Язык Python. Часть 2

Лекция 15

Пример прерывистых полос

```
fig, ax = plt.subplots()
ax.broken_barh([(110, 30), (150, 10)], (10, 9), facecolors=
ax.broken_barh([(10, 50), (100, 20), (130, 10)], (20, 9),
               facecolors=('tab:orange', 'tab:green', 'tab:
ax.set_ylim(5, 35)
ax.set_xlim(0, 200)
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=
```

Разберем

- Новый метод - broken_barh
- Первый параметр - список пар
- Пара состоит из x-координаты начала полоски
- И толщины

Разберем

- Второй параметр - просто пара
- Координата полоски по оси y и высота
- Цвет можно задавать одиночно
- А можно - списком

Разберем

- Указываем засечки по оси y
- И даем им строковые имена
- Визуализируем сетку (метод grid)
- А можно - списком
- Добавляем текстовую аннотацию

Несколько панелей отдельно

- Можно создать несколько экземпляров Axes
- И в них создавать свои элементы
- Например, графики
- Можно показывать взаимосвязанные явления
- Не создавая шума

Пример

```
 1 np.random.seed(19680801)
 2
 3 dt = 0.01
 4 t = np.arange(0, 30, dt)
 5 nse1 = np.random.randn(len(t))                 # white nois
 6 nse2 = np.random.randn(len(t))                 # white nois
 7
 8 # Two signals with a coherent part at 10 Hz and a random pa
 9 s1 = np.sin(2 * np.pi * 10 * t) + nse1
10 s2 = np.sin(2 * np.pi * 10 * t) + nse2
```

Пример (продолжение)

```
fig, axs = plt.subplots(2, 1)
axs[0].plot(t, s1, t, s2)
axs[0].set_xlim(0, 2)
axs[0].set_xlabel('Time')
axs[0].set_ylabel('s1 and s2')
axs[0].grid(True)

cxy, f = axs[1].cohere(s1, s2, 256, 1. / dt)
axs[1].set_ylabel('Coherence')

fig.tight_layout()

plt.show()
```

Разберем

- plt.subplots(2, 1) - создаем сетку 2 x 1 из Axes
- Второй возвращаемый элемент - список Axes
- cohere - когерентность (некий показатель взаимодействия сигналов)
- tight_layout - обеспечивает целостную прорисовку

Заполнение области цветом

- Можно обозначить область на Axes
- Как последовательность координат точек
- Определяющую многоугольник
- И заполнить цветом

Пример: рисуем снежинку

```
def koch_snowflake(order, scale=10):
    def _koch_snowflake_complex(order):
        if order == 0:
            # initial triangle
            angles = np.array([0, 120, 240]) + 90
            return scale / np.sqrt(3) * np.exp(np.deg2rad(a
        else:
            ZR = 0.5 - 0.5j * np.sqrt(3) / 3

            p1 = _koch_snowflake_complex(order - 1)  # star
            p2 = np.roll(p1, shift=-1)  # end points
            dp = p2 - p1  # connection vectors

            new_points = np.empty(len(p1) * 4, dtype=np.com
            new_points[::4] = p1
```

Пример: рисуем снежинку

```
x, y = koch_snowflake(order=12)

plt.figure(figsize=(8, 8))
plt.axis('equal')
plt.fill(x, y)
plt.show()
```

Разберем

- Создаем очень простой Figure
- Там не нужны панели с координатными осями
- Для этого есть функция - figure
- fill - заполняет многоугольник в этой области

Пример: несколько снежинок

```
x, y = koch_snowflake(order=2)

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(9, 3),
                                    subplot_kw={'aspect': 'e
ax1.fill(x, y)
ax2.fill(x, y, facecolor='lightsalmon', edgecolor='orangered
ax3.fill(x, y, facecolor='none', edgecolor='purple', linewid

plt.show()
```

Вернемся к линиям

- Линии могут быть пунктирными
- Пунктирность можно настраивать
- Последовательностью чисел четной длины
- Ненулевой

Вернемся к линиям

- Сначала длина линии
- Потом - длина пробела
- В простейшем случае - все
- Но можно продолжить
- Получится сложный пунктир

Пример

```
x = np.linspace(0, 10, 500)
y = np.sin(x)

plt.rc('lines', linewidth=2.5)
fig, ax = plt.subplots()

line1, = ax.plot(x, y, label='Using set_dashes() and set_da
line1.set_dashes([2, 2, 10, 2])  # 2pt line, 2pt break, 10p
line1.set_dash_capstyle('round')

line2, = ax.plot(x, y - 0.2, dashes=[6, 2], label='Using th

line3, = ax.plot(x, y - 0.4, dashes=[4, 4],
                 label='Using the dashes and gapcolor param

```

Есть предопределенные стили

- Задаются строками
- 'solid', 'dotted', 'dashed', 'dashdot'
- Посмотрим на демострацию разных стилей
- (Исходники в репозитории)

Изображения текста

- Есть типовые формы избражения текста
- Легенды, подписи под осями координат, возле диаграмм
- Есть произвольные формы
- Написать что-то каком-то месте изображения

Изображения текста

- В типовых не указываются координаты
- Вызываются методы типа set_title/set_xlabel
- Достаточно вызвать метод и передать текст
- Matplotlib разбирается сам, где его разместить

Изображения текста

- Для произвольного текста есть универсальный метод текст
- Ему передаем не только текст
- Еще координаты расположения
- И часто - уточняющие детали

Пример

```
# полная версия  - в репозитории

ax.text(3, 8, 'boxed italics text in data coords', style='i
        bbox={'facecolor': 'red', 'alpha': 0.5, 'pad': 10})

ax.text(2, 6, r'an equation: $E=mc^2$', fontsize=15)

ax.text(3, 2, 'Unicode: Institut f\374r Festk\366rperphysik

ax.text(0.95, 0.01, 'colored text in axes coords',
        verticalalignment='bottom', horizontalalignment='ri
        transform=ax.transAxes,
        color='green', fontsize=15)
#...
ax.set(xlim=(0, 10), ylim=(0, 10))
```

Разберем пример

- Первые два параметра - координаты x и y
- Последний вызов - особый случай
- Координаты - в терминах настроек Axes
- Координаты левого нижнего угла текста

Разберем пример

- Сам текст - третий параметр
- Очень много именованных параметров
- Задают настройки текста
- Полное описание - в документации

Разберем пример

- В тексте используется "математическое расширение" - mathext
- Простой язык, позволяющий писать формулы
- Действует '$' до следующего '$'
- Дроби, верхние и нижние индексы, греческие буквы, спецсимволы...
- Полное описание

Разберем пример

- В последнем сильно другие координаты
- И выглядят по-другому, и интерпретируются
- Все из-за параметра transform
- Он задает преобразование координат

Разберем пример

- Можно строить разные системы прямоугольных координат
- С разной начальной точкой
- И разным масштабом
- И одна точка будет иметь разные значения координат в ней

Разберем пример

- Matplotlib позволяет задавать свои преобразования координат
- Чтобы помещать текст в какой-либо Axes
- Но координаты указывать не в левом нижнем углу Axes
- И/или не в том масштабе, в каком заданы видимые оси

Разберем пример

- Можно задавать произвольные
- Но есть несколько предопределенных
- Одно из них Axes.transAxes
- У каждого экземпляра Axes оно свое

Разберем пример

- В этом преобразовании оси совпадают
- Меняется только масштаб
- Правая граница по оси X берется за 1
- Аналогично с верхней границей по оси Y

Разберем пример

- Еще есть Figure.trasnsFigure
- Свое у каждого экземпляра Figure
- Меняются оси (параллельный перенос) и масштаб
- За начало берется левый нижний угол Figure
- За 1 - границы Figure

Пример: водяной знак

```
 1 import numpy as np
 2 import matplotlib.pyplot as plt
 3
 4 np.random.seed(42)
 5
 6 fig, ax = plt.subplots()
 7 ax.plot(np.random.rand(20), '-o', ms=20, lw=2, alpha=0.7, m
 8 ax.grid()
 9
10 ax.text(0.5, 0.5, 'created with matplotlib', transform=ax.t
11         fontsize=40, color='gray', alpha=0.5,
12         ha='center', va='center', rotation=30)
13
14 plt.show()
```