

## Лабораторная работа № 11.

Тема: Построение графиков и визуализация данных с использованием NumPy и Matplotlib.

Matplotlib — это библиотека для создания статических, анимированных и интерактивных графиков в языке программирования Python. Она предоставляет широкие возможности для визуализации данных и используется в различных областях, таких как научные исследования, анализ данных, машинное обучение, и другие.

Вот несколько ключевых аспектов Matplotlib:

- |   |  |
|---|--|
| 1. <b>Простота использования:</b>             | Matplotlib обладает простым и интуитивно понятным интерфейсом, что делает ее доступной для широкого круга пользователей.                                     |
| 2. <b>Гибкость:</b>                           | Библиотека предоставляет множество возможностей для настройки графиков, включая выбор цвета, маркеров, шрифтов и многое другое.                              |
| 3. <b>Поддержка различных типов графиков:</b> | Matplotlib поддерживает создание различных типов графиков, таких как линейные графики, столбчатые диаграммы, круговые диаграммы, гистограммы, графики и т.д. |
| 4. <b>Интеграция с NumPy:</b>                 | Matplotlib хорошо интегрируется с библиотекой NumPy, что позволяет легко визуализировать данные, хранящиеся в массивах NumPy.                                |

Пример создания простого графика с использованием Matplotlib:

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

# Создание данных
x = np.linspace(0, 2 * np.pi, 100)
y = np.sin(x)

# Построение графика
plt.plot(x, y)
plt.title('Пример графика с Matplotlib')
plt.xlabel('X-ось')
plt.ylabel('Y-ось')

# Отображение графика
plt.show()
```

Краткий мануал по библиотеке **Matplotlib** вы найдете в файле `Python_matplotlib.pdf`.

**NumPy** (Numerical Python) — это библиотека для языка программирования Python, предназначенная для выполнения операций с многомерными массивами и матрицами, а также предоставляющая функциональность для работы с линейной алгеброй, случайными числами и другими математическими операциями. Вот несколько ключевых аспектов NumPy:

1. **Многомерные массивы:** NumPy предоставляет **numpy.ndarray**, который является объектом
  2. **Быстрые математические операции:** с помощью **numpy**, позволяя эффективно оперировать данными. Оптимизирует векторизованные математические операции, избегая явных циклов,
  3. делает их быстрыми и эффективными.
  4. **Линейная алгебра:** предоставляет функции для таких как умножение матриц, вычисление определителя и операций с собственными значениями.
- Библиотеки:** NumPy является основой для многих других научных вычислений.
- Интеграция с другими:** NumPy интегрируется с другими библиотеками, такими как pandas и matplotlib.

Пример использования NumPy для создания массива и выполнения операций:

```
import numpy as np

# Создание одномерного массива
arr = np.array([1, 2, 3, 4, 5])

# Умножение каждого элемента на 2
arr = arr * 2

# Вывод результата
print(arr)
```

Некоторые из методов **NumPy**, с примерами кода и комментариями:

**numpy.linspace(start, stop, num):** Создает массив равномерно распределенных значений от **start** до **stop** включительно с заданным числом элементов **num**. Он часто используется для создания оси x на графиках.

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

x = np.linspace(0, 10, 100) # Создание массива из 100 чисел между 0 и 10
y = np.sin(x) # Вычисление синуса каждого элемента массива x

plt.plot(x, y) # Построение графика
plt.show() # Отображение графика
```

**numpy.arange(start, stop, step):** Создает массив значений, начиная с **start**, заканчивая **stop** (не включительно), с шагом **step**. Он также часто используется для создания оси x на графиках.

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

x = np.arange(0, 10, 0.1) # Создание массива чисел с шагом 0.1 между 0 и 10
y = np.cos(x) # Вычисление косинуса каждого элемента массива x

plt.plot(x, y) # Построение графика
plt.show() # Отображение графика
```

**numpy.random:** предоставляет функции для генерации случайных чисел. Он может быть использован для создания случайных данных для построения графиков.

**numpy.random.normal** - это функция из подмодуля **numpy.random** библиотеки NumPy в Python, предназначенная для генерации случайных чисел из нормального распределения (гауссовского распределения).

```
import numpy as np

# Генерация случайных чисел из нормального распределения
mean = 0 # Среднее значение распределения
std_dev = 1 # Стандартное отклонение распределения
size = (3, 3) # Размерность массива случайных чисел

random_numbers = np.random.normal(mean, std_dev, size)
```

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

# Генерация 100 случайных чисел из нормального распределения
x = np.random.normal(0, 1, 100)
y = np.random.normal(0, 1, 100)

plt.scatter(x, y) # Построение точечного графика
plt.show() # Отображение графика
```

**numpy.random.rand** - это функция из подмодуля `numpy.random` библиотеки NumPy в Python, предназначенная для генерации случайных чисел из равномерного распределения в интервале [0, 1).

```
1 import numpy as np
2
3 # Генерация случайных чисел из равномерного распределения
4 size = (3, 3) # Размерность массива случайных чисел
5 random_numbers = np.random.rand(*size)
6 print(random_numbers)
7
```

Оболочка x

```
>>> %Run -c $EDITOR_CONTENT
[[0.40625179 0.66015179 0.23523556]
 [0.96799884 0.71258997 0.80533994]
 [0.38228954 0.08399615 0.42810239]]
```

```
1 import numpy as np
2
3 # Генерация случайных чисел из равномерного распределения
4 size = (10) # Размерность массива случайных чисел
5 random_numbers = np.random.rand(size)
6 print(random_numbers)
7
```

Оболочка x

```
>>> %Run -c $EDITOR_CONTENT
[0.47956144 0.7276545 0.70799375 0.24366125 0.3639987 0.39183938
 0.22141719 0.16136106 0.33075089 0.09757296]
```

**numpy.random.randint** - это функция из подмодуля `numpy.random` библиотеки NumPy в Python, предназначенная для генерации случайных целых чисел из заданного диапазона.

```
import numpy as np

# Генерация случайных целых чисел из заданного диапазона
low = 1 # Нижняя граница диапазона (включительно)
high = 10 # Верхняя граница диапазона (исключительно)
size = (3, 3) # Размерность массива случайных чисел

random_integers = np.random.randint(low, high, size)
```

**numpy.histogram**: Этот метод позволяет создать гистограмму на основе данных. Он разбивает данные на интервалы и подсчитывает количество значений в каждом интервале.

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

# Генерация 100 случайных чисел из нормального распределения
data = np.random.normal(0, 1, 100)

plt.hist(data, bins=10) # Построение гистограммы с 10 интервалами
plt.show() # Отображение графика
```

| Вариант 1   | Вариант 2  |
|---|--|
| Постройте график функции $y = x^2$  | Постройте график функции $y = 1 / x^3$   |
| В файле shop.csv находится статистика продаж, расходов, заработной платы и прибыли по месяцам. Постройте столбчатую диаграмму продаж по месяцам.                | В файле shop.csv находится статистика продаж, расходов, заработной платы и прибыли по месяцам. Постройте столбчатую диаграмму прибыли по месяцам.      |
| В файле shop.csv находится статистика продаж, расходов, заработной платы и прибыли по месяцам. Постройте два графика на одном поле: заработной платы и прибыли. | В файле shop.csv находится статистика продаж, расходов, заработной платы и прибыли по месяцам. Постройте два графика на одном поле: расходов и продаж. |