Introduction to coding with python

Workshop 2 – 20-09-2024

Willem Kroese – w.s.j.kroese@uu.nl – BBG 6.17
Institute of Marine and Atmospheric Research

Last time and today

- NumPy, Matplotlib and SciPy
- Questions?
- Today: Importing data with Pandas, NetCDF4 and xarray

Quick word about functions

$$e_s(T) = 6.1094\exp\left(\frac{17.625T}{T+243.04}\right),$$

+ Code + Markdown

```python
T = 15 #C
e_s = 6.1094 * np.exp(17.625 * (T+273.15) / ((T+273.15) + 243.04))
print('for T =15, e_s=',e_s)
```

✓ 0.0s Python

```
for T =15, e_s= 86743.29343409656
```

Quick word about functions

$$e_s(T) = 6.1094 \exp\left(\frac{17.625T}{T + 243.04}\right),$$

```python
T = 25 #C
e_s = 6.1094 * np.exp(17.625 * (T+273.15) / ((T+273.15) + 243.04))
print('for T =25, e_s=',e_s)
```

✓ 0.0s Python

```
for T =25, e_s= 100681.32170543664
```

Quick word about functions

```python
def clausius_clapeyron(T):
    '''input: temperature in (K)
      output: saturation vapor pressure in (hPa)'''
    return 6.1094 * np.exp(17.625 * T / (T + 243.04))
```

✓ 0.0s Python

```python
print(clausius_clapeyron(15+273.15), clausius_clapeyron(25+273.15))
```

✓ 0.0s Python

```
86743.29343409656 100681.32170543664
```

Quick word about functions

```
def clausius_clapeyron(T):
    '''input: temperature in (K)
       output: saturation vapor pressure in (hPa)'''
    return 6.1094 * np.exp(17.625 * T / (T + 243.04))
```

✓ 0.0s Python

```
print(clausius_clapeyron(15+273.15), clausius_clapeyron(25+273.15))
```

✓ 0.0s Python

```
86743.29343409656 100681.32170543664
```

Quick word about functions

```python
def clausius_clapeyron(T,unit='K'):
    '''input:
            temperature in (K) or (C)
            unit: string specifying the unit: 'K' or 'C'
        output:
            saturation vapor pressure in (hPa)'''
    if unit == 'K':
        return 6.1094 * np.exp(17.625 * T / (T + 243.04))
    elif unit == 'C':
        return 6.1094 * np.exp(17.625 * (T+273.15) / ((T+273.15) + 243.04))
    else:
        print('unit not recognized')
```

✓ 0.0s Python

```python
print(clausius_clapeyron(15,'C'), clausius_clapeyron(25,'C'))
print(clausius_clapeyron(15+273.15), clausius_clapeyron(25+273.15))
```

✓ 0.0s Python

```
86743.29343409656 100681.32170543664
86743.29343409656 100681.32170543664
```

Importing data

- NumPy has some built-in functions for importing data
- But there are some powerful libraries

- **Pandas** = importing, writing and processing tabular data (i.e. spreadsheets)
- **NetCDF4** = importing and writing NetCDF files
- **Xarray** = importing, writing and processing NetCDF files

Pandas

- **Series =** 1D labelled array
- **DataFrame =** 2D labelled array (table)
- Very useful features, e.g. Time Indexes!
- Indexing is different..

In [9]:
```python
# first we create a dictionary
data = {'age': [30, 15, 22],
        'height': [180, 155, 160],
        'weight': [70, 52, np.nan]}
df = pd.DataFrame(data, index=['Alice', 'Bob', 'Claire'])
df
```

Out[9]:

	age	height	weight
Alice	30	180	70.0
Bob	15	155	52.0
Claire	22	160	NaN

Pandas

- **Indexing:** 2 methods (`.loc[:,:]` `.iloc[:,:]`)

```
df
```

	age	height	weight
Alice	30	180	70.0
Bob	15	155	52.0
Claire	22	160	NaN

```
df.loc[:,'height':]
```

	height	weight
Alice	180	70.0
Bob	155	52.0
Claire	160	NaN

```
df.iloc[:,1:]
```

	height	weight
Alice	180	70.0
Bob	155	52.0
Claire	160	NaN

NetCDF4

- **NetCDF =** - efficient storage of multidimensional data
  - accessible with (almost) all programming languages
  - machine-independent (portable)
  - easy access of a subset
  - description of data included

- Import data with: `data = NetCDF4.Dataset('filename')`
- Print variables with: `data.variables`

```
longitude = winddata.variables['lon'][:]
latitude = winddata.variables['lat'][:]
time = winddata.variables['time'][:]
```

```
longitude = winddata['lon'][:]
latitude = winddata['lat'][:]
time = winddata['time'][:]
```

xarray

- Higher level interface for dealing with NetCDF files
- Similar to pandas, i.e. label bases indexing, built-in functions for analysis and plotting
- Two structures:
  - **DataArray** = 1 variable field, e.g. temperature(lon,lat,z,t)
  - **DataSet** = multiple DataArrays, that may share coordinates, e.g. temperature(lon,lat,z,t) and relative_humidity(lon,lat,z,t)
- For each variable, the dimensions are labelled:

  In NumPy, the dimensions were just 0,1,2, …
  here you can give them names, e.g., lon, lat, time

Notebooks

- **Pandas**: examples of built-in statistics and plotting and practice with weather station data from NOAA
- **NetCDF4**: practice with importing and interpreting output of wind model of the KNMI
- **xarray**: practice with temperature data from NASA

Notebooks and data are on Blackboard and Github. You need to unpack ‘data.zip’ to run them!

Pandas

- **Series =** 1D labelled array

Pandas

- **Series** = 1D labelled array

In [2]:
```
names = ['Alice', 'Bob', 'Claire']
values = [30, 15, 22]
ages = pd.Series(values, index=names)
ages
```

Out[2]:
```
Alice     30
Bob       15
Claire    22
dtype: int64
```

Pandas

- **Series** = 1D labelled array

In [2]:
```
names = ['Alice', 'Bob', 'Claire']
values = [30, 15, 22]
ages = pd.Series(values, index=names)
ages
```

Out[2]:
```
Alice     30
Bob       15
Claire    22
dtype: int64
```

Index → use this to call to data instead of indices

Pandas

- **Series =** 1D labelled array

- **DataFrame =** 2D labelled array (table)

Pandas

- **Series** = 1D labelled array

- **DataFrame** = 2D labelled array (table)

In [9]:

```
# first we create a dictionary
data = {'age': [30, 15, 22],
        'height': [180, 155, 160],
        'weight': [70, 52, np.nan]}
df = pd.DataFrame(data, index=['Alice', 'Bob', 'Claire'])
df
```

Out[9]:

	age	height	weight
Alice	30	180	70.0
Bob	15	155	52.0
Claire	22	160	NaN

Pandas

- **Series =** 1D labelled array

- **DataFrame =** 2D labelled array (table)

In [9]:

```python
# first we create a dictionary
data = {'age': [30, 15, 22],
        'height': [180, 155, 160],
        'weight': [70, 52, np.nan]}
df = pd.DataFrame(data, index=['Alice', 'Bob', 'Claire'])
df
```

Out[9]:

column

	age	height	weight
Alice	30	180	70.0
Bob	15	155	52.0
Claire	22	160	NaN

index

Pandas

- **Series** = 1D labelled array

- **DataFrame** = 2D labelled array (table)

In [9]:
```python
# first we create a dictionary
data = {'age': [30, 15, 22],
        'height': [180, 155, 160],
        'weight': [70, 52, np.nan]}
df = pd.DataFrame(data, index=['Alice', 'Bob', 'Claire'])
df
```

Out[9]:

	age	height	weight
Alice	30	180	70.0
Bob	15	155	52.0
Claire	22	160	NaN

column

index

NaN = not a number, to deal with missing data

Pandas

- **Indexing:** 2 methods (`.loc[:,:]` `.iloc[:,:]`)

```
df
```

	age	height	weight
Alice	30	180	70.0
Bob	15	155	52.0
Claire	22	160	NaN

```
df.loc[:,'height':]
```

	height	weight
Alice	180	70.0
Bob	155	52.0
Claire	160	NaN

```
df.iloc[:,1:]
```

	height	weight
Alice	180	70.0
Bob	155	52.0
Claire	160	NaN

Pandas

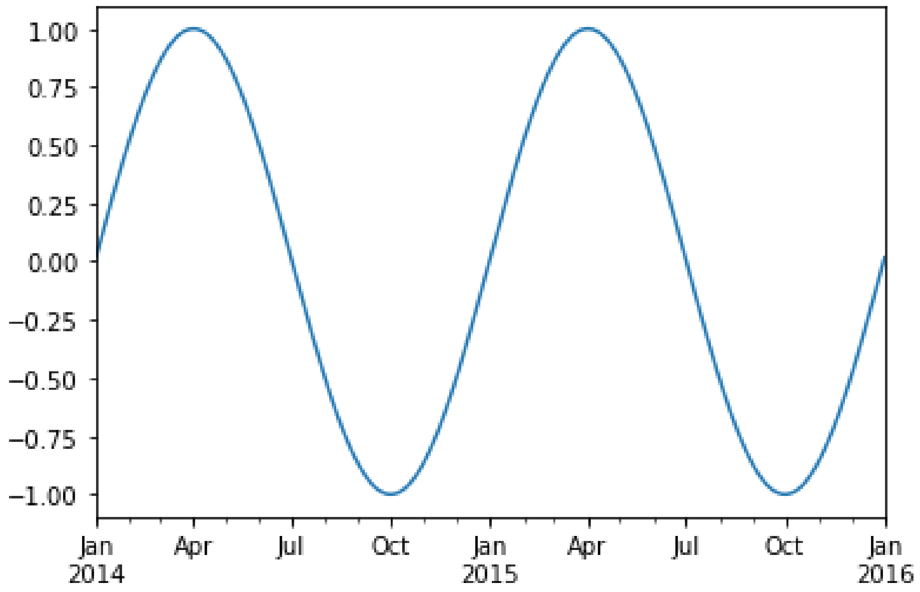
- Timearrays

In [29]:
```
two_years = pd.date_range(start='2014-01-01', end='2016-01-01', freq='D')
two_years
```

Out[29]:
```
DatetimeIndex(['2014-01-01', '2014-01-02', '2014-01-03', '2014-01-04',
               '2014-01-05', '2014-01-06', '2014-01-07', '2014-01-08',
               '2014-01-09', '2014-01-10',
               ...
               '2015-12-23', '2015-12-24', '2015-12-25', '2015-12-26',
               '2015-12-27', '2015-12-28', '2015-12-29', '2015-12-30',
               '2015-12-31', '2016-01-01'],
              dtype='datetime64[ns]', length=731, freq='D')
```

In [30]:
```
timeseries = pd.Series(np.sin(2 *np.pi *two_years.dayofyear / 365),
                       index=two_years)
timeseries.plot()
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6c63a1668>
```



Data in same folder as notebook? -> provide name

Otherwise give the path to the datafile

Pandas

In [34]:

```
df = pd.read_csv('data/sample.csv', sep=',', na_values=[9999.9, 999.9, 99.99])
df.head()
```

Out[34]:

	STATION	DATE	LATITUDE	LONGITUDE	ELEVATION	NAME	TEMP	TEMP_ATTRIBUTES	DEWP	DEWP_ATTRIBUTES	...	MXSPD	GUST
0	72565003017	01/01/2018	39.8328	-104.6575	1650.2	DENVER INTERNATIONAL AIRPORT, CO US	11.6	24	5.5	24	...	9.9	NaN
1	72565003017	02/01/2018	39.8328	-104.6575	1650.2	DENVER INTERNATIONAL AIRPORT, CO US	21.2	24	7.3	24	...	9.9	NaN
2	72565003017	03/01/2018	39.8328	-104.6575	1650.2	DENVER INTERNATIONAL AIRPORT, CO US	31.8	24	3.0	24	...	15.9	24.1
3	72565003017	04/01/2018	39.8328	-104.6575	1650.2	DENVER INTERNATIONAL AIRPORT, CO US	34.6	24	11.6	24	...	8.9	NaN
4	72565003017	05/01/2018	39.8328	-104.6575	1650.2	DENVER INTERNATIONAL AIRPORT, CO US	36.3	24	11.4	24	...	14.0	NaN

5 rows × 28 columns

Pandas

In [56]:

```
df.iloc[:,4:].describe() # start from the 5th column
```

Out[56]:

	TEMP	TEMP_ATTRIBUTES	DEWP	DEWP_ATTRIBUTES	SLP	SLP_ATTRIBUTES	STP	STP_ATTRIBUTES	VISIB	VISIB_ATTRI
count	365.000000	365.0	365.000000	365.0	364.000000	365.000000	365.000000	365.0	365.000000	
mean	51.808767	24.0	29.516164	24.0	1013.821429	22.619178	833.118904	24.0	9.369863	
std	18.124424	0.0	14.705844	0.0	7.087589	2.703308	5.014939	0.0	1.435316	
min	5.000000	24.0	-4.100000	24.0	996.900000	10.000000	819.300000	24.0	0.700000	
25%	36.400000	24.0	18.100000	24.0	1009.075000	23.000000	829.900000	24.0	9.700000	
50%	50.700000	24.0	28.500000	24.0	1013.500000	24.000000	833.500000	24.0	10.000000	
75%	68.000000	24.0	42.300000	24.0	1018.625000	24.000000	836.500000	24.0	10.000000	
max	85.300000	24.0	61.500000	24.0	1038.300000	24.000000	845.400000	24.0	10.000000	

Pandas

- Tutorial: examples of built-in statistics and plotting and practice with weather station data from NOAA
  Workshop 3a – pandas.ipynb

NetCDF4

- **NetCDF** = - efficient storage of multidimensional data
  - accessible with (almost) all programming languages
  - machine-independent (portable)
  - easy access of a subset
  - description of data included

NetCDF4

- Import data with: nc.Dataset('netcdf file')

NetCDF4

- Import data with: nc.Dataset('netcdf file')
- Data.variables

NetCDF4

- Import data with: nc.Dataset(‘netcdf file’)
- Data.variables

In [25]:

```
longitude = winddata.variables['lon'][:]
latitude = winddata.variables['lat'][:]
time = winddata.variables['time'][:]
```

NetCDF4

- Import data with: nc.Dataset('netcdf file')
- Data.variables

In [25]:
```
longitude = winddata.variables['lon'][:]
latitude = winddata.variables['lat'][:]
time = winddata.variables['time'][:]
```

Tutorial: practice with importing and interpreting output of wind model of the KNMI
Workshop 3b – netcdf4.ipynb

xarray

- Higher level interface for dealing with NetCDF files

xarray

- Higher level interface for dealing with NetCDF files
- Similar to pandas, i.e. label bases indexing, built-in functions for analysis and plotting

xarray

- Higher level interface for dealing with NetCDF files
- Similar to pandas, i.e. label bases indexing, built-in functions for analysis and plotting
- Two structures:
  - **DataArray** = 1 variable field, e.g. temperature(lon,lat,z,t)
  - **DataSet** = multiple DataArrays, that may share coordinates, e.g. temperature(lon,lat,z,t) and salinity(lon,lat,z,t)

xarray

- For each variable, the dimensions are labelled:

  In NumPy, the dimensions were just 0,1,2, …
  here you can give them names, e.g., lon, lat, time

xarray

- For each variable, the dimensions are labelled:

  In NumPy, the dimensions were just 0,1,2, …
  here you can give them names, e.g., lon, lat, time

Easy to extract variables at certain positions or time and perform computations on them

- Tutorial: practice with temperature data from NASA
  Workshop 3c – xarray.ipynb