PYTHON

Лекция 13

ПЛАН ЛЕКЦИИ

- Pandas
- MatPlotlib

MATPLOTLIB

- Библиотека для работы с изображениями
- Цели - совсем другие, чем у OpenCV
- Основная цель - графики, диаграммы и т.п.

КОНТЕКСТ

- matplotlib работает с абстрактной панелью
- Которая может быть куда-то интегрирована
- В простейшем варианте это может быть базовое десктоп-приложение
- Но можно интенрировать в браузер через Jupyter/IPython
- Мы фокусируемся на "внутренней" части, не на деталях интеграции

АНАЛОГ HELLO, WORLD

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 import matplotlib as mpl
 5
 6 fig, ax = plt.subplots()
 7 x_data = np.arange(1000, dtype=np.double) / 100
 8 ax.plot(x_data, np.sin(x_data))
 9
10 plt.show()
```

РАЗБЕРЕМ

- Логика рисунка определяется вызовом `ax.plot`
- `plt.show` - это прорисовка
- `ax` - объект, представляющий "холст"
- И еще есть `fig`, который не используем

КЛАССЫ MATPLOTLIB

- Класс `Artist` - абстракция того, что может быть прорисовано
- Примеры подклассов: `Text`, `Line2D`, `Axis`, `Axes`, `Figure`, много еще
- `Axes` - это составной визуальный объект
- Что-то типа школьной доски, на которой нарисована координатная ось

КЛАССЫ MATPLOTLIB

- `Axis` - класс, представляющий координатные оси
- `Figure` - это вся картина в целом
- Может включать в себя несколько `Axes`
- И даже другие `Figure`

ПОРАЗНООБРАЗНЕЕ

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 import matplotlib as mpl
 5
 6
 7 fig = plt.figure()
 8 fig, ax = plt.subplots()
 9 fig, axs = plt.subplots(2, 2)
10 fig, axs = plt.subplot_mosaic([['left', 'right_top'],
11                               ['left', 'right_bottom']])
12
13 plt.show()
```

ПОРАЗНООБРАЗНЕЕ

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 import matplotlib as mpl
 5
 6 fig = plt.figure(figsize=(2, 2), facecolor='lightskyblue',
 7                  layout='constrained')
 8 fig.suptitle('Figure')
 9 ax = fig.add_subplot()
10 ax.set_title('Axes', loc='left',
11               fontstyle='oblique', fontsize='medium')
12
13 plt.show()
```

ДВА ПОДХОДА

- Возможно два подхода
- OO-style: работаем чем методы объектов
- pyploy-style: работаем через функции модуля
- В большом приложении лучше OO-style, для быстрых экспериментов - pyplot

ПРИМЕР

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 import matplotlib as mpl
5
6 x = np.linspace(0, 2, 100)  # Sample data.
7
8 # ..............................
```

ПРИМЕР

```
# ..............................

fig, ax = plt.subplots(figsize=(5, 2.7),
                       layout='constrained')
ax.plot(x, x, label='linear')
ax.plot(x, x**2, label='quadratic')
ax.plot(x, x**3, label='cubic')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.set_title("Simple Plot")
ax.legend()

plt.show()
```

ПРИМЕР

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 import matplotlib as mpl
5
6 x = np.linspace(0, 2, 100)
7
8 # ....................................
```

ПРИМЕР

```
# ....................................

plt.figure(figsize=(5, 2.7), layout='constrained')
plt.plot(x, x, label='linear')
plt.plot(x, x**2, label='quadratic')
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()

plt.show()
```

СТИЛЬ ЛИНИЙ

- Можно уточнить многие атрибуты линии
- Особенно интересна параметризация типа линии
- Есть несколько предопределенных
- Но можно задавать и свои

ЛИНИИ

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 linestyle_str = [
 5      ('solid', 'solid'),      # Same as (0, ()) or '-'
 6      ('dotted', 'dotted'),    # Same as (0, (1, 1)) or ':'
 7      ('dashed', 'dashed'),    # Same as '--'
 8      ('dashdot', 'dashdot')]  # Same as '-.'
 9
10 # ........................
```

ЛИНИИ

```
# ........................
linestyle_tuple = [
     ('loosely dotted',         (0, (1, 10))),
     ('dotted',                 (0, (1, 1))),
     ('densely dotted',         (0, (1, 1))),
     ('long dash with offset', (5, (10, 3))),
     ('loosely dashed',         (0, (5, 10))),
     ('dashed',                 (0, (5, 5))),
     ('densely dashed',         (0, (5, 1))),
# ........................
```

ЛИНИИ

```
# ..........................
     ('loosely dashdotted',    (0, (3, 10, 1, 10))),
     ('dashdotted',            (0, (3, 5, 1, 5))),
     ('densely dashdotted',    (0, (3, 1, 1, 1))),

     ('dashdotdotted',         (0, (3, 5, 1, 5, 1, 5))),
     ('loosely dashdotdotted',
                           (0, (3, 10, 1, 10, 1, 10))),
     ('densely dashdotdotted', (0, (3, 1, 1, 1, 1, 1)))]

# ..........................
```

ЛИНИИ

```
# ........................

def plot_linestyles(ax, linestyles, title):
    X, Y = np.linspace(0, 100, 10), np.zeros(10)
    yticklabels = []

    for i, (name, linestyle) in enumerate(linestyles):
        ax.plot(X, Y+i, linestyle=linestyle,
                linewidth=1.5, color='black')
        yticklabels.append(name)

# ........................
```

ЛИНИИ

```
# ........................
    ax.set_title(title)
    ax.set(ylim=(-0.5, len(linestyles)-0.5),
           yticks=np.arange(len(linestyles)),
           yticklabels=yticklabels)
    ax.tick_params(left=False, bottom=False,
                   labelbottom=False)
    ax.spines[:].set_visible(False)

# ........................
```

ЛИНИИ

```
# ........................
    for i, (name, linestyle) in enumerate(linestyles):
        ax.annotate(repr(linestyle),
                    xy=(0.0, i),
                    xycoords=ax.get_yaxis_transform(),
                    xytext=(-6, -12),
                    textcoords='offset points',
                    color="blue", fontsize=8,
                    ha="right", family="monospace")
# ........................
```

ЛИНИИ

```
# ........................
fig, (ax0, ax1) = plt.subplots(2, 1, figsize=(10, 8))

plot_linestyles(ax0, linestyle_str[::-1],
                title='Named linestyles')
plot_linestyles(ax1, linestyle_tuple[::-1],
                title='Parametrized linestyles')

plt.tight_layout()

plt.show()
# ........................
```

SCATTER PLOT

- Хотим оценить взаимосвязь двух атрибутов
- Каждый объект представим точкой
- Координаты - значения атрибутов
- По форме "облака" можем оценить характер зависимости

ПРИМЕР

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 np.random.seed(19680801)
 5
 6 N = 50
 7 x = np.random.rand(N)
 8 y = np.random.rand(N)
 9 colors = np.random.rand(N)
10 area = (30 * np.random.rand(N))**2
11
12 plt.scatter(x, y, s=area, c=colors, alpha=0.5)
13 plt.show()
```

BAR CHART

- Популярная форма визуализации
- Используется для сравнения числовых показателей
- matplotlib предоставляет нужные инструменты

ПРИМЕР

```
 1 import matplotlib.pyplot as plt
 2
 3 fig, ax = plt.subplots()
 4
 5 fruits = ['apple', 'blueberry', 'cherry', 'orange']
 6 counts = [40, 100, 30, 55]
 7 bar_labels = ['red', 'blue', '_red', 'orange']
 8 bar_colors = ['tab:red', 'tab:blue',
 9               'tab:red', 'tab:orange']
10
11 # ......................
```

ПРИМЕР

```
# ....................

ax.bar(fruits, counts, label=bar_labels, color=bar_colors)

ax.set_ylabel('fruit supply')
ax.set_title('Fruit supply by kind and color')
ax.legend(title='Fruit color')

plt.show()
```

ПРИМЕР

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 species = (
 5     "Adelie\n $\\mu=$3700.66g",
 6     "Chinstrap\n $\\mu=$3733.09g",
 7     "Gentoo\n $\\mu=5076.02g$",
 8 )
 9
10 # ..............................
```

ПРИМЕР

```
# ..............................

weight_counts = {
    "Below": np.array([70, 31, 58]),
    "Above": np.array([82, 37, 66]),
}
width = 0.5

fig, ax = plt.subplots()
bottom = np.zeros(3)

# ......................
```

ПРИМЕР

```
# ..............................

for boolean, weight_count in weight_counts.items():
    p = ax.bar(species, weight_count, width,
               label=boolean, bottom=bottom)
    bottom += weight_count

ax.set_title("Number of penguins with "
             "above average body mass")
ax.legend(loc="upper right")

plt.show()
```

ПРИМЕР

```
import matplotlib.pyplot as plt
import numpy as np

species = ("Adelie", "Chinstrap", "Gentoo")
penguin_means = {
    'Bill Depth': (18.35, 18.43, 14.98),
    'Bill Length': (38.79, 48.83, 47.50),
    'Flipper Length': (189.95, 195.82, 217.19),
}

# ............................
```

ПРИМЕР

```
# ............................
x = np.arange(len(species))
width = 0.25
multiplier = 0

fig, ax = plt.subplots(layout='constrained')

for attribute, measurement in penguin_means.items():
    offset = width * multiplier
    rects = ax.bar(x + offset, measurement, width,
                   label=attribute)
    ax.bar_label(rects, padding=3)
    multiplier += 1
# ............................
```

ПРИМЕР

```
# ..............................
ax.set_ylabel('Length (mm)')
ax.set_title('Penguin attributes by species')
ax.set_xticks(x + width, species)
ax.legend(loc='upper left')
ax.set_ylim(0, 250)

plt.show()
```

ПРИМЕР

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 np.random.seed(19680801)
 5
 6 fig, ax = plt.subplots()
 7
 8 people = ('Tom', 'Dick', 'Harry', 'Slim', 'Jim')
 9 y_pos = np.arange(len(people))
10 performance = 3 + 10 * np.random.rand(len(people))
11 error = np.random.rand(len(people))
12
13 # ........................
```

ПРИМЕР

```
# ........................

ax.barh(y_pos, performance, xerr=error, align='center')
ax.set_yticks(y_pos, labels=people)
ax.invert_yaxis()  # labels read top-to-bottom
ax.set_xlabel('Performance')
ax.set_title('How fast do you want to go today?')

plt.show()
```

ПРИМЕР

```
 1 import matplotlib.pyplot as plt
 2
 3 # Horizontal bar plot with gaps
 4 fig, ax = plt.subplots()
 5 ax.broken_barh([(110, 30), (150, 10)],
 6                 (10, 9), facecolors='tab:blue')
 7 ax.broken_barh([(10, 50), (100, 20), (130, 10)], (20, 9),
 8         facecolors=('tab:orange', 'tab:green', 'tab:red'))
 9 ax.set_ylim(5, 35)
10 ax.set_xlim(0, 200)
11 # ..........................
```

ПРИМЕР

```
# ........................
ax.set_xlabel('seconds since start')
ax.set_yticks([15, 25], labels=['Bill', 'Jim'])
ax.grid(True)
ax.annotate('race interrupted', (61, 25),
            xytext=(0.8, 0.9), textcoords='axes fraction',
            arrowprops=dict(facecolor='black',
                            shrink=0.05),
            fontsize=16,
            horizontalalignment='right',
            verticalalignment='top')
plt.show()
```

3D

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 np.random.seed(19680801)
 5
 6 fig = plt.figure()
 7 ax = fig.add_subplot(projection='3d')
 8 x, y = np.random.rand(2, 100) * 4
 9 hist, xedges, yedges = np.histogram2d(
10     x, y, bins=4, range=[[0, 4], [0, 4]]
11 )
12 # ....................
```

3D

```
# ....................
xpos, ypos = np.meshgrid(
             xedges[:-1] + 0.25,
             yedges[:-1] + 0.25, indexing="ij")
xpos = xpos.ravel()
ypos = ypos.ravel()
zpos = 0

dx = dy = 0.5 * np.ones_like(zpos)
dz = hist.ravel()

ax.bar3d(xpos, ypos, zpos, dx, dy, dz, zsort='average')

plt.show()
```

ЛИНИИ

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 np.random.seed(19680801)
 5
 6 t = np.arange(0.0, 5.0, 0.1)
 7 s = np.exp(-t) + np.sin(2 * np.pi * t) + 1
 8 nse = np.random.normal(0.0, 0.3, t.shape) * s
 9
10 fig, (vax, hax) = plt.subplots(1, 2, figsize=(12, 6))
11 # ........................
```

ЛИНИИ

```
# ........................
vax.plot(t, s + nse, '^')
vax.vlines(t, [0], s)
vax.vlines([1, 2], 0, 1,
           transform=vax.get_xaxis_transform(),
           colors='r')
vax.set_xlabel('time (s)')
vax.set_title('Vertical lines demo')
hax.plot(s + nse, t, '^')
hax.hlines(t, [0], s, lw=2)
hax.set_xlabel('time (s)')
hax.set_title('Horizontal lines demo')

plt.show()
```

ЦВЕТА

```
 1 import math
 2 import matplotlib.pyplot as plt
 3 import matplotlib.colors as mcolors
 4 from matplotlib.patches import Rectangle
 5
 6 def plot_colortable(colors, *, ncols=4, sort_colors=True):
 7     cell_width = 212
 8     cell_height = 22
 9     swatch_width = 48
10     margin = 12
11 # ......................
```

ЦВЕТА

```
# ......................
    if sort_colors:
        names = sorted(
            colors,
            key=lambda c: \
              tuple(mcolors.rgb_to_hsv(mcolors.to_rgb(c)))
        )
    else:
        names = list(colors)

    n = len(names)
    nrows = math.ceil(n / ncols)
# ......................
```

ЦВЕТА

```
# ......................
    width = cell_width * ncols + 2 * margin
    height = cell_height * nrows + 2 * margin
    dpi = 72

    fig, ax = plt.subplots(
        figsize=(width / dpi, height / dpi), dpi=dpi)
    fig.subplots_adjust(margin/width, margin/height,
                        (width-margin)/width,
                        (height-margin)/height)
# ......................
```

ЦВЕТА

```
# ......................
    ax.set_xlim(0, cell_width * ncols)
    ax.set_ylim(cell_height * (nrows-0.5),
                -cell_height/2.)
    ax.yaxis.set_visible(False)
    ax.xaxis.set_visible(False)
    ax.set_axis_off()

    for i, name in enumerate(names):
        row = i % nrows
        col = i // nrows
        y = row * cell_height
# ......................
```

ЦВЕТА

```
1 # ..............................
2         swatch_start_x = cell_width * col
3         text_pos_x = cell_width * col + swatch_width + 7
4
5         ax.text(text_pos_x, y, name, fontsize=14,
6                 horizontalalignment='left',
7                 verticalalignment='center')
8 # ..............................
```

ЦВЕТА

```
# ..............................

        ax.add_patch(
            Rectangle(xy=(swatch_start_x, y-9),
                      width=swatch_width,
                      height=18,
                      facecolor=colors[name],
                      edgecolor='0.7')
        )

    return fig
# ..............................
```

ЦВЕТА

```
# ..............................

plot_colortable(mcolors.BASE_COLORS, ncols=3,
                sort_colors=False)
plot_colortable(mcolors.TABLEAU_COLORS, ncols=2,
                sort_colors=False)

plot_colortable(mcolors.CSS4_COLORS)
plt.show()
```

ТЕКСТ

- Текст может появляться в стандартизированных местах
- В маркировке оси или в заголовке графика
- Или в произвольном месте
- Можно графический текст
- Например, создавать dataset для приложения распознавания букв

ПРИМЕР

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 mu, sigma = 115, 15
 5 x = mu + sigma * np.random.randn(10000)
 6 fig, ax = plt.subplots(figsize=(5, 2.7),
 7                        layout='constrained')
 8 n, bins, patches = ax.hist(x, 50, density=True,
 9                            facecolor='C0', alpha=0.75)
10 # ................
```

ПРИМЕР

```
# ................

ax.set_xlabel('Length [cm]')
ax.set_ylabel('Probability')
ax.set_title('Aardvark lengths\n (not really)')
ax.text(75, .025, r'$\mu=115,\ \sigma=15$')
ax.axis([55, 175, 0, 0.03])
ax.grid(True)

plt.show()
```

ПОСЛОЖНЕЕ

```
1 import re
2 import subprocess
3 import sys
4
5 import matplotlib.pyplot as plt
6
7 mathtext_demos = {
8 # ................
```

ПОСЛОЖНЕЕ

```
# ..................
    "Header demo":
        r"$W^{3\beta}_{\delta_1 "
        r"\rho_1 \sigma_2} = "
        r"U^{3\beta}_{\delta_1 \rho_1} + "
        r"\frac{1}{8 \pi 2} "
        r"\int^{\alpha_2}_{\alpha_2} d "
        r"\alpha^\prime_2 \left[\frac{ "
        r"U^{2\beta}_{\delta_1 \rho_1} - "
        r"\alpha^\prime_2U^{1\beta}_"
        r"{\rho_1 \sigma_2} }{U^{0\beta}_"
        r"{\rho_1 \sigma_2}}\right]$",
# ..................
```

ПОСЛОЖНЕЕ

```
1 # ..............
2
3     "Subscripts and superscripts":
4         r"$\alpha_i > \beta_i,\ "
5         r"\alpha_{i+1}^j = "
6         r"{\rm sin}(2\pi f_j t_i) e^{-5 t_i/\tau},\ "
7         r"\ldots$",
8 # ..............
```

ПОСЛОЖНЕЕ

```
# ................
    "Fractions, binomials and stacked numbers":
        r"$\frac{3}{4},\ \binom{3}{4},"
        r"\ \genfrac{}{}{0}{}{3}{4},\ "
        r"\left(\frac{5 - \frac{1}{x}}"
        r"{4}\right),\ \ldots$",

    "Radicals":
        r"$\sqrt{2},\ \sqrt[3]{x},\ \ldots$",
# ................
```

ПОСЛОЖНЕЕ

```
# ................
    "Fonts":
        r"$\mathrm{Roman}\ , \ \mathit{Italic}\ ,"
        r" \ \mathtt{Typewriter} \ "
        r"\mathrm{or}\ "
        r"\mathcal{CALLIGRAPHY}$",

    "Accents":
        r"$\acute a,\ \bar a,\ \breve a,"
        r"\ \dot a,\ \ddot a, \ \grave a, \ "
        r"\hat a,\ \tilde a,\ \vec a,"
        r"\ \widehat{xyz},\ \widetilde{xyz},\ "
        r"\ldots$",
# ................
```

ПОСЛОЖНЕЕ

```
1 # ................
2     "Greek, Hebrew":
3         r"$\alpha,\ \beta,\ \chi,"
4         r"\ \delta,\ \lambda,\ \mu,\ "
5         r"\Delta,\ \Gamma,\ \Omega,"
6         r"\ \Phi,\ \Pi,\ \Upsilon,\ \nabla,\ "
7         r"\aleph,\ \beth,\ \daleth,\ \gimel,\ \ldots$",
8 # ................
```

ПОСЛОЖНЕЕ

```
# ..............

    "Delimiters, functions and Symbols":
        r"$\coprod,\ \int,\ \oint,\ \prod,\ \sum,\ "
        r"\log,\ \sin,\ \approx,\ \oplus,"
        r"\ \star,\ \varpropto,\ "
        r"\infty,\ \partial,\ \Re,"
        r"\ \leftrightsquigarrow, \ \ldots$",
}
n_lines = len(mathtext_demos)
# ..............
```

ПОСЛОЖНЕЕ

```
# ..............
def doall():
    mpl_grey_rgb = (51 / 255, 51 / 255, 51 / 255)

    fig = plt.figure(figsize=(7, 7))
    ax = fig.add_axes([0.01, 0.01, 0.98, 0.90],
                      facecolor="white", frameon=True)
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.set_title("Matplotlib's math rendering engine",
           color=mpl_grey_rgb, fontsize=14, weight='bold')
    ax.set_xticks([])
    ax.set_yticks([])
# ..............
```

ПОСЛОЖНЕЕ

```
# ..................
    line_axesfrac = 1 / n_lines
    full_demo = mathtext_demos['Header demo']
    ax.annotate(full_demo,
                xy=(0.5, 1. - 0.59 * line_axesfrac),
          color='tab:orange', ha='center', fontsize=20)

    for i_line, (title, demo) in
                          enumerate(mathtext_demos.items()):
        print(i_line, demo)
        if i_line == 0:
            continue
# ..................
```

ПОСЛОЖНЕЕ

```
# ................
        baseline = 1 - i_line * line_axesfrac
        baseline_next = baseline - line_axesfrac
        fill_color = ['white', 'tab:blue'][i_line % 2]
        ax.axhspan(baseline, baseline_next,
                   color=fill_color, alpha=0.2)
        ax.annotate(f'{title}:',
              xy=(0.06, baseline - 0.3 * line_axesfrac),
              color=mpl_grey_rgb, weight='bold')
        ax.annotate(demo,
              xy=(0.04, baseline - 0.75 * line_axesfrac),
              color=mpl_grey_rgb, fontsize=16)
    plt.show()
# ................
```

ПОСЛОЖНЕЕ

```
# ..............

if '--latex' in sys.argv:
    with open("mathtext_examples.ltx", "w") as fd:
        fd.write("\\documentclass{article}\n")
        fd.write("\\usepackage{amsmath, amssymb}\n")
        fd.write("\\begin{document}\n")
        fd.write("\\begin{enumerate}\n")
# ..............
```

ПОСЛОЖНЕЕ

```
# ................
        for s in mathtext_demos.values():
            s = re.sub(r"(?<!\\)\$", "$$", s)
            fd.write("\\item %s\n" % s)

        fd.write("\\end{enumerate}\n")
        fd.write("\\end{document}\n")

    subprocess.call(["pdflatex", "mathtext_examples.ltx"])
else:
    doall()
```

ИНТЕРАКТИВНОСТЬ

```
 1 import numpy as np
 2
 3 class PointBrowser:
 4     def __init__(self):
 5         self.lastind = 0
 6         self.text = ax.text(0.05, 0.95, 'selected: none',
 7                             transform=ax.transAxes, va='top')
 8         self.selected, = ax.plot([xs[0]], [ys[0]], 'o',
 9                                  ms=12, alpha=0.4,
10                                  color='yellow', visible=False)
11 # ..................
```

ИНТЕРАКТИВНОСТЬ

```
# ................
    def on_press(self, event):
        if self.lastind is None:
            return
        if event.key not in ('n', 'p'):
            return
        if event.key == 'n':
            self.lastind += 1
        else:
            self.lastind -= 1
        self.lastind = np.clip(self.lastind, 0,
                               len(xs) - 1)
        self.update()
# ................
```

ИНТЕРАКТИВНОСТЬ

```
# ................
    def on_pick(self, event):

        if event.artist != line:
            return True

        N = len(event.ind)
        if not N:
            return True

        x = event.mouseevent.xdata
        y = event.mouseevent.ydata
# ................
```

ИНТЕРАКТИВНОСТЬ

```
1 # ..................
2         distances = np.hypot(x - xs[event.ind],
3                              y - ys[event.ind])
4         indmin = distances.argmin()
5         dataind = event.ind[indmin]
6
7         self.lastind = dataind
8         self.update()
9 # ..................
```

ИНТЕРАКТИВНОСТЬ

```
 1 # ................
 2     def update(self):
 3         if self.lastind is None:
 4             return
 5 
 6         dataind = self.lastind
 7 
 8         ax2.clear()
 9         ax2.plot(X[dataind])
10 # ................
```

ИНТЕРАКТИВНОСТЬ

```
# ................
        ax2.text(0.05, 0.9,
                 f'mu={xs[dataind]:1.3f}\n'
                 f'sigma={ys[dataind]:1.3f}',
                 transform=ax2.transAxes, va='top')
        ax2.set_ylim(-0.5, 1.5)
        self.selected.set_visible(True)
        self.selected.set_data(xs[dataind], ys[dataind])

        self.text.set_text('selected: %d' % dataind)
        fig.canvas.draw()

# ................
```

ИНТЕРАКТИВНОСТЬ

```
# ................
if __name__ == '__main__':
    import matplotlib.pyplot as plt

    np.random.seed(19680801)

    X = np.random.rand(100, 200)
    xs = np.mean(X, axis=1)
    ys = np.std(X, axis=1)
# ................
```

ИНТЕРАКТИВНОСТЬ

```
# ..................
    fig, (ax, ax2) = plt.subplots(2, 1)
    ax.set_title('click on point to plot time series')
    line, = ax.plot(xs, ys, 'o',
                    picker=True, pickradius=5)

    browser = PointBrowser()
    fig.canvas.mpl_connect('pick_event', browser.on_pick)
    fig.canvas.mpl_connect('key_press_event',
                           browser.on_press)

    plt.show()
```

ИНТЕРАКТИВНОСТЬ

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 np.random.seed(19680801)
 5
 6 figsrc, axsrc = plt.subplots(figsize=(3.7, 3.7))
 7 figzoom, axzoom = plt.subplots(figsize=(3.7, 3.7))
 8 axsrc.set(xlim=(0, 1), ylim=(0, 1), autoscale_on=False,
 9           title='Click to zoom')
10 axzoom.set(xlim=(0.45, 0.55), ylim=(0.4, 0.6),
11            autoscale_on=False,
12            title='Zoom window')
13 # ................
```

ИНТЕРАКТИВНОСТЬ

```
1 # ..............
2
3 x, y, s, c = np.random.rand(4, 200)
4 s *= 200
5
6 axsrc.scatter(x, y, s, c)
7 axzoom.scatter(x, y, s, c)
8 # ..............
```

ИНТЕРАКТИВНОСТЬ

```
# ................
def on_press(event):
    if event.button != 1:
        return
    x, y = event.xdata, event.ydata
    axzoom.set_xlim(x - 0.1, x + 0.1)
    axzoom.set_ylim(y - 0.1, y + 0.1)
    figzoom.canvas.draw()

figsrc.canvas.mpl_connect('button_press_event', on_press)
plt.show()
```

PANDAS

- Ключевая структура - `DataFrame`
- Набор столбцов
- Каждый столбец - класс `Series`
- Содержимое типизировано, по аналогии с `numpy`

PANDAS

- Типы - целый, вещественный, строка
- И еще - категориальный, `timestamp`
- Типизация - единая внутри колонки
- `DataFrame` может хранить колонки разных типов

PANDAS

- Размер `Series` фиксирован
- Высота `DataFrame` - тоже
- Колонки можно вставлять прямо в `DataFrame`
- Как и удалять
- И менять значение в конкретных позициях

СОХРАНЕНИЕ

- Pandas поддерживает много разных форматов
- И много источников
- В том числе - запись DF в csv-файл
- И чтение из csv-файла

ПРИМЕР

```
1 data = [{'a': 1, 'b': 2},
2         {'a': 5, 'b': 10, 'c': 20}]
3 df = pd.DataFrame(data)
4
5 print([m for m in dir(pd.DataFrame())
6        if m.startswith('to_')])
7
8 df2 = df[['a', 'b']]
9 df2.to_csv('data2.csv')
```

ЧТЕНИЕ

- Типы данных при чтении pandas пытается определить эвристиками
- Если колонка похожа на целочисленную - будет `int`
- Если попадаются числа с десятичной точкой - будет `float`
- Иначе - строка
- Отдельно обрабатываются пропущенные значения

ПРИМЕР

```
1  data = [{'a': 1, 'b': 2},
2          {'a': 5, 'b': 10, 'c': 20}]
3  df = pd.DataFrame(data)
4
5  print([m for m in dir(pd.DataFrame())
6         if m.startswith('to_')])
7
8  df2 = df[['a', 'b']]
9  df2.to_csv('data2.csv')
10
11 df3 = pd.read_csv('data2.csv')
12 print(df3.dtypes)
```

PANDAS

- Возьмем реальный набор данных
- https://data.cityofnewyork.us/Transportation/2018-Yellow-Taxi-Trip-Data/t29m-gskq
- Скачаем вручную и прочитаем
- (pandas умеет и по URL читать)

ПРОЧИТАЕМ

```
1 import pandas as pd
2
3 data = pd.read_csv(
4     '2018_Yellow_Taxi_Trip_Data_20231108.csv',
5     nrows=1000000)
6
7 print(data.dtypes)
8
9 print(data.describe())
```

ОБРАБОТАЕМ ДАТЫ

```
import pandas as pd

data = pd.read_csv(
    '2018_Yellow_Taxi_Trip_Data_20231108.csv',
    parse_dates=['tpep_pickup_datetime',
                 'tpep_dropoff_datetime'],
    nrows=100000)

print(data.dtypes)

print(data.describe())
```

СТОЛБЕЦ

```
1 import pandas as pd
2
3 data = pd.read_csv(
4     '2018_Yellow_Taxi_Trip_Data_20231108.csv',
5     parse_dates=['tpep_pickup_datetime',
6                  'tpep_dropoff_datetime'],
7     nrows=50000)
8
9 # ..............................
```

СТОЛБЕЦ

```
 1 # ..........................
 2
 3 print(data['trip_distance'])
 4 print(data['trip_distance'].describe())
 5 print(data['trip_distance'][0])
 6 print(data['trip_distance'][2])
 7 print(sum(data['trip_distance'] > 1.0))
 8
 9 selected = data['trip_distance'] > 1.0
10 print(data['trip_distance'][selected])
11 print(data['trip_distance'][selected].describe())
```

СТОЛБЕЦ

```
1 import pandas as pd
2
3 data = pd.read_csv(
4     '2018_Yellow_Taxi_Trip_Data_20231108.csv',
5     parse_dates=['tpep_pickup_datetime',
6     'tpep_dropoff_datetime'],
7     nrows=50000)
8 # ..............................
```

СТОЛБЕЦ

```
# ..............................
print(data['tip_amount'])
print(data['tip_amount'].describe())
print(data['trip_distance'][0])
print(data['trip_distance'][2])
print(sum(data['trip_distance'] > 1.0))

selected = data['trip_distance'] > 1.0
print(data['tip_amount'][selected].describe())
```

ИЗМЕНЕНИЕ СТРУКТУРЫ

- Можно создать колонку через присваивание
- Можно удалить существующую
- Например, добавить долю чаевых в сумме поездки отдельным столбцом
- А сумму - удалить

СРЕЗ ПО СТРОКАМ

- Подход отличается от numpy
- Потому что размерности более "разнородны"
- И потому что есть два подхода к адресации строк: по номеру и по индексу
- Поэтому реализован подход через view-объект

СТОЛБЕЦ

```
 1 import pandas as pd
 2
 3 data = pd.read_csv(
 4     '2018_Yellow_Taxi_Trip_Data_20231108.csv',
 5     parse_dates=['tpep_pickup_datetime',
 6                  'tpep_dropoff_datetime'],
 7    nrows=50000)
 8
 9 print(data.iloc[10])
10 print(data.iloc[10]['extra'])
11 print(data.iloc[:10])
12 print(data.iloc[:10]['extra'])
```