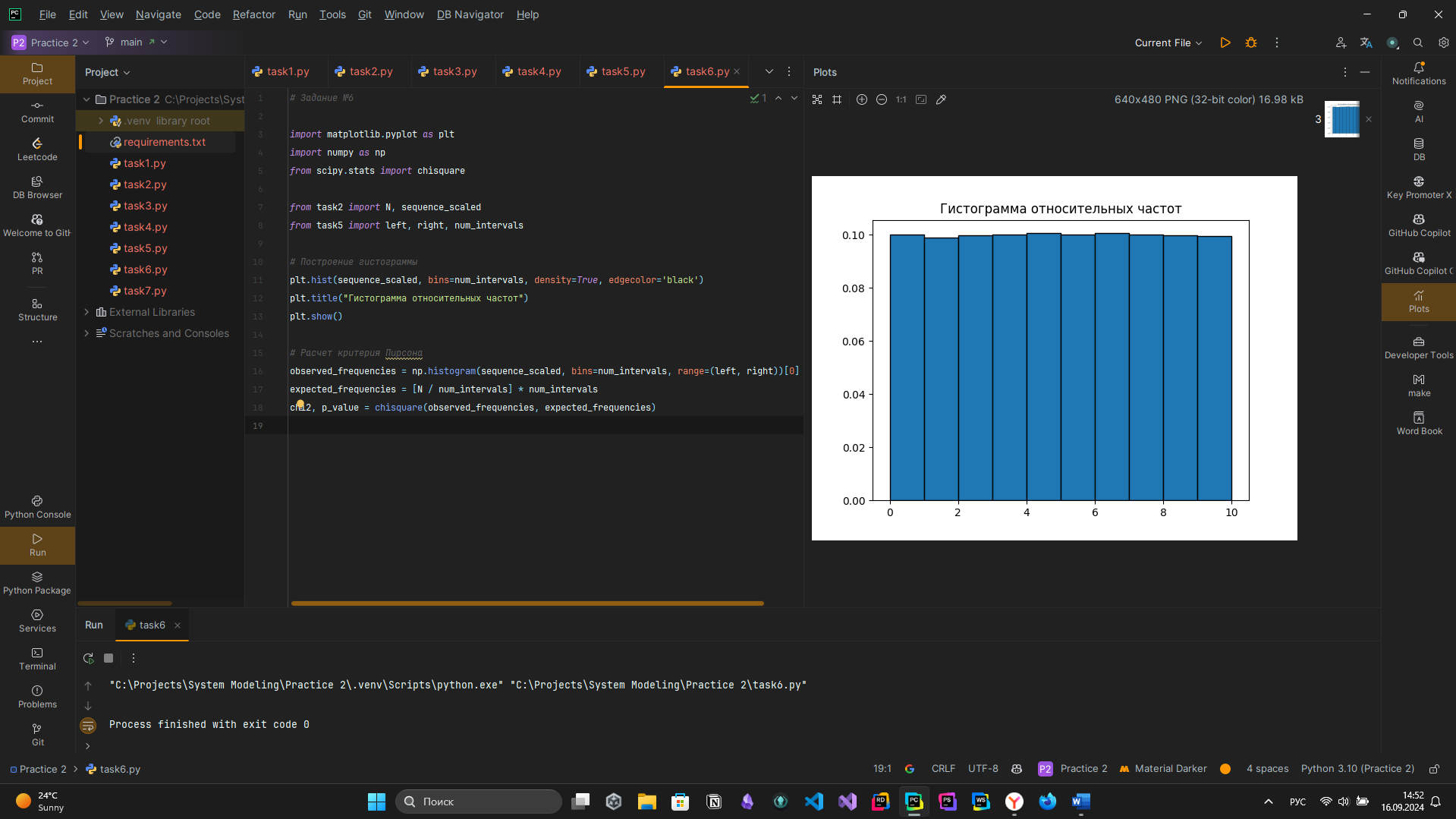
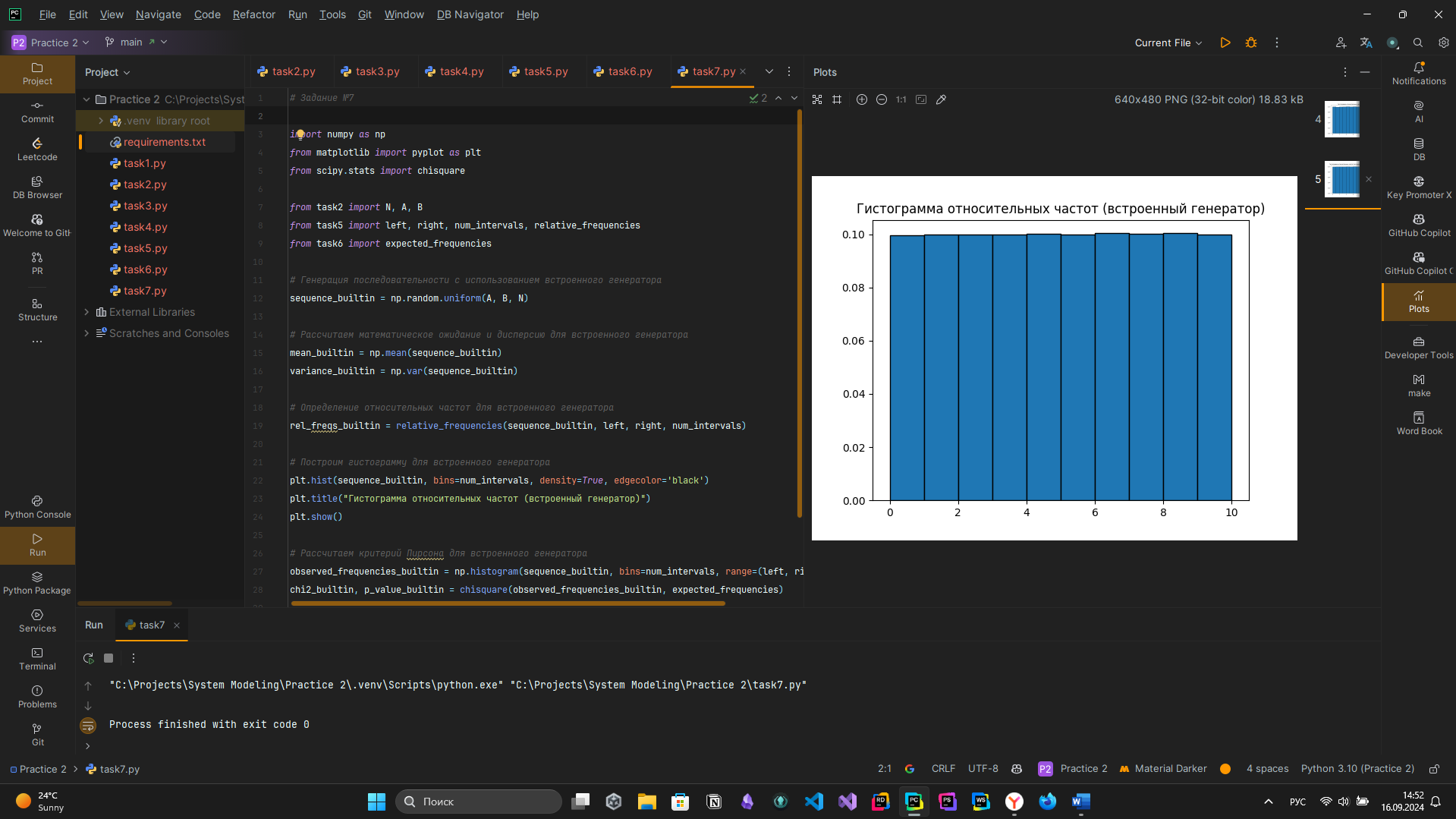
Практика 2

БСБО-09-23

Шутов Кирилл Сергеевич

Пример работы





Листинг кода

|  |
| --- |
| *# Задание №1* ***def*** *multiplicative\_rng(a, b, m, X0, N):  sequence = []  X = X0* ***for*** *\_* ***in*** *range(N):  X = (a \* X + b) % m  sequence.append(X / m) # нормализуем результат в интервале [0, 1]* ***return*** *sequence* |
| *# Задание №2* **from** task1 **import** multiplicative\_rng  *# Параметры* a = 22695477 b = 1 m = 2 \*\* 32 X0 = 1 N = 10 \*\* 6  *# Генерация последовательности* sequence = multiplicative\_rng(a, b, m, X0, N)  *# Преобразование в интервал [A, B]* A = 0 B = 10 sequence\_scaled = [A + (B - A) \* x **for** x **in** sequence] |
| *# Задание №3* **import** numpy **as** np  **from** task2 **import** sequence\_scaled **from** task2 **import** A, B  *# Рассчитываем математическое ожидание и дисперсию* mean\_empirical = np.mean(sequence\_scaled) variance\_empirical = np.var(sequence\_scaled)  *# Теоретические значения для равномерного распределения на [A, B]* mean\_theoretical = (A + B) / 2 variance\_theoretical = (B - A) \*\* 2 / 12  (mean\_empirical, variance\_empirical), (mean\_theoretical, variance\_theoretical) |
| *# Задание №4* **from** task2 **import** sequence   **def** find\_period(sequence):  seen = {}  **for** i, num **in** enumerate(sequence):  **if** num **in** seen:  **return** i - seen[num]  seen[num] = i  **return None** *# Определение периода* period = find\_period(sequence) |
| *# Задание №5* **from** task2 **import** sequence\_scaled   **def** relative\_frequencies(sequence, left, right, num\_intervals):  interval\_width = (right - left) / num\_intervals  intervals = [0] \* num\_intervals   **for** num **in** sequence:  **if** left <= num < right:  idx = int((num - left) / interval\_width)  intervals[idx] += 1   total = len(sequence)  relative\_freqs = [count / total **for** count **in** intervals]  **return** relative\_freqs   *# Пример вызова функции* left, right, num\_intervals = 0, 10, 10 rel\_freqs = relative\_frequencies(sequence\_scaled, left, right, num\_intervals) |
| *# Задание №6* **import** matplotlib.pyplot **as** plt **import** numpy **as** np **from** scipy.stats **import** chisquare  **from** task2 **import** N, sequence\_scaled **from** task5 **import** left, right, num\_intervals  *# Построение гистограммы* plt.hist(sequence\_scaled, bins=num\_intervals, density=**True**, edgecolor=**'black'**) plt.title(**"Гистограмма относительных частот"**) plt.show()  *# Расчет критерия Пирсона* observed\_frequencies = np.histogram(sequence\_scaled, bins=num\_intervals, range=(left, right))[0] expected\_frequencies = [N / num\_intervals] \* num\_intervals chi2, p\_value = chisquare(observed\_frequencies, expected\_frequencies) |
| *# Задание №7* **import** numpy **as** np **from** matplotlib **import** pyplot **as** plt **from** scipy.stats **import** chisquare  **from** task2 **import** N, A, B **from** task5 **import** left, right, num\_intervals, relative\_frequencies **from** task6 **import** expected\_frequencies  *# Генерация последовательности с использованием встроенного генератора* sequence\_builtin = np.random.uniform(A, B, N)  *# Рассчитаем математическое ожидание и дисперсию для встроенного генератора* mean\_builtin = np.mean(sequence\_builtin) variance\_builtin = np.var(sequence\_builtin)  *# Определение относительных частот для встроенного генератора* rel\_freqs\_builtin = relative\_frequencies(sequence\_builtin, left, right, num\_intervals)  *# Построим гистограмму для встроенного генератора* plt.hist(sequence\_builtin, bins=num\_intervals, density=**True**, edgecolor=**'black'**) plt.title(**"Гистограмма относительных частот (встроенный генератор)"**) plt.show()  *# Рассчитаем критерий Пирсона для встроенного генератора* observed\_frequencies\_builtin = np.histogram(sequence\_builtin, bins=num\_intervals, range=(left, right))[0] chi2\_builtin, p\_value\_builtin = chisquare(observed\_frequencies\_builtin, expected\_frequencies) |