

Анализ макроэкономических данных

Что нужно сделать?

Нужно написать скрипт для обработки csv-файла.

Макроэкономические данные звучит страшно, но ничего сложного нет.

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), отчёт 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.

Чтобы сфокусироваться на функционале формирования отчёта и не отвлекаться на рутинные задачи (обработку параметров скрипта, чтения файлов и вывод), можно использовать стандартную библиотеку `argparse` и `csv`, а для отображения в консоли - библиотеку `tabulate`.

Примеры файлов можно посмотреть [здесь](#), выглядят так:

```
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
```

Пример запуска скрипта:

```
$ python3 main.py --files dataset1.py dataset2.py --report average-gdp
```

	country	gdp
1	United States	23923.67
2	China	17810.33
3	Japan	4467.00
4	Germany	4138.33
5	India	3423.67
6	United Kingdom	3113.33
7	France	2834.67
8	Canada	2096.33
9	Russia	2077.67
10	Italy	2042.00
11	Brazil	1900.67
12	South Korea	1727.33
13	Australia	1637.00
14	Spain	1409.33
15	Mexico	1392.67
16	Indonesia	1274.33
17	Saudi Arabia	1016.33
18	Netherlands	1006.00
19	Turkey	927.33
20	Switzerland	845.00

Какие функциональные требования?

- 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

Какие не функциональные требования?

- 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

Как сдавать задание?

- присылайте ссылку на git репозиторий, ссылки на google drive или yandex не подходят
- присылайте примеры запуска скрипта, например:
 - можно сделать скриншот запуска скрипта и добавить его репозиторий, для примера работы можно использовать [эти](#) файлы. За приложенные примеры запуска ревьюер скажет вам спасибо и добавит баллы.
- перед отправкой ссылки на репозиторий проверьте, пожалуйста, что репозиторий публичный и его можно посмотреть

FAQ

- Можно ли использовать нейросети?
 - Рекомендуем не использовать. Сталкиваемся со случаями, когда кандидаты увлекаются нейросетями, чтобы сделать тестовое, а потом не проходят техническое интервью, потому что не понимают, почему нейросеть написала тот или иной код.
- Будет ли приниматься задание без тестов?
 - Нет, приниматься не будет. Наличие тестов входит в основные требования.
- Код покрыт тестами - это какой процент покрытия?
 - Можно ориентироваться на 80% покрытия по pytest-cov, можно больше, можно меньше, но главное чтобы был протестирован критически важный функционал.
- Можно ли использовать какие-то дополнительные библиотеки к pytest?
 - Да, всё что помогает вам тестировать код можно использовать.
- Можно ли пользоваться линтерами или форматтерами кода?
 - Да, можно использовать любой линтер и любой форматтер - это хорошая практика.
- Можно ли использовать что-то кроме pip для управления зависимостями?
 - Да, можно, приветствуется.
- Можно ли менять API, название и формат параметров скрипта?
 - Нет, менять нельзя, используем **files** и **report**.
- Нужно ли учитывать случаи, когда пользователь при запуске скрипта ввёл что-то не то?

- Да, нужно, считаем что содержимое файлов всегда валидно, но пользователь может запустить скрипт с любыми аргументами.
- Нужно ли учитывать случаи, когда файл с данными большой, например несколько Гб?
 - Нет, не нужно, ограничений по памяти нет, всё можно читать в память.
- Нужно ли писать комментарии в коде?
 - Если вы считаете что они нужны, то пишите.
- На что будет смотреть ревьюер при проверки задания?
 - Будет проверяться выполнение функциональных и нефункциональных требований, соблюдение общепринятых стандартов написания кода на python и архитектура, которая позволит добавлять новые отчёты.
- Новые отчёты которые будут добавляться будут использовать тот же формат файлов?
 - Да, формат будет такой же, новые колонки добавляться не будут.
- Нужно ли упаковывать приложение в docker?
 - Нет, не нужно.
- Нужно ли писать readme.md?
 - Да, но достаточно пара абзацев в несколько предложений. В него можно положить примеры запуска скрипта и написать то, что на ваш взгляд важно знать ревьюеру, например описание того как добавить новый отчет. Если будете писать readme, то пусть он будет небольшим, но по делу, чем большим и сгенерированным нейросетью.
- Можно ли задавать уточняющие вопросы если что-то не понятно?
 - Не стоит, оперативно ответить не сможем, формулировки проверялись несколькими людьми, если что-то вдруг не понятно, то делаем так как понимаем, в крайнем случае можно прямо в коде или readme комментарий для ревьюера оставить, он всё читает.