Задание 2b: Выделение областей в облаке точек с помощью Python и NumPy

Цель:

Научиться выделять (сегментировать) подмножества точек из облака по координатным ограничениям (bounding box), атрибутам (например, высоте), а также по произвольной геометрической области.

Задание:

1. Загрузка данных

Вариант A — сгенерировать искусственное облако:

```
import numpy as np

# Генерация облака точек: куб 100x100x100
points = np.random.uniform(0, 100, size=(100000, 3))
np.savetxt("synthetic_cloud.xyz", points)
```

Вариант B — использовать готовый файл `.xyz` или `.txt`:

```
points = np.loadtxt("your_cloud.xyz")  # Файл должен содержать X Y Z
```

---

2. Выделение области по координатным границам (Bounding Box)

Например, выделим подмножество точек, находящихся в кубе:

- `x ∈ [20, 50]`
- `y ∈ [30, 70]`
- `z ∈ [10, 40]`

```
def filter_by_bbox(points, xmin, xmax, ymin, ymax, zmin, zmax):
    mask = (
        (points[:, 0] >= xmin) & (points[:, 0] <= xmax) &
        (points[:, 1] >= ymin) & (points[:, 1] <= ymax) &
```

```
        (points[:, 2] >= zmin) & (points[:, 2] <= zmax)
    )
    return points[mask]

filtered = filter_by_bbox(points, 20, 50, 30, 70, 10, 40)
```

---

3. Выделение по высоте (например, только "высокие" точки)

Выделить все точки, где `Z > 80`:

```
high_points = points[points[:, 2] > 80]
```

---

4. Выделение по расстоянию до заданной точки

Оставить точки в радиусе `R` от точки `C = (50, 50, 50)`:

```
def filter_by_distance(points, center, radius):
    distances = np.linalg.norm(points - center, axis=1)
    return points[distances <= radius]

near_center = filter_by_distance(points, center=np.array([50, 50, 50]), radiu
```

---

5. Сохранение результатов

```
np.savetxt("bbox_filtered.xyz", filtered)
np.savetxt("high_points.xyz", high_points)
np.savetxt("near_center.xyz", near_center)
```

---

6. Визуализация (опционально, но желательно)

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def show_cloud(points, title="Point Cloud"):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(points[:, 0], points[:, 1], points[:, 2], s=0.3)
    ax.set_title(title)
    plt.show()
```

```
show_cloud(points, "Original Cloud")
show_cloud(filtered, "BBox Filtered")
show_cloud(high_points, "High Points")
```

Отчет:

- Python-скрипт (`.py`) или Jupyter ноутбук (`.ipynb`)
- Минимум 3 результата фильтрации (`.xyz`)
- Скриншоты визуализации (если делалась)
- Краткий текстовый отчёт:
  - Какие области выделены и почему
  - Количество точек до и после фильтрации
  - Выводы по сравнению результатов

Дополнительно (опционально):

- Реализуй фильтр по сложной 2D-области (например, внутри круга в плоскости XY)
- Используй pandas или scipy для более продвинутой фильтрации
