Цель работы.

Обработка набора данных средствами Python.

Задание.

1. С помощью примера кода сгенерировать CSV-файл с данными в 5 столбцов, количество записей 10^6.

2. Средствами Python прочитать содержимое CSV-файла.

3. Найти среднее значение по колонке 2.

4. Построить график значений для колонок 3 и 4 в зависимости от данных времени из колонки 1.

5. Для колонки 5 построить график медиан кортежей по 10^5 записей. По оси X откладываем номер кортежа.

Параметры для генерации функций приведены в таблице 1.

Таблица 1

Вариант	period	v1	v2	v3
1	100000	3000	1000	200.0
2	130000	2000	500	170.0
3	170000	3000	800	150.0
4	200000	4000	1100	130.0
5	110000	5000	1200	220.0
6	220000	2500	1000	210.0
7	400000	3000	1300	190.0

Теоретические положения.

В качестве основы использован пример кода из задания на языке программирования Python.

При выполнении задания используется типовой подход к моделированию и обработке временных рядов. Исходные данные формируются в виде CSV-файла, содержащего столбец времени и четыре вычисляемые функции. Параметры генерации (period, v1, v2, v3) выбираются согласно таблице 1 для соответствующего варианта.

Функции представляют собой комбинацию детерминированной составляющей и случайного шума. В частности, для func1 используется равномерная случайная добавка к базовому уровню; func2 задаётся синусоидальной зависимостью от времени с шумовой компонентой; func3 формируется как сумма func1 и func2; func4 дополнительно включает аргумент времени, поэтому имеет выраженный тренд роста

по оси времени. Такая структура данных позволяет проверить чтение/запись CSV, вычисление статистик и построение графиков.

При обработке данных выполняются: чтение файла построчно, расчёт среднего значения по второй колонке (func1), построение графиков func2 и func3 от времени, а также расчёт медиан func4 по блокам объёмом 10^5 записей. Медиана используется как устойчивая характеристика центральной тенденции при наличии случайной составляющей.

Ход работы:

Работа выполнена по варианту 6. Для удобства выполнения и проверки задания программа разбита на несколько отдельных скриптов: генерация исходных данных в CSV, вычисление среднего значения, построение графика для колонок 3 и 4 и построение графика медиан для колонки 5. Такое разбиение позволяет запускать этапы независимо и проще контролировать результат каждого шага.

1. Формирование исходного набора данных (CSV)

На первом этапе был сформирован файл test.csv, содержащий 5 столбцов и 10^6 строк. Генерация выполнялась скриптом Генерация CSV.py. В качестве времени используется целочисленная последовательность, начинающаяся с текущей метки времени UNIX (timestamp), что обеспечивает монотонный рост аргумента времени и удобную привязку данных к оси X при построении графиков. Инициализация времени и запуск генерации выполняются следующим образом:

```
1. start = int(datetime.datetime.now().timestamp())
2. generate_csv('test.csv', start, 10 ** 6, period, v1, v2, v3)
```

Параметры функций взяты из таблицы вариантов (вариант 6) и заданы явно в основной части скрипта:

```
1. period = 220000
2. v1 = 2500
3. v2 = 1000
4. v3 = 210.0
```

Структура CSV фиксирована: time, func1, func2, func3, func4. Запись выполняется через csv.DictWriter, что позволяет формировать строки по именам столбцов и

избегать ошибок с порядком колонок. В цикле по i формируются значения функций и добавляются в файл:

```
fieldnames = ['time', 'func1', 'func2', 'func3', 'func4']
...
writer.writerow({
    'time': start + i,
    'func1': func_rand(start + i),
    'func2': func_sin(start + i , period, v1, v2, v3),
    'func3': func_three(start + i , period, v1, v2, v3),
    'func4': func_four(start + i, period, v1, v2, v3)
})
```

Содержательно столбцы формируются так:

- func1 — случайная составляющая (базовый уровень + равномерная добавка), что задаёт разброс значений:

```
                res = 2500 + random.random() * 1000
```

- func2 — синусоидальная зависимость от времени с шумом. В формуле присутствуют параметры варианта (period, v1, v2, v3), где v1 задаёт уровень, v2 — амплитуду синусоидальной части, v3 — уровень случайной добавки:

```
            s = math.sin((2 * np.pi / period) * time)
            res = v1 + s * v2 + random.random() * v3
```

По итогу этапа сформирован файл test.csv, который далее используется во всех последующих пунктах задания. Блок схема алгоритма генерации представлена на Рисунке 1.

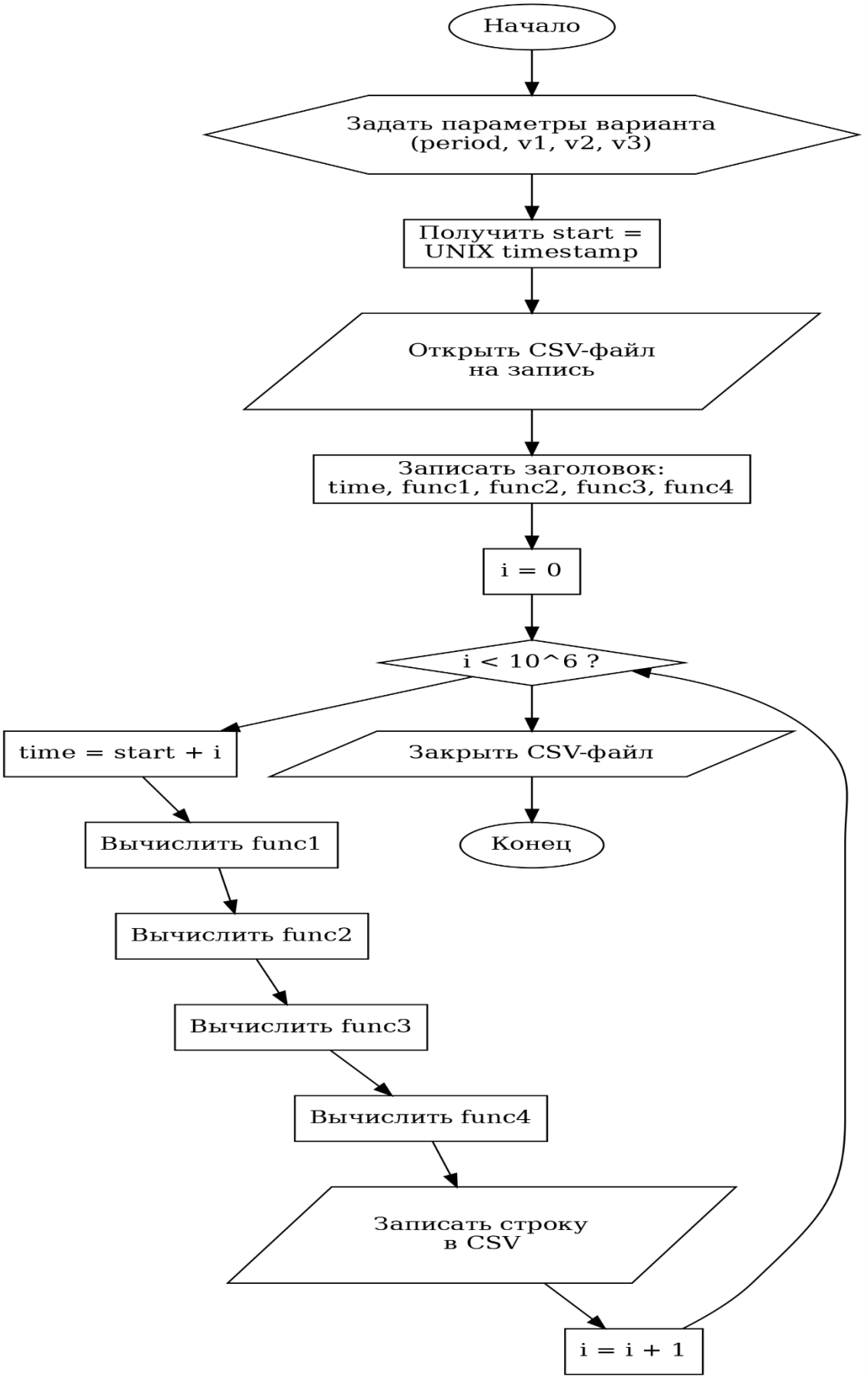

Рисунок 1. Блок-схема алгоритма генерации CSV-файла (формирование 10^6 записей со столбцами time, func1–func4).

2. Чтение CSV и вычисление среднего значения по колонке 2

Далее выполнено чтение сформированного файла средствами Python и вычислено среднее значение по **второй колонке**. В созданном CSV после time второй колонкой является func1, поэтому среднее считается по полю func1. Для этого используется скрипт Среднее по колонке 2.py.

Чтение организовано последовательно (построчно) через csv.DictReader, что удобно при большом объёме данных и не требует загрузки всего файла в память. В цикле суммируются значения и считается количество строк:

```
1. for row in reader:
2.     total += float(row['func1'])
3.     count += 1
```

После прохода по файлу среднее вычисляется как отношение суммы к числу элементов:

```
1. return total / count
```

Результат выводится в консоль, что позволяет быстро проверить корректность чтения файла и получить численное значение среднего:

```
1. print("Среднее по колонке func1:", avg)
```

3. Построение графика значений для колонок 3 и 4 по времени

Следующий пункт задания — построить графики значений колонок 3 и 4 в зависимости от данных времени (колонка 1). В CSV колонке 3 соответствует func2, колонке 4 — func3. Построение выполнено скриптом График значений для колонок 3 и 4.py.

Сначала выполняется чтение файла и формирование массивов значений по времени и двум функциям:

```
1. times.append(float(row['time']))
2. col3.append(float(row['func2']))
3. col4.append(float(row['func3']))
```

Затем строятся две кривые на одном графике, для визуального сравнения поведения func2 и func3 на одном интервале времени:

```
1. plt.plot(times, col3, label='func2 (колонка 3)')
2. plt.plot(times, col4, label='func3 (колонка 4)')
```

Для корректного оформления добавлены подписи осей, заголовок, легенда и сетка. График сохраняется в PNG с разрешением 300 dpi:

```
1. plt.savefig("График значений для колонок 3 и 4.png", dpi=300)
```

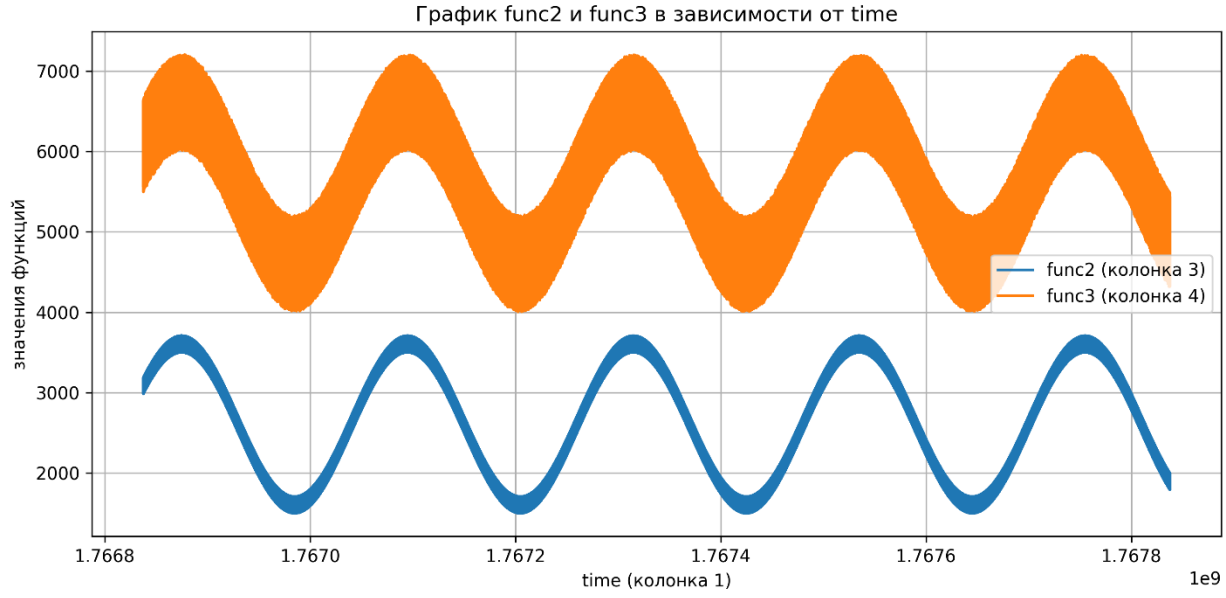

Рисунок 2 — График функций func2 и func3 в зависимости от времени.

4. Построение графика медиан для колонки 5 по блокам 10^5

Заключительный пункт задания — для колонки 5 (то есть func4) построить график медиан по кортежам (блокам) объёмом 10^5 записей, откладывая по оси X номер кортежа. Для этого используется скрипт Строим график медиан.py. Размер блока задан как 100 000, а число блоков — 10, что соответствует общему объёму 10^6 записей:

```
1. block_size = 100_000
2. blocks = [[] for _ in range(10)]
```

При чтении файла значения func4 распределяются по блокам по индексу строки:

```
1. value = float(row['func4'])
2. block_index = index // block_size
3. if block_index < 10:
4.     blocks[block_index].append(value)
5. index += 1
```

После заполнения блоков для каждого рассчитывается медиана стандартной функцией statistics.median, затем строится график медиан по номерам блоков 1…10:

```
1. medians = [statistics.median(block) for block in blocks]
2. ...
3. plt.plot(range(1, 11), medians, marker='o')
4.
```

График оформляется подписями и сохраняется в PNG:

plt.savefig ("График медиан func4.png", dpi=300)

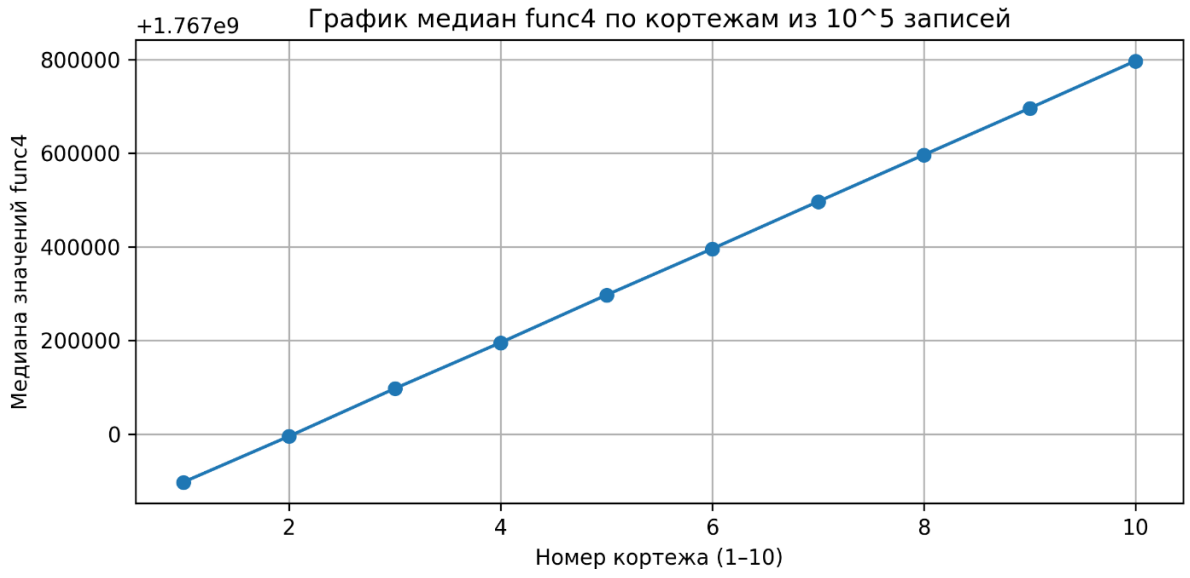

Рисунок 3 — График медиан func4 по блокам из 10^5 записей.

В результате выполнены все пункты задания: сгенерирован CSV-файл заданного формата и объёма, выполнено чтение данных, рассчитано среднее по второй колонке (func1), построен график зависимостей для колонок 3 и 4 от времени, а также построен график медиан для колонки 5 по блокам 10^5 записей. Полученные изображения графиков сохранены в файлы PNG.

Вывод:

В рамках выполнения ИДЗ №1 (вариант 6) были реализованы и последовательно выполнены все пункты задания по обработке набора данных средствами Python. На основе заданных параметров сформирован CSV-файл test.csv, содержащий 10^6 записей и 5 столбцов (временная шкала и четыре функции). После генерации выполнено чтение файла и получено среднее значение по второй колонке (func1), что подтверждает корректность структуры CSV и пригодность данных для дальнейшей обработки.

Для анализа поведения функций построены графики значений func2 и func3 в зависимости от времени, что позволяет наглядно сравнить характер изменения данных в третьей и четвёртой колонках. Также для пятой колонки (func4) выполнено разбиение на 10 блоков по 10^5 значений и построен график медиан по каждому блоку. Такой подход даёт обобщённую оценку центрального уровня значений на больших объёмах данных и позволяет сравнить блоки между собой. Все графики сохранены в формате PNG и использованы в отчёте как иллюстрации результатов. Код модулей представлен в приложениях к отчету.

Таким образом, поставленная цель работы — обработка набора данных средствами Python — достигнута, а требуемые вычисления и визуализация выполнены в полном объёме.

Приложения.

Код исполняемого файла Генерация CSV.py

```
import datetime
import csv
import math
import random
import numpy as np


def func_rand(time:float):
    # """Генерирует случайное значение для столбца func1"""
    res = 2500 + random.random() * 1000
    return res

def func_sin(time:float, period:int = 220000, v1:int = 2500, v2:float = 1000, v3:float = 210.0):
    # """Синусоидальная функция с шумом для столбца func2"""
    s = math.sin((2 * np.pi / period) * time)
    res = v1 + s * v2 + random.random() * v3
    return res

def func_three(time:float, period:int = 220000, v1:int = 2500, v2:float = 1000, v3:float = 210.0):
    # """Комбинация func_sin и func_rand для столбца func3"""
    res = func_sin(time, period, v1, v2, v3) + func_rand(time)
    return res

def func_four(time:float, period:int = 220000, v1:int = 2500, v2:float = 1000, v3:float = 210.0):
    # """time + func_three + случайный шум — данные для столбца func4"""
    res = time + func_three(time, period, v1, v2, v3) + func_rand(time)
    return res


def generate_csv(filename:str, start:int, count:int, period:int, v1:int, v2:float, v3:float):
    # """Генерирует CSV-файл с пятью функциями по заданным параметрам"""
    with open(filename, 'w', newline='') as csvfile:
        fieldnames = ['time', 'func1', 'func2', 'func3', 'func4']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        writer.writeheader()
        for i in range(count):
            writer.writerow({
                'time': start + i,
                'func1': func_rand(start + i),
                'func2': func_sin(start + i , period, v1, v2, v3),
                'func3': func_three(start + i , period, v1, v2, v3),
                'func4': func_four(start + i, period, v1, v2, v3)
            })


if __name__ == '__main__':
    start = int(datetime.datetime.now().timestamp())

    # параметры варианта 6
    period = 220000
    v1 = 2500
    v2 = 1000
    v3 = 210.0

    generate_csv('test.csv', start, 10 ** 6, period, v1, v2, v3)
    print("CSV-файл 'test.csv' успешно сгенерирован.")
```

Код исполняемого файла График значений для колонок 3 и 4.py

```
import csv
import matplotlib.pyplot as plt

def plot_columns(filename: str):
    times = []
    col3 = []  # func2
    col4 = []  # func3

    with open(filename, 'r') as f:
        reader = csv.DictReader(f)
        for row in reader:
            times.append(float(row['time']))
            col3.append(float(row['func2']))
            col4.append(float(row['func3']))

    # Строим графики
    plt.figure(figsize=(10, 5))
    plt.plot(times, col3, label='func2 (колонка 3)')
    plt.plot(times, col4, label='func3 (колонка 4)')

    plt.xlabel('time (колонка 1)')
    plt.ylabel('значения функций')
    plt.title('График func2 и func3 в зависимости от time')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.tight_layout()
    plt.savefig("График значений для колонок 3 и 4.png", dpi=300)
    plt.show()


if __name__ == '__main__':
    plot_columns('test.csv')
    print("График успешно сохранён ")
```

Код исполняемого файла Среднее по колонке 2.py

```
import csv

def read_and_avg(filename: str):
    total = 0.0
    count = 0

    with open(filename, 'r') as f:
        reader = csv.DictReader(f)

        for row in reader:
            total += float(row['func1'])
            count += 1

    return total / count


if __name__ == '__main__':
    avg = read_and_avg('test.csv')
    print("Среднее по колонке func1:", avg)

```

Код исполняемого файла Строим график медиан.py

```
import csv
import statistics
import matplotlib.pyplot as plt

def plot_medians(filename: str):
    block_size = 100_000        # 10^5 записей
    blocks = [[] for _ in range(10)]  # 10 кортежей

    with open(filename, 'r') as f:
        reader = csv.DictReader(f)

        index = 0
        for row in reader:
            value = float(row['func4'])
            block_index = index // block_size
            if block_index < 10:
                blocks[block_index].append(value)
            index += 1

    # вычисляем медианы каждого блока
    medians = [statistics.median(block) for block in blocks]

    # строим график
    plt.figure(figsize=(8, 4))
    plt.plot(range(1, 11), medians, marker='o')
    plt.xlabel("Номер кортежа (1–10)")
    plt.ylabel("Медиана значений func4")
    plt.title("График медиан func4 по кортежам из 10^5 записей")
    plt.grid(True)
    plt.tight_layout()
    plt.tight_layout()
    plt.savefig("График медиан func4.png", dpi=300)
    plt.show()


if __name__ == '__main__':
    plot_medians('test.csv')
    print("График медиан func4 успешно построен.")
```