Лекция 10

Анализ данных в python

- Основы Jupyter
- Numpy
- Pandas
- Визуализация

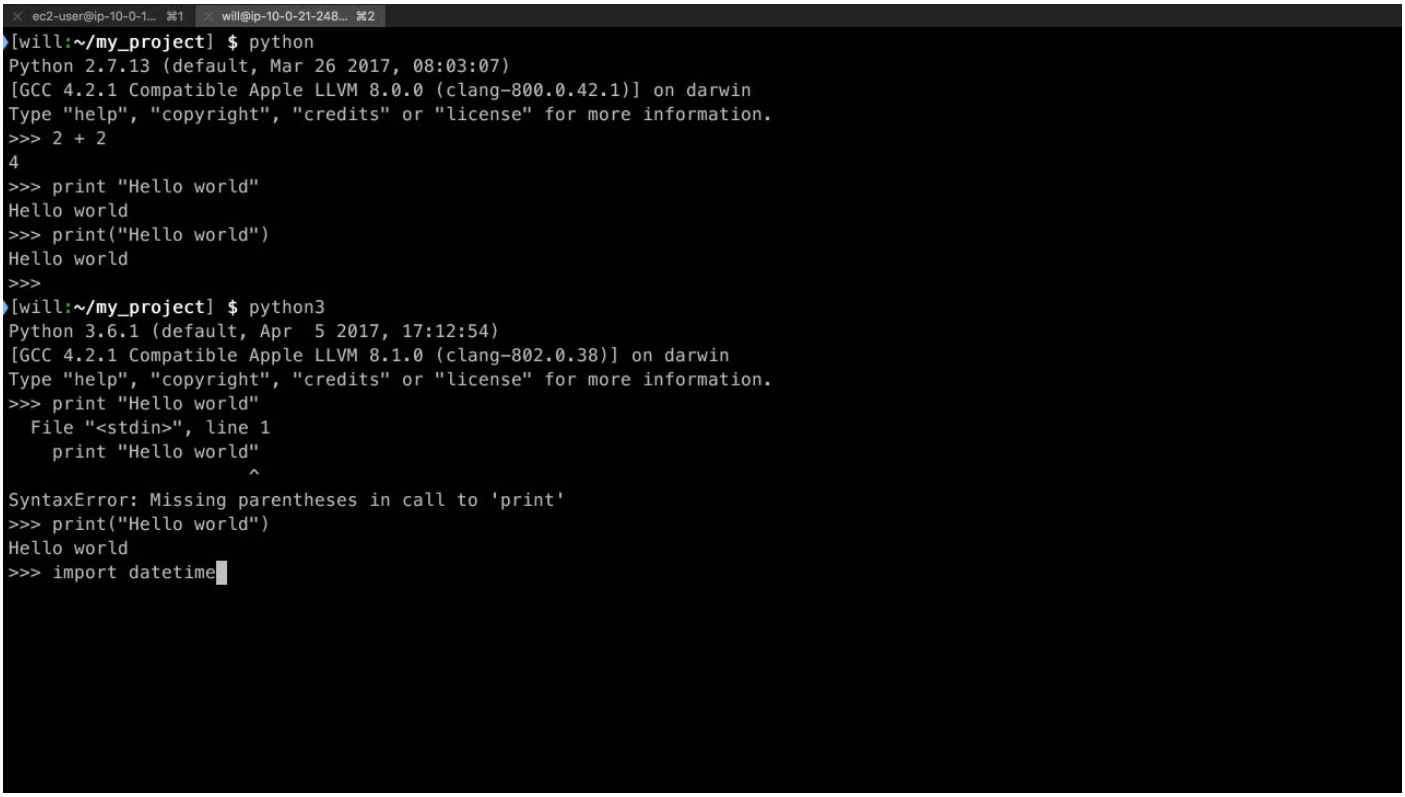
Jupyter Notebook Basics

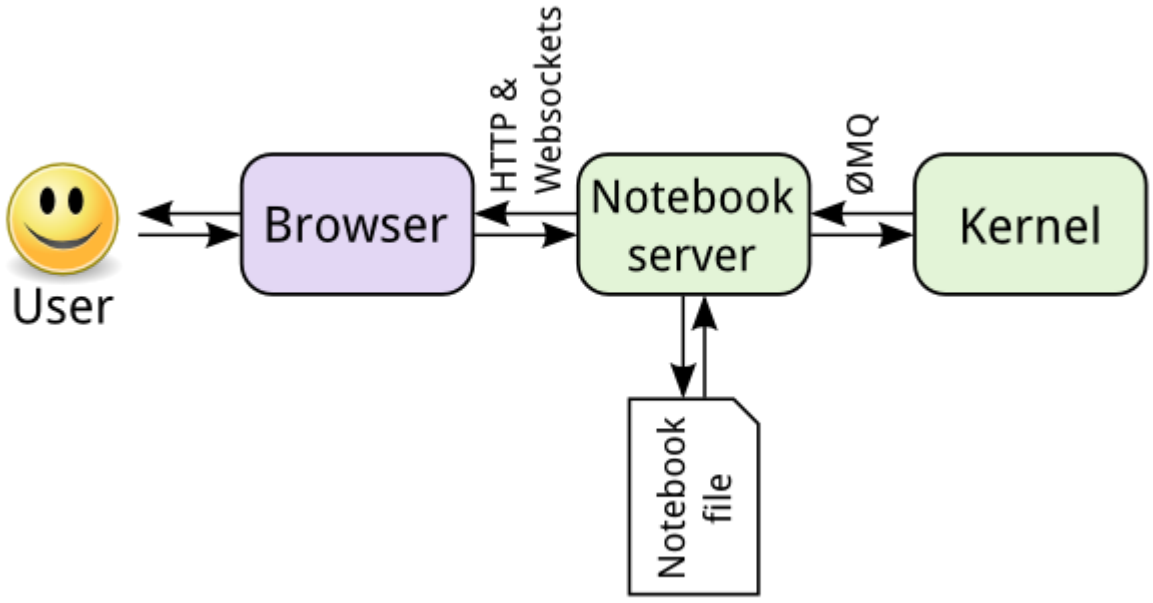
- Что такое Jupyter Notebook и как с ним работать
- Виды ячеек: код, markdown, формулы
- Магические команды
- Shell-команды
- Jupyter в IDE

```
pip install jupyter
jupyter notebook
```

Архитектура

REPL - Read-Evaluate-Print Loop





https://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html
(https://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html)

Jupyter + Cloud

binder colab DATALORE kaggle

Интерактивная среда разработки

In [165]:
```
a = 10
b = 3
a + b
```

Out[165]: 13

In [166]:
```
def is_odd(num):
    return bool(num % 2)

is_odd(17)
```

Out[166]:
```
True
```

In [169]:
```
from math import pi
pi
```

Out[169]:
```
3.1415926536
```

Shortcuts

`Esc` - enables Command Mode
`Enter` - enables Edit Mode
`h` - in Command Mode shows keyboard shortcuts help

In [ ]:
```
a = 10
b = 3
a + b
```

Markdown

Header Level 1

...

Header Level 4

This is bold text
And this is Italic
`Here's an example of Monospace font`

1. Item 1
2. Item 2
    - sub item 1
    - sub item 2
3. Item 3

HTML

LaTeX

$$f'(a) = \lim_{x \to a} \frac{f(x) - f(a)}{x - a}$$

Магические команды

Справка по командам

In [170]:
```
%lsmagic
```

Out[170]:
```
Available line magics:
%alias  %alias_magic  %autoawait  %autocall  %automagic  %autosave  %bookmark
%cd  %clear  %cls  %colors  %conda  %config  %connect_info  %copy  %ddir  %de
bug  %dhist  %dirs  %doctest_mode  %echo  %ed  %edit  %env  %gui  %hist  %his
tory  %killbgscripts  %ldir  %less  %load  %load_ext  %loadpy  %logoff  %logo
n  %logstart  %logstate  %logstop  %ls  %lsmagic  %macro  %magic  %matplotlib
%mkdir  %more  %notebook  %page  %pastebin  %pdb  %pdef  %pdoc  %pfile  %pinf
o  %pinfo2  %pip  %popd  %pprint  %precision  %prun  %psearch  %psource  %pus
hd  %pwd  %pycat  %pylab  %qtconsole  %quickref  %recall  %rehashx  %reload_e
xt  %ren  %rep  %rerun  %reset  %reset_selective  %rmdir  %run  %save  %sc  %
set_env  %store  %sx  %system  %tb  %time  %timeit  %unalias  %unload_ext  %w
ho  %who_ls  %whos  %xdel  %xmode

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%capture  %%cmd  %%debug  %%file  %%html  %%java
script  %%js  %%latex  %%markdown  %%perl  %%prun  %%pypy  %%python  %%python
2  %%python3  %%ruby  %%script  %%sh  %%svg  %%sx  %%system  %%time  %%timeit
%%writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```

In [171]:
```
%quickref
```

In [172]:
```
%alias_magic?
```

In [174]:
```
range?
```

% vs %%

In [175]:
```
# single line commands
%time [x**2 for x in range(1000)][:7];
```

```
Wall time: 500 µs
```

Out[175]:
```
[0, 1, 4, 9, 16, 25, 36]
```

In [176]:
```
%%time
# cell magic command (no blank line or comment at first line)
import time
for _ in range(100):
    time.sleep(0.01)
```

```
Wall time: 1.02 s
```

Работа с кодом

In [181]:
```
%who
```

```
A         B         a         a_observations  a_slice         b         calc_factori
al        columns             df
effective_series              effectiveness   envs    full_df         group    hosp
itals     idx1      idx2      is_odd
mask      np        patients          pd      pi      sbn     sns     time     valu
e
x         y
```

In [182]:
```
%%writefile factorial.py
def calc_factorial(n):
    if n == 1:
        return n
    else:
        return n * calc_factorial(n-1)
```

```
Overwriting factorial.py
```

In [ ]:
```
%ls .
```

In [184]:
```
%pycat factorial.py
```

In [186]:
```
# %load factorial.py
def calc_factorial(n):
    if n == 1:
        return n
    else:
        return n * calc_factorial(n-1)
```

In [187]:
```
%who function
```

```
calc_factorial   is_odd
```

In [188]:
```
calc_factorial(5)
```

Out[188]:
```
120
```

shell-команды

In [189]:
```
!ls .
```

```
0 Jupyter Basics
09 Exceptions & Context Managers
10 Numpy & Pandas
10_numpy_pandas.ipynb
11 Testing
12 Speedup
factorial.py
```

Задание 1

Выведите значение константы `pi`, которую выше импортировали из модуля `math` с двумя знаками после запятой

In [ ]:
```

```

Задание 2

Каково значение переменной окружения `PATH` на вашей системе?

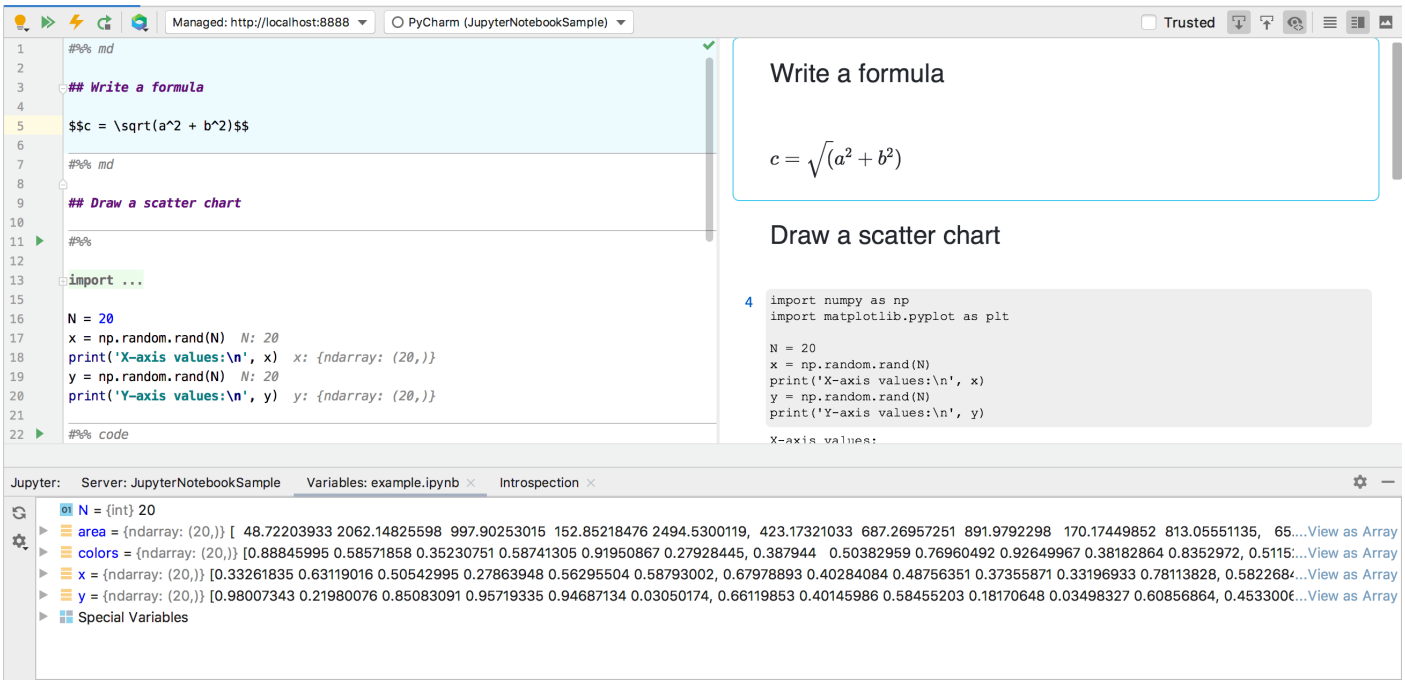
In [ ]:
```

```

Задание 3

Какая версия python установлена в вашем окружении?

```
In [ ]:
```

Jupyter + IDE



NumPy

Библиотека для научных вычислений

- Эффективные многомерные массивы
- Высокоуровневые математические функции
- Возможность интегрировать код на C/C++, Fortran

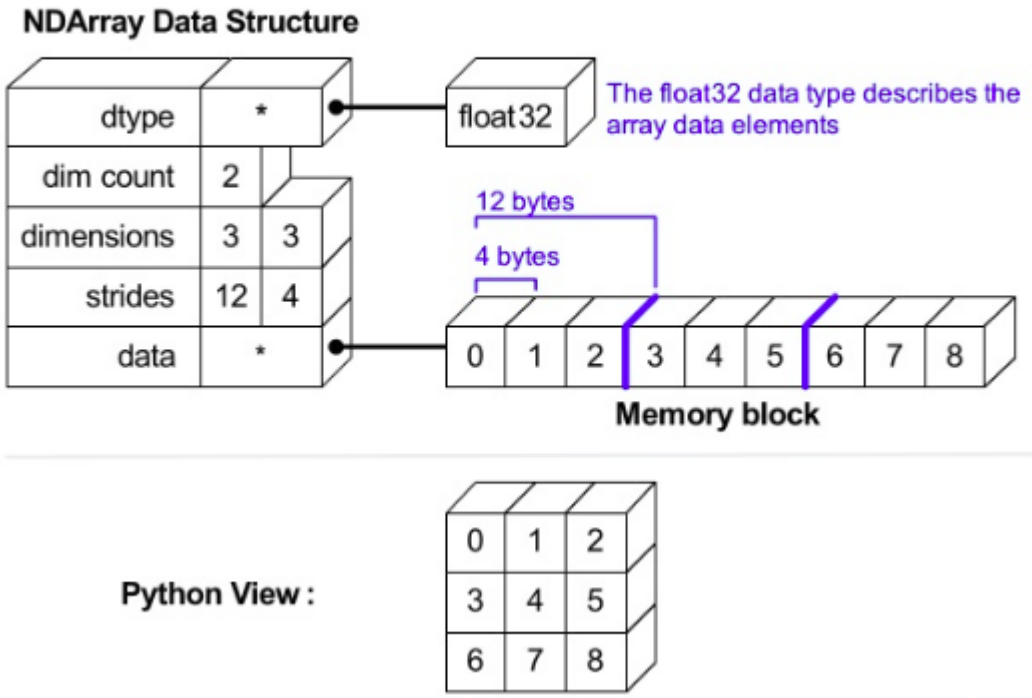
```
In [190]: import numpy as np
          np.__version__
```

```
Out[190]: '1.18.1'
```

numpy.ndarray

Многомерный массив элементов **одного типа** (чаще всего числового), занимающий непрерывный участок памяти

Array Data Structure



In [191]:
```
a = np.array([1, 2, 3, 4, 5, 6, 7, 8])
```

In [192]:
```
a.dtype
```

Out[192]:
```
dtype('int32')
```

In [193]:
```
a.astype(np.float32)
```

Out[193]:
```
array([1., 2., 3., 4., 5., 6., 7., 8.], dtype=float32)
```

In [194]:
```
a.shape
```

Out[194]:
```
(8,)
```

In [195]:
```
a.reshape((2, 4))
```

Out[195]:
```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

In [198]:
```
a.shape = 4, 2
a
```

Out[198]:
```
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
```

In [199]:
```
a.size
```

Out[199]:
```
8
```

In [200]:
```
a.ndim
```

Out[200]:
```
2
```

Специальные виды массивов

In [201]:
```
np.zeros((3, 3))
```

Out[201]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [202]:
```
np.empty(5)
```

Out[202]:
```
array([7.2911220196e-304, 8.2890460585e-317, 4.9406564584e-324,
       0.0000000000e+000, 2.1219957910e-314])
```

In [203]:
```
np.ones((2, 5), dtype=np.int8)
```

Out[203]:
```
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]], dtype=int8)
```

In [204]:
```
np.random.rand(3, 3)
```

Out[204]:
```
array([[0.2148639211, 0.3993791727, 0.2778990899],
       [0.5832160177, 0.9839696927, 0.3610251097],
       [0.7258782609, 0.5974873135, 0.0254303019]])
```

Доступ к элементам

Схоже со списками в python: индексы, слайсы, обход в цикле

In [205]:
```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
A
```

Out[205]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
In [206]: A[0]
```

```
Out[206]: array([1, 2, 3])
```

```
In [207]: A[1, 1]
```

```
Out[207]: 5
```

```
In [208]: A[0:2]
```

```
Out[208]: array([[1, 2, 3],
                 [4, 5, 6]])
```

```
In [209]: A[0:2, 0:2]
```

```
Out[209]: array([[1, 2],
                 [4, 5]])
```

```
In [210]: A[:, 2]
```

```
Out[210]: array([3, 6, 9])
```

Значения массива можно изменять используя оба способа - и индексы, и слайсы

```
In [211]: A[0, 0] = 100
```

```
In [212]: A[:, 2] = [0, 0, 0]
```

```
In [213]: A
```

```
Out[213]: array([[100,   2,   0],
                 [  4,   5,   0],
                 [  7,   8,   0]])
```

Memory view

Слайс numpy не создает копию данных (в отличие от списков python), а возвращает так называемый view. Это значит, что если поменять значение в слайсе, то значение в оригинальном массиве также изменится

```
In [214]: a_slice = A[0:2, 0:2]
          a_slice[0, 0] = -22
          A
```

```
Out[214]: array([[-22,   2,   0],
                 [  4,   5,   0],
                 [  7,   8,   0]])
```

Fancy indexing

Обращение к элементам массива используй другой массив (целочисленный или булевый)

In [215]:
```
A = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]])
A
```

Out[215]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

In [216]:
```
idx1 = np.array([0, 1, 2, 3])
idx2 = np.array([0, 1, 2, 3])
```

In [217]:
```
A[idx1, idx2]
```

Out[217]:
```
array([ 1,  6, 11, 16])
```

In [218]:
```
A
```

Out[218]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

In [220]:
```
idx1 = [
    [0, 1],
    [3, 2]
]

idx2 = [
    [0, 2],
    [1, 1]
]
```

In [224]:
```
A[idx1, idx2]
```

Out[224]:
```
array([[ 1,  7],
       [14, 10]])
```

In [225]:
```
B = np.array([1, 2, 3, 4, 5, 6])
mask = np.array([True, False, False, False, True, False])
B[mask]
```

Out[225]:
```
array([1, 5])
```

In [226]:
```
A > 2
```

Out[226]:
```
array([[False, False,  True,  True],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True]])
```

In [227]:
```
A[A > 5]
```

Out[227]:
```
array([ 6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```

Базовые операции

Арифметические операции

выполняются поэлементно

In [228]:
```
a = np.array([20, 30, 40, 50])
b = np.arange(4)
a, b
```

Out[228]:
```
(array([20, 30, 40, 50]), array([0, 1, 2, 3]))
```

In [229]:
```
a - b
```

Out[229]:
```
array([20, 29, 38, 47])
```

In [230]:
```
a ** b
```

Out[230]:
```
array([     1,     30,   1600, 125000], dtype=int32)
```

In [231]:
```
10 * a
```

Out[231]:
```
array([200, 300, 400, 500])
```

In [232]:
```
a > 35
```

Out[232]:
```
array([False, False,  True,  True])
```

Произведение матриц

операция @ или метод dot()

In [233]:
```
A = np.array( [[1,1],
               [0,1]] )
B = np.array( [[2,0],
               [3,4]] )
```

In [234]:
```
# Поэлементное умножение матриц
A * B
```

Out[234]:
```
array([[2, 0],
       [0, 4]])
```

In [235]:
```
# Произведение матриц
A @ B
```

Out[235]:
```
array([[5, 4],
       [3, 4]])
```

In [236]:
```
# ... или
A.dot(B)
```

Out[236]:
```
array([[5, 4],
       [3, 4]])
```

Унарные операции

реализованы как методы класса ndarray

In [237]:
```
a = np.random.random((2,3))
a
```

Out[237]:
```
array([[0.6317556974, 0.6890465697, 0.2038007117],
       [0.1330713379, 0.0508935241, 0.4152276312]])
```

In [238]:
```
a.sum()
```

Out[238]:
```
2.1237954719620546
```

In [239]:
```
a.min()
```

Out[239]:
```
0.050893524110887656
```

In [240]:
```
a.max()
```

Out[240]:
```
0.6890465697247758
```

По умолчанию эти функции просматривают элементы массива как одномерный список.
С помощью аргумента axis можно указать, вдоль какой оси выполняются вычисления.

In [241]:
```
b = np.arange(12).reshape(3,4)
b
```

Out[241]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [242]:
```
# сумма элементов в каждом столбце
b.sum(axis=0)
```

Out[242]:
```
array([12, 15, 18, 21])
```

In [243]:
```
# сумма в каждой строке
b.sum(axis=1)
```

Out[243]:
```
array([ 6, 22, 38])
```

Общие математические функции

In [244]:
```
a = np.arange(3)
a
```

Out[244]:
```
array([0, 1, 2])
```

In [245]:
```
np.exp(a)
```

Out[245]:
```
array([1.          , 2.7182818285, 7.3890560989])
```

In [246]:
```
np.sqrt(a)
```

Out[246]:
```
array([0.          , 1.          , 1.4142135624])
```

Broadcasting

Правила преобразования размерности массивов, чтобы выполнение поэлементных операций было возможно.
https://numpy.org/devdocs/user/basics.broadcasting.html
(https://numpy.org/devdocs/user/basics.broadcasting.html)

In [247]:
```
x = np.ones((4, 3))
x.shape
```

Out[247]:
```
(4, 3)
```

In [248]:
```
y = np.random.rand(3)
y.shape
```

Out[248]:
```
(3,)
```

In [249]:
```
(x * y).shape
```

Out[249]:
```
(4, 3)
```

Pandas

Библиотека для преобразования и анализа данных в форме таблиц и временных рядов

- множество форматов для импорта/экспорта данных: csv, xml, sql базы данных
- очистка, группировки, объединиение дата-сетов
- производительность

In [250]:
```
import pandas as pd
pd.__version__
```

Out[250]:
```
'0.25.3'
```

pd.Series

Проиндексированный одномерный массив - сопоставляет каждому значению массива уникальный ключ

Представим, мы работаем с некоторыми данными о результатах тестирования лекарства для нормализации давления.
В качестве индекса будем использовать "имена" пациентов.
В массиве хранится информация о том, эффективно ли лекарство для конкретного пациента.

In [251]:
```
patients = ["a", "b", "c", "d", "e", "f"]
effectiveness = [True, True, True, True, False, False]

effective_series = pd.Series(effectiveness, index=patients)
effective_series
```

Out[251]:
```
a     True
b     True
c     True
d     True
e    False
f    False
dtype: bool
```

Если нужно больше информации, одного массива уже недостаточно. Такие данные удобно хранить в табличной форме.

pd.DataFrame

Проиндексированя таблица. Можно рассматривать DataFrame как словарь элементов Series

В роли индекса - снова имена пациентов.
В каждой колонке хранится значение отдельного признака для всех пациентов.
Каждая строка - набор всех признаков для конкретного пациента.

In [252]:
```
columns = {
    'gender': ['F', 'M', 'M', 'M', 'F', 'M'],
    'sys_initial': [120, 126, 130, 115, 150, 117],
    'dia_initial': [75, 85, 90, 87, 90, 74],
    'sys_final': [115, 123, 130, 118, 130, 121],
    'dia_final': [70, 82, 92, 87, 85, 74],
    'drug_admst': effectiveness
}

df = pd.DataFrame(columns, index=patients)
```

In [253]:
```
df
```

Out[253]:

	gender	sys_initial	dia_initial	sys_final	dia_final	drug_admst
a	F	120	75	115	70	True
b	M	126	85	123	82	True
c	M	130	90	130	92	True
d	M	115	87	118	87	True
e	F	150	90	130	85	False
f	M	117	74	121	74	False

In [254]:
```
type(df['sys_initial'])
```

Out[254]:
```
pandas.core.series.Series
```

In [255]:
```
type(df.loc['a'])
```

Out[255]:
```
pandas.core.series.Series
```

Базовый анализ

In [256]:
```
df.shape
```

Out[256]:
```
(6, 6)
```

In [257]:
```
df.head(2)
```

Out[257]:

	gender	sys_initial	dia_initial	sys_final	dia_final	drug_admst
a	F	120	75	115	70	True
b	M	126	85	123	82	True

In [258]:
```
df.tail(2)
```

Out[258]:

	gender	sys_initial	dia_initial	sys_final	dia_final	drug_admst
e	F	150	90	130	85	False
f	M	117	74	121	74	False

In [259]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6 entries, a to f
Data columns (total 6 columns):
gender         6 non-null object
sys_initial    6 non-null int64
dia_initial    6 non-null int64
sys_final      6 non-null int64
dia_final      6 non-null int64
drug_admst     6 non-null bool
dtypes: bool(1), int64(4), object(1)
memory usage: 454.0+ bytes
```

In [260]:
```
df.dia_initial.value_counts()
```

Out[260]:
```
90    2
87    1
85    1
74    1
75    1
Name: dia_initial, dtype: int64
```

In [261]:
```
df.dia_initial.sort_values()
```

Out[261]:
```
f    74
a    75
b    85
d    87
c    90
e    90
Name: dia_initial, dtype: int64
```

Индексация

In [262]:
```
df.loc['b']
```

Out[262]:
```
gender            M
sys_initial     126
dia_initial      85
sys_final       123
dia_final        82
drug_admst     True
Name: b, dtype: object
```

In [263]:
```
df.iloc[1]
```

Out[263]:
```
gender            M
sys_initial     126
dia_initial      85
sys_final       123
dia_final        82
drug_admst     True
Name: b, dtype: object
```

In [264]:
```
df.dia_initial
```

Out[264]:
```
a    75
b    85
c    90
d    87
e    90
f    74
Name: dia_initial, dtype: int64
```

In [265]:
```
df.loc['c', 'sys_final']
```

Out[265]:
```
130
```

In [266]:
```
df.iloc[2, 2]
```

Out[266]:
```
90
```

In [267]:
```
df.loc[:, ['sys_final', 'dia_final']]
```

Out[267]:

	sys_final	dia_final
a	115	70
b	123	82
c	130	92
d	118	87
e	130	85
f	121	74

In [268]:
```
df[(df.sys_initial <= 120) & (df.sys_final > 115)]
```

Out[268]:

	gender	sys_initial	dia_initial	sys_final	dia_final	drug_admst
d	M	115	87	118	87	True
f	M	117	74	121	74	False

groupBy

Возвращает итератор, каждый элемент которого - фрейм с соответствующим значением

In [269]:
```
type(df.groupby('drug_admst'))
```

Out[269]:
```
pandas.core.groupby.generic.DataFrameGroupBy
```

In [270]:
```
for value, group in df.groupby('drug_admst'):
    print("Value: {}".format(value))
    print("Group DataFrame:")
    print(group)
```

```
Value: False
Group DataFrame:
  gender  sys_initial  dia_initial  sys_final  dia_final  drug_admst
e      F          150           90        130         85       False
f      M          117           74        121         74       False
Value: True
Group DataFrame:
  gender  sys_initial  dia_initial  sys_final  dia_final  drug_admst
a      F          120           75        115         70        True
b      M          126           85        123         82        True
c      M          130           90        130         92        True
d      M          115           87        118         87        True
```

In [271]:
```
df.groupby('drug_admst').agg({
    'sys_initial': ['min', 'max'],
    'sys_final': np.mean
})
```

Out[271]:

	sys_initial		sys_final
	min	max	mean
drug_admst			
False	117	150	125.5
True	115	130	121.5

Joining

In [272]:
```
hospitals = pd.DataFrame({
        "name" : ["City 1", "City 2", "City 3"],
        "address" : ["Address 1", "Address 2", "Address 3"],
        "city": ["City 1", "City 2", "City 3"]
    }, index=["H1", "H2", "H3"])

hospitals
```

Out[272]:

	name	address	city
H1	City 1	Address 1	City 1
H2	City 2	Address 2	City 2
H3	City 3	Address 3	City 3

In [273]:
```
df['hospital_id'] = ['H1', 'H2', 'H2', 'H3', 'H3', 'H3']
df
```

Out[273]:

	gender	sys_initial	dia_initial	sys_final	dia_final	drug_admst	hospital_id
a	F	120	75	115	70	True	H1
b	M	126	85	123	82	True	H2
c	M	130	90	130	92	True	H2
d	M	115	87	118	87	True	H3
e	F	150	90	130	85	False	H3
f	M	117	74	121	74	False	H3

In [274]:
```
full_df = df.join(hospitals, on='hospital_id')
full_df
```

Out[274]:

	gender	sys_initial	dia_initial	sys_final	dia_final	drug_admst	hospital_id	name	address
a	F	120	75	115	70	True	H1	City 1	Address 1
b	M	126	85	123	82	True	H2	City 2	Address 2
c	M	130	90	130	92	True	H2	City 2	Address 2
d	M	115	87	118	87	True	H3	City 3	Address 3
e	F	150	90	130	85	False	H3	City 3	Address 3
f	M	117	74	121	74	False	H3	City 3	Address 3

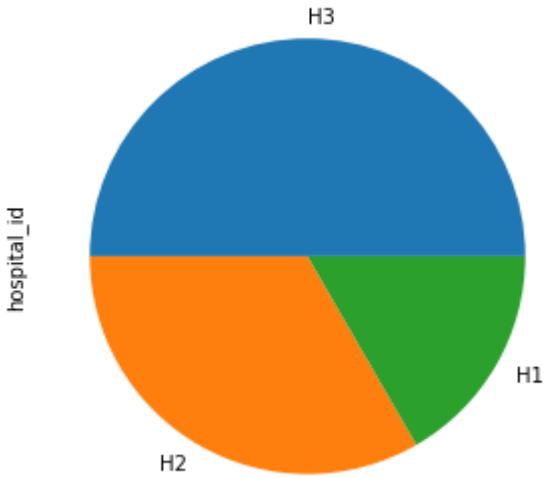
Визуализация

matplotlib, seaborn
Pandas позволяет быстро создавать графики, используя API matplotlib.

Круговая диаграмма

```
In [275]: full_df['hospital_id'].value_counts().plot.pie(figsize=(5, 5))
```

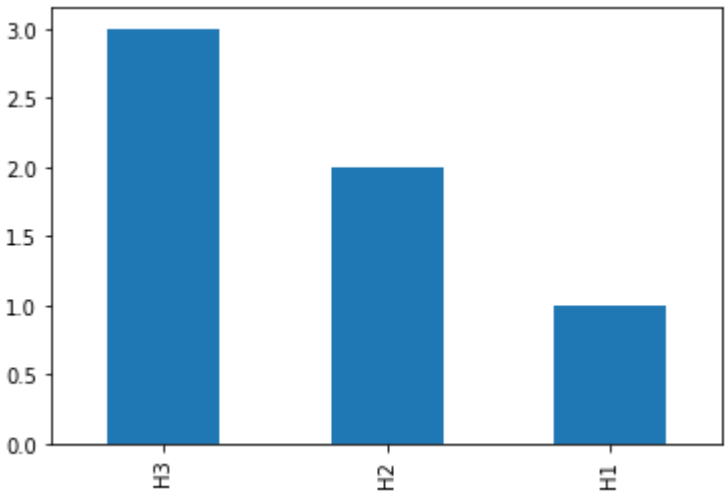
```
Out[275]: <matplotlib.axes._subplots.AxesSubplot at 0x2592958a7c8>
```



Столбчатая диаграмма

```
In [276]: full_df['hospital_id'].value_counts().plot.bar()
```

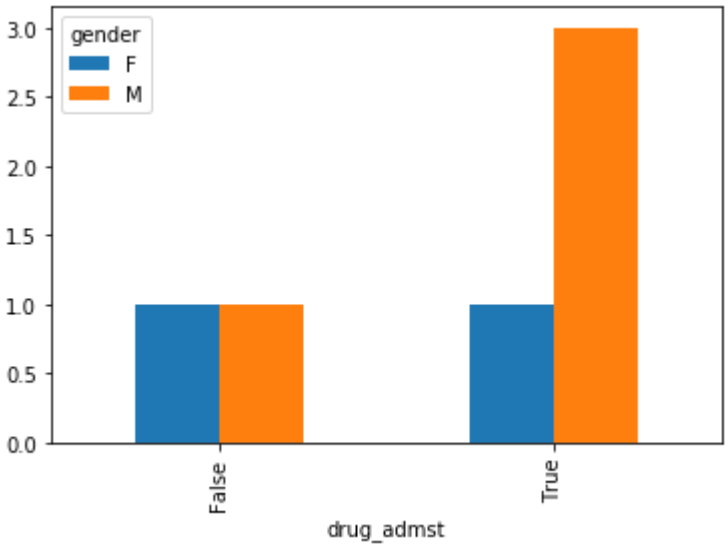
```
Out[276]: <matplotlib.axes._subplots.AxesSubplot at 0x259297c9e08>
```



Столбчатая диаграмма #2

In [279]:
```
pd.crosstab(df.drug_admst, df.gender).plot.bar()
```

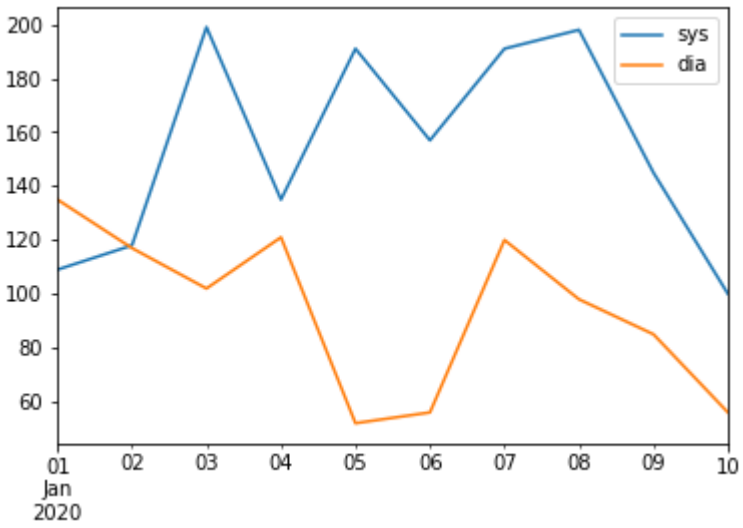
Out[279]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25929754f88>
```



Линейный график

In [281]:
```
columns = {
    'sys': np.random.randint(low=100, high=200, size=10),
    'dia': np.random.randint(low=50, high=150, size=10),
}

a_observations = pd.DataFrame(columns, index=pd.date_range('1/1/2020', periods
=10))
a_observations.plot()
```

Out[281]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x259298ea548>
```



Домашнее задание

Во вкладке **Files** в группе **Webinars** прикреплен файл _olimpic*medals.csv* с данными о медалях на Олимпийских играх за 1896-2008 годы.
Ссылка:
https://epam.sharepoint.com/sites/EPAMNNPythonLab/Shared%20Documents/Webinars/olimpic_medals.csv (https://epam.sharepoint.com/sites/EPAMNNPythonLab/Shared%20Documents/Webinars/olimpic_medals.csv)
Скачайте .csv файл себе на диск, загрузите данные в DataFrame (функция _read*csv*) и ответьте на следующие вопросы, используя функционал pandas:

1. Сколько медалей выиграл Jesse Owens в 1936?
2. Какая страна выйграла большинство золотых медалей мужчинами в бадминтоне?
3. Какие три страны выйграли большинство медалей в последние годы (с 1984 по 2008)?
4. Покажите мужчин золотых медалистов по 100m. Выведите результаты по убыванию года выйгрыша. Покажите город в котором проходила олимпиала, год, имя атлета и страну за которую он выступал.
5. Как много медалей было выйграно мужчинами и женщинами в истории олимпиады. Как много золотых, серебрянных и бронзовых медалей было выйграно каждым полом?
6. Используя groupby(), постройте график числа всех медалей выйгранных на каждой олимпиаде.
7. Создайте список показывающий число всех медалей выйгранных каждой страной в течение всей истории олимпийских игр. Для каждой страны, необходимо показать год первой и последней заработанной медали.
8. Атлеты выйгравшие медали в Beijing на дистанции 100m или 200m
9. Постройте график числа золотых медалей выйгранных США мужчинами и женщинами в атлетике.
10. Постройте график 5 атлетов которые выйграли большинство золотых медалей.
11. Покажите суммарное количество медалей выйгранных странами в последних олимпийских играх.
12. Постройте таблицу в которой по годам всех олимпиад покажите топовых атлетов США(1 атлет на год) по общему количеству медалей. Включите дисциплину атлета.

Форма чтобы засабмитить задание: https://epa.ms/pyhomework (https://epa.ms/pyhomework)