Exercises 1 - example solutions

In [1]:
```
import numpy as N
import pandas as P
```

1.1

In [2]:
```
# (assuming that numbers are one-based indices)

a = P.Series([5, 8, 7, 6, 7, 8])
b = P.Series([1.3, 2.1, 1.7, 1.1, 1.4, 2.3])
c = P.Series(['y', 'y', 'n', 'y', 'n', 'n'])

#

df = P.concat([a, b, c], axis=1)
print(df, end="\n\n")

# 1.

print(df.iloc[2, 1], end="\n\n")

# 2.

print(df.iloc[3,], end="\n\n")

# 3.

print(df.iloc[1:5, -2:], end="\n\n")

# 4.

print(df.transpose(), end="\n\n")
```

```
   0    1  2
0  5  1.3  y
1  8  2.1  y
2  7  1.7  n
3  6  1.1  y
4  7  1.4  n
5  8  2.3  n

1.7

0      6
1    1.1
2      y
Name: 3, dtype: object

     1  2
1  2.1  y
2  1.7  n
3  1.1  y
4  1.4  n

     0    1    2    3    4    5
0    5    8    7    6    7    8
1  1.3  2.1  1.7  1.1  1.4  2.3
2    y    y    n    y    n    n
```

1.2

In [3]:
```
# documentation from allbp.names
columns = ['age',
           'sex',
           'on thyroxine',
           'query on thyroxine',
           'on antithyroid medication',
           'sick',
           'pregnant',
           'thyroid surgery',
           'I131 treatment',
           'query hypothyroid',
           'query hyperthyroid',
           'lithium',
           'goitre',
           'tumor',
           'hypopituitary',
           'psych',
           'TSH measured',
           'TSH',
           'T3 measured',
           'T3',
           'TT4 measured',
           'TT4',
           'T4U measured',
           'T4U',
           'FTI measured',
           'FTI',
           'TBG measured',
           'TBG',
           'referral source',
           'CLASS']
categorical_idx = list(range(1, 16)) + [16, 18, 20, 22, 24, 26, 28, 29]
quantitative_idx = [0, 17, 19, 21, 23, 25, 27]
```

```
# 2.

df = P.read_csv('allbp.data',
                na_values=['?'],
                names=columns)

# remove extra characters at the end of line
df.CLASS = df.CLASS.str.split('.', expand=True).iloc[:, 0]

# convert to categorical
for i in categorical_idx:
    name = columns[i]
    df[name] = P.Categorical(df[name])

# 3.

print(df.shape, end="\n\n")

# 2800 observations, 30 variables

# 4.

print(df.isna().sum(), end="\n\n")

# one value is missing from 'age'
# hundreds of values are missing from 'sex', 'TSH', 'T3', etc.
```

```
(2800, 30)

age                            1
sex                          110
on thyroxine                   0
query on thyroxine             0
on antithyroid medication      0
sick                           0
pregnant                       0
thyroid surgery                0
I131 treatment                 0
query hypothyroid              0
query hyperthyroid             0
lithium                        0
goitre                         0
tumor                          0
hypopituitary                  0
psych                          0
TSH measured                   0
TSH                          284
T3 measured                    0
T3                           585
TT4 measured                   0
TT4                          184
T4U measured                   0
T4U                          297
FTI measured                   0
FTI                          295
TBG measured                   0
TBG                         2800
referral source                0
CLASS                          0
dtype: int64
```

1.3

In [4]:
```
indicator_idx = list(range(2, 16)) + [16, 18, 20, 22, 24, 26]
quantitative_idx = [17, 19, 21, 23, 25, 27]

# 1.

counts = (df.iloc[:, indicator_idx] == 't').sum(axis=0)
result = counts / df.shape[0]
print(result, end="\n\n")

# 2.

subset = df.iloc[:, quantitative_idx]
result = (subset**2).sum(axis=0) / subset.notna().sum(axis=0)
print(result, end="\n\n")

# 3.

result = (df['T3'] / df['TT4']).mean()
print(result, end="\n\n")
```

```
on thyroxine                 0.117857
query on thyroxine           0.014286
on antithyroid medication    0.012143
sick                         0.039286
pregnant                     0.014643
thyroid surgery              0.013929
I131 treatment               0.017143
query hypothyroid            0.058214
query hyperthyroid           0.061786
lithium                      0.005000
goitre                       0.008929
tumor                        0.025357
hypopituitary                0.000357
psych                        0.048214
TSH measured                 0.898571
T3 measured                  0.791071
TT4 measured                 0.934286
T4U measured                 0.893929
FTI measured                 0.894643
TBG measured                 0.000000
dtype: float64

TSH      481.725148
T3         4.780147
TT4    13148.934755
T4U        1.033601
FTI    13354.902248
TBG             NaN
dtype: float64

0.019673501919946535
```

1.4

In [5]:
```
# 1.

df = P.read_csv('purchases.csv')
```

```
# 2.

print(df.sex.value_counts(), end="\n\n")
df.loc[df.sex == 'nale', 'sex'] = 'male'

print(df.location.value_counts(), end="\n\n")
df.loc[df.location == '33100', 'location'] = 'Tampere'
df.loc[df.location == '20100', 'location'] = 'Turku'

print(df.describe(), end="\n\n")
df.loc[df.retention_time < 0, 'retention_time'] = N.NaN

# 3.

df.purchases.fillna(0, inplace=True)

# 4.

simple = df.copy()
simple.retention_time = simple.retention_time.fillna(simple.retention_time.median()

# 5.

bonus = df.copy()
fn = lambda s: s.fillna(s.median())
bonus.retention_time = bonus.groupby(['sex', 'location']).retention_time.transform(
```

```
male      112
female     87
nale        1
Name: sex, dtype: int64

Helsinki    92
Turku       70
Tampere     27
33100        8
20100        3
Name: location, dtype: int64

        purchases  retention_time
count  191.000000      177.000000
mean     5.010471       48.879661
std      2.323766       46.141814
min      0.000000      -14.900000
25%      3.000000       13.600000
50%      5.000000       32.900000
75%      7.000000       72.400000
max     13.000000      237.800000
```

1.5

In [6]:

```
# (these are open-ended questions; the code below is just an example what could be

# 1.

df = P.read_csv('bikes.data')

# 2.

# check obviously invalid values
```

```
print((df.duration < 0).value_counts(), end="\n\n")
print((df.distance < 0).value_counts(), end="\n\n")

# check possibly invalid values

# (assistance should stop at 25 km/h)
print(((df.distance / df.duration) > 7).value_counts(), end="\n\n")

# check values that might indicate irrelevant records

print((df.duration == 0).value_counts(), end="\n\n")
print((df.distance == 0).value_counts(), end="\n\n")

print((df.distance < 100).value_counts(), end="\n\n")

# 3.

# select trips with more than 100 m and 60 s

df = df.loc[(df.distance > 100) & (df.duration > 60), :]
```

```
False    1774
Name: duration, dtype: int64

False    1735
True       39
Name: distance, dtype: int64

False    1768
True        6
dtype: int64

False    1774
Name: duration, dtype: int64

False    1517
True      257
Name: distance, dtype: int64

False    1425
True      349
Name: distance, dtype: int64
```