# **Задание 2.3 NumPy Обработка матрицы**

Вариант 19

Выполнить обработку элементов прямоугольной матрицы 𝐴, имеющей 𝑛 строк и 𝑚 столбцов. Добавить к матрице столбец чисел и вставить его под номером 𝑘.

Пользователь вводит 3 целых числа: n — количество строк в матрице, m — количество столбцов в матрице, k — индекс, после которого будет добавлен новый столбец. Дальше код создает прямоугольную матрицу случайных целых чисел от 1 до 1000 размером n строк на m столбцов. Затем код создает новый одномерный массив случайных целых чисел от 1 до 1000 длиной n, который будет добавлен в качестве нового столбца в матрицу. Далее код разбивает матрицу на две части: левую часть, состоящую из столбцов от 0 до k не включительно, и правую часть, состоящую из столбцов от k до m. Затем код использует функцию np.hstack() для объединения левой части, нового столбца и правой части в одну матрицу. Наконец, выводится исходная матрица и новая матрица с добавленным столбцом.

n, m, k = map(int, input().split())

Ввод: 6 7 5

matrix = np.random.randint(1, 1000, (n, m))

print(matrix)

array([[576, 814, 714, 685, 449, 208, 646],

[427, 744, 678, 416, 405, 940, 250],

[239, 563, 432, 422, 169, 457, 386],

[ 11, 789, 542, 35, 217, 843, 725],

[263, 971, 946, 288, 163, 761, 167],

[996, 455, 867, 501, 826, 456, 540]])

new = np.random.randint(1, 1000, n)

print(new)

array([163, 882, 48, 705, 236, 424])

left\_part = matrix[:, :k]

right\_part = matrix[:, k:]

ma = np.hstack((left\_part, new.reshape(n, 1), right\_part))

print(ma)

array([[576, 814, 714, 685, 449, 163, 208, 646],

[427, 744, 678, 416, 405, 882, 940, 250],

[239, 563, 432, 422, 169, 48, 457, 386],

[ 11, 789, 542, 35, 217, 705, 843, 725],

[263, 971, 946, 288, 163, 236, 761, 167],

[996, 455, 867, 501, 826, 424, 456, 540]])

**1**

*Гипотеза*: Результаты ОГЭ лучше в лицеях и гимназиях по сравнению с ООШ и СОШ. В специализированных школах и школах-интернатах с углубленным изучением предметов результаты тоже выше, нежели в ООШ и СОШ.

Переменные it1, it2, it3, it4, it5, it6 – это средние значения оценок по каждому типу образовательного учреждения. Переменная labels содержит названия типов образовательных учреждений, а sizes - средние значения оценок для каждого типа. Параметр colors задает цвета для каждого сектора диаграммы, а параметр autopct задает формат вывода процентного соотношения для каждого сектора.

Круговая диаграмма(рисунок 27) позволяет быстро сравнить средние значения оценок в различных типах образовательных учреждений и понять, какой тип образовательного учреждения имеет лучшие результаты.

labels = ['Лицеи', 'Гимназии', 'СОШ', 'ООШ', 'СОШ с углубленным изучением отдельных предметов', 'Школа-интернат']

sizes = [it1.mean(), it2.mean(), it3.mean(), it4.mean(), it5.mean(), it6.mean()]

purplec = ['#f3e5f5', '#e1bee7', '#ce93d8', '#ba68c8', '#9c27b0', '#6a1b9a']

plt.figure(figsize=(10, 7))

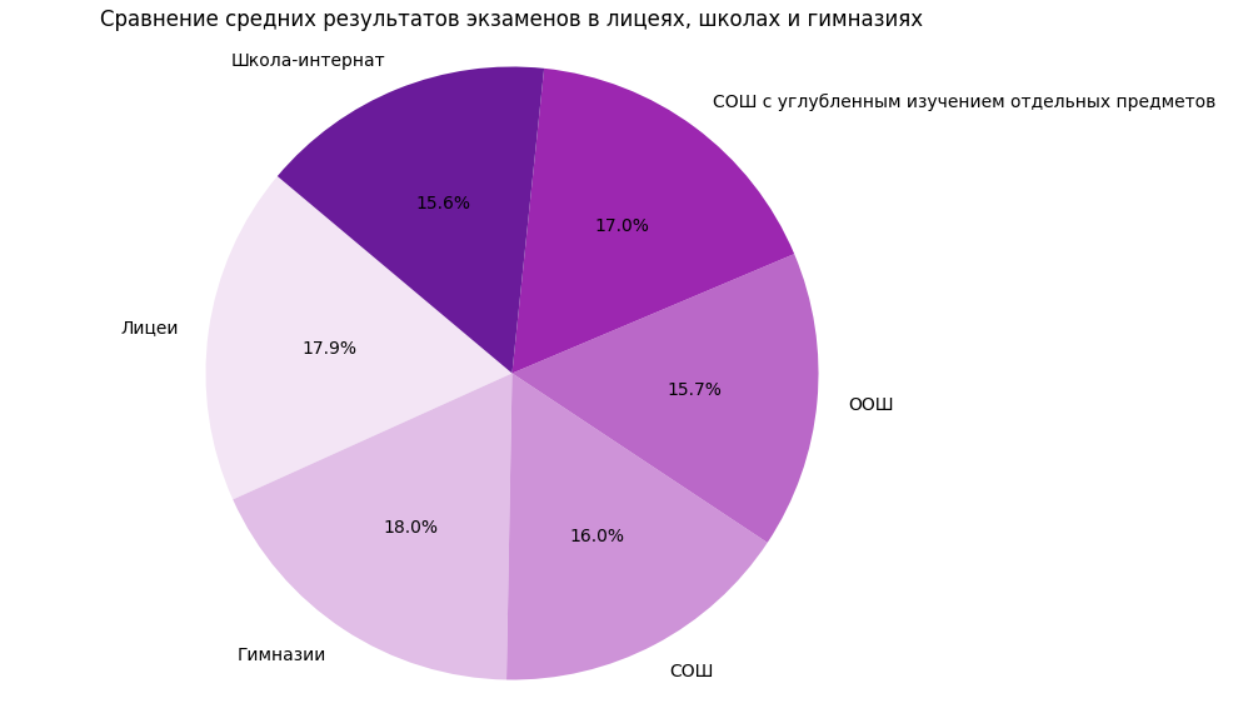
plt.pie(sizes, labels=labels, colors=purplec, autopct='%1.1f%%', startangle=140)

plt.axis('equal')

plt.title('Сравнение средних результатов экзаменов в лицеях, школах и гимназиях')

plt.show()

Диаграмма:



Итог: гипотеза о том, что результаты выпускников гимназий и лицеев выше, чем у выпускников обычных школ, подтвердилась. Кроме того, результаты выпускников средней общеобразовательной школы-интерната с углубленным изучением предметов оказались ниже, чем в СОШ и ООШ. Это, вероятно, связано с разницей в количестве учеников в каждом учебном заведении.

**2**

Гипотеза: Задания части В были структурированы с намерением увеличивать сложность по мере их продвижения.

itog - общая таблица, где есть и результаты ученика, и в каком районе, был получен результат. tot – список, в котором хранится общее кол-во плюсов по каждой позиции части В. total\_rows — общее количество строк в itog, percentages — список процентов, где каждый элемент x из tot (сумма плюсов по каждой позиции) делится на total\_rows и умножается на 100. Построение линейного графика для отображения процента выполнения по каждой позиции, с подписями осей и сеткой для наглядности.

total\_rows = len(itog)

percentages = [(x / total\_rows) \* 100 for x in tot]

plt.figure(figsize=(10, 6))

plt.plot(range(1, len(percentages)+1), percentages, color='mediumorchid')

plt.title('Процент выполнения по позициям')

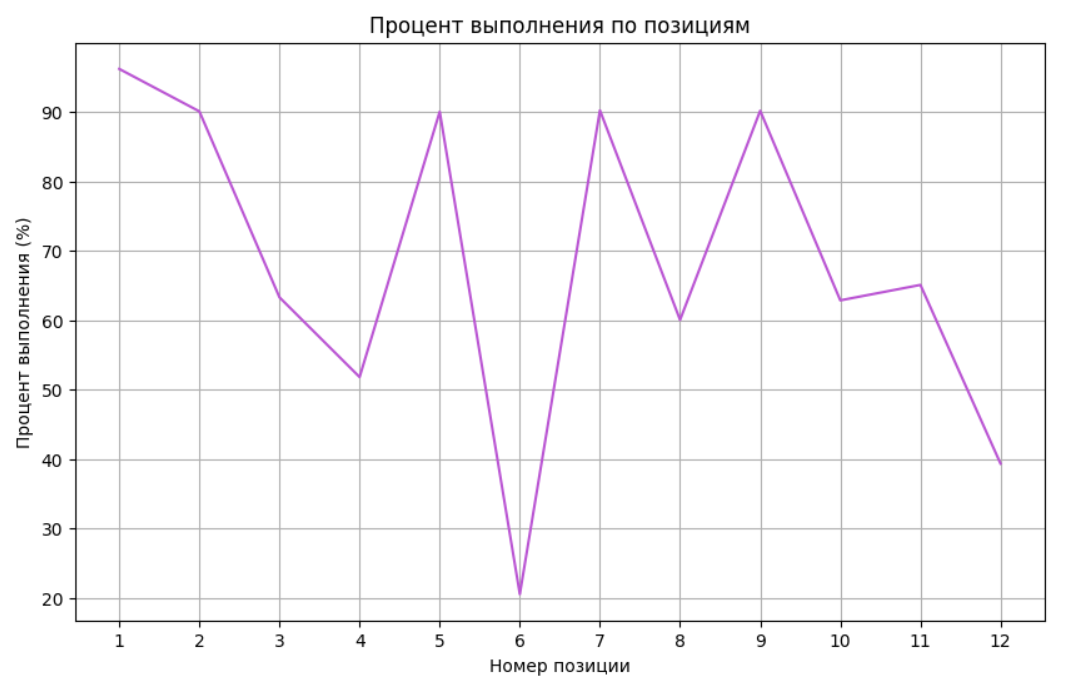
plt.xlabel('Номер позиции')

plt.ylabel('Процент выполнения (%)')

plt.xticks(range(1, 13))

plt.grid(True)

plt.show()



Однако, анализ данных показывает, что задания не всегда были последовательно расположены по сложности, хотя общая тенденция возрастания сложности первых заданий сохраняется (с 1 по 4). Задания 1, 2, 5, 7 и 9 оказались самыми простыми для участников. Это говорит о том, что данные задания были понятны и доступны для большинства. Задание 3, 4, 8, 10 и 11 оказались немного сложнее.

Задания 6 и 12 стали одними из самых сложных. Выводы о сложности: Самыми лёгкими были задания 1, 2, 5, 7 и 9, которые выполнило наибольшее количество участников. Самыми сложными оказались задания 6 и 12, которые выполнило наименьшее количество человек.