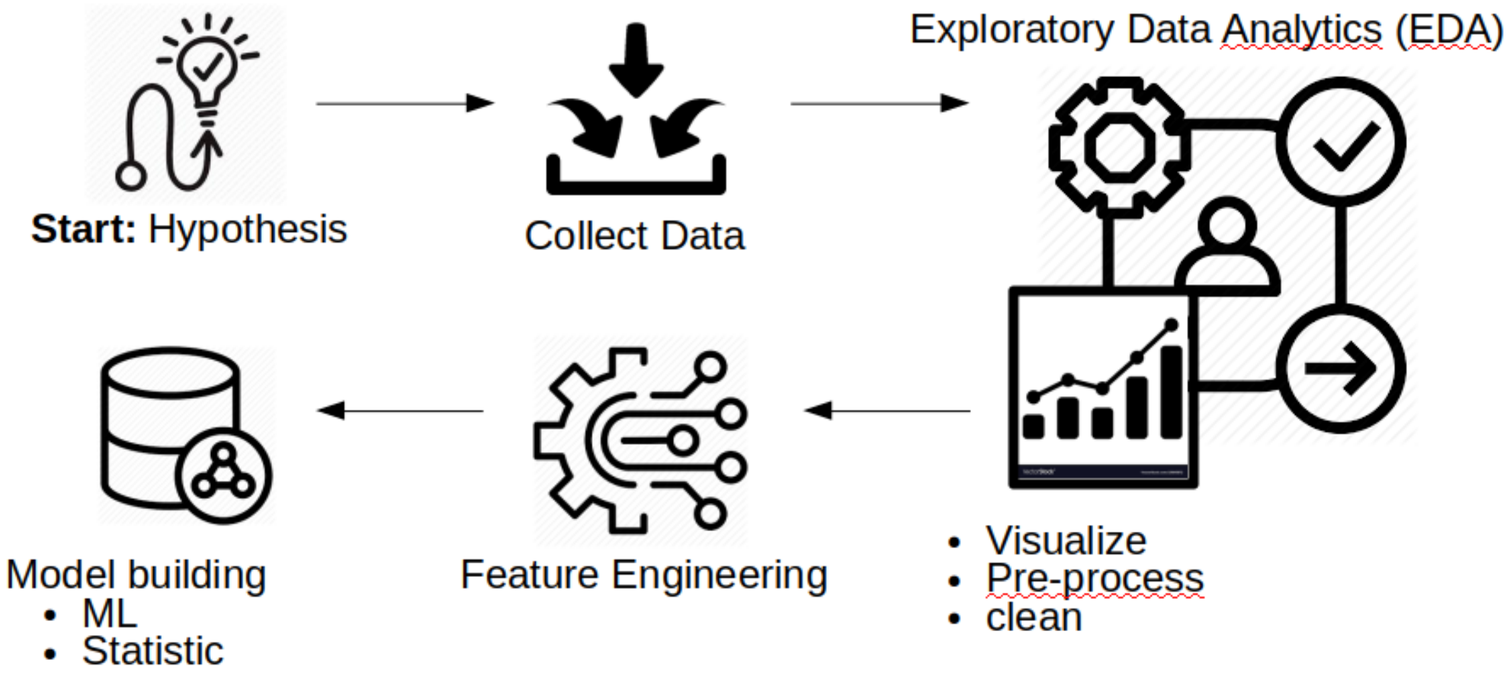
Data Wrangling

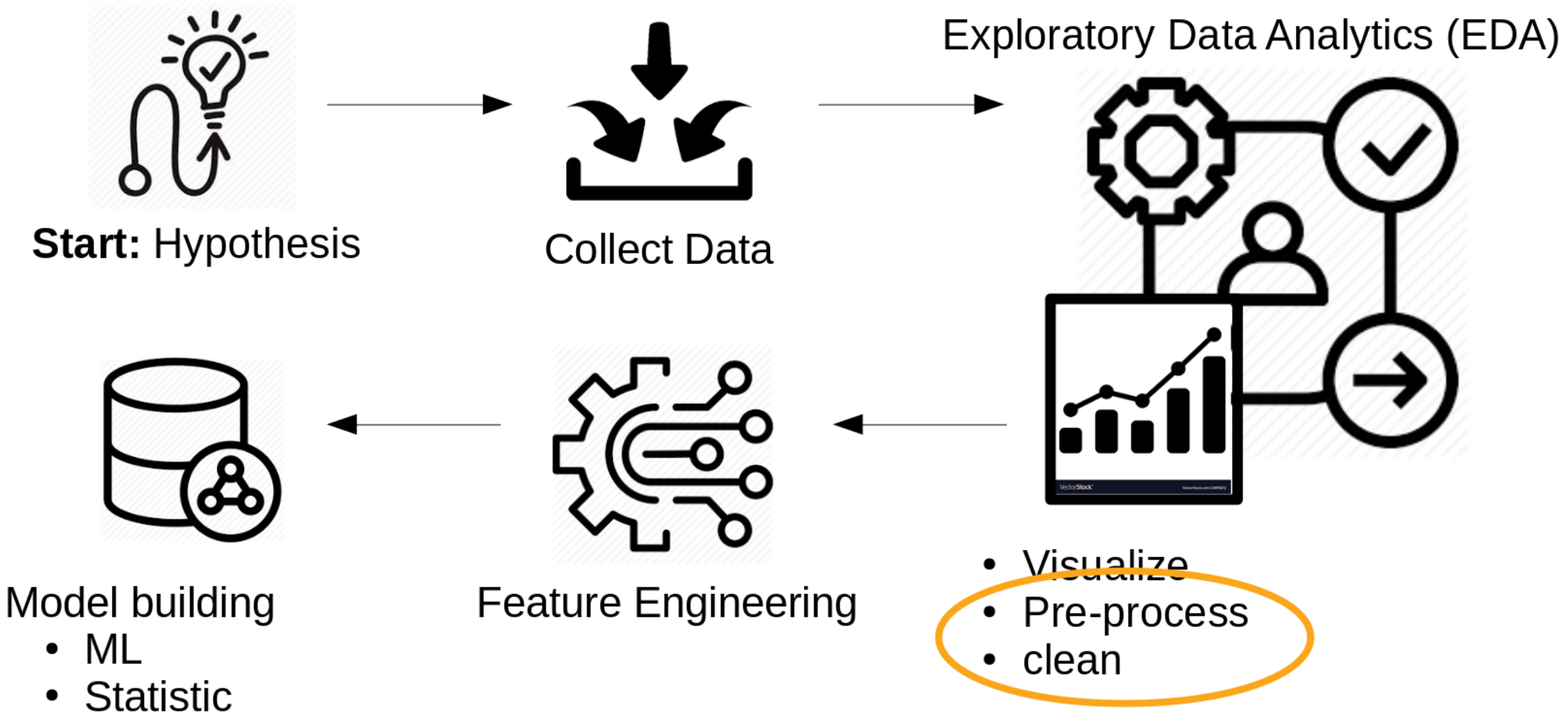
Outline

- Data Science Processing Pipeline
- What is ***Data Wrangling***?
    - Stages of ***Data Wrangling***
- Short Introduction to ***Pandas***
- ***Wrangling*** by Use Cases (Lab session)

Data Science Processing Pipeline



Data Science Processing Pipeline

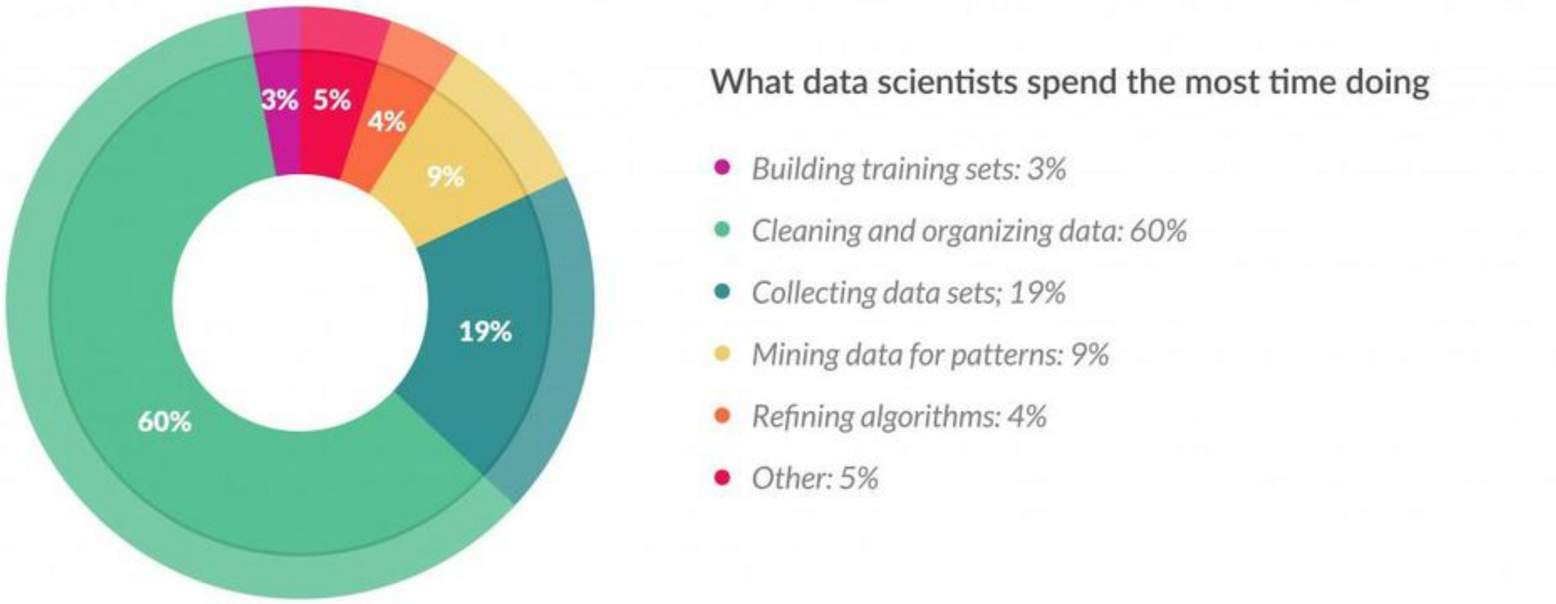


What is *Data Wrangling* ?

What is *Data Wrangling* ?

Definition:

Data wrangling, sometimes referred to as data munging, is the process of transforming and mapping data from one "raw" data form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes such as analytics. [wikipedia]



[source: study by forbes.com: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#]

Phases of *Data Wrangling*

- (Scrape)
- Clean
- Transform
- Merge
- Reshape -> Rectify

Phases of *Data Wrangling*

- (Scrape): ***get data from sensors, internet, databases, ...***
- Clean
- Transform
- Merge
- Reshape -> Rectify

Phases of *Data Wrangling*

- (Scrape)
- Clean : ***remove "bad data"***
- Transform
- Merge
- Reshape -> Rectify

Phases of *Data Wrangling*

- (Scrape)
- Clean
- Transform : ***change/correct data formats, recompute, ...***
- Merge
- Reshape -> Rectify

Phases of *Data Wrangling*

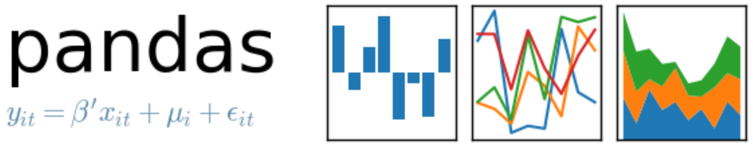
- (Scrape)
- Clean
- Transform
- Merge: ***combine and connect data sources***
- Reshape -> Rectify

Phases of *Data Wrangling*

- (Scrape)
- Clean
- Transform
- Merge
- Reshape -> Rectify: ***output: vectors, arrays, tables***

Wrangling in Python with Pandas

Started as *"**spread sheets for python**"* - now has become one of the most important ***Data Wrangling*** and **EDA** tools in ***Python***



pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python has long been great for data munging and preparation, but less so for data analysis and modeling. pandas helps fill this gap, enabling you to carry out your entire data analysis workflow in Python without having to switch to a more domain specific language like **R**.[pandas website]

Pandas Documentation

- Pandas website: https://pandas.pydata.org/
- Pandas user guide: http://pandas.pydata.org/pandas-docs/stable/user_guide/index.html
- Pandas API documentation: http://pandas.pydata.org/pandas-docs/stable/reference/index.html
- VERY USEFULL: Pandas Cheat Sheet: https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf

Pandas in a Nutshell

In [1]:
```
#import the pandas module
import pandas as pd #naming convention for pandas is pd
```

Pandas in a Nutshell

In [1]:
```
#import the pandas module
import pandas as pd #naming convention for pandas is pd
```

The central element of *Pandas* is the *DataFrame*

- spreadsheet like data structure
- rectifies data into tables
- database like functionality
- arrray compatible

Pandas Features

- Data in- and export
- DataFrame (DF) data structure with functionality of
    - spreadsheet
    - relational data base
- DF Statistcs
- DF Visualization
- Rich library of ***wrangling*** methods

Reading CSV and Excel sheets:

d=pd.read_csv("path"):

- pd.read_csv() is the function to read the CSV(Comma separated values) file from your computer.
- In the function you have to pass "path" of the CSV file under quote.
- Store the dataframe in any variable,here i stored it in variable "d".
- read_csv() function makes the CSV file into dataframe so that you can access it just like a disctionary.

d=pd.read_excel("path") :

- It is same as the read_csv() but it reads excel sheet or file.

Importing Weather data

In [2]:
```
#get data from GitHub -> https://github.com/keuperj/DATA
!git clone https://github.com/keuperj/DATA.git
```

```
fatal: destination path 'DATA' already exists and is not an empty directory.
```

In [3]:
```
d=pd.read_csv('DATA/weather.csv')
# returning dataframe object
d.head()
#printing dataframe d in table format
```

Out[3]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

First interaction with the data

How many rows are in my DataFrame ?

In [4]:
```
print(len(d))
# there are 96453 rows
```

```
96453
```

Getting the first n-rows of dataset

- To see the first five rows call head() function with dataframe object.

  for example-

  ```
  d.head()
  ```

- If you want to view first n-rows

  ```
  d.head(n)
  ```

In [5]:
```
d.head() # first five rows
```

Out[5]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

In [6]:
```
d.head(9) # to print first n=9 rows
```

Out[6]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.
5	2006-04-01 05:00:00.000 +0200	Partly Cloudy	rain	9.222222	7.111111	0.85	13.9587	258.0	14.9569	0.0	1016.66	Partly cloudy throughout the day.
6	2006-04-01 06:00:00.000 +0200	Partly Cloudy	rain	7.733333	5.522222	0.95	12.3648	259.0	9.9820	0.0	1016.72	Partly cloudy throughout the day.
7	2006-04-01 07:00:00.000 +0200	Partly Cloudy	rain	8.772222	6.527778	0.89	14.1519	260.0	9.9820	0.0	1016.84	Partly cloudy throughout the day.
8	2006-04-01 08:00:00.000 +0200	Partly Cloudy	rain	10.822222	10.822222	0.82	11.3183	259.0	9.9820	0.0	1017.37	Partly cloudy throughout the day.

Getting the last n-rows of dataset

- Call the tail() function

  ```
  d.tail()
  ```

  It will show only last five rows of dataframe d.
- If you want to see last n-rows -

  ```
  d.tail(n)
  ```

In [7]:
```
d.tail() #last five rows
```

Out[7]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.66	Partly cloudy starting in the morning.
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.16	Partly cloudy starting in the morning.

In [8]:
```
d.tail(9) # last 9 rows
```

Out[8]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
96444	2016-09-09 15:00:00.000 +0200	Partly Cloudy	rain	31.083333	29.616667	0.28	15.5043	40.0	16.1000	0.0	1014.17	Partly cloudy starting in the morning.
96445	2016-09-09 16:00:00.000 +0200	Partly Cloudy	rain	31.083333	29.611111	0.28	13.8943	40.0	16.1000	0.0	1013.97	Partly cloudy starting in the morning.
96446	2016-09-09 17:00:00.000 +0200	Partly Cloudy	rain	30.766667	29.311111	0.28	14.2163	24.0	15.5526	0.0	1013.83	Partly cloudy starting in the morning.
96447	2016-09-09 18:00:00.000 +0200	Partly Cloudy	rain	28.838889	27.850000	0.32	12.2038	21.0	16.1000	0.0	1014.07	Partly cloudy starting in the morning.
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.66	Partly cloudy starting in the morning.
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.16	Partly cloudy starting in the morning.

Simple Slicing in a DataFrame

- Slicing works very similar to Numpy
- Suppose you want to get 10 rows of the dataframe ranging from row 20 to 30.

```
d[20:31]
```

In [9]:
```
d[20:31]
```

Out[9]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
20	2006-04-01 20:00:00.000 +0200	Mostly Cloudy	rain	11.550000	11.550000	0.77	7.3899	147.0	11.0285	0.0	1015.85	Partly cloudy throughout the day.
21	2006-04-01 21:00:00.000 +0200	Mostly Cloudy	rain	11.183333	11.183333	0.76	4.9266	160.0	9.9820	0.0	1015.77	Partly cloudy throughout the day.
22	2006-04-01 22:00:00.000 +0200	Partly Cloudy	rain	10.116667	10.116667	0.79	6.6493	163.0	15.8263	0.0	1015.40	Partly cloudy throughout the day.
23	2006-04-01 23:00:00.000 +0200	Mostly Cloudy	rain	10.200000	10.200000	0.77	3.9284	152.0	14.9569	0.0	1015.51	Partly cloudy throughout the day.
24	2006-04-10 00:00:00.000 +0200	Partly Cloudy	rain	10.422222	10.422222	0.62	16.9855	150.0	15.8263	0.0	1014.40	Mostly cloudy throughout the day.
25	2006-04-10 01:00:00.000 +0200	Partly Cloudy	rain	9.911111	7.566667	0.66	17.2109	149.0	15.8263	0.0	1014.20	Mostly cloudy throughout the day.
26	2006-04-10 02:00:00.000 +0200	Mostly Cloudy	rain	11.183333	11.183333	0.80	10.8192	163.0	14.9569	0.0	1008.71	Mostly cloudy throughout the day.
27	2006-04-10 03:00:00.000 +0200	Partly Cloudy	rain	7.155556	5.044444	0.79	11.0768	180.0	15.8263	0.0	1014.47	Mostly cloudy throughout the day.
28	2006-04-10 04:00:00.000 +0200	Partly Cloudy	rain	6.111111	4.816667	0.82	6.6493	161.0	15.8263	0.0	1014.45	Mostly cloudy throughout the day.
29	2006-04-10 05:00:00.000 +0200	Partly Cloudy	rain	6.788889	4.272222	0.83	13.0088	135.0	14.9569	0.0	1014.49	Mostly cloudy throughout the day.
30	2006-04-10 06:00:00.000 +0200	Mostly Cloudy	rain	7.261111	5.155556	0.85	11.1734	141.0	6.1985	0.0	1014.52	Mostly cloudy throughout the day.

In [10]:
```
#slicing with step size
d[20:30:2]
```

Out[10]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
20	2006-04-01 20:00:00.000 +0200	Mostly Cloudy	rain	11.550000	11.550000	0.77	7.3899	147.0	11.0285	0.0	1015.85	Partly cloudy throughout the day.
22	2006-04-01 22:00:00.000 +0200	Partly Cloudy	rain	10.116667	10.116667	0.79	6.6493	163.0	15.8263	0.0	1015.40	Partly cloudy throughout the day.
24	2006-04-10 00:00:00.000 +0200	Partly Cloudy	rain	10.422222	10.422222	0.62	16.9855	150.0	15.8263	0.0	1014.40	Mostly cloudy throughout the day.
26	2006-04-10 02:00:00.000 +0200	Mostly Cloudy	rain	11.183333	11.183333	0.80	10.8192	163.0	14.9569	0.0	1008.71	Mostly cloudy throughout the day.
28	2006-04-10 04:00:00.000 +0200	Partly Cloudy	rain	6.111111	4.816667	0.82	6.6493	161.0	15.8263	0.0	1014.45	Mostly cloudy throughout the day.

In [11]:
```
#inverse index
d[-15:-10]
```

Out[11]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
96438	2016-09-09 09:00:00.000 +0200	Partly Cloudy	rain	22.138889	22.138889	0.65	7.7763	30.0	16.1000	0.0	1015.46	Partly cloudy starting in the morning.
96439	2016-09-09 10:00:00.000 +0200	Partly Cloudy	rain	22.872222	22.872222	0.59	6.4239	49.0	16.1000	0.0	1015.65	Partly cloudy starting in the morning.
96440	2016-09-09 11:00:00.000 +0200	Partly Cloudy	rain	27.072222	27.022222	0.42	12.0106	49.0	15.5526	0.0	1015.44	Partly cloudy starting in the morning.
96441	2016-09-09 12:00:00.000 +0200	Partly Cloudy	rain	28.866667	28.216667	0.37	13.9265	61.0	16.1000	0.0	1015.35	Partly cloudy starting in the morning.
96442	2016-09-09 13:00:00.000 +0200	Partly Cloudy	rain	30.994444	29.972222	0.33	15.6170	70.0	16.1000	0.0	1014.86	Partly cloudy starting in the morning.

BUT: accesing a single element is different!

```
In [12]: d[3]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~/anaconda3/lib/python3.7/site-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   2896             try:
-> 2897                 return self._engine.get_loc(key)
   2898             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 3

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-12-0acadf17a380> in <module>
----> 1 d[3]

~/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py in __getitem__(self, key)
   2993             if self.columns.nlevels > 1:
   2994                 return self._getitem_multilevel(key)
-> 2995             indexer = self.columns.get_loc(key)
   2996             if is_integer(indexer):
   2997                 indexer = [indexer]

~/anaconda3/lib/python3.7/site-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   2897                 return self._engine.get_loc(key)
   2898             except KeyError:
-> 2899                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2900         indexer = self.get_indexer([key], method=method, tolerance=tolerance)
   2901         if indexer.ndim > 1 or indexer.size > 1:
```

In [13]:
```
#use iloc instead
d.iloc[3]
```

Out[13]:
```
Formatted Date                  2006-04-01 03:00:00.000 +0200
Summary                                         Partly Cloudy
Precip Type                                              rain
Temperature (C)                                       8.28889
Apparent Temperature (C)                              5.94444
Humidity                                                 0.83
Wind Speed (km/h)                                     14.1036
Wind Bearing (degrees)                                    269
Visibility (km)                                       15.8263
Loud Cover                                                  0
Pressure (millibars)                                  1016.41
Daily Summary                 Partly cloudy throughout the day.
Name: 3, dtype: object
```

In [14]:
```
#check with head
d.head()
```

Out[14]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

Accesing the particular column of dataframe :

- You can access the particular column of the dataframe just my mentioning its name under quote with dataframe d.
  for example you want to access Humidity column e.i,

```
d['Humidity']
```

In [15]:
```
d['Humidity'].head(10)
#Here i have applied head() function just to see only first 10 values because humidity
#column has too many values
```

Out[15]:
```
0    0.89
1    0.86
2    0.89
3    0.83
4    0.83
5    0.85
6    0.95
7    0.89
8    0.82
9    0.72
Name: Humidity, dtype: float64
```

Finding min, max and average of a column:

- **min() :** To find minimum of a column
- **max() :** To find maximum of a column
- **mean() :** To find average of a column

Find minimum and maximum of Humidity column-

Finding min, max and average of a column:

- **min() :** To find minimum of a column
- **max() :** To find maximum of a column
- **mean() :** To find average of a column

Find minimum and maximum of Humidity column-

In [16]:
```
d['Humidity'].min()
#Printing minimum value of Humidity
```

Out[16]:
```
0.0
```

Finding min, max and average of a column:

- **min() :** To find minimum of a column
- **max() :** To find maximum of a column
- **mean() :** To find average of a column

Find minimum and maximum of Humidity column-

In [16]:
```
d['Humidity'].min()
#Printing minimum value of Humidity
```

Out[16]:
```
0.0
```

In [17]:
```
d['Humidity'].max()
#Print maximum value Humidity
```

Out[17]:
```
1.0
```

Finding min, max and average of a column:

- **min() :** To find minimum of a column
- **max() :** To find maximum of a column
- **mean() :** To find average of a column

Find minimum and maximum of Humidity column-

In [16]:
```
d['Humidity'].min()
#Printing minimum value of Humidity
```

Out[16]:
```
0.0
```

In [17]:
```
d['Humidity'].max()
#Print maximum value Humidity
```

Out[17]:
```
1.0
```

In [18]:
```
d['Humidity'].mean()
#Print average value Humidty
```

Out[18]:
```
0.7348989663358906
```

Conditional statements

- `d["your column"]["your condition"]`
  It will return all the values in which your condition holds true.

  Examples :

- Find temp when Humidity is minimum

```
d['Temperature(C)'][d['Humidity']==d['Humidity'].min() ]
```

- Find temp when Humidity is maximum

```
d['Temperature(C)'][d['Humidity']==d['Humidity'].max() ]
```

In [21]:
```
d['Temperature (C)'][d['Humidity']==d['Humidity'].min()]
```

Out[21]:
```
19958    -1.111111
28101   -15.000000
28103   -15.555556
28110   -13.888889
29627     1.111111
54840   -15.555556
54858   -15.000000
54870   -16.111111
54872   -15.000000
54873   -13.888889
55086   -12.777778
55088   -12.777778
55349   -12.222222
55350   -12.222222
55352   -12.222222
55412   -17.777778
55469   -17.777778
55472   -17.777778
55473   -16.111111
55481   -12.777778
55508   -16.111111
55511   -17.777778
Name: Temperature (C), dtype: float64
```

In [22]:
```
d['Temperature (C)'][d['Humidity']==d['Humidity'].max()]
```

Out[22]:
```
319      11.972222
342       7.155556
390       7.688889
535       5.933333
536      10.405556
            ...
95907    14.905556
96123     7.594444
96148     8.838889
96363    12.338889
96364    12.872222
Name: Temperature (C), Length: 2890, dtype: float64
```

In the same way you can apply various condition and analyse it hidden features

Replacing NaN with specific value

- Here I am replacing all NaN value with 0.

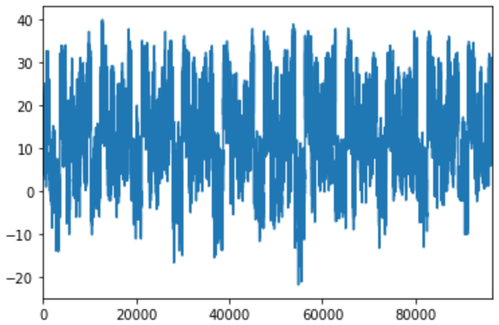
In [23]:
```
d.fillna(0,inplace=True)
```

Visualization with Pandas

Pandas also has build in visualization and plotting methods

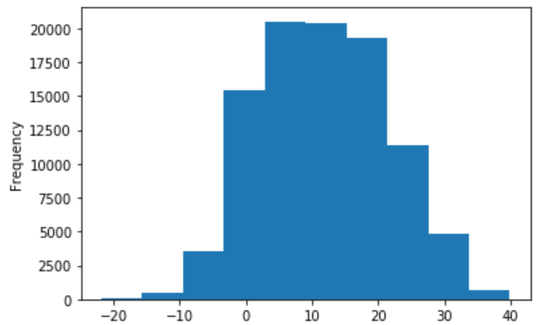
In [20]:
```
# example simple plot
import matplotlib.pyplot as plt
%matplotlib inline
d['Temperature (C)'].plot()
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0612e19990>
```



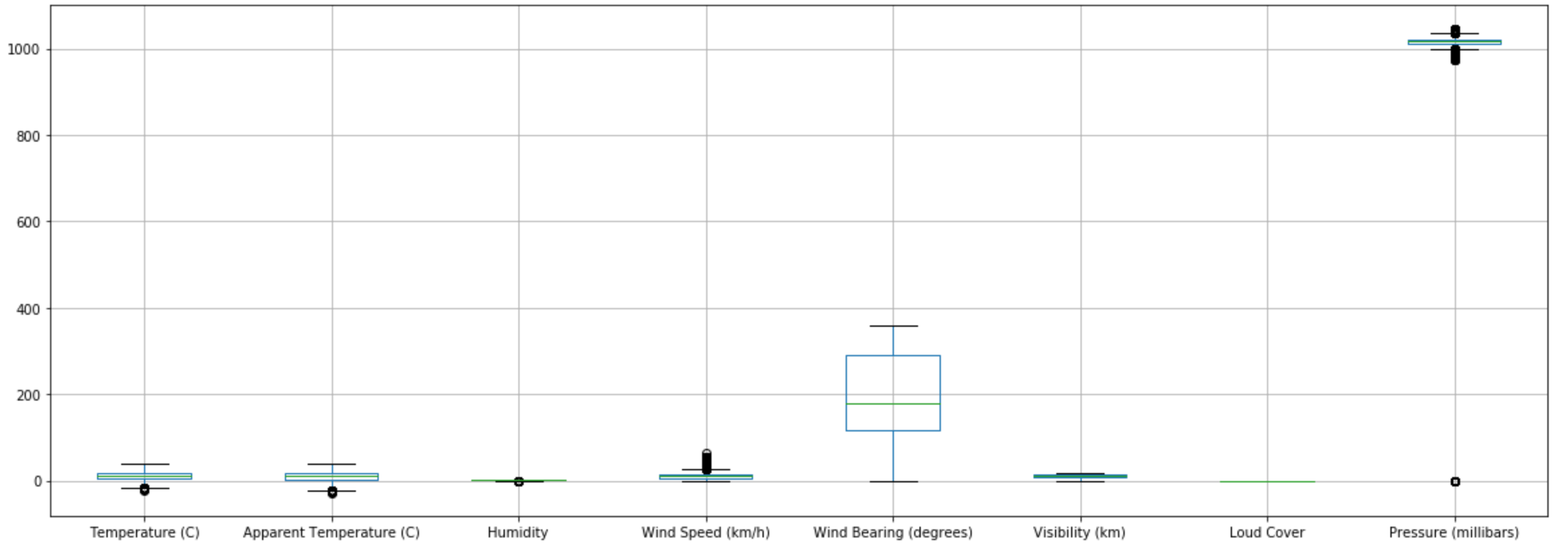
In [21]:
```
# histogram
d['Temperature (C)'].plot.hist()
```

Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f060ee6fe10>
```



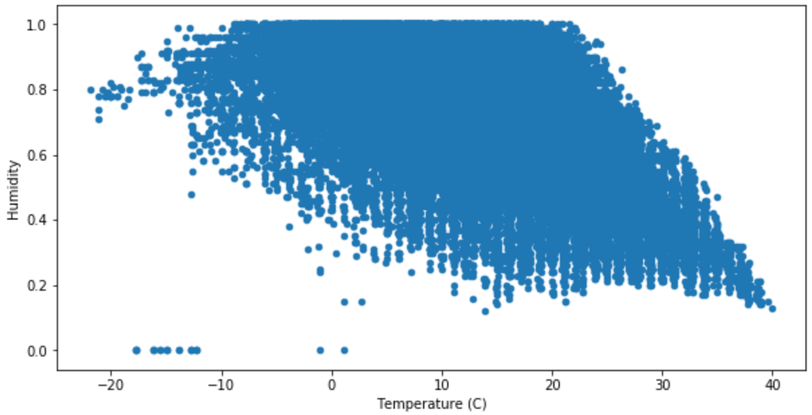
In [23]:
```
# boxplot over all numerical variables
plt.figure(figsize=(20,7)) #pandas uses plt, so we can use plt methods to manipulate the output
d.boxplot()
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7f060eaf3490>

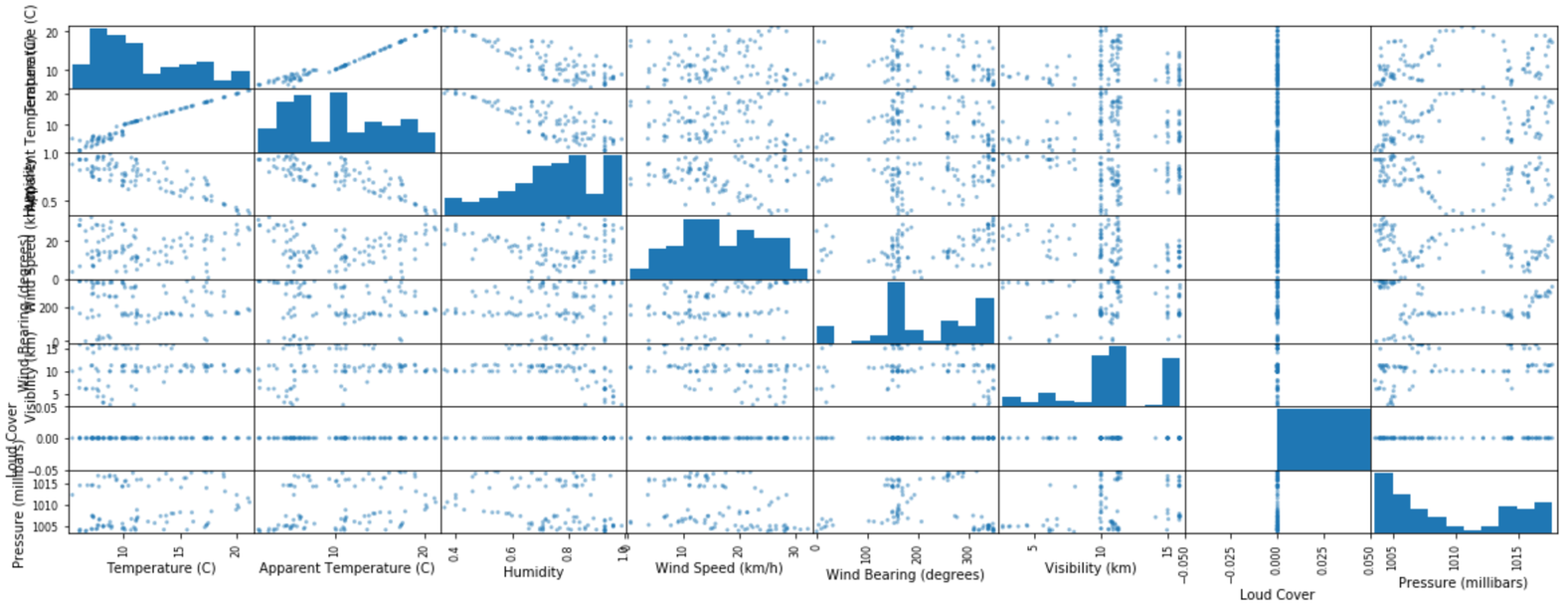


In [24]:
```
#scatter plot
d.plot.scatter('Temperature (C)','Humidity',figsize=(10,5))
```

Out[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f060ea23710>
```



In [27]:
```
#finaly a realy helpfull tool to get a first look at data - takes some time to compute on full data
output=pd.plotting.scatter_matrix(d[1:100],figsize=(20,7))
```



Detailted introduction in the Lab session!

- With wrangling use cases ...