

Тестовое задание
по теме: «Анализ макроэкономических данных»

Выполнил: Алексеенко Андрей Евгеньевич

2026 г.

СОДЕРЖАНИЕ

ЗАДАНИЕ	3
ТРЕБОВАНИЯ К ПРОГРАММЕ	3
ССЫЛКА НА РЕПОЗИТОРИЙ	3
КОД ПРОГРАММЫ.....	4
main.py	4
analysis.py	5
read.py	6
test_analysis.py	7
test_main.py	9
sample_data.csv	10
РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ.....	10
РЕЗУЛЬТАТ ТЕСТИРОВАНИЯ	11
ВЫВОД	11

ЗАДАНИЕ

Написать скрипт для обработки csv-файла. Скрипт читает файлы с экономическими данными по странам (см. примеры ниже) и формирует отчеты. Нужно сформировать всего один отчет, в котором будет среднее ВВП по странам (см. пример ниже). Отчёт включает в себя список стран и среднее ВВП (среднее арифметическое по колонке gdp), отчёты сортируются по убыванию ВВП. Название файлов (может быть несколько) и название отчета передается в виде параметров `--files` и `--report` (в нашем случае это `average-gdp`). Отчёт формируется по всем переданным файлам, а не по каждому отдельно.

ТРЕБОВАНИЯ К ПРОГРАММЕ

- можно передать пути к файлам через `--files`
- можно указать название отчета через `--report` (`average-gdp`)
- в консоль (не в файл) выводится отчёт в виде таблицы
- для всего кроме тестов и вывода в консоль, можно использовать только стандартную библиотеку, например:
 - для обработки параметров скрипта нельзя использовать `click`, можно `argparse`
 - для чтения файлов нельзя использовать `pandas`, но можно `csv`
- в архитектуру заложена возможность добавления новых отчётов, как раз через параметр `--report`, например, если захотим посмотреть как изменилась безработица за несколько лет или размер популяции по континентам, то отчет можно будет быстро добавить
- код покрыт тестами, написанными на `pytest`
- для тестов можно использовать любые дополнительные библиотеки
- код соответствует:
 - общепринятым стандартам написания проектов на `python`
 - общепринятому стилю

ССЫЛКА НА РЕПОЗИТОРИЙ

<https://github.com/andrey5e/task>

КОД ПРОГРАММЫ

main.py:

```
import argparse
import sys
from tabulate import tabulate
from read import CSVDataReader
from analysis import get_analyzer

# Разбор аргументов командной строки
def parse_arguments():
    parser = argparse.ArgumentParser(
        description='Анализ макроэкономических данных',
        formatter_class=argparse.RawDescriptionHelpFormatter,
    )

    parser.add_argument(
        '--files',
        nargs='+',
        required=True,
    )

    parser.add_argument(
        '--report',
        required=True,
        choices=['average-gdp'],
    )

    return parser.parse_args()

def main():
    # Разбираем аргументы
    args = parse_arguments()

    # Получаем анализатор
    analyzer = get_analyzer(args.report)
    if not analyzer:
        print(f"Ошибка: Неизвестный тип отчёта '{args.report}'")
        print(f"Доступные отчеты: average-gdp")
        sys.exit(1)

    # Чтение данных
    print(f"Чтение данных из {len(args.files)} файла(ов)")
    reader = CSVDataReader()
    data = reader.read_multiple_files(args.files)

    if not data:
        print("Нет данных в указанных файлах")
        sys.exit(1)
    print(f"Загружено {len(data)} строк данных")

    # Анализ данных
    print(f"Генерация отчета '{args.report}'...")
```

```

results = analyzer.analyze(data)

if not results:
    print("Нет результатов для отображения")
    sys.exit(0)

# Подготовка данных для таблиц
table_data = []
for result in results:
    if args.report == 'average-gdp':
        table_data.append([result['country'], result['average_gdp']])

# Вывод результатов
print("\nРезультаты:\n")
print(tabulate(
    table_data,
    headers=analyzer.get_headers(),
    tablefmt='grid',
    floatfmt='.2f',
    numalign='right'
))

print(f"\nОтчет сформирован. Найдены данные для {len(results)} стран.")

if __name__ == '__main__':
    main()

```

analysis.py:

```

from collections import defaultdict
from typing import List, Dict, Any, Optional
from abc import ABC, abstractmethod

# Класс для анализаторов
class BaseAnalyzer(ABC):
    @abstractmethod
    # Анализ данных
    def analyze(self, data: List[Dict[str, Any]]) -> List[Dict[str, Any]]:
        pass

    @abstractmethod
    # Возврат названия отчета
    def get_report_name(self) -> str:
        pass

    # Возврат заголовков таблицы для вывода
    def get_headers(self) -> List[str]:
        return ["Страна", "Значение"]

class GPDAverageAnalyzer(BaseAnalyzer):
    # Анализатор для расчета среднего ВВП по странам
    def analyze(self, data: List[Dict[str, Any]]) -> List[Dict[str, Any]]:
        if not data:

```

```

        return []

    # Группировка значений ВВП по странам
    country_gdp = defaultdict(list)

    for row in data:
        country = row.get('country', '').strip()
        if not country: # Пропуск строк без страны
            continue

        try:
            gdp = float(row.get('gdp', 0))
            if gdp > 0:
                country_gdp[country].append(gdp)
        except (ValueError, TypeError):
            # Пропуск некорректных значений ВВП
            continue

    # Рассчёт средних значений
    results = []
    for country, gdps in country_gdp.items():
        if gdps:
            avg_gdp = sum(gdps) / len(gdps)
            results.append({
                'country': country,
                'average_gdp': round(avg_gdp, 2)
            })

    # Сортируем по среднему ВВП по убыванию
    return sorted(results, key=lambda x: x['average_gdp'], reverse=True)

def get_report_name(self) -> str:
    return "average-gdp"

def get_headers(self) -> List[str]:
    return ["Страна", "Средний ВВП"]

# Получение соответствующего анализатора для указанного отчета
def get_analyzer(report_name: str) -> Optional[BaseAnalyzer]:
    analyzers = {
        'average-gdp': GDPAverageAnalyzer(),
    }

    return analyzers.get(report_name)

```

read.py:

```

import csv
import os
from typing import List, Dict, Any

# Чтение CSV файлов с экономическими данными
class CSVDataReader:

```

```

# Чтение CSV файла
def read_file(self, file_path: str) -> List[Dict[str, Any]]:
    if not os.path.exists(file_path):
        raise FileNotFoundError(f"Файл не найден: {file_path}")
    data = []
    with open(file_path, 'r', encoding='utf-8') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            # Очистка данных (удаление лишних пробелов)
            cleaned_row = {}
            for key, value in row.items():
                if key is not None:
                    cleaned_key = key.strip()
                    cleaned_value = value.strip() if value else ''
                    cleaned_row[cleaned_key] = cleaned_value
            data.append(cleaned_row)

    return data

# Чтение нескольких CSV файлов и объединение их данных
def read_multiple_files(self, file_paths: List[str]) -> List[Dict[str, Any]]:
    all_data = []
    for file_path in file_paths:
        try:
            file_data = self.read_file(file_path)
            all_data.extend(file_data)
            print(f"Загружено {len(file_data)} строк из {file_path}")
        except (FileNotFoundError, csv.Error) as e:
            print(f"Не удалось прочитать {file_path}: {e}")

    return all_data

```

test_analysis.py:

```

import pytest
from analysis import GDPAverageAnalyzer, get_analyzer

class TestGDPAverageAnalyzer:
    @pytest.fixture
    # Создание экземпляра анализатора для тестирования
    def analyzer(self):
        return GDPAverageAnalyzer()

    # Тест с пустыми данными
    def test_empty_data(self, analyzer):
        assert analyzer.analyze([]) == []

    # Тест с одной страной и одним годом
    def test_single_country_single_year(self, analyzer):
        data = [{'country': 'USA', 'gdp': '25462'}]
        result = analyzer.analyze(data)
        assert len(result) == 1

```

```

    assert result[0]['country'] == 'USA'
    assert result[0]['average_gdp'] == 25462.0

# Тест с одной страной и несколькими годами
def test_single_country_multiple_years(self, analyzer):
    data = [
        {'country': 'USA', 'gdp': '25462'},
        {'country': 'USA', 'gdp': '23315'},
        {'country': 'USA', 'gdp': '22994'}
    ]
    result = analyzer.analyze(data)
    assert len(result) == 1
    expected_avg = (25462 + 23315 + 22994) / 3
    assert result[0]['average_gdp'] == round(expected_avg, 2)

# Тест с несколькими странами
def test_multiple_countries(self, analyzer):
    data = [
        {'country': 'USA', 'gdp': '30000'},
        {'country': 'USA', 'gdp': '20000'},
        {'country': 'China', 'gdp': '25000'},
        {'country': 'Germany', 'gdp': '15000'}
    ]
    result = analyzer.analyze(data)
    assert len(result) == 3

    # Проверка сортировки по убыванию
    assert result[0]['country'] == 'USA'
    assert result[1]['country'] == 'China'
    assert result[2]['country'] == 'Germany'

    assert result[0]['average_gdp'] == 25000.0
    assert result[1]['average_gdp'] == 25000.0
    assert result[2]['average_gdp'] == 15000.0

# Тест с некорректными значениями ВВП
def test_invalid_gdp_values(self, analyzer):
    data = [
        {'country': 'USA', 'gdp': '25462'},
        {'country': 'USA', 'gdp': 'invalid'}, # Некорректное значение
        {'country': 'USA', 'gdp': ''}, # Пустое значение
        {'country': 'USA', 'gdp': '22994'},
        {'country': 'China', 'gdp': '-1000'}, # Отрицательное значение (пропуск)
        {'country': 'China', 'gdp': '17963'}
    ]
    result = analyzer.analyze(data)
    assert len(result) == 2

    # USA должно усреднить только валидные значения
    usa_result = next(r for r in result if r['country'] == 'USA')
    assert usa_result['average_gdp'] == (25462 + 22994) / 2

    # China должно учесть только положительное значение
    china_result = next(r for r in result if r['country'] == 'China')

```



```

    assert china_result['average_gdp'] == 17963.0

# Тест с отсутствующей страной
def test_missing_country(self, analyzer):
    data = [
        {'gdp': '1000'}, # Нет страны
        {'country': 'USA', 'gdp': '2000'},
        {'country': '', 'gdp': '3000'} # Пустая страна
    ]
    result = analyzer.analyze(data)
    assert len(result) == 1
    assert result[0]['country'] == 'USA'
    assert result[0]['average_gdp'] == 2000.0

# Тест функции получения анализатора
def test_get_analyzer():
    # Существующий анализатор
    analyzer = get_analyzer('average-gdp')
    assert analyzer is not None
    assert isinstance(analyzer, GDPAverageAnalyzer)

    # Несуществующий анализатор
    assert get_analyzer('invalid-report') is None

```

test_main.py:

```

import pytest
from unittest.mock import patch
from main import parse_arguments

# Тест разбора валидных аргументов
def test_parse_arguments_valid():
    with patch('sys.argv', ['main.py', '--files', 'data.csv', '--report', 'average-gdp']):
        args = parse_arguments()
        assert args.files == ['data.csv']
        assert args.report == 'average-gdp'

# Тест с несколькими файлами
def test_parse_arguments_multiple_files():
    with patch('sys.argv', ['main.py', '--files', 'file1.csv', 'file2.csv', '--report', 'average-gdp']):
        args = parse_arguments()
        assert args.files == ['file1.csv', 'file2.csv']
        assert args.report == 'average-gdp'

# Тест с отсутствующим обязательным аргументом --files
def test_parse_arguments_missing_files():
    with patch('sys.argv', ['main.py', '--report', 'average-gdp']):
        with pytest.raises(SystemExit):
            parse_arguments()

# Тест с отсутствующим обязательным аргументом --report
def test_parse_arguments_missing_report():

```

```

with patch('sys.argv', ['main.py', '--files', 'data.csv']):
    with pytest.raises(SystemExit):
        parse_arguments()

# Тест с неверным типом отчета
def test_parse_arguments_invalid_report():
    with patch('sys.argv', ['main.py', '--files', 'data.csv', '--report', 'invalid']):
        with pytest.raises(SystemExit):
            parse_arguments()

```

sample_data.csv:

```

country,year,gdp,gdp_growth,inflation,unemployment,population,continent
United States,2023,25462,2.1,3.4,3.7,339,North America
United States,2022,23315,2.1,8.0,3.6,338,North America
United States,2021,22994,5.9,4.7,5.3,337,North America
China,2023,17963,5.2,2.5,5.2,1425,Asia
China,2022,17734,3.0,2.0,5.6,1423,Asia
China,2021,17734,8.4,1.0,5.1,1420,Asia
Germany,2023,4086,-0.3,6.2,3.0,83,Europe
Germany,2022,4072,1.8,8.7,3.1,83,Europe
Germany,2021,4257,2.6,3.1,3.6,83,Europe
Japan,2023,4231,1.9,2.8,2.6,125,Asia
Japan,2022,4152,1.7,2.5,2.7,126,Asia
Japan,2021,4089,2.0,2.2,2.8,126,Asia
France,2023,3049,1.2,5.2,7.1,68,Europe
France,2022,2987,2.5,5.8,7.3,68,Europe
France,2021,2957,6.8,2.1,7.9,68,Europe

```

РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

```

● PS C:\Users\Андрей\Desktop\StafIT> py main.py --files data/sample_data.csv --report average-gdp

Чтение данных из 1 файла(ов)

Загружено 15 строк из data/sample_data.csv

Генерация отчета 'average-gdp'

        Результаты анализа
+-----+-----+
| Страна      | Средний ВВП |
+=====+=====+
| United States | 23923.67 |
+-----+-----+
| China        | 17810.33 |
+-----+-----+
| Japan        | 4157.33 |
+-----+-----+
| Germany      | 4138.33 |
+-----+-----+
| France       | 2997.67 |
+-----+-----+

Отчёт сформирован. Представлены данные для 5 стран.

```

РЕЗУЛЬТАТ ТЕСТИРОВАНИЯ

```
PS C:\Users\Андрей\Desktop\StafIT> pytest tests/ -v
===== test session starts =====
platform win32 -- Python 3.13.3, pytest-7.4.0, pluggy-1.6.0 -- C:\Users\Андрей\AppData\Local\Programs\Python\Python313\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Андрей\Desktop\StafIT
plugins: anyio-4.11.0, Faker-38.2.0, cov-4.1.0
collected 12 items

tests/test_analysis.py::TestGDPAverageAnalyzer::test_empty_data PASSED [ 8%]
tests/test_analysis.py::TestGDPAverageAnalyzer::test_single_country_single_year PASSED [ 16%]
tests/test_analysis.py::TestGDPAverageAnalyzer::test_single_country_multiple_years PASSED [ 25%]
]
tests/test_analysis.py::TestGDPAverageAnalyzer::test_multiple_countries PASSED [ 33%]
tests/test_analysis.py::TestGDPAverageAnalyzer::test_invalid_gdp_values PASSED [ 41%]
tests/test_analysis.py::TestGDPAverageAnalyzer::test_missing_country PASSED [ 50%]
tests/test_analysis.py::test_get_analyzer PASSED [ 58%]
tests/test_main.py::test_parse_arguments_valid PASSED [ 66%]
tests/test_main.py::test_parse_arguments_multiple_files PASSED [ 75%]
tests/test_main.py::test_parse_arguments_missing_files PASSED [ 83%]
tests/test_main.py::test_parse_arguments_missing_report PASSED [ 91%]
tests/test_main.py::test_parse_arguments_invalid_report PASSED [100%]

===== 12 passed in 0.15s =====
```

ВЫВОД

Результатом проделанной работы является разработанный скрипт, читающий CSV-файлы с экономическими данными через параметры --files, формирует отчёт average-gdp со средним ВВП по странам, сортирует результаты по убыванию и выводит таблицу в консоль. Реализовано тестирование pytest (включая тесты для пустых данных, некорректных значений и множественных файлов), используя стандартные библиотеки (argparse, csv) для обработки данных, что удовлетворяет функциональным и нефункциональным требованиям данного задания.