

PyWeather

API Documentation

March 27, 2006

Contents

Contents	1
1 Package weather	3
1.1 Modules	3
1.2 Variables	3
2 Package weather.services	4
2.1 Modules	4
3 Module weather.services.test	5
3.1 Functions	5
3.2 Variables	5
4 Module weather.services.wunderground	6
4.1 Class Publisher	6
4.1.1 Methods	6
4.1.2 Class Variables	6
5 Package weather.stations	7
5.1 Modules	7
6 Module weather.stations.davis	8
6.1 Class VantagePro	8
6.1.1 Methods	8
7 Module weather.stations.validate	9
7.1 Class Validator	9
7.1.1 Methods	9
8 Module weather.test_all	10
8.1 Functions	10
9 Package weather.units	11
9.1 Modules	11
10 Module weather.units.astro	12
10.1 Functions	12
10.2 Variables	12

CONTENTS	CONTENTS
11 Module weather.units.precip	13
11.1 Variables	13
12 Module weather.units.pressure	14
12.1 Functions	14
12.2 Variables	16
13 Module weather.units.temp	18
13.1 Functions	18
13.2 Variables	19
14 Module weather.units.test_pressure	20
14.1 Functions	20
14.2 Variables	20
14.3 Class TestCase	20
14.3.1 Methods	20
15 Module weather.units.test_temp	23
15.1 Functions	23
15.2 Variables	23
15.3 Class TestCase	23
15.3.1 Methods	23
16 Module weather.units.test_wind	25
16.1 Functions	25
16.2 Variables	25
16.3 Class TestCase	25
16.3.1 Methods	25
17 Module weather.units.wind	27
17.1 Functions	27
17.2 Variables	28
Index	29

a collection of meteorological conversion functions, station readers, and publishing modules

1.1 Modules

- **services** (*Section 2, p. 4*)
 - **test**: Demonstrates how to call the wunderground publisher. (*Section 3, p. 5*)
 - **wunderground**: WUnderground.com Publisher (*Section 4, p. 6*)
- **stations** (*Section 5, p. 7*)
 - **davis** (*Section 6, p. 8*)
 - **validate** (*Section 7, p. 9*)
- **test_all** (*Section 8, p. 10*)
- **units** (*Section 9, p. 11*)
 - **astro** (*Section 10, p. 12*)
 - **precip**: precipitation related conversions (*Section 11, p. 13*)
 - **pressure**: pressure related conversion functions (*Section 12, p. 14*)
 - **temp**: temperature related conversion functions (*Section 13, p. 18*)
 - **test_pressure**: Unit tests the pressure module. (*Section 14, p. 20*)
 - **test_temp**: Unit tests the temp module. (*Section 15, p. 23*)
 - **test_wind**: Unit tests the pressure module. (*Section 16, p. 25*)
 - **wind**: wind related conversion functions (*Section 17, p. 27*)

1.2 Variables

Name	Description
__version__	Value: '0.7.0'

Package weather.services

2.1 Modules

- **test:** Demonstrates how to call the wunderground publisher.
(Section 3, p. 5)
- **wunderground:** WUnderground.com Publisher
(Section 4, p. 6)

Demonstrates how to call the wunderground publisher. See the wunderground module for more information.

3.1 Functions

`main()`

`usage()`

`wunderground()`

3.2 Variables

Name	Description
<code>__author__</code>	Value: 'Christopher Blunck'
<code>__email__</code>	Value: 'chris@wxnet.org'
<code>__revision__</code>	Value: '\$Revision: 1.6 \$'
<code>__usage__</code>	Value: '\n python \$0\n'

WUnderground.com Publisher

Abstract: The class contained within this module allows python programs to publish weather conditions to the wunderground.com servers. That is, this class encapsulates the wire protocol wunderground.com supports and allows application developers to insulate themselves against changes in the wunderground.com wire protocol.

If the `rtfreq` parameter is passed to the `Publisher` constructor, posting of the current conditions will go to the "real time updater" service provided by wunderground.com. The `rtfreq` optional parameter passed to the constructor is a float that represents the number of seconds between observations.

```
Usage: >>> publisher = Publisher() >>> publisher.set(30.12, 28.52, 53.0, 44.6, 0.0, time.gmtime(), 0, 0, 0)
>>> response = publisher.publish('MyUserName', 'MyPassword') >>> print '%s: %s' % (response.status, response.reason)
```

Notes on arguments to `Publisher.set()`: <float> pressure: in inches of Hg <float> dewpt: in Fahrenheit <float> humidity: between 0.0 and 100.0 inclusive <float> tempf: in Fahrenheit <time tuple> dateutc: 9 value time tuple in UTC (e.g. `time.gmtime()`) <float> windgust: in mph <float> winddir: in degrees, between 0.0 and 100.0 <string> clouds: unknown at this time (email me if you know!) <string> weather: unknown at this time (email me if you know!)

Developers Notes: It appears that even if you provide an invalid username and password, a status of 200, and a reason of "OK" is returned.

Author: Christopher Blunck (chris@wxnet.org) Date: 2006-03-27

4.1 Class Publisher

Publishes weather data to the wunderground.com servers. See module documentation for additional information and usage idioms.

4.1.1 Methods

<code>__init__(self, rtfreq=None)</code>
<code>publish(self, username, password, debug=False)</code>
<code>set(self, pressure, dewpoint, humidity, tempf, rainin, dateutc, windgust, windspeed, winddir, clouds='NA', weather='NA')</code>

4.1.2 Class Variables

Name	Description
<code>REALTIME_SERVER</code>	Value: <code>'rtupdate.wunderground.com'</code>
<code>STD_SERVER</code>	Value: <code>'weatherstation.wunderground.com'</code>
<code>URI</code>	Value: <code>'/weatherstation/updateweatherstation.php'</code>

Package `weather.stations`

5.1 Modules

- `davis` (*Section 6, p. 8*)
- `validate` (*Section 7, p. 9*)

Module `weather.stations.davis`

6.1 Class `VantagePro`

A class capable of reading raw (binary) weather data from a vantage pro console and parsing it into usable scalar (integer/long/real) values.

The data read from the console is in binary format, and must be converted to hex using a least-ordered nybble strategy. The hex is in fixed-length format, and values can be extracted using an offset and length strategy (e.g. outside humidity starts at position 56 and is 2 bytes long).

6.1.1 Methods

<code>__init__(self, device, start='c4f4f4')</code>
--

<code>get_field(self, start, len)</code>

returns the value in the field specified by the starting offset and of the length provided. this value is computed by first reversing the hex data, followed by conversion of the hex data to a long.

<code>parse(self)</code>

read and parse a set of data read from the console. after the data is parsed it is available in the fields variable.
--

Module `weather.stations.validate`

7.1 Class Validator

7.1.1 Methods

```
__init__(self, fields)
```

```
get_value(self, field, default)
```

```
validate(self)
```

8 Module weather.test_all

8.1 Functions

```
main()
```

9.1 Modules

- **astro** (*Section 10, p. 12*)
- **precip**: precipitation related conversions
(*Section 11, p. 13*)
- **pressure**: pressure related conversion functions
(*Section 12, p. 14*)
- **temp**: temperature related conversion functions
(*Section 13, p. 18*)
- **test_pressure**: Unit tests the pressure module.
(*Section 14, p. 20*)
- **test_temp**: Unit tests the temp module.
(*Section 15, p. 23*)
- **test_wind**: Unit tests the pressure module.
(*Section 16, p. 25*)
- **wind**: wind related conversion functions
(*Section 17, p. 27*)

10.1 Functions

`daylight(lat, long, tz, day, month, year)`

`degrees_to_radians(degrees)`

`radians_to_degrees(radians)`

10.2 Variables

Name	Description
zenith	Value: -0.014540000000000001

precipitation related conversions

11.1 Variables

Name	Description
__author__	Value: 'Christopher Blunck'
__email__	Value: 'chris@wxnet.org'
__revision__	Value: '\$Revision: 1.6 \$'
__usage__	Value: ''

12 Module weather.units.pressure

pressure related conversion functions

12.1 Functions

atm_to_in32(*atm*)

Atmospheres (atm) to inches of mercury @32F (inHg32)

atm_to_in60(*atm*)

Atmospheres (atm) to inches of mercury @60F (inHg60)

atm_to_lb_sqin(*atm*)

Atmospheres (atm) to pounds/square inch (lb/in**2)

atm_to_mb(*atm*)

Atmospheres (atm) to millibars (mb)

atm_to_pa(*atm*)

Atmospheres (atm) to pascals (Pa)

hpa_to_inches(*hpa*)

hpa_to_mb(*hpa*)

Hectopascals (hPa) to millibars (mb)

in32_to_atm(*inches*)

Inches of mercury @32F (inHg32) to millibars (mb)

in32_to_lbs(*inches*)

Inches of mercury @32F (inHg32) to pounds/square inch (lb/in**2)

in32_to_mb(*inches*)

Inches of mercury @32F (inHg32) to millibars (mb)

in60_to_atm(*inches*)

Inches of mercury @60F (inHg60) to millibars (mb)

<i>Module weather.units.pressure</i>	<i>Functions</i>
in60_to_lbs (<i>inches</i>)	Inches of mercury @60F (inHg60) to pounds/square inch (lb/in**2)
in60_to_mb (<i>inches</i>)	Inches of mercury @60F (inHg60) to atmospheres (atm)
kpa_to_mb (<i>hpa</i>)	Kilopascals (kPa) to millibars (mb)
lb_sqft_to_mb (<i>lbs</i>)	Pounds/square foot (lb/ft**2) to millibars (mb)
lb_sqin_to_atm (<i>lbs</i>)	Pounds/square inch (lb/in**2) to atmospheres (atm)
lb_sqin_to_mb (<i>lbs</i>)	Pounds/square inch (lb/in**2) to millibars (mb)
lb_sqin_to_mm32 (<i>lbs</i>)	Pounds/square inch (lb/in**2) to inches of mercury @32F (inHg32)
lb_sqin_to_mm60 (<i>lbs</i>)	Pounds/square inch (lb/in**2) to inches of mercury @60F (inHg60)
mb_to_atm (<i>mb</i>)	Millibars (mb) to atmospheres (atm)
mb_to_hpa (<i>mb</i>)	Millibars (mb) to hectopascals (hPa)
mb_to_in32 (<i>mb</i>)	Millibars (mb) to inches of mercury @32F (inHg60)
mb_to_in60 (<i>mb</i>)	Millibars (mb) to inches of mercury @60F (inHg60)
mb_to_kpa (<i>mb</i>)	Millibars (mb) to kilopascals (kPa)

Module <code>weather.units.pressure</code>	Variables
<code>mb_to_lb_sqft</code> (<i>mb</i>)	Millibars (mb) to pounds/square foot (lb/ft**2)
<code>mb_to_lb_sqin</code> (<i>mb</i>)	Millibars (mb) to pounds/square inch (lb/in**2)
<code>mb_to_mm32</code> (<i>mb</i>)	Millibars (mb) to millimeters of mercury @32F (mmHg)
<code>mb_to_mm60</code> (<i>mb</i>)	Millibars (mb) to millimeters of mercury @60F (mmHg)
<code>mb_to_n_sqm</code> (<i>mb</i>)	Millibars (mb) to newtons/square meter (N/m**2)
<code>mb_to_pa</code> (<i>mb</i>)	Millibars (mb) to pascals (Pa)
<code>mm32_to_mb</code> (<i>mm32</i>)	Millimeters of mercury @32F (mmHg) to millibars (mb)
<code>mm60_to_mb</code> (<i>mm60</i>)	Millimeters of mercury @60F (mmHg) to millibars (mb)
<code>n_sqm_to_mb</code> (<i>nsqm</i>)	Newtons/square meter (N/m**2) to millibars (mb)
<code>pa_to_atm</code> (<i>pa</i>)	Pascals (Pa) to atmospheres (atm)
<code>pa_to_mb</code> (<i>pa</i>)	Pascals (Pa) to millibars (mb)

12.2 Variables

Name	Description
<code>__author__</code>	Value: 'Christopher Blunck'
<code>__email__</code>	Value: 'chris@wxnet.org'

continued on next page

<i>Module weather</i>	<i>Dynamic</i>	<i>pressure</i>	Description	<i>Variables</i>
<code>--revision--</code>	Value:	<code>'\$Revision: 1.6 \$'</code>		
<code>--usage--</code>	Value:	<code>'this module should not be run via the command line'</code>		

13 Module `weather.units.temp`

temperature related conversion functions

13.1 Functions

`calc_dewpoint`(*temp*, *hum*)

calculates the dewpoint via the formula from weatherwise.org return the dewpoint in degrees F.

`calc_heat_index`(*temp*, *hum*)

calculates the heat index based upon temperature (in F) and humidity.
http://www.srh.noaa.gov/bmx/tables/heat_index.html
returns the heat index in degrees F.

`calc_humidity`(*temp*, *dewpoint*)

calculates the humidity via the formula from weatherwise.org return the relative humidity

`calc_wind_chill`(*t*, *windspeed*, *windspeed10min=*None)

calculates the wind chill value based upon the temperature (F) and wind.
returns the wind chill in degrees F.

`celsius_to_fahrenheit`(*c*)

Degrees Celsius (C) to degrees Fahrenheit (F)

`celsius_to_kelvin`(*c*)

Degrees Celsius (C) to degrees Kelvin (K)

`celsius_to_rankine`(*c*)

Degrees Celsius (C) to degrees Rankine (R)

`fahrenheit_to_celsius`(*f*)

Degrees Fahrenheit (F) to degrees Celsius (C)

`fahrenheit_to_kelvin`(*f*)

Degrees Fahrenheit (F) to degrees Kelvin (K)

`fahrenheit_to_rankine`(*f*)

Degrees Fahrenheit (F) to degrees Rankine (R)

Module <code>weather.units.temp</code>	Variables
<code>kelvin_to_celsius(<i>k</i>)</code>	Degrees Kelvin (K) to degrees Celsius (C)
<code>kelvin_to_fahrenheit(<i>k</i>)</code>	Degrees Kelvin (K) to degrees Fahrenheit (F)
<code>kelvin_to_rankine(<i>k</i>)</code>	Degrees Kelvin (K) to degrees Rankine (R)
<code>rankine_to_celsius(<i>r</i>)</code>	Degrees Rankine (R) to degrees Celsius (C)
<code>rankine_to_fahrenheit(<i>r</i>)</code>	Degrees Rankine (R) to degrees Fahrenheit (F)
<code>rankine_to_kelvin(<i>r</i>)</code>	Degrees Rankine (R) to degrees Kelvin (K)

13.2 Variables

Name	Description
<code>__author__</code>	Value: 'Christopher Blunck'
<code>__email__</code>	Value: 'chris@wxnet.org'
<code>__revision__</code>	Value: '\$Revision: 1.6 \$'
<code>__usage__</code>	Value: 'this module should not be run via the command line'

Unit tests the pressure module.

14.1 Functions

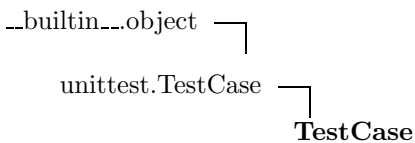
`main()`

`usage()`

14.2 Variables

Name	Description
<code>__author__</code>	Value: 'Christopher Blunck'
<code>__email__</code>	Value: 'chris@wxnet.org'
<code>__revision__</code>	Value: '\$Revision: 1.6 \$'
<code>__usage__</code>	Value: '\n python \$0\n'

14.3 Class `TestCase`



14.3.1 Methods

`setUp(self)`
 Hook method for setting up the test fixture before exercising it.
 Overrides: `unittest.TestCase.setUp` `exitit`(inherited documentation)

`tearDown(self)`
 Hook method for deconstructing the test fixture after testing it.
 Overrides: `unittest.TestCase.tearDown` `exitit`(inherited documentation)

`test_atm_to_in32(self)`

`test_atm_to_in60(self)`

`test_atm_to_lb_sqin(self)`

`test_atm_to_mb(self)`

`test_atm_to_pa(self)``test_hpa_to_mb(self)``test_in32_to_atm(self)``test_in32_to_lbs(self)``test_in32_to_mb(self)``test_in60_to_atm(self)``test_in60_to_lbs(self)``test_in60_to_mb(self)``test_kpa_to_mb(self)``test_lb_sqft_to_mb(self)``test_lb_sqin_to_atm(self)``test_lb_sqin_to_mb(self)``test_lb_sqin_to_mm32(self)``test_lb_sqin_to_mm60(self)``test_mb_to_atm(self)``test_mb_to_hpa(self)``test_mb_to_in32(self)``test_mb_to_in60(self)``test_mb_to_kpa(self)``test_mb_to_lb_sqft(self)``test_mb_to_lb_sqin(self)``test_mb_to_mm32(self)``test_mb_to_mm60(self)`

Module `weather.units.test.pressure` Class `TestCase`

`test__mb_to_n_sqm(self)`

`test__mb_to_pa(self)`

`test__mm32_to_mb(self)`

`test__mm60_to_mb(self)`

`test__n_sqm_to_mb(self)`

`test__pa_to_atm(self)`

`test__pa_to_mb(self)`

Inherited from object: `__delattr__`, `__getattr__`, `__hash__`, `__reduce__`, `__reduce_ex__`, `__setattr__`

Inherited from type: `__new__`

Inherited from `TestCase`: `__init__`, `__call__`, `__repr__`, `__str__`, `assert_`, `assertAlmostEqual`, `assertAlmostEquals`, `assertEqual`, `assertEquals`, `assertNotAlmostEqual`, `assertNotAlmostEquals`, `assertNotEqual`, `assertNotEquals`, `assertRaises`, `countTestCases`, `debug`, `defaultTestResult`, `fail`, `failIf`, `failIfAlmostEqual`, `failIfEquals`, `failUnless`, `failUnlessAlmostEqual`, `failUnlessEqual`, `failUnlessRaises`, `id`, `run`, `shortDescription`

Unit tests the temp module.

15.1 Functions

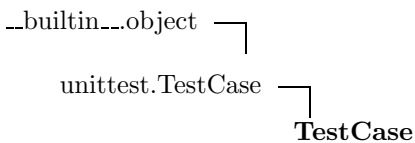
`main()`

`usage()`

15.2 Variables

Name	Description
<code>__author__</code>	Value: 'Christopher Blunck'
<code>__email__</code>	Value: 'chris@wxnet.org'
<code>__revision__</code>	Value: '\$Revision: 1.6 \$'
<code>__usage__</code>	Value: '\n python \$0\n'

15.3 Class `TestCase`



15.3.1 Methods

`setUp(self)`
 Hook method for setting up the test fixture before exercising it.
 Overrides: `unittest.TestCase.setUp` `exitit`(inherited documentation)

`tearDown(self)`
 Hook method for deconstructing the test fixture after testing it.
 Overrides: `unittest.TestCase.tearDown` `exitit`(inherited documentation)

`test__calc_heat_index(self)`

`test__calc_wind_chill(self)`

`test__celsius_to_fahrenheit(self)`

`test__celsius_to_kelvin(self)`

Module `weather.units.test.temp` Class `TestCase`

`test__celsius_to_rankine(self)`

`test__dewpoint(self)`

`test__fahrenheit_to_celsius(self)`

`test__fahrenheit_to_kelvin(self)`

`test__fahrenheit_to_rankine(self)`

`test__humidity(self)`

`test__kelvin_to_celsius(self)`

`test__kelvin_to_fahrenheit(self)`

`test__kelvin_to_rankine(self)`

`test__rankine_to_celsius(self)`

`test__rankine_to_fahrenheit(self)`

`test__rankine_to_kelvin(self)`

Inherited from object: `__delattr__`, `__getattr__`, `__hash__`, `__reduce__`, `__reduce_ex__`, `__setattr__`

Inherited from type: `__new__`

Inherited from `TestCase`: `__init__`, `__call__`, `__repr__`, `__str__`, `assert_`, `assertAlmostEqual`, `assertAlmostEquals`, `assertEqual`, `assertEquals`, `assertNotAlmostEqual`, `assertNotAlmostEquals`, `assertNotEqual`, `assertNotEquals`, `assertRaises`, `countTestCases`, `debug`, `defaultTestResult`, `fail`, `failIf`, `failIfAlmostEqual`, `failIfEquals`, `failUnless`, `failUnlessAlmostEqual`, `failUnlessEqual`, `failUnlessRaises`, `id`, `run`, `shortDescription`

Unit tests the pressure module.

16.1 Functions

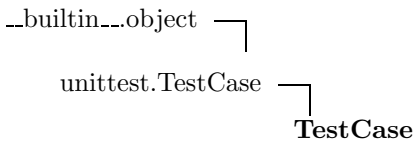
`main()`

`usage()`

16.2 Variables

Name	Description
<code>__author__</code>	Value: 'Christopher Blunck'
<code>__email__</code>	Value: 'chris@wxnet.org'
<code>__revision__</code>	Value: '\$Revision: 1.6 \$'
<code>__usage__</code>	Value: '\n python \$0\n'

16.3 Class `TestCase`



16.3.1 Methods

`setUp(self)`
 Hook method for setting up the test fixture before exercising it.
 Overrides: `unittest.TestCase.setUp` `exitit`(inherited documentation)

`tearDown(self)`
 Hook method for deconstructing the test fixture after testing it.
 Overrides: `unittest.TestCase.tearDown` `exitit`(inherited documentation)

`test_ft_min_to_mph(self)`

`test_ft_sec_to_knots(self)`

`test_ft_sec_to_mph(self)`

`test_km_hr_to_knots(self)`

`test__knots_to_ft_sec(self)``test__knots_to_km_hr(self)``test__knots_to_m_sec(self)``test__knots_to_mph(self)``test__knots_to_nmph(self)``test__m_sec_to_knots(self)``test__m_sec_to_mph(self)``test__mph_to_ft_min(self)``test__mph_to_ft_sec(self)``test__mph_to_km_hr(self)``test__mph_to_knots(self)``test__mph_to_m_sec(self)``test__nmph_to_knots(self)`**Inherited from object:** `__delattr__`, `__getattr__`, `__hash__`, `__reduce__`, `__reduce_ex__`, `__setattr__`**Inherited from type:** `__new__`**Inherited from `TestCase`:** `__init__`, `__call__`, `__repr__`, `__str__`, `assert_`, `assertAlmostEqual`, `assertAlmostEquals`, `assertEqual`, `assertEquals`, `assertNotAlmostEqual`, `assertNotAlmostEquals`, `assertNotEqual`, `assertNotEquals`, `assertRaises`, `countTestCases`, `debug`, `defaultTestResult`, `fail`, `failIf`, `failIfAlmostEqual`, `failIfEquals`, `failUnless`, `failUnlessAlmostEqual`, `failUnlessEqual`, `failUnlessRaises`, `id`, `run`, `shortDescription`

17 Module weather.units.wind

wind related conversion functions

17.1 Functions

ft_min_to_mph(*ft*)

Feet/minute (ft/min) to miles/hour (mph)

ft_sec_to_knots(*ft*)

Feet/second (ft/s) to knots (kt)

ft_sec_to_mph(*ft*)

Feet/second (ft/s) to miles/hour (mph)

km_hr_to_knots(*km*)

Kilometers/hour (kph) to knots (kt)

km_hr_to_mph(*km*)

Kilometers/hour (kph) to miles/hour (mph)

knots_to_ft_sec(*kts*)

Knots (kt) to feet/second (ft/s)

knots_to_km_hr(*kts*)

Knots (kt) to kilometers/hour (kph)

knots_to_m_sec(*kts*)

Knots (kt) to meters/second (m/s)

knots_to_mph(*kts*)

Knots (kt) to miles/hour (mph)

knots_to_nmph(*kts*)

Knots (kt) to nautical miles/hour (nmph)

m_sec_to_knots(*m*)

Meters/second (m/s) to knots (kt)

Module <code>weather.units.wind</code>	Variables
<code>m_sec_to_mph(<i>m</i>)</code>	Meters/second (m/s) to miles/hour (mph)
<code>mph_to_ft_min(<i>mph</i>)</code>	Miles/hour (mph) to feet/minute (ft/min)
<code>mph_to_ft_sec(<i>mph</i>)</code>	Miles/hour (mph) to feet/second (ft/s)
<code>mph_to_km_hr(<i>mph</i>)</code>	Miles/hour (mph) to kilometers/hour (kph)
<code>mph_to_knots(<i>mph</i>)</code>	Miles/hour (mph) to knots (kt)
<code>mph_to_m_sec(<i>mph</i>)</code>	Miles/hour (mph) to meters/second (m/s)
<code>nmph_to_knots(<i>nmph</i>)</code>	Nautical miles/hour (nmph) to knots (kt)

17.2 Variables

Name	Description
<code>__author__</code>	Value: 'Christopher Blunck'
<code>__email__</code>	Value: 'chris@wxnet.org'
<code>__revision__</code>	Value: '\$Revision: 1.6 \$'
<code>__usage__</code>	Value: 'this module should not be run via the command line'

Index

- weather (*package*), 2
- weather.services (*package*), 3
- weather.services.test (*module*), 4
 - main (*function*), 4
 - usage (*function*), 4
 - wunderground (*function*), 4
- weather.services.wunderground (*module*), 5
 - Publisher (*class*), 5
 - __init__ (*method*), 5
 - publish (*method*), 5
 - set (*method*), 5
- weather.stations (*package*), 6
- weather.stations.davis (*module*), 7
 - VantagePro (*class*), 7
 - __init__ (*method*), 7
 - get_field (*method*), 7
 - parse (*method*), 7
- weather.stations.validate (*module*), 8
 - Validator (*class*), 8
 - __init__ (*method*), 8
 - get_value (*method*), 8
 - validate (*method*), 8
- weather.test_all (*module*), 9
 - main (*function*), 9
- weather.units (*package*), 10
- weather.units.astro (*module*), 11
 - daylight (*function*), 11
 - degrees_to_radians (*function*), 11
 - radians_to_degrees (*function*), 11
- weather.units.precip (*module*), 12
- weather.units.pressure (*module*), 13–16
 - atm_to_in32 (*function*), 13
 - atm_to_in60 (*function*), 13
 - atm_to_lb_sqin (*function*), 13
 - atm_to_mb (*function*), 13
 - atm_to_pa (*function*), 13
 - hpa_to_inches (*function*), 13
 - hpa_to_mb (*function*), 13
 - in32_to_atm (*function*), 13
 - in32_to_lbs (*function*), 13
 - in32_to_mb (*function*), 13
 - in60_to_atm (*function*), 13
 - in60_to_lbs (*function*), 13
 - in60_to_mb (*function*), 14
 - kpa_to_mb (*function*), 14
 - lb_sqft_to_mb (*function*), 14
 - lb_sqin_to_atm (*function*), 14
 - lb_sqin_to_mb (*function*), 14
 - lb_sqin_to_mm32 (*function*), 14
 - lb_sqin_to_mm60 (*function*), 14
 - mb_to_atm (*function*), 14
 - mb_to_hpa (*function*), 14
 - mb_to_in32 (*function*), 14
 - mb_to_in60 (*function*), 14
 - mb_to_kpa (*function*), 14
 - mb_to_lb_sqft (*function*), 14
 - mb_to_lb_sqin (*function*), 15
 - mb_to_mm32 (*function*), 15
 - mb_to_mm60 (*function*), 15
 - mb_to_n_sqm (*function*), 15
 - mb_to_pa (*function*), 15
 - mm32_to_mb (*function*), 15
 - mm60_to_mb (*function*), 15
 - n_sqm_to_mb (*function*), 15
 - pa_to_atm (*function*), 15
 - pa_to_mb (*function*), 15
- weather.units.temp (*module*), 17–18
 - calc_dewpoint (*function*), 17
 - calc_heat_index (*function*), 17
 - calc_humidity (*function*), 17
 - calc_wind_chill (*function*), 17
 - celsius_to_fahrenheit (*function*), 17
 - celsius_to_kelvin (*function*), 17
 - celsius_to_rankine (*function*), 17
 - fahrenheit_to_celsius (*function*), 17
 - fahrenheit_to_kelvin (*function*), 17
 - fahrenheit_to_rankine (*function*), 17
 - kelvin_to_celsius (*function*), 17
 - kelvin_to_fahrenheit (*function*), 18
 - kelvin_to_rankine (*function*), 18
 - rankine_to_celsius (*function*), 18
 - rankine_to_fahrenheit (*function*), 18
 - rankine_to_kelvin (*function*), 18
- weather.units.test_pressure (*module*), 19–21
 - main (*function*), 19
 - TestCase (*class*), 19–21
 - setUp (*method*), 19
 - tearDown (*method*), 19
 - test_atm_to_in32 (*method*), 19
 - test_atm_to_in60 (*method*), 19
 - test_atm_to_lb_sqin (*method*), 19
 - test_atm_to_mb (*method*), 19
 - test_atm_to_pa (*method*), 19
 - test_hpa_to_mb (*method*), 20
 - test_in32_to_atm (*method*), 20
 - test_in32_to_lbs (*method*), 20
 - test_in32_to_mb (*method*), 20
 - test_in60_to_atm (*method*), 20
 - test_in60_to_lbs (*method*), 20
 - test_in60_to_mb (*method*), 20

-
- test_kpa_to_mb (*method*), 20
 - test_lb_sqft_to_mb (*method*), 20
 - test_lb_sqin_to_atm (*method*), 20
 - test_lb_sqin_to_mb (*method*), 20
 - test_lb_sqin_to_mm32 (*method*), 20
 - test_lb_sqin_to_mm60 (*method*), 20
 - test_mb_to_atm (*method*), 20
 - test_mb_to_hpa (*method*), 20
 - test_mb_to_in32 (*method*), 20
 - test_mb_to_in60 (*method*), 20
 - test_mb_to_kpa (*method*), 20
 - test_mb_to_lb_sqft (*method*), 20
 - test_mb_to_lb_sqin (*method*), 20
 - test_mb_to_mm32 (*method*), 20
 - test_mb_to_mm60 (*method*), 20
 - test_mb_to_n_sqm (*method*), 20
 - test_mb_to_pa (*method*), 21
 - test_mm32_to_mb (*method*), 21
 - test_mm60_to_mb (*method*), 21
 - test_n_sqm_to_mb (*method*), 21
 - test_pa_to_atm (*method*), 21
 - test_pa_to_mb (*method*), 21
 - usage (*function*), 19
 - weather.units.test_temp (*module*), 22–23
 - main (*function*), 22
 - TestCase (*class*), 22–23
 - setUp (*method*), 22
 - tearDown (*method*), 22
 - test_calc_heat_index (*method*), 22
 - test_calc_wind_chill (*method*), 22
 - test_celsius_to_fahrenheit (*method*), 22
 - test_celsius_to_kelvin (*method*), 22
 - test_celsius_to_rankine (*method*), 22
 - test_dewpoint (*method*), 23
 - test_fahrenheit_to_celsius (*method*), 23
 - test_fahrenheit_to_kelvin (*method*), 23
 - test_fahrenheit_to_rankine (*method*), 23
 - test_humidity (*method*), 23
 - test_kelvin_to_celsius (*method*), 23
 - test_kelvin_to_fahrenheit (*method*), 23
 - test_kelvin_to_rankine (*method*), 23
 - test_rankine_to_celsius (*method*), 23
 - test_rankine_to_fahrenheit (*method*), 23
 - test_rankine_to_kelvin (*method*), 23
 - usage (*function*), 22
 - weather.units.test_wind (*module*), 24–25
 - main (*function*), 24
 - TestCase (*class*), 24–25
 - setUp (*method*), 24
 - tearDown (*method*), 24
 - test_ft_min_to_mph (*method*), 24
 - test_ft_sec_to_knots (*method*), 24
 - test_ft_sec_to_mph (*method*), 24
 - test_km_hr_to_knots (*method*), 24
 - test_km_hr_to_mph (*method*), 24
 - test_knots_to_ft_sec (*method*), 25
 - test_knots_to_km_hr (*method*), 25
 - test_knots_to_m_sec (*method*), 25
 - test_knots_to_mph (*method*), 25
 - test_knots_to_nmph (*method*), 25
 - test_m_sec_to_knots (*method*), 25
 - test_m_sec_to_mph (*method*), 25
 - test_mph_to_ft_min (*method*), 25
 - test_mph_to_ft_sec (*method*), 25
 - test_mph_to_km_hr (*method*), 25
 - test_mph_to_knots (*method*), 25
 - test_mph_to_m_sec (*method*), 25
 - test_nmph_to_knots (*method*), 25
 - usage (*function*), 24
 - weather.units.wind (*module*), 26–27
 - ft_min_to_mph (*function*), 26
 - ft_sec_to_knots (*function*), 26
 - ft_sec_to_mph (*function*), 26
 - km_hr_to_knots (*function*), 26
 - km_hr_to_mph (*function*), 26
 - knots_to_ft_sec (*function*), 26
 - knots_to_km_hr (*function*), 26
 - knots_to_m_sec (*function*), 26
 - knots_to_mph (*function*), 26
 - knots_to_nmph (*function*), 26
 - m_sec_to_knots (*function*), 26
 - m_sec_to_mph (*function*), 26
 - mph_to_ft_min (*function*), 27
 - mph_to_ft_sec (*function*), 27
 - mph_to_km_hr (*function*), 27
 - mph_to_knots (*function*), 27
 - mph_to_m_sec (*function*), 27
 - nmph_to_knots (*function*), 27