# Скрипт подсчёта зарплаты сотрудников

## Что нужно сделать?

Нужно написать скрипт, который читает данные сотрудников из файлов в формате csv и формирует простой отчет по заработной плате (см. ниже примеры). В скрипт можно передать несколько файлов и тип отчета который нужно сформировать, в данном случае отчёт по зарплатам **payout**. Файлы на вход всегда в формате **csv**, валидные и без ошибок. Название отчета передается через параметр **--report**. Реализовать нужно только отчёт по зарплатам, но желательно заложить возможность добавления новых отчётов, например если захочется посмотреть среднюю ставку в час по отделам то это можно будет быстро добавить.

##### Пример файла csv с данными сотрудников:

```

id,email,name,department,hours\_worked,hourly\_rate

1,alice@example.com,Alice Johnson,Marketing,160,50

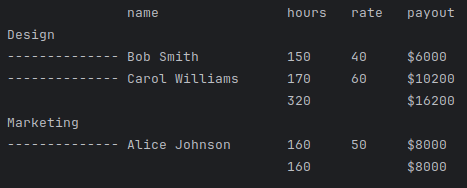
2,bob@example.com,Bob Smith,Design,150,40

3,carol@example.com,Carol Williams,Design,170,60

```

##### Пример отчёта payout, можно вывести прямо в консоль:

```



```

##### Пример запуска скрипта:

```

python3 main.py data1.csv data2.csv data3.csv --report payout  
```

Примеры файлов с данными сотрудников можно скачать [тут](https://drive.google.com/drive/folders/1x_L6MWtA9PRVhkkIjol53EuBTIecN-5f?usp=drive_link). Название колонки hourly\_rate может быть разными в разных файлах, а именно hourly\_rate, rate, salary. Вот такой вот экспорт, при это одинаковый порядок колонок не гарантируется, как в жизни, каждый менеджер сделал по своему (:

## Какие функциональные требования?

* можно передать пути к файлам
* можно указать название отчета
* можно сформировать отчет payout

## Какие не функциональные требования?

* для всего кроме тестов, можно использовать только стандартную библиотеку, например, для работы с параметрами скрипта нельзя использовать click, но можно использовать argparse, а для чтения csv нельзя использовать pandas
* нельзя использовать библиотеку csv
* код покрыт тестами написанных 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, тестовые присланные в google drive или yandex диск проверяться не будут
* присылайте примеры запуска приложения, например:
  + можно сделать скриншот запуска скрипта и добавить его репозиторий
  + можно сделать скриншот сформированного отчета и добавить его репозиторий, для примера формирования отчета можно использовать [эти](https://drive.google.com/drive/folders/1x_L6MWtA9PRVhkkIjol53EuBTIecN-5f) три файла примера.
* перед отправкой ссылки на репозиторий проверьте, пожалуйста, что репозиторий публичный и его можно посмотреть

## Как получить дополнительные баллы за тестовое?

Это дополнительные требования, их не обязательно реализовывать, но они помогут проявить себя. Если эти требования будут реализованы, хотя бы частично - это даст дополнительные баллы:

* есть обработка случаев, когда пользователь при запуске скрипта указал что-то не то
* в архитектуру заложена возможность быстрого добавления новых отчётов
* в коде используются аннотации

## FAQ

* Входные файлы всегда в формате csv?
  + Да, всегда. Выходной формат json, но в идеале должна быть возможность легко добавить к json новый формат если потребуется. Легко означает что не надо переписывать пол проекта чтобы добавить новый формат, а лишь дописать новый функционал или внести незначительные изменения.
* 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](https://pypi.org/project/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 или форматтер - это хорошая практика.
* Можно ли использовать pandas?
  + Нет, он не входит в стандартную библиотеку.
* Почему нельзя использовать библиотеку csv, она же стандартная?
  + Библиотеку csv здесь экономит буквально 15 минут, но её отсутствие позволит больше поработать напрямую со строками.
* Нужно ли писать комментарии в коде?
  + Если вы считаете что они нужны, то пишите.
* Нужно ли писать README.md?
  + Писать не обязательно, но его наличие это хороший тон, например в README.md можно описать как добавить новый отчет, где какие методы или классы нужно для этого написать или поменять.