pandas.pydata.org

*PANDAS = **Pan**el **Da**ta*

Основные типы данных в PANDAS

- DataFrame – 2D, таблица
- Series – 1D, столбец таблицы

PANDAS: ввод/вывод

- ✔ Pickle
- ✔ Текст (csv и т.п.)
- ✔ Clipboard
- ✔ Excel
- ✔ JSON
- ✔ HTML
- ✔ HDFStore: PyTables (HDF5)
- ✔ Feather
- ✔ Parquet
- ✔ SAS
- ✔ SQL
- ✔ Google BigQuery
- ✔ STATA

DataFrame

```
>>> import pandas as pd

>>> letters = ['b','c','a','o','l']
... numbers = [98, 55, 77, 68, 73]

>>> qqq=list(zip(letters, numbers))

>>> qqq
[('b', 98), ('c', 55), ('a', 77), ('o', 68), ('l', 73)]

>>> pd.DataFrame(data=qqq)
   0   1
0  b  98
1  c  55
2  a  77
3  o  68
4  l  73

>>> pd.DataFrame(data=qqq, columns=['A','B'])
   A   B
0  b  98
1  c  55
2  a  77
3  o  68
4  l  73
```

DataFrame

```
>>> data = pd.read_csv('4.5_year.csv')
```

```
>>> data
```

	time	latitude	longitude	depth	mag	magType	nst	...	horizontalError	depthError	magError	magNst	status
0	2017-11-08T04:50:51.960Z	24.0923	91.2855	29.51	4.9	mb	NaN	...	4.0	4.4	0.057	96.0	reviewed
1	2017-11-08T07:23:21.810Z	6.2840	125.9510	119.51	5.1	mb	NaN	...	8.5	5.3	0.041	194.0	reviewed
2	2017-11-08T07:35:11.540Z	-21.8761	-179.3823	593.71	5.7	mww	NaN	...	9.3	3.6	0.073	18.0	reviewed
3	2017-11-08T07:54:41.860Z	-21.6877	168.5539	8.29	5.0	mb	NaN	...	5.3	4.0	0.088	41.0	reviewed
4	2017-11-08T10:29:01.060Z	-17.9528	-178.3860	572.39	4.5	mb	NaN	...	9.6	8.9	0.085	41.0	reviewed
5	2017-11-08T15:28:23.300Z	14.9382	-94.2246	10.00	4.6	mb	NaN	...	6.3	1.9	0.052	110.0	reviewed
6	2017-11-08T18:24:43.100Z	34.2187	141.6414	17.67	4.6	mb	NaN	...	6.2	3.9	0.080	47.0	reviewed
7	2017-11-08T19:17:06.960Z	-6.5071	104.6146	35.00	5.0	mww	NaN	...	8.2	1.9	0.075	17.0	reviewed
8	2017-11-08T20:07:11.750Z	-21.6942	168.4960	9.61	4.7	mb	NaN	...	5.2	4.8	0.099	31.0	reviewed

```
>>> data.describe()
```

	latitude	longitude	depth	mag	nst	gap	dmin	rms	horizontalError	depthError	magError	magN
count	7140.000000	7140.000000	7140.000000	7140.000000	77.000000	7096.000000	7094.000000	7140.000000	7099.000000	7140.000000	6996.000000	7067.0000
mean	-0.512592	39.246295	75.394602	4.816458	35.857143	92.511594	4.467442	0.843987	8.375568	4.107207	0.090806	63.0544
std	28.165799	121.948322	138.467815	0.375673	15.799496	45.672022	5.575528	0.250141	3.139039	3.119286	0.047423	80.4340
min	-65.885300	-179.994400	-1.080000	4.500000	11.000000	8.000000	0.001376	0.080000	0.100000	0.000000	0.019000	2.0000
25%	-20.343725	-71.735575	10.000000	4.600000	30.000000	58.000000	1.438250	0.670000	6.300000	1.900000	0.058000	19.0000
50%	-5.739350	96.121450	13.720000	4.700000	36.000000	87.000000	2.741500	0.830000	8.000000	2.000000	0.081000	34.0000
75%	19.398708	142.555600	63.210000	4.900000	38.000000	121.000000	5.330750	1.000000	10.200000	6.200000	0.113000	71.0000
max	86.893200	179.995200	670.810000	8.200000	90.000000	340.000000	58.528000	1.600000	32.800000	31.610000	0.910000	854.0000

DataFrame.describe(percentiles=None, include=None, exclude=None)

DataFrame

```
>>> data.shape
(7140, 22)

>>> data.columns
Index(['time', 'latitude', 'longitude', 'depth', 'mag', 'magType', 'nst',
       'gap', 'dmin', 'rms', 'net', 'id', 'updated', 'place', 'type',
       'horizontalError', 'depthError', 'magError', 'magNst', 'status',
       'locationSource', 'magSource'],
      dtype='object')

>>> data.values
array([['2017-11-08T04:50:51.960Z', 24.0923, 91.2855, ..., 'reviewed',
        'us', 'us'],
       ['2017-11-08T07:23:21.810Z', 6.284, 125.95100000000001, ...,
        'reviewed', 'us', 'us'],
       ['2017-11-08T07:35:11.540Z', -21.8761, -179.3823, ..., 'reviewed',
        'us', 'us'],
       ...,
       ['2018-11-08T08:16:13.030Z', -27.8369, -71.2439, ..., 'reviewed',
        'us', 'us'],
       ['2018-11-08T08:30:16.710Z', 2.3671, 127.2763, ..., 'reviewed',
        'us', 'us'],
       ['2018-11-08T09:02:48.300Z', -15.2742, -173.305, ..., 'reviewed',
        'us', 'us']], dtype=object)

>>> type(data.values)
numpy.ndarray
```

DataFrame

```
>>> data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7140 entries, 0 to 7139
Data columns (total 22 columns):
time               7140 non-null object
latitude           7140 non-null float64
longitude          7140 non-null float64
depth              7140 non-null float64
mag                7140 non-null float64
magType            7140 non-null object
nst                77 non-null float64
gap                7096 non-null float64
dmin               7094 non-null float64
rms                7140 non-null float64
net                7140 non-null object
id                 7140 non-null object
updated            7140 non-null object
place              7140 non-null object
type               7140 non-null object
horizontalError    7099 non-null float64
depthError         7140 non-null float64
magError           6996 non-null float64
magNst             7067 non-null float64
status             7140 non-null object
locationSource     7140 non-null object
magSource          7140 non-null object
dtypes: float64(12), object(10)
memory usage: 1.2+ MB
```

DataFrame

```
>>> data['type'].describe()
count             7140
unique               2
top       earthquake
freq              7078
Name: type, dtype: object

>>> data['type'].unique()
array(['earthquake', 'volcanic eruption'],
dtype=object)
```

DataFrame – доступ по индексу

```
>>> data.iloc[3:5,0:10]

                        time  latitude  longitude   depth  mag magType
nst   gap   dmin   rms
3  2017-11-08T07:54:41.860Z  -21.6877   168.5539    8.29  5.0      mb
NaN  91.0  0.528  0.97
4  2017-11-08T10:29:01.060Z  -17.9528  -178.3860  572.39  4.5      mb
NaN  81.0  3.398  0.82


>>> data.loc[3:5, ('time', 'depth', 'mag')]

                        time   depth  mag
3  2017-11-08T07:54:41.860Z    8.29  5.0
4  2017-11-08T10:29:01.060Z  572.39  4.5
5  2017-11-08T15:28:23.300Z   10.00  4.6
```

DataFrame – фильтрация

```
>>> data[data.mag > 7.8]
>>>
                          time  latitude  longitude   depth  mag magType  nst   ga
1439  2018-01-23T09:31:40.890Z   56.0039  -149.1658   14.06  7.9     mww  NaN  30.
5216  2018-08-19T00:19:40.670Z  -18.1125  -178.1530  600.00  8.2     mww  NaN  13.
5757  2018-09-06T15:49:18.710Z  -18.4743   179.3502  670.81  7.9     mww  NaN  12.
```

Сортировка

```
>>> data.sort_values(['mag', 'time'], ascending=False)
```

	time	latitude	longitude	depth	mag	magType	nst	gap	...	type	horizontalError	depthError	magErr
5216	2018-08-19T00:19:40.670Z	-18.112500	-178.153000	600.00	8.2	mww	NaN	13.0	...	earthquake	9.60	1.90	0.0
5757	2018-09-06T15:49:18.710Z	-18.474300	179.350200	670.81	7.9	mww	NaN	12.0	...	earthquake	9.50	2.80	0.0
1439	2018-01-23T09:31:40.890Z	56.003900	-149.165800	14.06	7.9	mww	NaN	30.0	...	earthquake	6.00	2.80	0.0
6268	2018-09-28T10:02:45.240Z	-0.255600	119.845100	20.00	7.5	mww	NaN	13.0	...	earthquake	5.80	1.80	0.0
2021	2018-02-25T17:44:44.140Z	-6.069900	142.753600	25.21	7.5	mww	NaN	17.0	...	earthquake	7.20	3.00	0.0
1210	2018-01-10T02:51:33.290Z	17.482500	-83.520000	19.00	7.5	mww	NaN	12.0	...	earthquake	5.80	1.70	0.0
5412	2018-08-21T21:31:47.600Z	10.773100	-62.901900	146.82	7.3	mww	NaN	15.0	...	earthquake	7.50	3.40	0.0
96	2017-11-12T18:18:17.180Z	34.910900	45.959200	19.00	7.3	mww	NaN	33.0	...	earthquake	4.90	1.70	0.0
1890	2018-02-16T23:39:39.280Z	16.385500	-97.978700	22.00	7.2	mww	NaN	82.0	...	earthquake	4.30	1.80	0.0
5574	2018-08-29T03:51:56.100Z	-22.029500	170.126200	21.43	7.1	mww	NaN	23.0	...	earthquake	5.80	2.80	0.0

```
>>> d1 = data.groupby('type')
    d1['depth'].mean()
```

```
type
earthquake           76.052924
volcanic eruption     0.239677
Name: depth, dtype: float64
```

```
>>> d1['depth'].agg([np.mean, np.std])
```

type	mean	std
earthquake	76.052924	138.893451
volcanic eruption	0.239677	0.451816

Сводные таблицы

```
>>>: df = pd.DataFrame({'A': ['one', 'one', 'two', 'three'] * 3,
    'B': ['A', 'B', 'C'] * 4,
    'C': ['foo', 'foo', 'foo', 'bar', 'bar', 'bar'] * 2,
    'D': np.random.randn(12),
    'E': np.random.randn(12)})

>>>: df
        A  B    C         D         E
0     one  A  foo -0.969174 -0.671771
1     one  B  foo -0.830265  1.129879
2     two  C  foo  0.212933 -0.147176
3   three  A  bar  1.073545  1.515654
4     one  B  bar  0.616847 -0.726561
5     one  C  bar  0.436268 -0.051810
6     two  A  foo  0.726210  0.614463
7   three  B  foo  0.074684  0.038751
8     one  C  foo  0.346208  0.041024
9     one  A  bar -0.764860  0.716271
10    two  B  bar -0.640135  1.279182
11  three  C  bar -2.125061 -0.518470
```

Сводные таблицы

```
>>>: pd.pivot_table(df, values='E', index=['A', 'B'], columns=['C'])
C             bar       foo
A     B
one   A  0.716271 -0.671771
      B -0.726561  1.129879
      C -0.051810  0.041024
three A  1.515654       NaN
      B       NaN  0.038751
      C -0.518470       NaN
two   A       NaN  0.614463
      B  1.279182       NaN
      C       NaN -0.147176
```

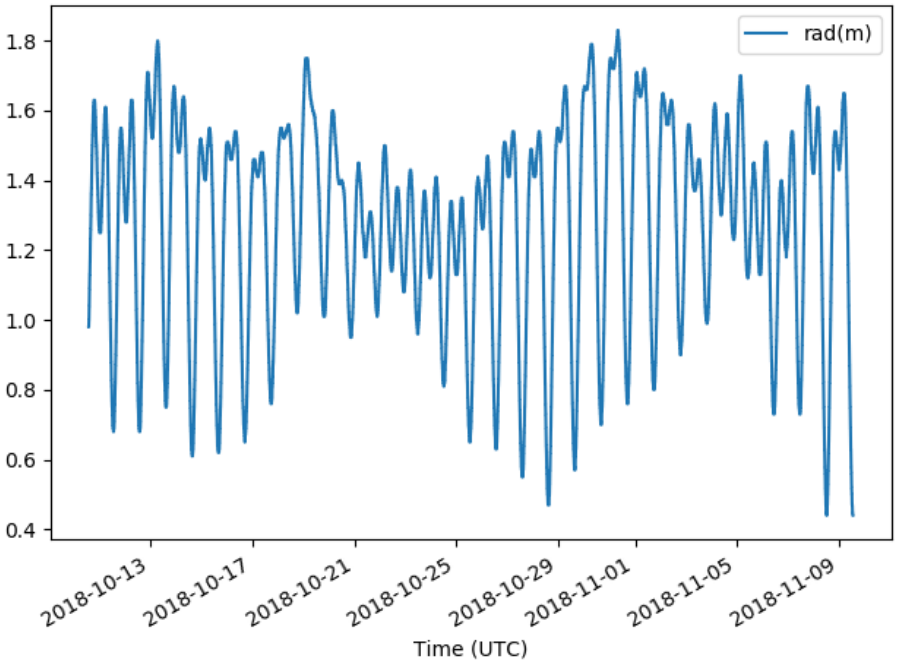
Переименование и удаление стобцов

```
df = pd.DataFrame({'x':[1,3,2], 'y':
[2,4,1]})

d1 = df.rename(columns={'y':'z'})
df.rename(columns={'y':'z'}, inplace
= True)

df.drop(1, axis=0, inplace=True)
del df['x']
df.drop('x', axis=1)
```

Работа с индексами

```
>>> nik = pd.read_html('/tmp/bgraph.htm', header=0, index_col=0)
>>> d = nik[0]
>>> d
```

```
                     rad(m)
Time (UTC)
2018-10-10 14:03:00    0.98
2018-10-10 14:04:00    0.98
2018-10-10 14:05:00    0.98
2018-10-10 14:06:00    0.99
2018-10-10 14:07:00    0.99
2018-10-10 14:08:00    0.99
2018-10-10 14:09:00    0.99
2018-10-10 14:10:00    1.00
```

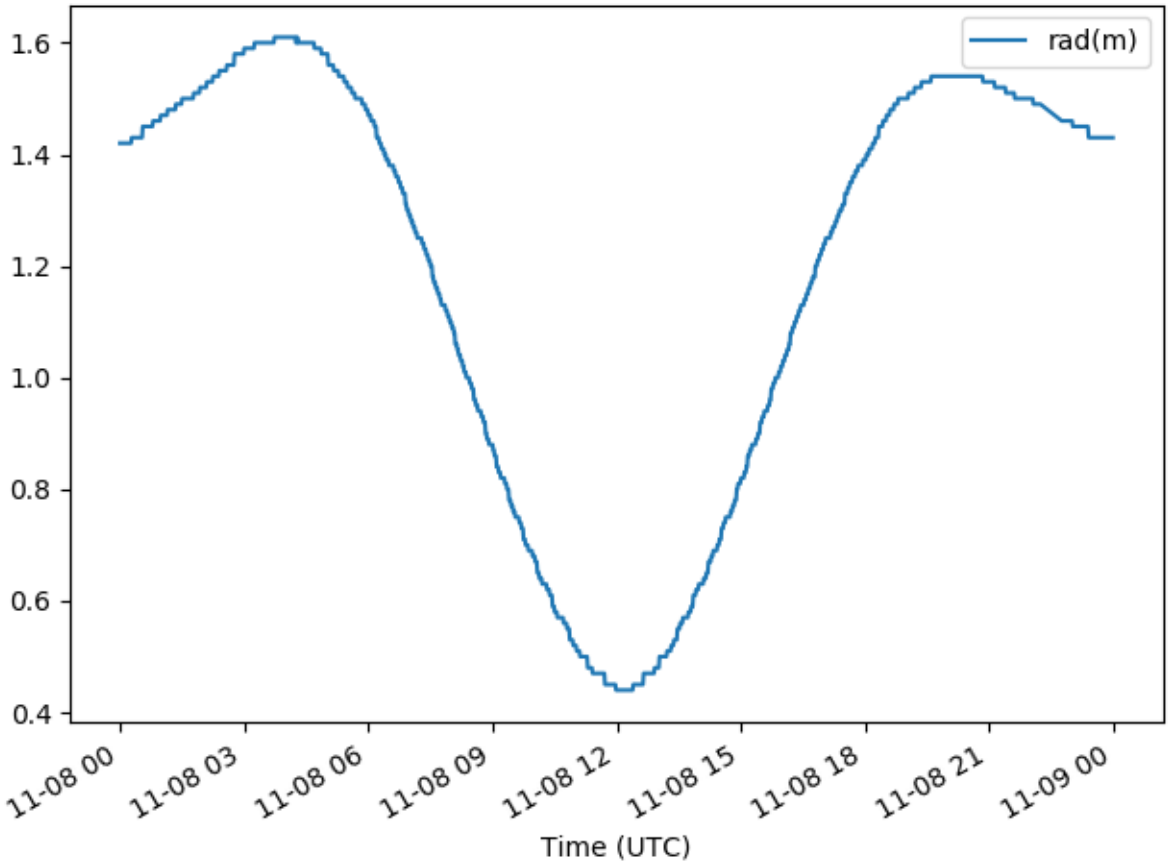



```
>>> d.index
```

```
Index(['2018-10-10 14:03:00', '2018-10-10 14:04:00', '2018-10-10 14:05:00',
       '2018-10-10 14:06:00', '2018-10-10 14:07:00', '2018-10-10 14:08:00',
       '2018-10-10 14:09:00', '2018-10-10 14:10:00', '2018-10-10 14:11:00',
       '2018-10-10 14:12:00',
       '2018-11-09 12:44:00'],
      dtype='object', name='Time (UTC)', length=43091)
```

```
>>> d.index = pd.to_datetime(d.index)
>>> d.index

DatetimeIndex(['2018-10-10 14:03:00', '2018-10-10 14:04:00',
               '2018-10-10 14:05:00', '2018-10-10 14:06:00',
               '2018-11-09 12:43:00', '2018-11-09 12:44:00'],
              dtype='datetime64[ns]', name='Time (UTC)', length=43091, freq=None)
```

```
>>> d['2018-11-08']

                     rad(m)
Time (UTC)
2018-11-08 00:00:00    1.42
2018-11-08 00:01:00    1.42
2018-11-08 00:02:00    1.42
2018-11-08 00:03:00    1.42
2018-11-08 00:04:00    1.42
2018-11-08 00:05:00    1.42
2018-11-08 00:06:00    1.42
2018-11-08 00:07:00    1.42
2018-11-08 00:08:00    1.42
2018-11-08 00:09:00    1.42
2018-11-08 00:10:00    1.42
2018-11-08 00:11:00    1.42
2018-11-08 00:12:00    1.42
```




```
>>> d['2018-11-08'].plot()
 <matplotlib.axes._subplots.AxesSubplot at 0x7f1c12d0d550>
>>> plt.show()
```

rolling: скользящие окна

DataFrame.rolling(window, min_periods=None, center=False,

win_type=None, on=None, axis=<no_default>,

closed=None, step=None, method='single')

```
nik[0].rolling(5000, win_type='triang').mean().plot()
```