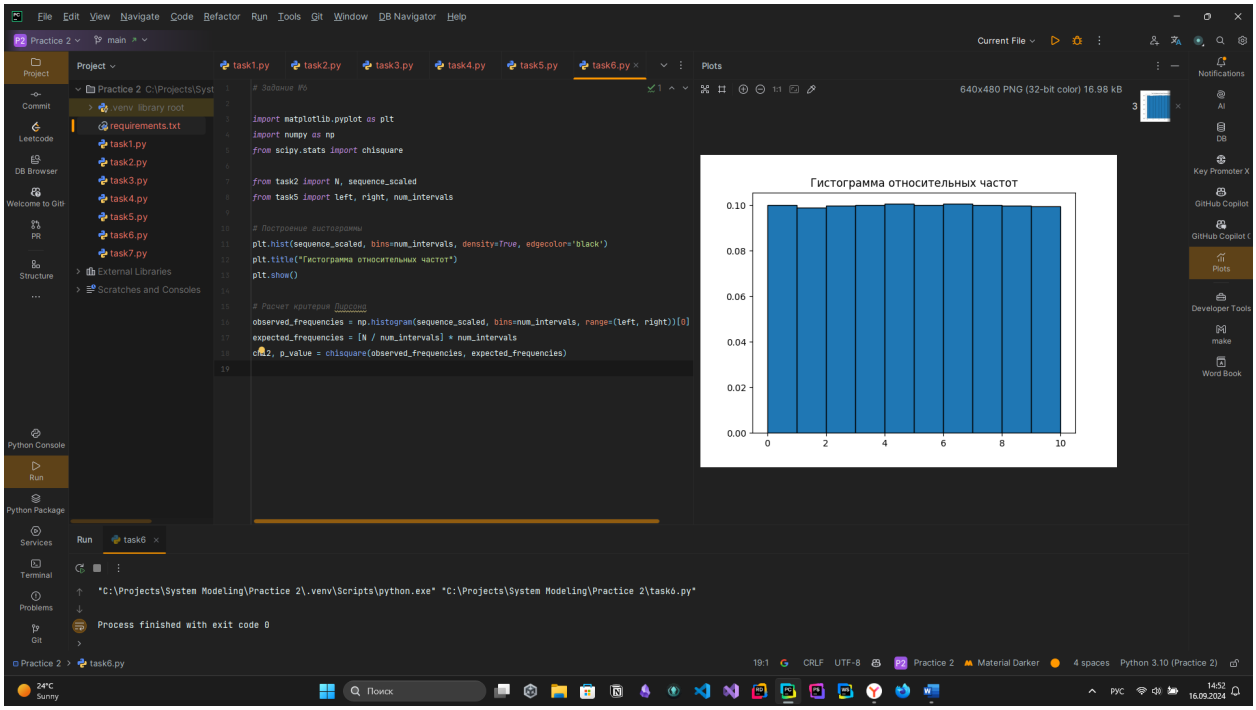
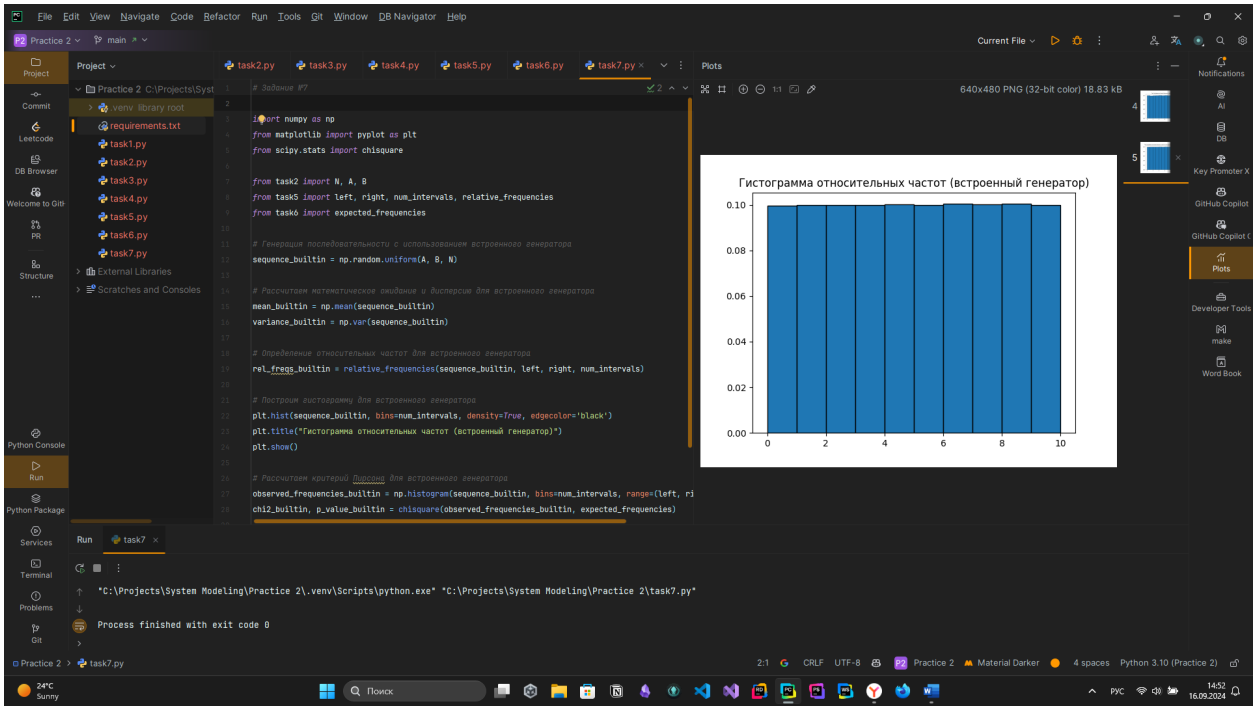
Практика 2

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)
```