Мухаметшин А.Р. ББМО-01-25

```
!git clone https://github.com/neuralcomputer/ML_School.git
```

```
Cloning into 'ML_School'...
remote: Enumerating objects: 94, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 94 (delta 5), reused 0 (delta 0), pack-reused 79 (from 1)
Receiving objects: 100% (94/94), 33.83 MiB | 16.90 MiB/s, done.
Resolving deltas: 100% (29/29), done.
```

```
import pandas as pd # подключим библиотеку
import numpy as np
s = pd.Series(data=[10, "11", ['a',12], 'ppp', 14, 42], # данные
index=[2.1, '2', 'два', 2, 2.1, -2]) # их индексы
s # здесь 5 ячее
```

	0
2.1	10
2	11
два	[a, 12]
2	ppp
2.1	14
-2	42

dtype: object

```
print(s[2]) # здесь вернется подпоследовательность элементов у которых индекс н
print('\n')
print(s['2']) # это ячейка с названием '2', это совершенно другой индекс, отлич
print('\n')
print(s['два']) # это ячейка с названием 'два', третья ячейка, в которой записа
print('\n')
print(s[-2]) # это ячейка с названием -2, пятая ячейка
```

```
ppp


11


['a', 12]


42
```

```
s['hello'] # такого нет
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.12/dist-packages/pandas/core/indexes/base.py in
get_loc(self, key)
   3804         try:
-> 3805             return self._engine.get_loc(casted_key)
   3806         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine._get_loc_duplicates()

index.pyx in pandas._libs.index.IndexEngine._maybe_get_bool_indexer()

index.pyx in pandas._libs.index._unpack_bool_indexer()

KeyError: 'hello'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
                                    3 frames
/usr/local/lib/python3.12/dist-packages/pandas/core/indexes/base.py in
get_loc(self, key)
   3810             ):
   3811                 raise InvalidIndexError(key)
-> 3812             raise KeyError(key) from err
   3813         except TypeError:
   3814             # If we have a listlike key, _check_indexing_error will
raise

KeyError: 'hello'
```

```
s[2.1] # а такой есть, но это не номер, а название индекса
```

	0
2.1	10
2.1	14

dtype: object

```
s
```

	0
2.1	10
2	11
два	[a, 12]
2	ppp
2.1	14
-2	42

dtype: object

```
s[1:3] # срез, вторая и третья ячейки, здесь это номера, а не названия.
```

	0
2	11
два	[a, 12]

dtype: object

```
s[1:2]
```

	0
2	11

dtype: object

```
s[1] # это ошибка, нет элемента с названием 1
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.12/dist-packages/pandas/core/indexes/base.py in
get_loc(self, key)
   3804         try:
-> 3805             return self._engine.get_loc(casted_key)
   3806         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine._get_loc_duplicates()

index.pyx in pandas._libs.index.IndexEngine._maybe_get_bool_indexer()

index.pyx in pandas._libs.index._unpack_bool_indexer()

KeyError: 1

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
```

3 frames

```
/usr/local/lib/python3.12/dist-packages/pandas/core/indexes/base.py in
get_loc(self, key)
   3810             ):
   3811                 raise InvalidIndexError(key)
-> 3812             raise KeyError(key) from err
   3813         except TypeError:
   3814             # If we have a listlike key, _check_indexing_error will
raise

KeyError: 1
```

```
v = pd.Series(data=[10, "11", ['a',12], 'ppp', 14], # данные
index=['a', 'f', 'c', 'd', 'e']) # их индексы
v['a':'d']
# v['a':'g'] # так работать не будет, потому что индекса 'g' нет.
```

	0
a	10
f	11
c	[a, 12]
d	ppp

dtype: object

```
df = pd.DataFrame([[10, 'aaa'], [s, 21], [30, 31]])
df
```

	0	1
0	10	aaa
1	2.1 10 2 11 два [a, 12] 2...	21
2	30	31

Далее: New interactive sheet

```
df = pd.DataFrame([[10, 'aaa'], [s, 21], [30, 31]],
columns=['невторой',2],
index=[1,'1','один'])
df
```

	невторой	2
1	10	aaa
1	2.1 10 2 11 два [a, 12] 2...	21
один	30	31

Далее: New interactive sheet

```
s0=df['невторой']
s1=df['невторой']['один']
s1
```

```
30
```

```
df.columns=['невторой','second']
#df.columns=['index','second']
df
```

	невторой	second
1	10	aaa
1	2.1 10 2 11 два [a, 12] 2...	21
один	30	31

Далее: New interactive sheet

```
df.columns=['столбец 1',2] # изменяем названия столбцов
df['столбец 1']
```

	столбец 1
1	10
1	2.1 10 2 11 два [a, 12] 2...
один	30

dtype: object

```
# а вот с такими названиями работать не будет
#df.столбец 1
df.'столбец 1'
```

```
  File "/tmp/ipython-input-4282008195.py", line 3
    df.'столбец 1'
       ^
SyntaxError: invalid syntax
```

```
df.columns=['index','second']
df['index']
```

	index
1	10
1	2.1 10 2 11 два [a, 12] 2...
один	30

dtype: object

```
# так ошибку не выдает, но выдает что-то не то. Это потому, что index уже опред
df.index
```

```
Index([1, '1', 'один'], dtype='object')
```

```
print(df.index)
print(df.columns)
print(df.values)
print(type(df.values))
```

```
Index([1, '1', 'один'], dtype='object')
Index(['index', 'second'], dtype='object')
[[10 'aaa']
 [2.1         10
  2           11
  два    [a, 12]
  2          ppp
  2.1         14
  -2          42
```

```
  dtype: object  21]
 [30 31]]
<class 'numpy.ndarray'>
```

```
df[1:2]
```

	index	second
1	2.1 10 2 11 два [a, 12] 2...	21

```
df
```

	index	second
1	10	aaa
1	2.1 10 2 11 два [a, 12] 2...	21
один	30	31

Далее: New interactive sheet

```
df.iloc[1,0]
```

	0
2.1	10
2	11
два	[a, 12]
2	ppp
2.1	14
-2	42

dtype: object

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

```
21
```

```
np.random.seed(123)
s = pd.Series(np.random.normal(size=10))
print(s)
ind=s>0
print(ind)
r=s[ind]
```

```
print(r)
```

```
0   -1.085631
1    0.997345
2    0.282978
3   -1.506295
4   -0.578600
5    1.651437
6   -2.426679
7   -0.428913
8    1.265936
9   -0.866740
dtype: float64
0    False
1     True
2     True
3    False
4    False
5     True
6    False
7    False
8     True
9    False
dtype: bool
1    0.997345
2    0.282978
5    1.651437
8    1.265936
dtype: float64
```

```
url = 'https://datahub.io/core/s-and-p-500-companies-financials/r/constituents-
#url = 'https://raw.githubusercontent.com/datasets/s-and-p-500-companies-financ
file='constituents-financials_csv.csv'
data = pd.read_csv(url, sep=',')
data
```

	Symbol	Name	Sector	Price	Price/Earnings	Dividend Yield	Earni
0	MMM	3M	Industrial Conglomerates	152.20	21.286713	0.0199	
1	AOS	A. O. Smith	Building Products	67.30	18.539946	0.0196	
2	ABT	Abbott Laboratories	Health Care Equipment	127.93	16.744764	0.0202	
3	ABBV	AbbVie	Biotechnology	183.90	64.300700	0.0373	
4	ACN	Accenture	IT Consulting & Other Services	384.95	32.294464	0.0170	
...	...	...	...	...	...	...	
498	XYL	Xylem Inc.	Industrial Machinery & Supplies & Components	124.04	35.643677	0.0118	
499	YUM	Yum! Brands	Restaurants	130.50	24.392525	0.0205	
500	ZBRA	Zebra Technologies	Electronic Equipment & Instruments	391.94	53.397820	NaN	
501	ZBH	Zimmer Biomet	Health Care Equipment	109.48	20.893131	0.0087	
502	ZTS	Zoetis	Pharmaceuticals	170.90	32.124058	0.0116	

503 rows × 14 columns

Далее: New interactive sheet

```
data.shape # сколько строк и столбцов?
```

```
(503, 14)
```

```
# выводит первые несколько строк
data.head()
```

	Symbol	Name	Sector	Price	Price/Earnings	Dividend Yield	Earnings/
0	MMM	3M	Industrial Conglomerates	152.20	21.286713	0.0199	
1	AOS	A. O. Smith	Building Products	67.30	18.539946	0.0196	
2	ABT	Abbott Laboratories	Health Care Equipment	127.93	16.744764	0.0202	
3	ABBV	AbbVie	Biotechnology	183.90	64.300700	0.0373	
4	ACN	Accenture	IT Consulting & Other Services	384.95	32.294464	0.0170	

Далее: New interactive sheet

```
# как называются столбцы?
data.columns
```

```
Index(['Symbol', 'Name', 'Sector', 'Price', 'Price/Earnings', 'Dividend Yield',
       'Earnings/Share', '52 Week Low', '52 Week High', 'Market Cap', 'EBITDA',
       'Price/Sales', 'Price/Book', 'SEC Filings'],
      dtype='object')
```

```
# количество строк
len(data)
```

```
503
```

```
# из них строк с компаниями из отрасли Industrials (столбец Sector)
induk=data['Sector']=='Industrial Conglomerates'
sum(induk)
```

```
2
```

```
# группируем по значениям столбцов Sector и
group=data.groupby('Sector')
group.size() # считаем количество строк, оказавшихся в каждой группе
```

Sector	0
Advertising	2
Aerospace & Defense	12
Agricultural & Farm Machinery	1
Agricultural Products & Services	2
Air Freight & Logistics	4
...	...
Tobacco	2
Trading Companies & Distributors	2
Transaction & Payment Processing Services	8
Water Utilities	1
Wireless Telecommunication Services	1

127 rows × 1 columns

dtype: int64

```
data.count()
```

	0
Symbol	503
Name	503
Sector	503
Price	500
Price/Earnings	475
Dividend Yield	404
Earnings/Share	499
52 Week Low	500
52 Week High	500
Market Cap	500
EBITDA	470
Price/Sales	499
Price/Book	468
SEC Filings	503

dtype: int64

```
group.count() # число раз, когда значение группы встречалось
# может быть меньше, чем число строк, если значение было пропущено в файле
```

Sector	Symbol	Name	Price	Price/Earnings	Dividend Yield	Earnings/Share
Advertising	2	2	2	2	2	2
Aerospace & Defense	12	12	12	11	9	12
Agricultural & Farm Machinery	1	1	1	1	1	1
Agricultural Products & Services	2	2	2	2	2	2
Air Freight & Logistics	4	4	4	4	4	4
...	...	...	...	...	...	...
Tobacco	2	2	2	2	2	2
Trading Companies & Distributors	2	2	2	2	2	2
Transaction & Payment Processing Services	8	8	8	8	5	8
Water Utilities	1	1	1	1	1	1
Wireless Telecommunication Services	1	1	1	1	1	1

127 rows × 13 columns

```
#group.Price.median() # медиана для групп по числовому столбцу Price
# тут-то нам и пригодилось обращение к столбцу как к атрибуту
group['Price'].median() # альтернативное написание
```

Sector	Price
Advertising	57.730
Aerospace & Defense	207.790
Agricultural & Farm Machinery	476.560
Agricultural Products & Services	63.680
Air Freight & Logistics	113.905
...	...
Tobacco	91.215
Trading Companies & Distributors	415.650
Transaction & Payment Processing Services	195.065
Water Utilities	124.640
Wireless Telecommunication Services	232.970

127 rows × 1 columns

dtype: float64

```
# каждую группу можно посмотреть, вот состав группы Materials
group.get_group('Tobacco')
```

	Symbol	Name	Sector	Price	Price/Earnings	Dividend Yield	Earnings/Shar
21	MO	Altria	Tobacco	52.23	7.877828	0.0775	6.6
374	PM	Philip Morris International	Tobacco	130.20	20.699522	0.0414	6.2

```
group.Name.unique()
```

Sector	Name
Advertising	[Interpublic Group of Companies (The), Omnicom...
Aerospace & Defense	[Axon Enterprise, Boeing, GE Aerospace, Genera...
Agricultural & Farm Machinery	[Deere & Company]
Agricultural Products & Services	[Archer Daniels Midland, Bunge Global]
Air Freight & Logistics	[C.H. Robinson, Expeditors International, FedE...
...	...
Tobacco	[Altria, Philip Morris International]
Trading Companies & Distributors	[Fastenal, United Rentals]
Transaction & Payment Processing Services	[Corpay, Fidelity National Information Service...
Water Utilities	[American Water Works]
Wireless Telecommunication Services	[T-Mobile US]

127 rows × 1 columns

dtype: object

```
letter="A"
for i in group.Symbol:
  print('{}: {}'.format(i[0],(i[1].str.get(0)==letter).any()))
```

```
Advertising: False
Aerospace & Defense: True
Agricultural & Farm Machinery: False
Agricultural Products & Services: True
Air Freight & Logistics: False
Apparel Retail: False
Apparel, Accessories & Luxury Goods: False
Application Software: True
Asset Management & Custody Banks: True
Automobile Manufacturers: False
Automotive Parts & Equipment: True
Automotive Retail: True
Biotechnology: True
Brewers: False
Broadcasting: False
Broadline Retail: True
Building Products: True
Cable & Satellite: False
Cargo Ground Transportation: False
Casinos & Gaming: False
Commodity Chemicals: False
Communications Equipment: True
```

```
Computer & Electronics Retail: False
Construction & Engineering: False
Construction Machinery & Heavy Transportation Equipment: False
Construction Materials: False
Consumer Electronics: False
Consumer Finance: True
Consumer Staples Merchandise Retail: False
Copper: False
Data Center REITs: False
Data Processing & Outsourced Services: False
Distillers & Vintners: False
Distributors: False
Diversified Banks: False
Diversified Support Services: True
Drug Retail: False
Electric Utilities: True
Electrical Components & Equipment: True
Electronic Components: True
Electronic Equipment & Instruments: False
Electronic Manufacturing Services: False
Environmental & Facilities Services: False
Fertilizers & Agricultural Chemicals: False
Financial Exchanges & Data: False
Food Distributors: False
Food Retail: False
Footwear: False
Gas Utilities: True
Gold: False
Health Care Distributors: False
Health Care Equipment: True
Health Care Facilities: False
Health Care REITs: False
Health Care Services: False
Health Care Supplies: True
Health Care Technology: False
Heavy Electrical Equipment: False
```

```
# своя функция, которая вычитает минимальное значение из максимального
def max_min(arr):
  return arr.max() - arr.min()
# считаем по группам для столбца Price свою функцию и среднее.
result=group.Price.agg([max_min, 'mean'])
result
```

Sector	max_min	mean
Advertising	58.12	57.730000
Aerospace & Defense	1276.83	361.176667
Agricultural & Farm Machinery	0.00	476.560000
Agricultural Products & Services	24.90	63.680000
Air Freight & Logistics	165.38	148.042500
...	...	...
Trading Companies & Distributors	684.82	415.650000
& Payment Processing Services	473.96	243.843750

Далее: New interactive sheet

```
data['Price']
```

	Price
0	152.20