*outfilename=None*)

Write or overwrite this .MAK file

class davies.compass.**UTMLocation**(*easting*, *northing*, *elevation=0.0*, *zone=0*, *datum=None*, *convergence=0.0*)
Represents a UTM-based coordinate for fixed stations.

class davies.compass.**UTMDatum**
Enumeration of common geographic datums.

class davies.compass.**DatFile**(*name=None*, *filename=None*)
Representation of a Compass .DAT File. A DatFile is a container for *Survey* objects.

See: Compass Survey Data File Format

Variables

- **name** – (string) the DatFile's "name", not necessarily related to its filename
- **filename** – (string) underlying .DAT file's filename
- **surveys** – (list of *Survey*)

add_survey(*survey*)
Add a *Survey* to surveys.

excluded_length
Surveyed length which does not count toward the included total

included_length
Surveyed length, not including "excluded" shots

length
Total surveyed length.

static read(*fname*)
Read a .DAT file and produce a *Survey*

write(*outfname=None*)
Write or overwrite a *Survey* to the specified .DAT file

class davies.compass.**Survey**(*name=''*, *date=None*, *comment=''*, *team=''*, *declination=0.0*, *file_format=None*, *corrections=(0.0, 0.0, 0.0)*, *corrections2=(0.0, 0.0)*, *cave_name=''*, *shot_header=()*, *shots=None*)
Representation of a Compass Survey object. A Survey is a container for *Shot* objects.

See: Compass Survey Data File Format

Variables

- **file_format** – (str) format string which defines how Compass will view/edit underlying survey data; setting this property will in turn set all the other file format properties listed below; should be a string of 11 - 13 characters
- **bearing_units** – (chr) 'D'
- **length_units** – (chr)
- **passage_units** – (chr)
- **inclination_units** – (chr)
- **passage_dimension_order** – (list of chr)
- **shot_item_order** – (list of chr)
- **backsight** – (chr)
- **lrud_association** – (chr)

add_shot(*shot*)
Add a shot dictionary to `shots`, applying this survey's magnetic declination

excluded_length
Surveyed length which does not count toward the included total

included_length
Surveyed length, not including "excluded" shots

length
Total surveyed length, regardless of exclusion flags.

class davies.compass.**Shot**(**args*, ***kwargs*)
Representation of a single shot in a Compass Survey.

See: Compass Survey Data File Format

__init__(**args*, ***kwargs*)

Parameters

- **FROM** – (str) from station
- **TO** – (str) to station
- **BEARING** – (float) forward compass in **decimal degrees**
- **AZM2** – (float) back compass in **decimal degrees**
- **INC** – (float) forward inclination in **decimal degrees**
- **INC2** – (float) back inclination in **decimal degrees**
- **LENGTH** – (float) distance in **decimal feet**
- **FLAGS** – (collection of *Exclude*) shot exclusion flags
- **COMMENTS** – (str) text comments, up to 80 characters long
- **declination** – (float) magnetic declination in decimal degrees

Variables **declination** – (float) set or get magnetic declination adjustment

azm
Corrected azimuth, taking into account backsight, declination, and compass corrections.

flags
Shot exclusion flags as a *set*

inc
Corrected inclination, taking into account backsight and clino corrections.

length
Corrected distance, taking into account tape correction.

class davies.compass.**Exclude**
Shot flags

class davies.compass.**CompassProjectParser**(*projectfile*)
Parser for Compass .MAK project files.

__init__(*projectfile*)

Parameters **projectfile** – (string) filename

parse()
Parse our project file and return *Project* object or raise *ParseException*.

class davies.compass.**CompassDatParser**(*datfilename*)
Parser for Compass .DAT data files

__init__(*datfilename*)

Parameters **datfilename** – (string) filename

parse()
Parse our data file and return a *DatFile* or raise *ParseException*.

exception davies.compass.**ParseException**
Exception raised when parsing fails.

davies.compass.plt

davies.compass.plt: Module for parsing and working with Compass .PLT plot files

class davies.compass.plt.**Plot**(*name=None*)
Compass .PLT plot file. A Plot is a container for *Segment* objects.

add_fixed_point(*name*, *coordinate*)
Add a (Y, X, Z) tuple to fixed_points.

add_loop(*n*, *common_sta*, *from_sta*, *to_sta*, *stations*)
Add a loop tuple to loops.

add_segment(*segment*)
Add a *Segment* to segments.

set_bounds(*ymin*, *ymax*, *xmin*, *xmax*, *zmin*, *zmax*, *edist=None*)
Set Y,X,Z bounds for the plot.

class davies.compass.plt.**Segment**(*name=None*, *date=None*, *comment=None*)
Compass .PLT segment. A segment is a container for Command objects.

add_command(*command*)
Add a Command to commands.

set_bounds(*ymin*, *ymax*, *xmin*, *xmax*, *zmin*, *zmax*)
Set Y,X,Z bounds for the segment.

class davies.compass.plt.**MoveCommand**(*y*, *x*, *z*, *name*, *l*, *r*, *u*, *d*, *edist*, *flags=None*)
Compass .PLT plot command for moving the "plotting pen" to a specified Y,X,Z coordinate.

class davies.compass.plt.**DrawCommand**(*y*, *x*, *z*, *name*, *l*, *r*, *u*, *d*, *edist*, *flags=None*)
Compass .PLT plot command for drawing a line segment between two points.

class davies.compass.plt.**CompassPltParser**(*pltfilename*, *strict_mode=False*)
Parser for Compass .PLT plot files.

__init__(*pltfilename*, *strict_mode=False*)

Parameters **pltfilename** – string filename

parse()
Parse our .PLT file and return *Plot* object or raise ParseException.

davies.pockettopo

davies.pockettopo: Module for parsing and working with exported PocketTopo survey data

class davies.pockettopo.**TxtFile**(*name=None*, *length_units='m'*, *angle_units=360*)
Representation of a PocketTopo .TXT File. A TxtFile is a container for *Survey* objects.

Variables

- **name** – (string) the TxtFile's "name"
- **length_units** – (string) *m* (default) or *feet*
- **angle_units** – (int) *360* for degrees (default) or *400* for grads
- **surveys** – (list of *Survey*)
- **reference_points** – (dict of UTMLocation by station)

add_reference_point(*station*, *utm_location*)
Add a UTMLocation to reference_points.

add_survey(*survey*)
Add a *Survey* to surveys.

length
Total surveyed length.

static read(*fname*, *merge_duplicate_shots=False*, *encoding='windows-1252'*)
Read a PocketTopo .TXT file and produce a *TxtFile* object which represents it

class davies.pockettopo.**Survey**(*name=None*, *date=None*, *comment=None*, *declination=0.0*, *cave_name=None*, *length_units='m'*, *angle_units=360*, *shots=None*)
Representation of a PocketTopo Survey object. A Survey is a container for *Shot* objects.

add_shot(*shot*)
Add a Shot to shots, applying our survey's declination to it.

length
Total surveyed cave length, not including splays.

total_length
Total surveyed length including splays.

class davies.pockettopo.**MergingSurvey**(*name=None*, *date=None*, *comment=None*, *declination=0.0*, *cave_name=None*, *length_units='m'*, *angle_units=360*, *shots=None*)
Representation of a PocketTopo Survey object. A Survey is a container for *Shot* objects. This Survey implementation merges "duplicate" shots into a single averaged shot.

PocketTopo (and DistoX) convention is to use triple forward shots for mainline survey. When adding a new shot to this class with *add_shot()*, if we detect that the previous shot was between the same two stations, we average values and merge the two together instead of appending the duplicate shot. We use a "running" mean algorithm, so that this feature works for any number of subsequent duplicate shots (two, three, four...).

add_shot(*shot*)
Add a shot dictionary to shots, applying our survey's declination, and optionally averaging and merging with duplicate previous shot.

class davies.pockettopo.**Shot**(**args*, ***kwargs*)
Representation of a single shot in a PocketTopo Survey.

Parameters

- **FROM** – (str) from station
- **TO** – (str) optional to station

- **LENGTH** – (float) distance
- **AZM** – (float) compass
- **INC** – (float) inclination
- **COMMENT** – (str)
- **declination** – (float) optional

Variables **declination** – (float) set or get the applied magnetic declination for the shot

azm
Corrected azimuth, taking into account declination.

inc
Corrected inclination.

is_splay
Is this shot a "splay shot"?

length
Corrected distance.

class davies.pockettopo.**PocketTopoTxtParser**(*txtfilename*, *merge_duplicate_shots=False*, *encoding='windows-1252'*)

Parses the PocketTopo .TXT file format

parse()
Produce a *TxtFile* object from the .TXT file

CHAPTER 6

Indices and Tables

- genindex
- modindex
- search

Python Module Index

d

davies.compass, 11
davies.compass.plt, 14
davies.pockettopo, 14

Index

Symbols

__init__() (davies.compass.CompassDatParser method), 14
__init__() (davies.compass.CompassProjectParser method), 13
__init__() (davies.compass.Shot method), 13
__init__() (davies.compass.plt.CompassPltParser method), 14

A

add_command() (davies.compass.plt.Segment method), 14
add_fixed_point() (davies.compass.plt.Plot method), 14
add_linked_file() (davies.compass.Project method), 11
add_linked_station() (davies.compass.Project method), 11
add_loop() (davies.compass.plt.Plot method), 14
add_reference_point() (davies.pockettopo.TxtFile method), 15
add_segment() (davies.compass.plt.Plot method), 14
add_shot() (davies.compass.Survey method), 12
add_shot() (davies.pockettopo.MergingSurvey method), 15
add_shot() (davies.pockettopo.Survey method), 15
add_survey() (davies.compass.DatFile method), 12
add_survey() (davies.pockettopo.TxtFile method), 15
azm (davies.compass.Shot attribute), 13
azm (davies.pockettopo.Shot attribute), 16

C

CompassDatParser (class in davies.compass), 13
CompassPltParser (class in davies.compass.plt), 14
CompassProjectParser (class in davies.compass), 13

D

DatFile (class in davies.compass), 12
davies.compass (module), 11
davies.compass.plt (module), 14
davies.pockettopo (module), 14
DrawCommand (class in davies.compass.plt), 14

E

Exclude (class in davies.compass), 13
excluded_length (davies.compass.DatFile attribute), 12
excluded_length (davies.compass.Survey attribute), 13

F

flags (davies.compass.Shot attribute), 13

I

inc (davies.compass.Shot attribute), 13
inc (davies.pockettopo.Shot attribute), 16
included_length (davies.compass.DatFile attribute), 12
included_length (davies.compass.Survey attribute), 13
is_splay (davies.pockettopo.Shot attribute), 16

L

length (davies.compass.DatFile attribute), 12
length (davies.compass.Shot attribute), 13
length (davies.compass.Survey attribute), 13
length (davies.pockettopo.Shot attribute), 16
length (davies.pockettopo.Survey attribute), 15
length (davies.pockettopo.TxtFile attribute), 15

M

MergingSurvey (class in davies.pockettopo), 15
MoveCommand (class in davies.compass.plt), 14

P

parse() (davies.compass.CompassDatParser method), 14
parse() (davies.compass.CompassProjectParser method), 13
parse() (davies.compass.plt.CompassPltParser method), 14
parse() (davies.pockettopo.PocketTopoTxtParser method), 16
ParseException, 14
Plot (class in davies.compass.plt), 14

PocketTopoTxtParser (class in davies.pockettopo), 16
Project (class in davies.compass), 11

R

read() (davies.compass.DatFile static method), 12
read() (davies.compass.Project static method), 11
read() (davies.pockettopo.TxtFile static method), 15

S

Segment (class in davies.compass.plt), 14
set_base_location() (davies.compass.Project method), 11
set_bounds() (davies.compass.plt.Plot method), 14
set_bounds() (davies.compass.plt.Segment method), 14
Shot (class in davies.compass), 13
Shot (class in davies.pockettopo), 15
Survey (class in davies.compass), 12
Survey (class in davies.pockettopo), 15

T

total_length (davies.pockettopo.Survey attribute), 15
TxtFile (class in davies.pockettopo), 14

U

UTMDatum (class in davies.compass), 12
UTMLocation (class in davies.compass), 11

W

write() (davies.compass.DatFile method), 12
write() (davies.compass.Project method), 11