Язык Python. Часть 2

Лекция 14

Matplotlib

- Библиотека для рисования
- Ориентирована на графики, диаграммы
- Может работать в разных средах
- Простейший вариант - приложение в среде с поддержкой GUI

Пример

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 x = np.linspace(0, 2, 100)
 5
 6 fig, ax = plt.subplots(figsize=(5, 2.7), layout='constraine
 7 ax.plot(x, x, label='linear')
 8 ax.plot(x, x**2, label='quadratic')
 9 ax.plot(x, x**3, label='cubic')
10 ax.set_xlabel('x label')
11 ax.set_ylabel('y label')
12 ax.set_title("Simple Plot")
13 ax.legend()
```

Разберем пример

- linspace - функция numpy
- Создаем ndarray
- Размерностью 1
- 100 элементов
- Делим отрезок [0, 2] на 100 частей

Разберем пример

- matplotlib нужны исходные данные для прорисовки
- Обычно это одномерные коллекции
- Или двумерные
- Могут быть стандартные питоновские
- Часто - структуры numpy и pandas

subplots

- Дальше идет функция subplots
- Она возвращает пару значений
- Первое даже не используется в примере
- Над вторым вызывается много методов

Некоторые понятия

- В matplotplib много классов
- Они увязаны в иерархию наследования
- Ближе к корню - общие понятия и сущности
- Ближе к листьям - более конкретные

Некоторые понятия

- Artist - абстрактное понятие
- Все, что прорисовывается
- Представлено классом
- Очень много классов наследуются от Artist
- Прямо или косвенно

Некоторые понятия

- Figure - изображение в целом
- Наследуется от Artist (через FigureBase)
- Содержит свойства и методы, унаследованные от Artist
- А также свои
- Или от FigureBase

Некоторые понятия

- Первый элемент возвращаемой пары - Figure
- Второй элемент - типа Axes
- Axes - тоже наследуется от Artist
- Это панель для прорисовки графиков

Axes

- Метод plot - добавляет график на панель
- Первый параметр - координаты по оси x
- Второй - по оси y
- Это панель для прорисовки графиков

Axes

- label - метка
- Проявляется в легенде
- Уберем ax.legend() - пропадет легенда
- Остальное - установка декораций

Поэксперементируем

- Радикально снизим количество элементов в linspace
- Например, до 10 или 5
- Нелинейные графики станут "угловатыми"
- Прорисовываются точки, и соединяются линиями
- Чем плотнее, тем больше похоже на график

Поэксперементируем

- Поменяем местами точки
- Графики станут странными
- Никто не сортирует точки
- Фактически - рисуется ломаная

Пример

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 x = np.linspace(0, 2, 5)
 5
 6 x = x[[1, 0, 3, 2, 4]]
 7
 8 fig, ax = plt.subplots(figsize=(5, 2.7), layout='constraine
 9 ax.plot(x, x, label='linear')
10 ax.plot(x, x**2, label='quadratic')
11 ax.plot(x, x**3, label='cubic')
12 ax.set_xlabel('x label')
13 ax.set_ylabel('y label')
14 ax.set_title("Simple Plot")
15 ax.legend()
```

Пример на другие функции

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

x = np.linspace(0, 20, 2000)

print(np.sin(x))

fig, ax = plt.subplots(layout='constrained')
ax.plot(x, np.sin(x), label='sin')
ax.plot(x, np.cos(x), label='cos')
ax.plot(x, np.sin(x) ** 2, label='sin^2')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.set_title("Simple Plot")
ax.legend()
```

Диаграммы

- Графики - не единственная форма визуализации
- Пример: фрукты
- Хотим визулализировать количество проданного
- Будем рисовать полоски пропорционально количеству

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', 'tab:red', 'tab:orange
 9
10 ax.bar(fruits, counts, label=bar_labels, color=bar_colors)
11
12 ax.set_ylabel('fruit supply')
13 ax.set_title('Fruit supply by kind and color')
14 ax.legend(title='Fruit color')
15
```

Разберемся

- Общая структура - та же
- Создаем Figure и Axes
- Добавляем элементы через Axes
- Внешнее обрамление похоже
- Только основной визуальный элемент - другой

Разовьем идею

- В каждой полоске хотим визуализировать части
- Например, каждая полоска - продажи товара
- Но по каждому товару есть разбивка покупателей
- Например, на мужчин и женщин
- Хотим визуализировать

Пример

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 species = ('Adelie', 'Chinstrap', 'Gentoo')
 5 sex_counts = {
 6     'Male': np.array([73, 34, 61]),
 7     'Female': np.array([73, 34, 58]),
 8 }
 9 width = 0.6
10
11 fig, ax = plt.subplots()
12 bottom = np.zeros(3)
13
14 for sex, sex_count in sex_counts.items():
15     p = ax.bar(species, sex_count, width, label=sex, bottom
```

Разберемся

- Используем параметр bottom
- Вызываем ax.bar дважды
- Первый раз - с общим значением bottom
- Второй раз - у каждого свое

Другие варианты

- Не обязательно полоскам быть вертикальными
- Попробуеи горизонтально
- Есть отдельный метод у Axes
- Называется barh

Пример

```
 1 np.random.seed(19680801)
 2
 3 people = ('Tom', 'Dick', 'Harry', 'Slim', 'Jim')
 4 y_pos = np.arange(len(people))
 5 performance = 3 + 10 * np.random.rand(len(people))
 6 error = np.random.rand(len(people))
 7
 8 fig, ax = plt.subplots()
 9
10 hbars = ax.barh(y_pos, performance, xerr=error, align='cent
11 ax.set_yticks(y_pos, labels=people)
12 ax.invert_yaxis()
13 ax.set_xlabel('Performance')
14 ax.set_title('How fast do you want to go today?')
15
```

Разберемся

- Основа похожа на прошлые примеры
- Добавляем инверсию данных - invert_yaxis
- Инвертируются оба массива
- Устанавливаем лимит по оси x

Группировка полосок

- Был пример с двумя цветами в одной полоске
- А если вариантов больше, чем два ?
- Можно развить тот вариант
- Но иногда визуально больше подходит другой

Группировка полосок

- Поставить несколько полосок вплотную
- Относящихся к одному показателю "первого уровня"
- А относящиеся к другим показателям - в других группах
- На отдалении

Пример настройки меток

```
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),
}

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
```

Разберем

- Похоже на случай двух цветов
- Только здесь не bottom меняем, а offset
- Через изменение multiplier
- На каждой итерации рисуем по 3 полоски