Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Новосибирский национальный исследовательский государственный университет»

# БИБЛИОТЕКА MATPLOTLIB И ПАКЕТ PYPLOT. ВИЗУАЛИЗАЦИЯ ДАННЫХ

ВИЗУАЛИЗАЦИЯ ДАННЫХ

СЛУЧАЙНОЕ БЛУЖДАНИЕ

Гаврилов Денис Андреевич, преподаватель кафедры СИ ФИТ НГУ

Инструменты визуализации данных.
Библиотека matplotlib.

Построение простого графика.

Настройка параметров графика.

Встроенные стили отображения графика.

# ВИЗУАЛИЗАЦИЯ ДАННЫХ

# БИБЛИОТЕКА MATPLOTLIB

**Matplotlib** — библиотека на языке программирования Python для визуализации данных при помощи, преимущественно, двумерной графики.

С помощью **matplotlib** можно строить диаграммы, графики, диаграммы разброса данных и т.д.

# БИБЛИОТЕКА MATPLOTLIB

В библиотеке **Matplotlib** присутствует множество пакетов для различных нужд.

В рамках текущего курса, мы рассмотрим работу с пакетом **pyplot**.

**Pyplot** - это набор функций, позволяющих строить графики и диаграммы схожим с MATLAB образом.

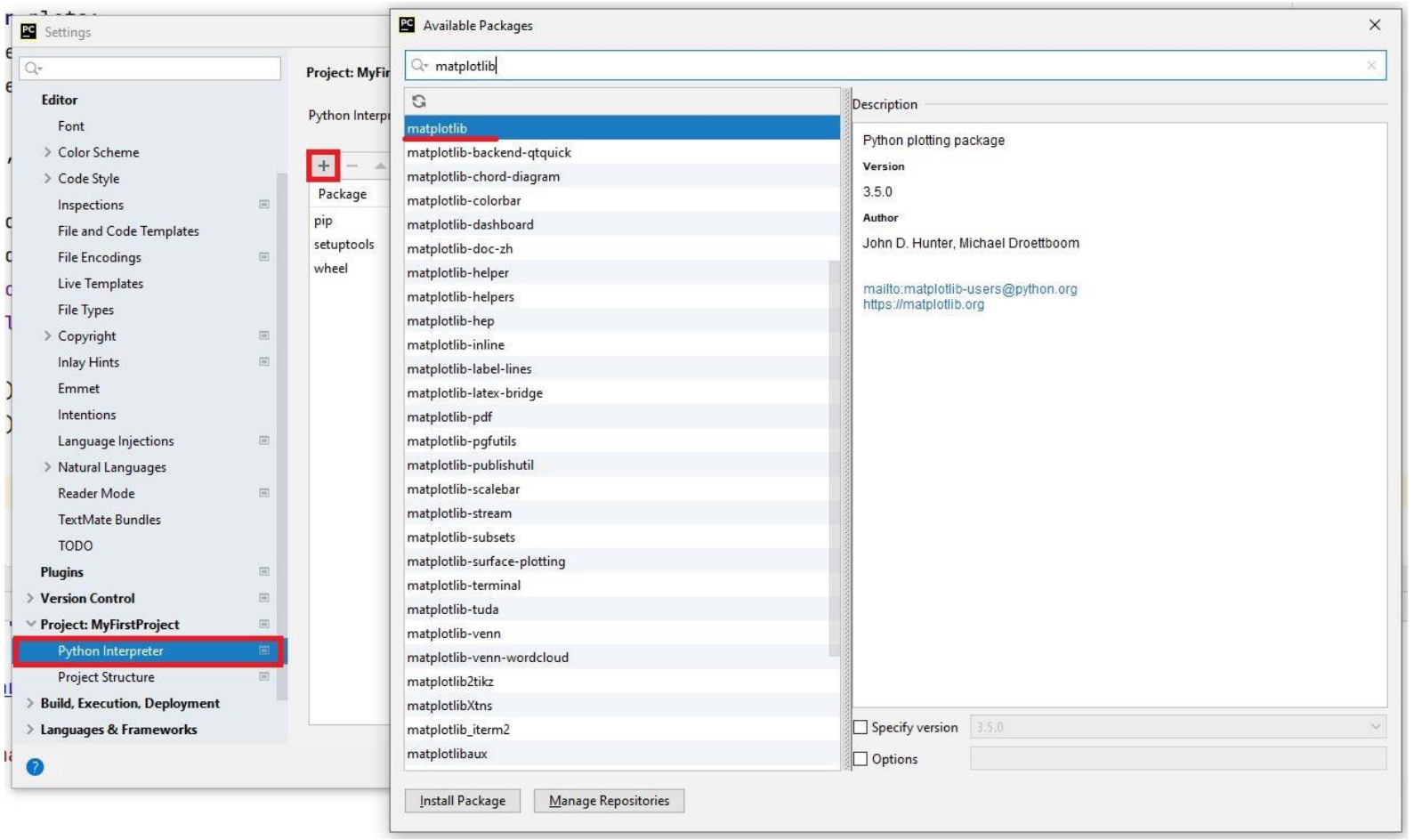
Каждая функция **pyplot** отвечает за различные параметры конструируемого графика: например, разметка области построения графика, построение элементов графика, установка меток и легенды и т.д..

# УСТАНОВКА MATPLOTLIB

Изначально библиотека **Matplotlib** недоступна для использования.

- Для установки **Matplotlib**, нужно войти в меню *File|Settings|Project Interpreter* и найти библиотеку **Matplotlib** в списке (или воспользоваться графой поиска).
- После выбора библиотеки, её можно загрузить, нажав *Install Package* внизу экрана.

# УСТАНОВКА MATPLOTLIB



# ПОСТРОЕНИЕ ПРОСТОГО ГРАФИКА

```
import matplotlib.pyplot as plt

heights = [2, 75, 89, 4, 35, 15]

fig, ax = plt.subplots()

ax.plot(heights)

plt.show()
```

- Подключение пакета *pyplot* из модуля *matplotlib*;
- Переобозначение полного названия пакета (*matplotlib.pyplot*) сокращенным именем *plt*;
- Ввод исходных данных для дальнейшего отображения;

# ПОСТРОЕНИЕ ПРОСТОГО ГРАФИКА

```
import matplotlib.pyplot as plt

heights = [2, 75, 89, 4, 35, 15]

fig, ax = plt.subplots()

ax.plot(heights)

plt.show()
```

- Определение рисунка (figure) и набора графиков, содержащихся на данном рисунке (plots);
- Иногда возникает необходимость размещения нескольких графиков на одной иллюстрации;
- Пример:

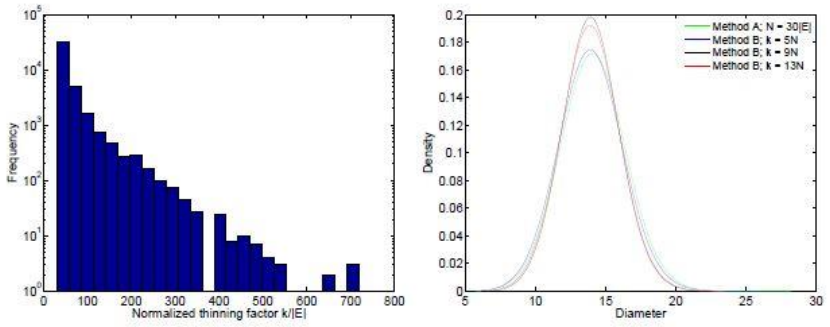



**Fig. 4.** Left: The normalized thinning factor $k/|E|$ for the Epinions graph, as calculated for the 40,574 sampled edges. We see that the most thinning factors are lie in $(10|E|, 100|E|)$. Right: Plot of the graph diameter and distribution generated using Method A (with $N = 30|E|$) and Method B (with $k$ equal to various multiples of $N$). We see that the distributions are very similar.

# ПОСТРОЕНИЕ ПРОСТОГО ГРАФИКА

```
import matplotlib.pyplot as plt

heights = [2, 75, 89, 4, 35, 15]

fig, ax = plt.subplots()

ax.plot(heights)

plt.show()
```

- Определение рисунка (figure) и набора графиков, содержащихся на данном рисунке (plots);
  - Иногда возникает необходимость размещения нескольких графиков на одной иллюстрации;
- Данный код формирует рисунок с единственным графиком на нем;

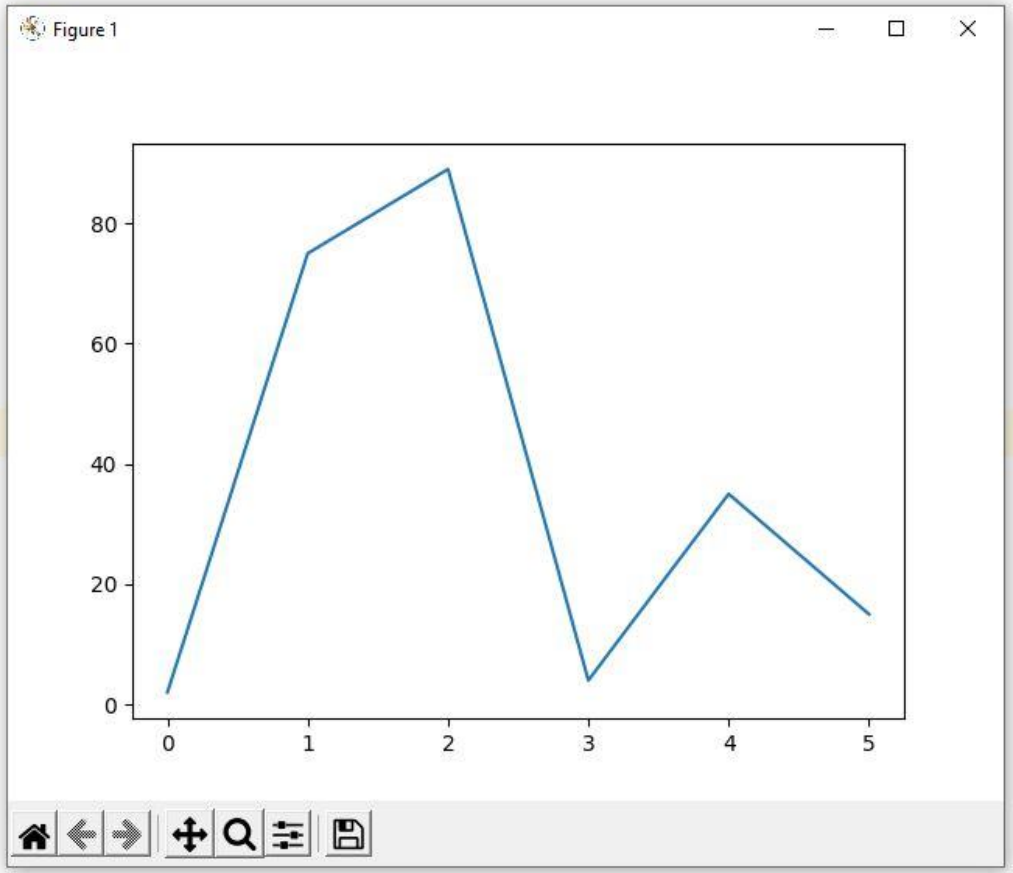
# ПОСТРОЕНИЕ ПРОСТОГО ГРАФИКА

```
import matplotlib.pyplot as plt

heights = [2, 75, 89, 4, 35, 15]

fig, ax = plt.subplots()

ax.plot(heights)

plt.show()
```

- Метод *plot()* строит графическое представление (диаграмму) для заданных чисел;
  - Построение происходит на графике, у которого данный метод был вызван;
- Вызов *plt.show()* открывает окно просмотра и выводит на него рисунок с построенными диаграммами;

# ПОСТРОЕНИЕ ПРОСТОГО ГРАФИКА

```
1  import matplotlib.pyplot as plt
2
3  heights = [2, 75, 89, 4, 35, 15]
4
5  fig, ax = plt.subplots()
6
7  ax.plot(heights)
8
9  plt.show()
10
```



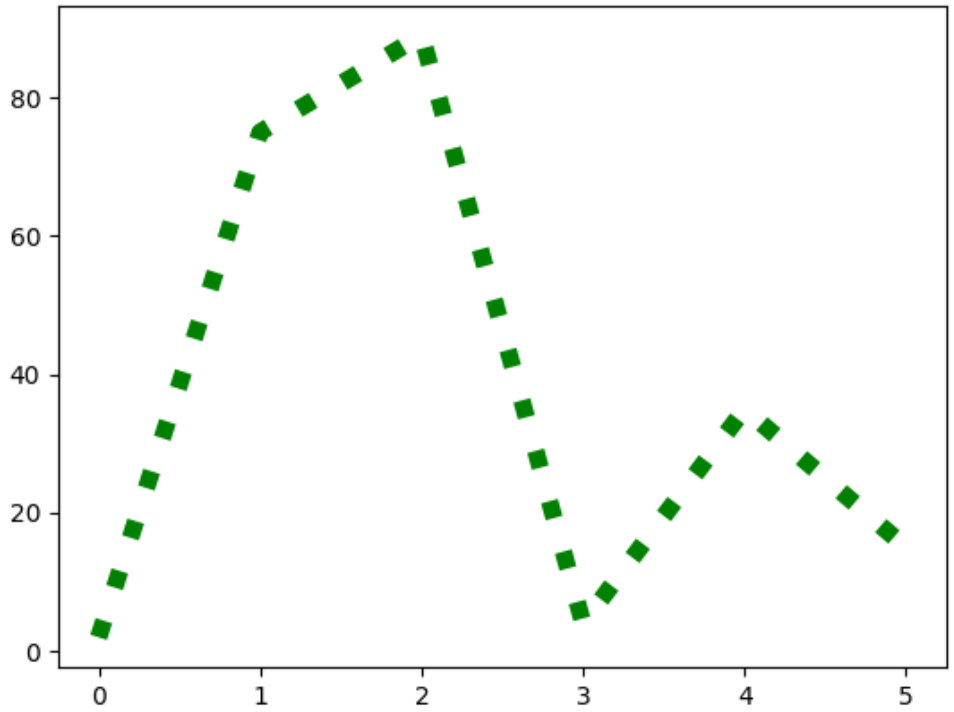
# НАСТРОЙКА ПАРАМЕТРОВ: ТОЛЩИНА, ЦВЕТ, ШТРИХ

```
import matplotlib.pyplot as plt

heights = [2, 75, 89, 4, 35, 15]

fig, ax = plt.subplots()

ax.plot(heights,
        linestyle=(0, (1, 2)),
        linewidth=7,
        color="Green")

plt.show()
```

- *linestyle* – стиль линии графика;
  - Можно задать как кортежем значений (кодом), так и строковым названием (для заранее заготовленных стилей);
  - В данном примере эквивалентно *linestyle="dotted"*;
- *linewidth* – толщина линии графика;
- *color* – цвет линии графика;
  - Можно задать как строковым названием (для некоторых цветов), так и RGBA-кодом;
  - В данном примере эквивалентно *color="#007700ff";*

# НАСТРОЙКА ПАРАМЕТРОВ: ТОЛЩИНА, ЦВЕТ, ШТРИХ

```
import matplotlib.pyplot as plt

heights = [2, 75, 89, 4, 35, 15]

fig, ax = plt.subplots()

ax.plot(heights,
        linestyle=(0, (1, 2)),
        linewidth=7,
        color="Green")

plt.show()
```



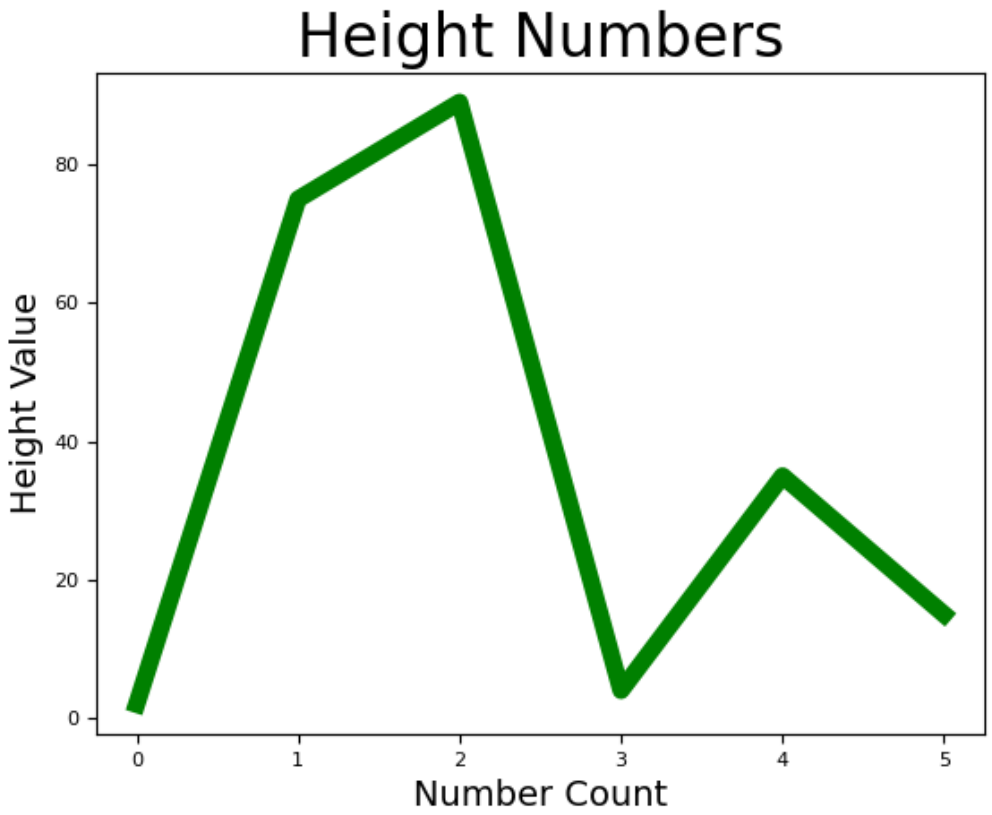
# НАСТРОЙКА ПАРАМЕТРОВ: ЗАГОЛОВКИ, МЕТКИ ОСЕЙ

```
import matplotlib.pyplot as plt

heights = [2, 75, 89, 4, 35, 15]

fig, ax = plt.subplots()

ax.plot(heights, linewidth=7, color="Green")

ax.set_title("Height Numbers", fontsize=24)
ax.set_xlabel("Number Count", fontsize=14)
ax.set_ylabel("Height Value", fontsize=14)
ax.tick_params(axis='both', labelsize=8)

plt.show()
```

- *set_title()* – задает название графика;
- *set_xlabel()* – задает название оси *X*;
- *set_ylabel()* – задает название оси *Y*;
- *tick_params()* – задает параметры числовых отметок на осях;
  - *labelsize / fontsize* – задает размер шрифта;

# НАСТРОЙКА ПАРАМЕТРОВ: ЗАГОЛОВКИ, МЕТКИ ОСЕЙ

```
import matplotlib.pyplot as plt

heights = [2, 75, 89, 4, 35, 15]

fig, ax = plt.subplots()

ax.plot(heights, linewidth=7, color="Green")

ax.set_title("Height Numbers", fontsize=24)
ax.set_xlabel("Number Count", fontsize=14)
ax.set_ylabel("Height Value", fontsize=14)
ax.tick_params(axis='both', labelsize=8)

plt.show()
```



# ВСТРОЕННЫЕ СТИЛИ ГРАФИКОВ

Для быстрого комплексного форматирования графика можно воспользоваться готовыми встроенными стилями.

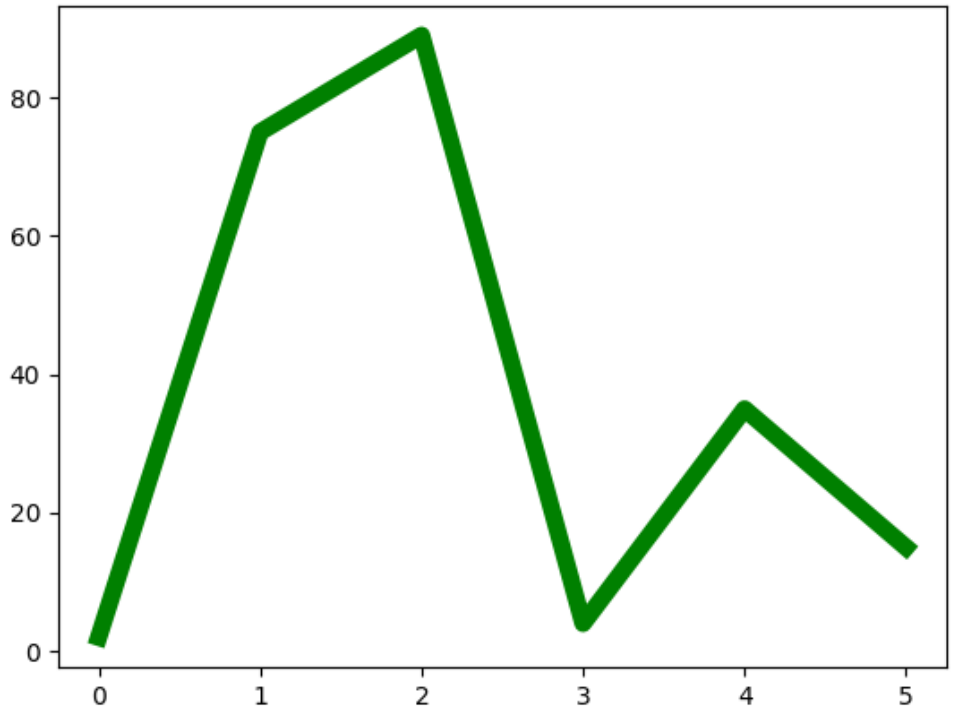
Для того, чтобы узнать список доступных стилей, можно выполнить следующий код:

```
import matplotlib.pyplot as plt

print(plt.style.available)
```

['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn',..

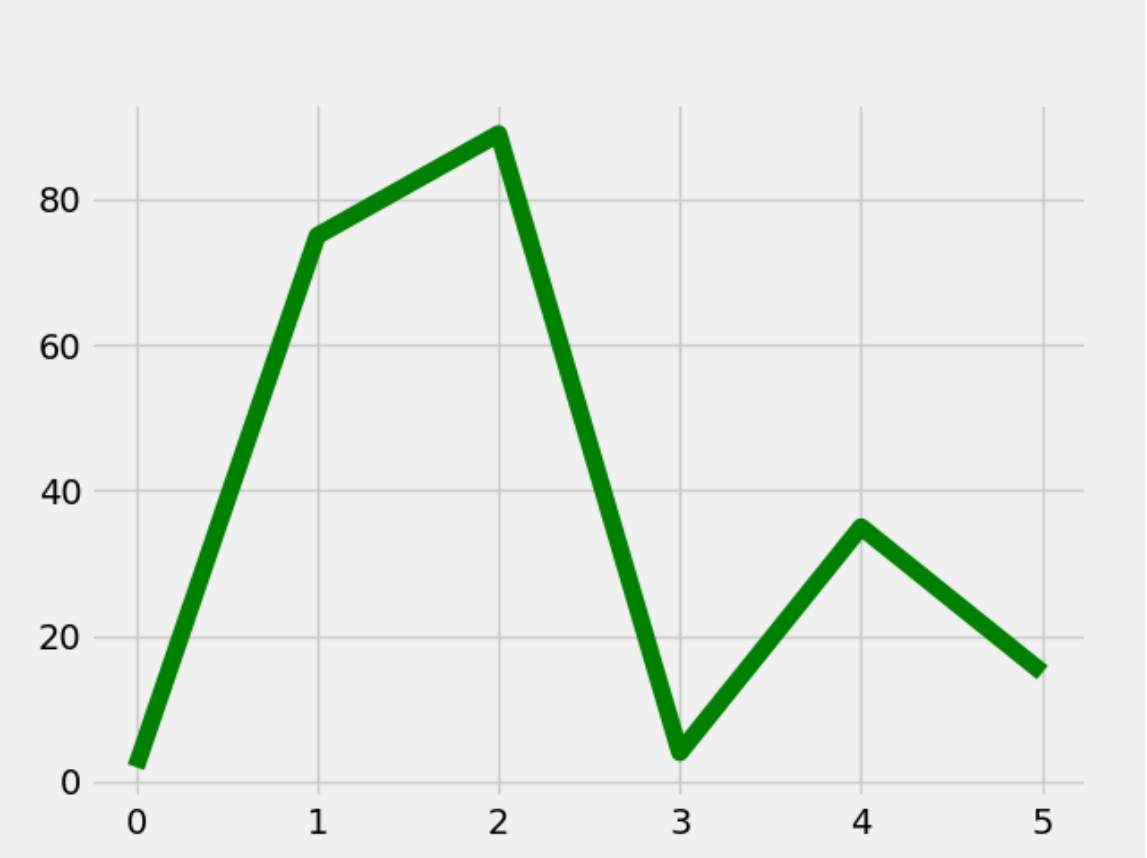
# ВСТРОЕННЫЕ СТИЛИ ГРАФИКОВ

```
import matplotlib.pyplot as plt

heights = [2, 75, 89, 4, 35, 15]

plt.style.use('grayscale')

fig, ax = plt.subplots()

ax.plot(heights, linewidth=7, color="Green")

plt.show()
```



# ВСТРОЕННЫЕ СТИЛИ ГРАФИКОВ

```
import matplotlib.pyplot as plt

heights = [2, 75, 89, 4, 35, 15]

plt.style.use('fivethirtyeight')

fig, ax = plt.subplots()

ax.plot(heights, linewidth=7, color="Green")

plt.show()
```



Случайное блуждание.

Подготовка данных:

определение класса.

Подготовка данных:

определение методов.

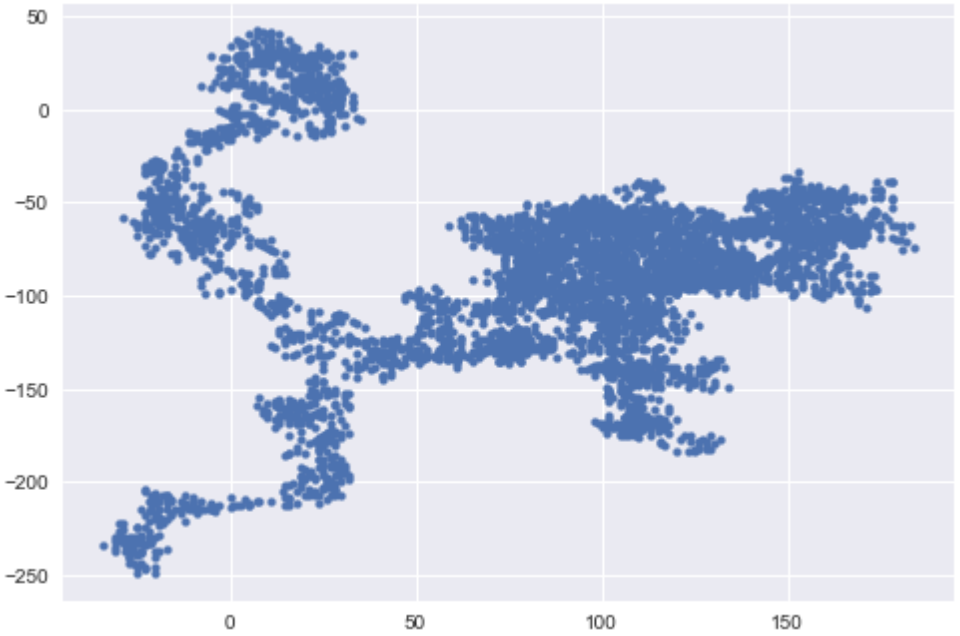
Визуализация данных.

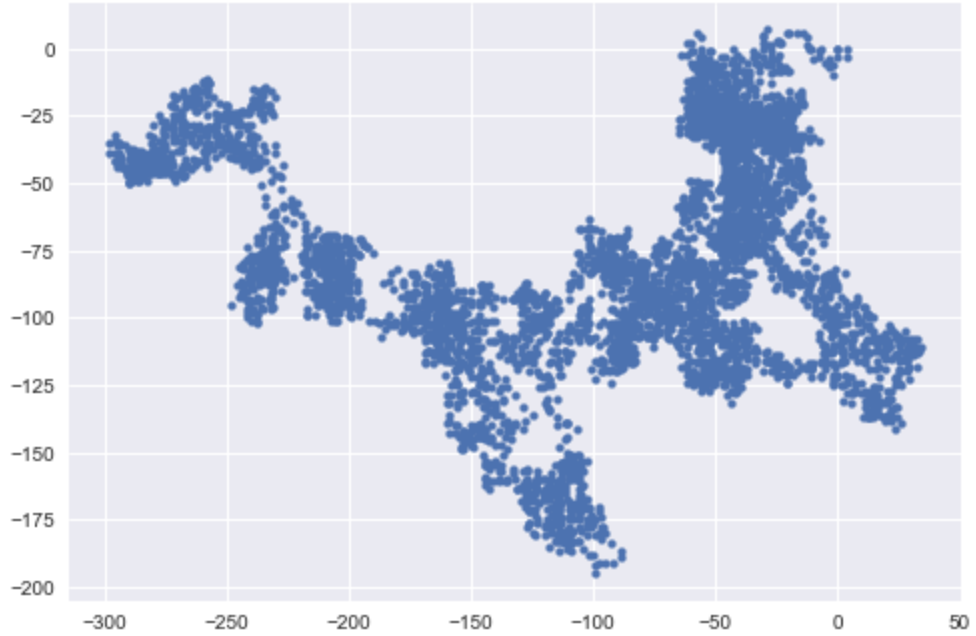
# ПРИМЕР ВИЗУАЛИЗАЦИИ; СЛУЧАЙНОЕ БЛУЖДАНИЕ

# СЛУЧАЙНОЕ БЛУЖДАНИЕ

**Случайным блужданием** (*random walk*) называется путь, не имеющий четкого направления и определяющийся серией случайных решений.

Примеры:



# ПОДГОТОВКА ДАННЫХ: ОПРЕДЕЛЕНИЕ КЛАССА

```
import matplotlib.pyplot as plt
from random import choice


class RandomWalk():

    def __init__(self, num_points=5000):
        self.num_points = num_points

        self.x_values = [0]
        self.y_values = [0]

. . .
```

# Подключение необходимых библиотек;
# Random – позволяет работать со случайными значениями

# Определение класса

# Определение конструктора

# Инициализация параметров случайного блуждания;
# Число точек, списки координат точек

# ПОДГОТОВКА ДАННЫХ: ОПРЕДЕЛЕНИЕ МЕТОДОВ

```
def fill_walk(self):

    while len(self.x_values) < self.num_points:

        x_direction = choice([1, -1])
        x_distance = choice([0, 1, 2, 3, 4])
        x_step = x_direction*x_distance

        y_direction = choice([1, -1])
        y_distance = choice([0, 1, 2, 3, 4])
        y_step = y_direction*y_distance

        . . .
```

```
# Определение метода, формирующего набор точек
# случайного блуждания

# Цикл – пока заданное число точек не будет сформировано

# Вычисление случайного шага по координате X;
# Шаг состоит из случайного направления (вправо/влево),
# а также случайной длины (от 0 до 4)

# Аналогично для координаты Y
```

# ПОДГОТОВКА ДАННЫХ: ОПРЕДЕЛЕНИЕ МЕТОДОВ

```
. . .

if x_step == 0 and y_step == 0:
    continue



x = self.x_values[-1] + x_step
y = self.y_values[-1] + y_step

self.x_values.append(x)
self.y_values.append(y)
```

# Если шаги по обеим координатам оказались нулевыми,

# Повторить предыдущий шаг

# Увеличение последних сформированных координат

# на вычисленные шаги,

# и добавление новых координат (новой точки) в списки

# ВИЗУАЛИЗАЦИЯ ДАННЫХ: СПОСОБ ОТОБРАЖЕНИЯ

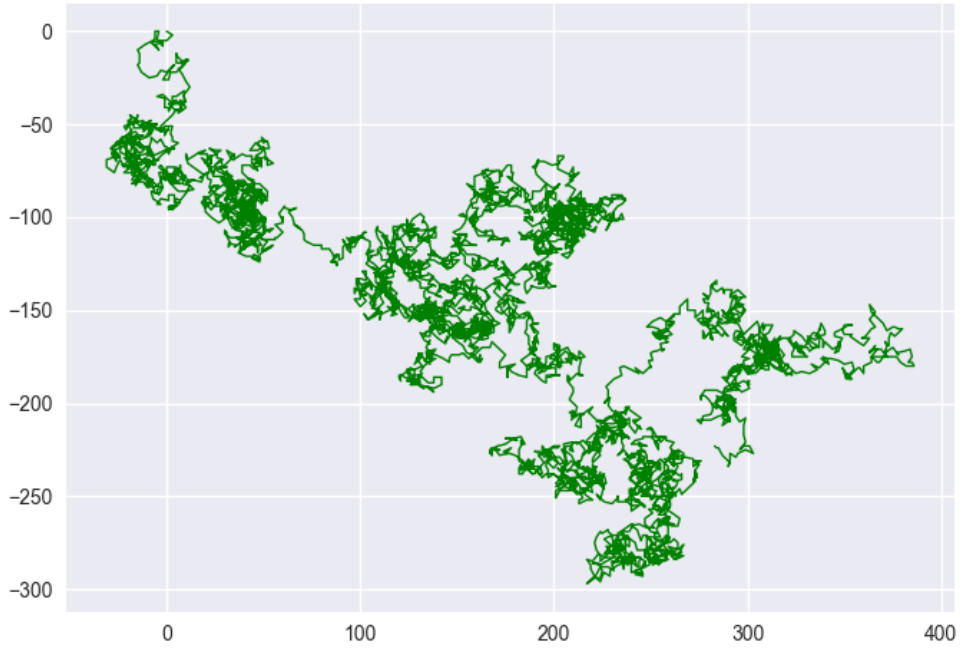
```
rw = RandomWalk()
rw.fill_walk()

plt.style.use('seaborn')

fig, ax = plt.subplots()

ax.plot(rw.x_values, rw.y_values,
        linewidth=1, color="Green")

plt.show()
```

- В случае использования *plot()* для построения диаграммы, мы получаем график, координатные точки которого соединены линией;

- В качестве альтернативы, мы можем построить диаграмму, отображающую распределение точек без порядкового связывания;
  - Для этого следует воспользоваться методом *scatter()*;

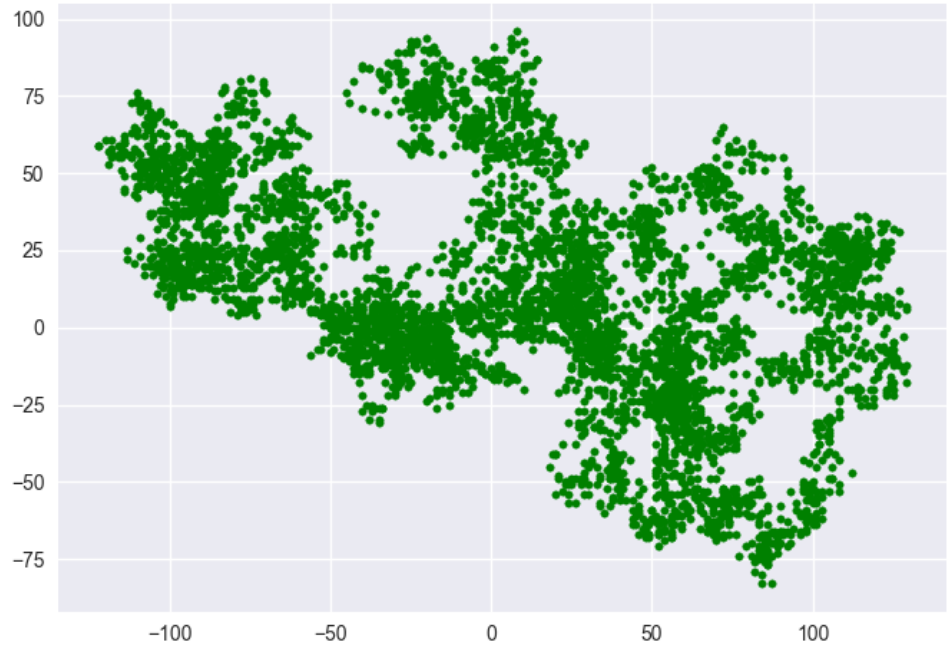
# ВИЗУАЛИЗАЦИЯ ДАННЫХ: СПОСОБ ОТОБРАЖЕНИЯ

```
rw = RandomWalk()
rw.fill_walk()

plt.style.use('seaborn')

fig, ax = plt.subplots()

ax.plot(rw.x_values, rw.y_values,
        linewidth=1, color="Green")

plt.show()
```



# ВИЗУАЛИЗАЦИЯ ДАННЫХ: СПОСОБ ОТОБРАЖЕНИЯ

```
rw = RandomWalk()
rw.fill_walk()

plt.style.use('seaborn')

fig, ax = plt.subplots()

ax.scatter(rw.x_values, rw.y_values, c='green', s=15)

plt.show()
```



# ВИЗУАЛИЗАЦИЯ ДАННЫХ: ЦВЕТОВЫЕ КАРТЫ

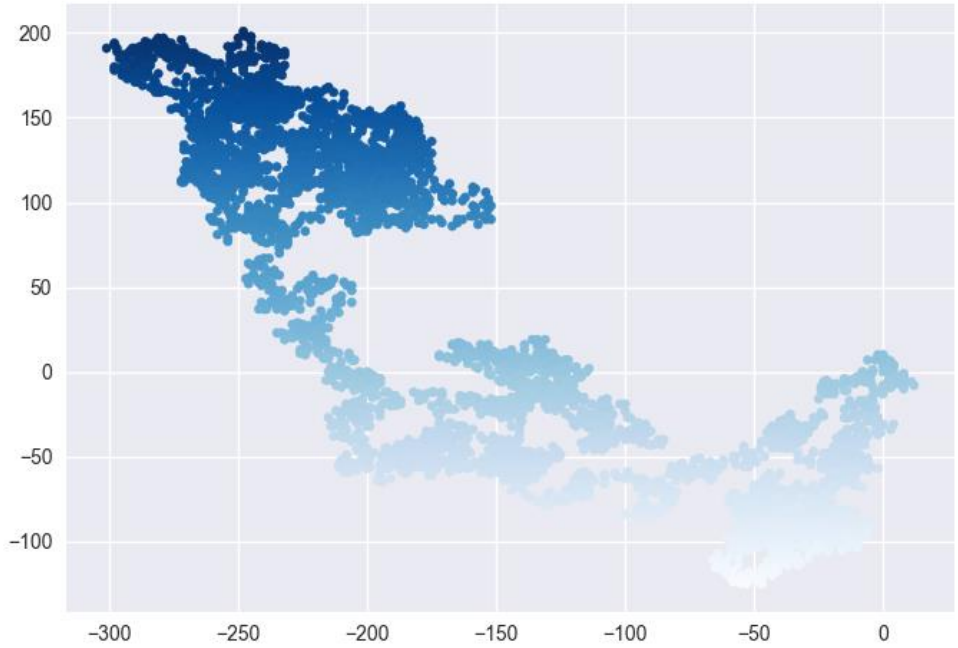
```
rw = RandomWalk()
rw.fill_walk()

plt.style.use('seaborn')

fig, ax = plt.subplots()

ax.scatter(rw.x_values, rw.y_values,
        c=rw.y_values, cmap=plt.cm.Blues, s=20)

plt.show()
```

- Цветовые карты используются в визуализациях для выделения закономерностей в данных.
- Например, малые значения можно выделить одним цветом, а большие – другим.
- Модуль *pyplot* включает набор встроенных цветовых карт:

  'magma', 'inferno', 'plasma', 'viridis', 'cividis', 'twilight', 'twilight_shifted', 'turbo', 'Blues', 'BrBG', 'BuGn', 'BuPu',..

# ВИЗУАЛИЗАЦИЯ ДАННЫХ: ЦВЕТОВЫЕ КАРТЫ

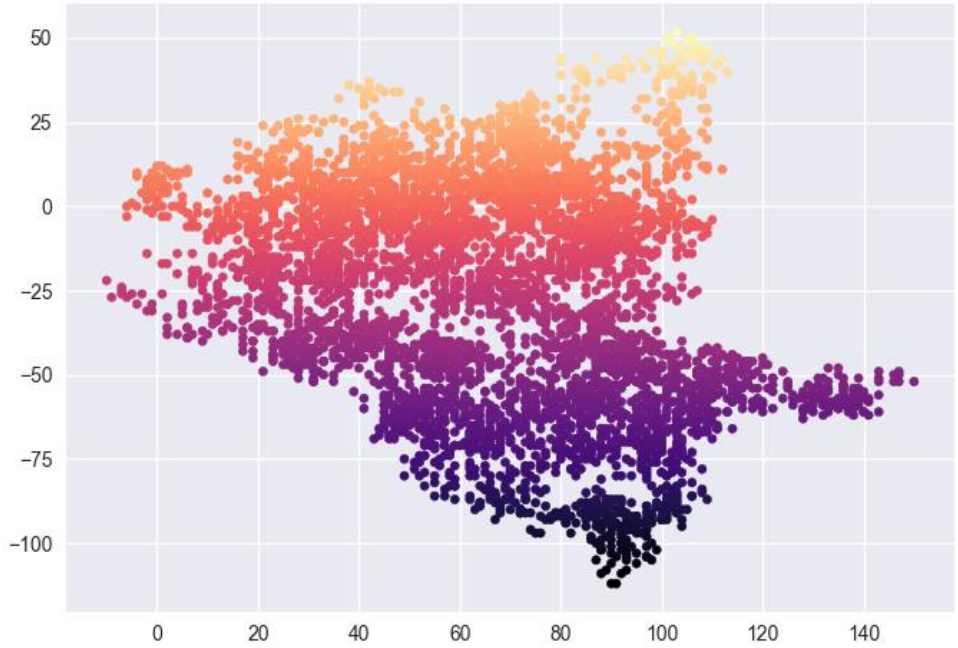
```
rw = RandomWalk()
rw.fill_walk()

plt.style.use('seaborn')

fig, ax = plt.subplots()

ax.scatter(rw.x_values, rw.y_values,
        c=rw.y_values, cmap=plt.cm.Blues, s=20)

plt.show()
```




# c – определяет значения,
# по которым будет происходить цветовая градация

# ВИЗУАЛИЗАЦИЯ ДАННЫХ: ЦВЕТОВЫЕ КАРТЫ

```
rw = RandomWalk()
rw.fill_walk()

plt.style.use('seaborn')

fig, ax = plt.subplots()

ax.scatter(rw.x_values, rw.y_values,
        c=rw.y_values, cmap=plt.cm.Blues, s=20)

plt.show()
```




# c – определяет значения,
# по которым будет происходить цветовая градация

# БЛАГОДАРЮ ЗА ВНИМАНИЕ!

Гаврилов Денис Андреевич, преподаватель кафедры СИ ФИТ НГУ