datapoint-python Documentation

Release v0.9.5+0.g0fc8c26.dirty

Jacob Tomlinson, Emlyn Price

Jan 16, 2020

Contents

CHAPTER 1

Installation

DataPoint for Python can be installed like any other Python module.

It is available on PyPI and the source is available on GitHub.

1.1 Pip

Pip makes Python package installation simple. For the latest stable version just fire up your terminal and run:

```
pip install datapoint
```

or for the very latest code from the repository's master branch run:

```
pip install git+git://github.com/ejep/datapoint-python.git@master
```

and to upgrade it in the future:

```
pip install git+git://github.com/ejep/datapoint-python.git@master --upgrade
```

1.2 Easy Install

Or if you really feel the need then you can use easy_install.

```
easy_install datapoint
```

But you probably shouldn't.

1.3 Source

You can also install from the source in GitHub.

First checkout the GitHub repository (or you can download the zip and extract it).

```
git clone https://github.com/ejep/datapoint-python.git datapoint-python
```

Navigate to that directory

```
cd datapoint-python
```

Then run the setup

```
python setup.py install
```

1.4 Windows

- Install python - you can see supported versions in the readme
- Install the appropriate setuptools python extension for your machine
- Install the appropriate pip python extension for your machine
- Add pip to your environment variables:
  - Run **Start > Edit the environment variables for your account**
  - Create a new variable:
  - **name** pip
  - **value** the path to **pip.exe** (this should be something like `C:\Python27\Scripts`)
- From the command line run **pip install datapoint**

CHAPTER 2

Getting started

Getting started with DataPoint for Python is simple and you can write a simple script which prints out data in just 6 lines of Python.

2.1 API Key

To access DataPoint you need to register with the Met Office and get yourself an API key. The process is simple and just ensures that you don't abuse the service.

2.2 Connecting to DataPoint

Now that we have an API key we can import the module:

```
import datapoint
```

And create a connection to DataPoint:

```
conn = datapoint.connection(api_key="aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee")
```

This creates a Manager Object which manages our connection and interacts with DataPoint for us, we'll discuss Manager Objects in depth later but for now you just need to know that it looks after your API key and has a load of methods to return data from DataPoint.

2.3 Getting data from DataPoint

So now that we have our Manager Object with a connection to DataPoint we can request some data. Our goal is to request some forecast data but first we need to know the site ID for the location we want data for. Luckily the Manager Object has a method to return a Site Object, which contains the ID among other things, from a specified latitude and longitude.

We can simply request a Site Object like so:

```
site = conn.get_nearest_forecast_site(51.500728, -0.124626)
```

For now we're just going to use this object to get us our forecast but you'll find more information about what the Site Object contains later.

Let's call another of the Manager Object's methods to give us a Forecast Object for our site:

```
forecast = conn.get_forecast_for_site(site.id, "3hourly")
```

We've given this method two parameters, the site ID for the forecast we want and also a forecast type of "3hourly". We'll discuss the forecast types later on.

This Forecast Object which has been returned to us contains lots of information which we will cover in a later section, right now we're just going to get the Timestep Object which represents right this minute:

```
current_timestep = forecast.now()
```

This Timestep Object contains many different details about the weather but for now we'll just print out the weather text.

```
print current_timestep.weather.text
```

And there you have it. If you followed all the steps you should have printed out the current weather for your chosen location.

2.4 Further Examples

For more code examples please have a look in the examples folder in the GitHub project.

CHAPTER 3

Locations

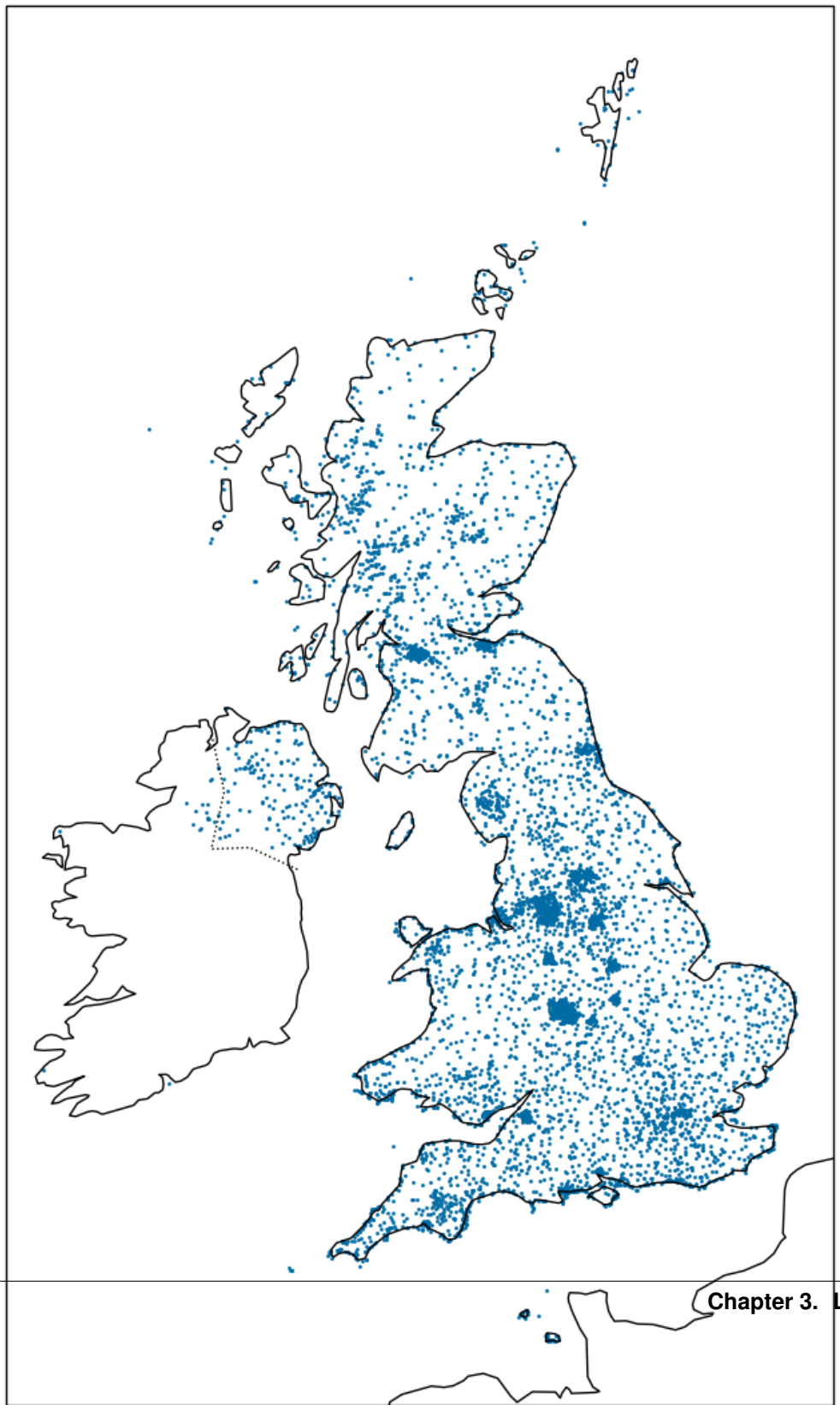
There are around 6000 sites available for forecasts, and around 150 sites available for observations.

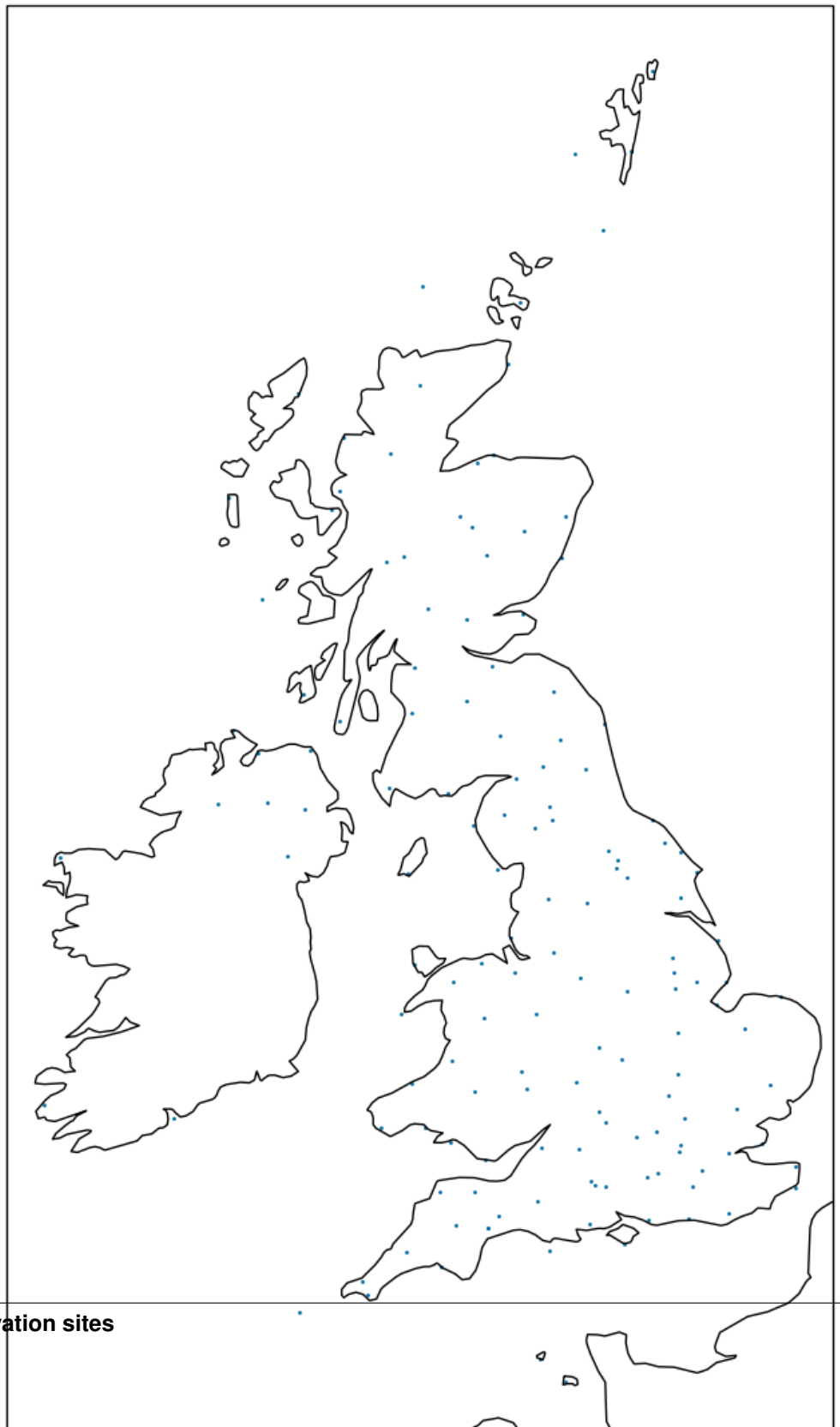
3.1 Forecast sites

The locations of the sites available for which forecasts are shown on the map below. Forecasts are restricted to locations within 30 km of the nearest site. This includes all of the United Kingdom. Most of the UK is within 10 km of the nearest site.

3.2 Observation sites

The locations of the sites available for which observations are shown on the map below.





CHAPTER 4

Objects

DataPoint for Python makes use of objects for almost everything. There are 6 different objects which will be returned by the module.

The diagram below shows the main classes used by the library, and how to move between them.

4.1 Manager

The object which stores your API key and has methods to access the API.

4.1.1 Attributes

attribute	type
api_key	string
call_response	dict
forecast_sites_last_update	float
forecast_sites_last_request	list of Site objects
forecast_sites_update_time	int
observation_sites_last_update	float
observation_sites_last_request	list of Site objects
observation_sites_update_time	int

4.1.2 Methods

get_forecast_sites()

Returns a list of available forecast sites.

- returns: list of Site objects

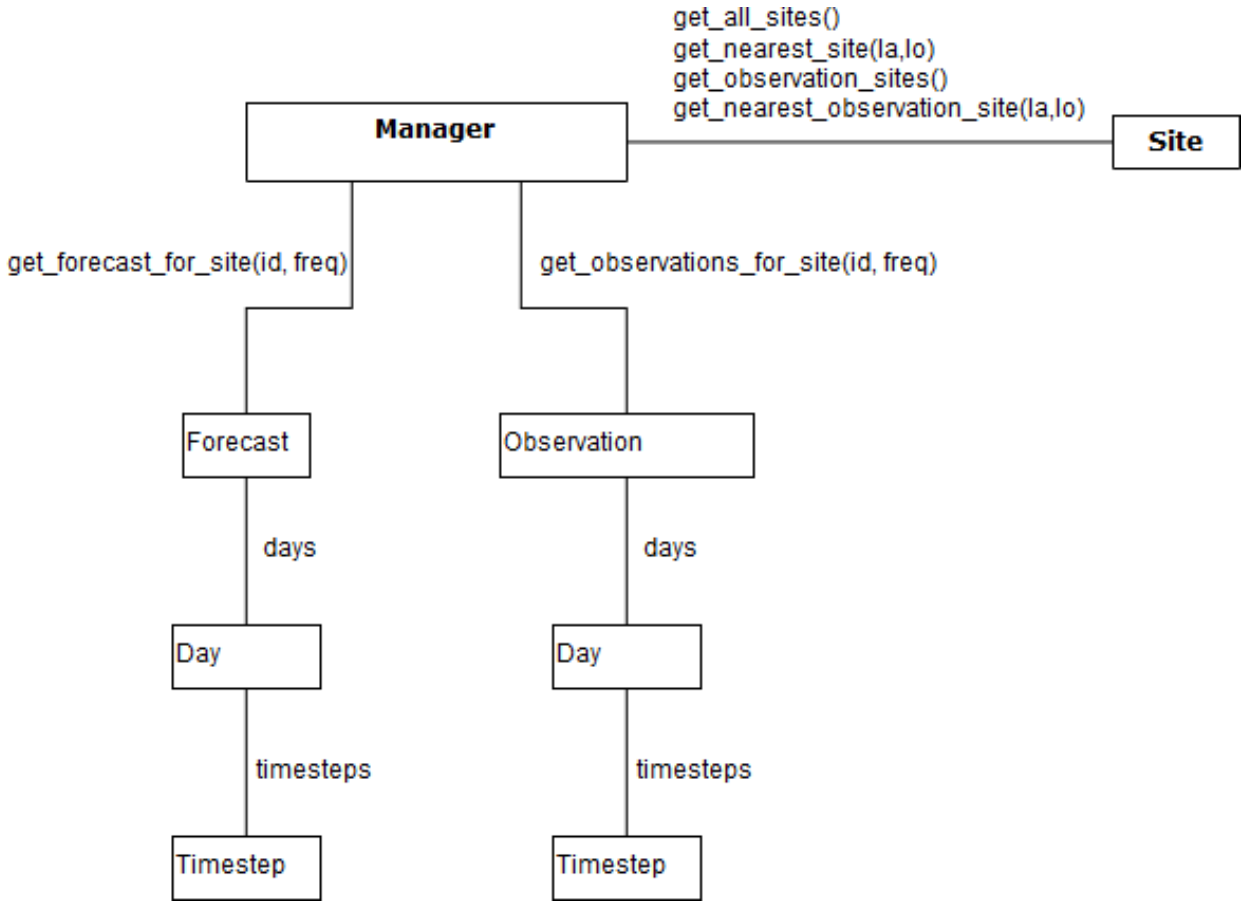


Fig. 1: classes

get_nearest_forecast_site(latitude=False, longitude=False)

Returns the nearest Site object to the specified coordinates which can provide a forecast.

- param latitude: int or float
- param longitude: int or float
- returns: Site

get_forecast_for_site(site_id, frequency="daily")

Get a forecast for the provided site. A frequency of "daily" will return two timesteps: "Day" and "Night". A frequency of "3hourly" will return 8 timesteps: 0, 180, 360 … 1260 (minutes since midnight UTC)

- param site_id: string or unicode
- param frequency: string ("daily" or "3hourly")
- returns: Forecast

get_observation_sites()

Returns a list of sites for which observations are available.

- returns: list of Site objects

get_nearest_observation_site(longitude=False, latitude=False)

Returns the nearest Site object to the specified coordinates.

- param longitude: int or float
- param latitude: int or float
- returns: Site

get_observations_for_site(site_id, frequency='hourly')

Get the observations for the provided site. Only hourly observations are available, and provide the last 24 hours of data.

- param site_id: string or unicode
- param frequency: string ("daily" or "3hourly")
- returns: Observation

4.2 Site

An object containing details about a specific forecast site.

4.2.1 Attributes

attribute	type
api_key	string
name	unicode
id	unicode
elevation	unicode
latitude	unicode
longitude	unicode
nationalPark	unicode
region	unicode
unitaryAuthArea	unicode

4.3 Forecast

An object with properties of a single forecast and a list of Day objects.

4.3.1 Attributes

attribute	type
api_key	string
data_date	datetime
continent	unicode
country	unicode
name	unicode
longitude	unicode
latitude	unicode
id	unicode
elevation	unicode
days	list of Day objects

4.3.2 Methods

now()

Get the current timestep from this forecast

- returns: Timestep (or False)

4.4 Observation

An object with the properties of a single observation and a list of Day objects.

4.4.1 Attributes

attribute	type
api_key	string
data_date	datetime
continent	unicode
country	unicode
name	unicode
longitude	unicode
latitude	unicode
id	unicode
elevation	unicode
days	list of Day objects

4.4.2 Methods

now()

Get the current timestep from this observation

- returns: Timestep

4.5 Day

An object with properties of a single day and a list of Timestep objects.

4.5.1 Attributes

attribute	type
api_key	string
date	datetime
timesteps	list of Timestep objects

4.6 Timestep

An object with each forecast property (wind, temp, etc) for a specific time, in the form of Element objects.

4.6.1 Attributes

attribute	type
api_key	string
name	string
date	datetime
weather	Element
temperature	Element
feels_like_temperature	Element
wind_speed	Element
wind_direction	Element
wind_gust	Element
visibility	Element
uv	Element
precipitation	Element
humidity	Element

4.6.2 Methods

elements()

Get a list of element objects in the Timestep.

- returns: List of element objects

4.7 Element

An object with properties about a specific weather element.

4.7.1 Attributes

attribute	type
id	string
value	int, float or string
units	unicode
text	string or None