Лабораторная работа по Pandas №4

Выполнил: студент группы М8О-103М-20 Воробьев А.В.

---

1. Импортировать модули sys numpy pandas:

In [2]:
```
import sys
import numpy as np
import pandas as pd
```

---

2. Создать объект pandas Series из листа, объекта NumPy, и словаря:

In [3]:
```
src_list = list('abcde')
src_arr = np.arange(5)
src_dict = dict(zip(src_list, src_arr))

s1 = pd.Series(src_list)
s2 = pd.Series(src_arr)
s3 = pd.Series(src_dict)

print(f'Made of  list:\n{s1}\n----------')
print(f'Made of  NumPy object:\n{s2}\n----------')
print(f'Made of  dictionary:\n{s3}\n----------')
```

```
Made of  list:
0    a
1    b
2    c
3    d
4    e
dtype: object
----------
Made of  NumPy object:
0    0
1    1
2    2
3    3
4    4
dtype: int32
----------
Made of  dictionary:
a    0
b    1
c    2
d    3
e    4
dtype: int64
----------
```

---

3. Преобразовать объект Series в DataFrame:

In [4]:
```
s = pd.Series({'a': 'one', 'b': 'two', 'c': 'three'})
print(f'Series object:\n{s}\n----------')
```

```
print(f'Start type:\t{type(s)}\n----------')

s = s.to_frame()
print(f'DataFrame object:\n{s}\n----------')
print(f'End type:\t{type(s)}\n----------')
```

```
Series object:
a      one
b      two
c    three
dtype: object
----------
Start type:     <class 'pandas.core.series.Series'>
----------
DataFrame object:
       0
a    one
b    two
c  three
----------
End type:       <class 'pandas.core.frame.DataFrame'>
----------
```

---

4. Создать объект Series, преобразовать в DataFrame и объединить несколько объектов Series в Dataframe:

In [7]:

```
s1 = pd.Series(list('abcdefghij'))
s2 = pd.Series(np.arange(10))

# Вариант 1
df1 = pd.concat([s1, s2], axis=1)
print(f'Вариант 1:\n{df1}\n----------')

# Вариант 2
df2 = pd.DataFrame({'column 1': s1, 'column 2': s2})
print(f'Вариант 2:\n{df2}')
```

```
Вариант 1:
   0  1
0  a  0
1  b  1
2  c  2
3  d  3
4  e  4
5  f  5
6  g  6
7  h  7
8  i  8
9  j  9
----------
Вариант 2:
  column 1  column 2
0        a         0
1        b         1
2        c         2
3        d         3
4        e         4
5        f         5
6        g         6
7        h         7
8        i         8
9        j         9
```

---

5. Присвоить имя индексу объекта Series:

In [37]:
```
s = pd.Series({'a': 1, 'b': 2, 'c': 3})

s.name = 'my_name'

print(f'Series object:\n{s}')
```

```
Series object:
a    1
b    2
c    3
Name: my_name, dtype: int64
```

---

6. Присвоить имя индексу объекта Series. Получить элементы объекта Series A, которых нет в объекте Series B.

In [9]:
```
s1 = pd.Series([1, 2, 3, 4, 5])
print(f'Series A:\n{s1}\n----------')

s2 = pd.Series([4, 5, 6, 7, 8])
print(f'Series B:\n{s2}\n----------')
```

```
Series A:
0    1
1    2
2    3
3    4
4    5
dtype: int64
----------
Series B:
0    4
1    5
2    6
3    7
4    8
dtype: int64
----------
```

6.1. Возвратить вместе с индексами:

In [10]:
```
ans = s1[~s1.isin(s2)]
print(f'Series elements A which are not in B (with indices):\n{ans}\n----------')
```

```
Series elements A which are not in B (with indices):
0    1
1    2
2    3
dtype: int64
----------
```

6.2 Возвратить значения:

In [12]:
```
ans2 = np.setdiff1d(s1, s2, assume_unique=False)
print(f'Series elements A which are not in B (only values):\n{ans2}')
```

```
Series elements A which are not in B (only values):
[1 2 3]
```

---

7. Получить не пересекающиеся элементы в двух объектах Series:

In [39]:
```
s1 = pd.Series(np.random.randint(0, 15, 5))
print(f'Series A:\n{s1}\n----------')

s2 = pd.Series(np.random.randint(5, 20, 5))
print(f'Series B:\n{s2}\n----------')
```

```
Series A:
0     8
1     8
2     4
3    10
4     7
dtype: int32
----------
Series B:
0    10
1    11
2    19
3    19
4     6
dtype: int32
----------
```

7.1. Получить объединенный Series без повтороений:

In [40]:
```
s_union = pd.Series(np.union1d(s1, s2))
print(s_union)
```

```
0     4
1     6
2     7
3     8
4    10
5    11
6    19
dtype: int32
```

7.2. Получить пересекающиеся данные:

In [41]:
```
s_intersect = pd.Series(np.intersect1d(s1, s2))
print(s_intersect)
```

```
0    10
dtype: int32
```

7.3. Отобрать все данные, кроме пересекающихся:

In [42]:
```
ans = s_union[~s_union.isin(s_intersect)]
print(ans)
```

```
0     4
```

```
1     6
2     7
3     8
5    11
6    19
dtype: int32
```

7.4. Возвратить значения:

In [43]:
```
ans2 = np.setxor1d(s1, s2, assume_unique=False)
print(ans2)
```

```
[ 4  6  7  8 11 19]
```

The first is done.

But this is just the beginning...
