Python - Analiza danych z modułem PANDAS

www.udemy.com (http://www.udemy.com) (R)

S02-L004 - LAB - Atrybuty Data Series

1. Zaimportuj moduły: pandas, numpy, matplotlib (tylko pyplot), math i nadaj im standardowe aliasy

2. Zaimportuj moduł random z aliasem rnd, a potem:
   - Zadeklaruj zmieną dataAsFloatList jako listę.
   - Napisz pętlę, która 100000 doda do listy element wyliczony wzorem i*rnd.random() gdzie i to kolejny numer pętli
   - Następnie w oparciu o listę dataAsFloatList utwórz obiekt series i zapisz go w zmiennej dataAsFloatSeries.

3. Wyświetl następujące atrybuty dataAsFloatSeries:
   - size
   - nbytes
   - shape
   - axes
   - dtype
   - index
   - is_unique
   - is_monotonic

4. Powtórz kroki z punktu (2) z drobnymi zmianami:
   - Zadeklaruj zmieną dataAsStringList jako listę.
   - Napisz pętlę, która 100000 doda do listy element wyliczony wzorem str(i*rnd.random()) gdzie i to kolejny numer pętli
   - Następnie w oparciu o listę dataAsStringList utwórz obiekt series i zapisz go w zmiennej dataAsStringSeries.

5. Podobnie jak poprzednio wyświetl atrybuty dataAsStringSeries:
   - size
   - nbytes
   - dtype

6. Czy Twoim zdaniem nbytes dla obu obiektów Data Series powinny być takie same czy inne? A jakie są?

7. Wyjaśnienie dla punktu (6). Atrybut nbytes "szacuje zajętość pamięci bardzo z grubsza". Ta metoda jest szybka ale niedokładna. Jeśli chcesz dokładnie policzyć ilość zajmowanej pamięci skorzystaj z metody (metody dokładnie omawiamy w kolejnej lekcji):

```
dataAsFloatSeries.memory_usage(deep=True)
dataAsStringSeries.memory_usage(deep=True)
```

Rozwiązania:

Poniżej znajdują się propozycje rozwiązań zadań. Prawdopodobnie istnieje wiele dobrych rozwiązań, dlatego jeżeli rozwiązujesz zadania samodzielnie, to najprawdopodobniej zrobisz to inaczej, może nawet lepiej :) Możesz pochwalić się swoimi rozwiązaniami w sekcji Q&A

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math as math
```

In [2]:
```
import random as rnd
dataAsFloatList = []
for i in range(100000):
    dataAsFloatList.append(i*rnd.random())
dataAsFloatSeries = pd.Series(dataAsFloatList)
```

In [3]:
```
print('size: \t',dataAsFloatSeries.size)
print('nbytes:\t',dataAsFloatSeries.nbytes)
print('shape:\t',dataAsFloatSeries.shape)
print('axes:\t',dataAsFloatSeries.axes)
print('dtype:\t',dataAsFloatSeries.dtype)
print('index:\t',dataAsFloatSeries.index)
print('unique:\t',dataAsFloatSeries.is_unique)
print('monotonic:\t',dataAsFloatSeries.is_monotonic)
```

```
size:    100000
nbytes:  800000
shape:   (100000,)
axes:    [RangeIndex(start=0, stop=100000, step=1)]
dtype:   float64
index:   RangeIndex(start=0, stop=100000, step=1)
unique:  True
monotonic:       False
```

In [4]:
```
import random as rnd
dataAsStringList = []
for i in range(100000):
    dataAsStringList.append(str(i*rnd.random()))
dataAsStringSeries = pd.Series(dataAsStringList)
```

In [5]:
```
print('size: \t',dataAsStringSeries.size)
print('nbytes:\t',dataAsStringSeries.nbytes)
print('dtype:\t',dataAsStringSeries.dtype)
```

```
size:    100000
nbytes:  800000
dtype:   object
```

In [6]:
```
dataAsFloatSeries.memory_usage(deep=True)
```

Out[6]:
```
800080
```

In [7]:
```
dataAsStringSeries.memory_usage(deep=True)
```

Out[7]:
```
7434796
```