cw1-b-dane-dataFrame-basic

March 15, 2022

[3]:
```
# dataFrame is a fundamental type/concept for dataAnalysis with Python/pandas
# df corresponds to 2D decision/data table concept. Each column can have␣
 ↪different type
# both rows and columns have their indexes
# df can be viewed as a dictionary of series that share the same index
# (also higher dimensional tensors than 2D can be represented by df with use of␣
 ↪hierarchical index)

import pandas as pd

# manual initialisation of df (with dictionary)
dataDic = {'city':['Warsaw', 'Warsaw', 'Kraków', 'Kraków' ], 'year':
 ↪[2020,2021,2021,2022], 'population':[1.7,1.8,0.7,0.8]}
df = pd.DataFrame(dataDic)
# (index is default and column order as given)
df
```

[3]:

	city	year	population
0	Warsaw	2020	1.7
1	Warsaw	2021	1.8
2	Kraków	2021	0.7
3	Kraków	2022	0.8

[4]:
```
# can provide new order of columns and new columns (that will be filled with␣
 ↪NaN)
pd.DataFrame(dataDic, columns = ['population','city','area'])
```

[4]:

	population	city	area
0	1.7	Warsaw	NaN
1	1.8	Warsaw	NaN
2	0.7	Kraków	NaN
3	0.8	Kraków	NaN

[5]:
```
# can provide index (must be the same length as the data)
df2 = pd.DataFrame(dataDic, index = ['a','b','c','d'])
df2
```

[5]:
```
     city  year  population
a  Warsaw  2020         1.7
b  Warsaw  2021         1.8
c  Kraków  2021         0.7
d  Kraków  2022         0.8
```

[6]:
```
# name and index name
df2.name = 'Population'
df2.index.name = 'case'
df2
```

[6]:
```
         city  year  population
case
a      Warsaw  2020         1.7
b      Warsaw  2021         1.8
c      Kraków  2021         0.7
d      Kraków  2022         0.8
```

[7]:
```
df2.columns
```

[7]:
```
Index(['city', 'year', 'population'], dtype='object')
```

[8]:
```
# a bcolumn can be selected by name or as df's attribute (only for valid␣
 ↪variable names - e.g. without spaces, etc.)
df['population']
```

[8]:
```
0    1.7
1    1.8
2    0.7
3    0.8
Name: population, dtype: float64
```

[9]:
```
df.population
```

[9]:
```
0    1.7
1    1.8
2    0.7
3    0.8
Name: population, dtype: float64
```

[10]:
```
# the column name cannot overwrite the build-in method of the df:
df3 = pd.DataFrame(dataDic, columns = ['columns','year','pop'])
df3
```

[10]:
```
  columns  year  pop
0     NaN  2020  NaN
1     NaN  2021  NaN
```

```
2     NaN  2021  NaN
3     NaN  2022  NaN
```

[11]:
```
df3.columns
```

[11]:
```
Index(['columns', 'year', 'pop'], dtype='object')
```

[12]:
```
# the overwriting attribute name can be retrieved with the second formb
df3['columns']
```

[12]:
```
0    NaN
1    NaN
2    NaN
3    NaN
Name: columns, dtype: object
```

[13]:
```
# retrieving rows with 'loc'
df2.loc['a']
```

[13]:
```
city          Warsaw
year            2020
population       1.7
Name: a, dtype: object
```

[14]:
```
# columns can be assigned new value (scalar or vector)
df3['pop'] = 'over500K'
df3
```

[14]:
```
  columns  year       pop
0     NaN  2020  over500K
1     NaN  2021  over500K
2     NaN  2021  over500K
3     NaN  2022  over500K
```

[15]:
```
# column can be assigned a sequence of proper length
import numpy as np
df2['year'] = np.arange(2019,2023)
df2
```

[15]:
```
        city  year  population
case
a     Warsaw  2019         1.7
b     Warsaw  2020         1.8
c     Kraków  2021         0.7
d     Kraków  2022         0.8
```

[16]:
```
# column values can be assigned with a Series by index alignment (gaps will be␣
↪NaNs)
df2['population'] = pd.Series(['1.6','0.6'], index = ['a','c'])
df2
```

[16]:

case	city	year	population
a	Warsaw	2019	1.6
b	Warsaw	2020	NaN
c	Kraków	2021	0.6
d	Kraków	2022	NaN

[17]:
```
# column deletion with 'del' (as in dictionary)
df2['historical capital'] = df2.city == 'Kraków'
df2
```

[17]:

case	city	year	population	historical capital
a	Warsaw	2019	1.6	False
b	Warsaw	2020	NaN	False
c	Kraków	2021	0.6	True
d	Kraków	2022	NaN	True

[18]:
```
del df2['historical capital']
df2
```

[18]:

case	city	year	population
a	Warsaw	2019	1.6
b	Warsaw	2020	NaN
c	Kraków	2021	0.6
d	Kraków	2022	NaN

[19]:
```
# the column returned by indexing is not a copy but a 'view' that enables␣
↪in-place modifiction.
# A copy can be obtained with 'copy' method of Series

# df can be also initialised as: dict/list of dict/series (4 variants) and list␣
↪of lists/tuples, numpy 2d barray, etc.
```

[20]:
```
# transposition
df2.T
```

[20]:

case	a	b	c	d
city	Warsaw	Warsaw	Kraków	Kraków
year	2019	2020	2021	2022
population	1.6	NaN	0.6	NaN

[21]:
```
# transposition does not modify the df
df2
```

[21]:
```
         city  year population
case
a      Warsaw  2019        1.6
b      Warsaw  2020        NaN
c      Kraków  2021        0.6
d      Kraków  2022        NaN
```

[22]:
```
# columns' name
df2.columns.name = 'attributes'
df2
```

[22]:
```
attributes    city  year population
case
a           Warsaw  2019        1.6
b           Warsaw  2020        NaN
c           Kraków  2021        0.6
d           Kraków  2022        NaN
```

[23]:
```
# values are returned as ndarray
df2['year'].values
```

[23]:
```
array([2019, 2020, 2021, 2022])
```

[24]:
```
#(with the most specific type of objects)
df2.values
```

[24]:
```
array([['Warsaw', 2019, '1.6'],
       ['Warsaw', 2020, nan],
       ['Kraków', 2021, '0.6'],
       ['Kraków', 2022, nan]], dtype=object)
```

[26]:
```
# index is immutable
ser = pd.Series(range(3), index = ['alfa','beta','gamma'])
ind = ser.index
ind
# ind['alfa'] = 'alpha' <- error (immutable)
```

[26]:
```
Index(['alfa', 'beta', 'gamma'], dtype='object')
```

[33]:
```
# index, besides being array-like, is also set-like
df2, 'b' in df2.index
```

[33]:
```
(attributes    city  year population
 case
```

```
a           Warsaw  2019        1.6
b           Warsaw  2020        NaN
c           Kraków  2021        0.6
d           Kraków  2022        NaN,
True)
```

[38]:
```
# index can contain duplicates
nonUniqueIndex = pd.Index([1,3,3,5])
nonUniqueIndex
# selections with non-unique labels return all occurences for such label
df2.index = nonUniqueIndex
df2.loc[3]
# various set-like operations can be done on indexes:
# append, difference, union, intersection, drop, insert, is_unique,␣
 ↪is_monotonic, unique, delete, etc.
```

[38]:
```
attributes    city  year population
3           Warsaw  2020        NaN
3           Kraków  2021        0.6
```