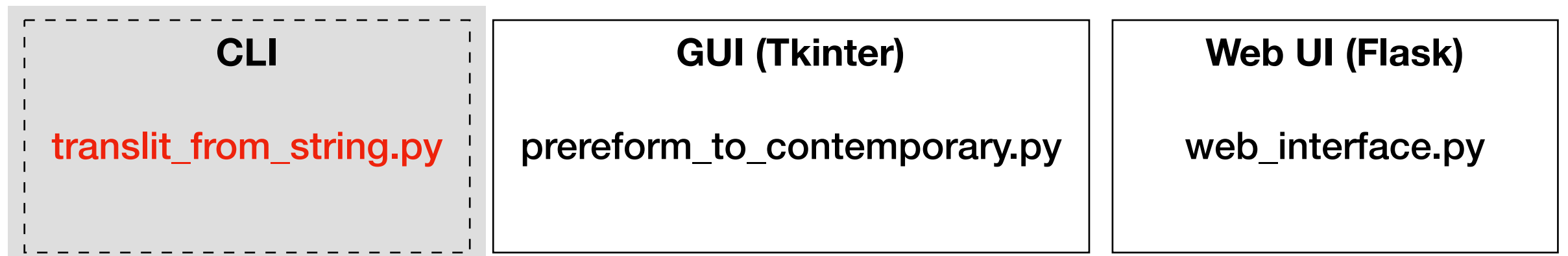


Предлагаемая примерная разбивка работ по проекту Prereform2Contemporary

Вариант для обсуждения, 25.07.2020

Подготовлен AlexSkrrn

Текущая структура проекта (стр. 1 из 2)

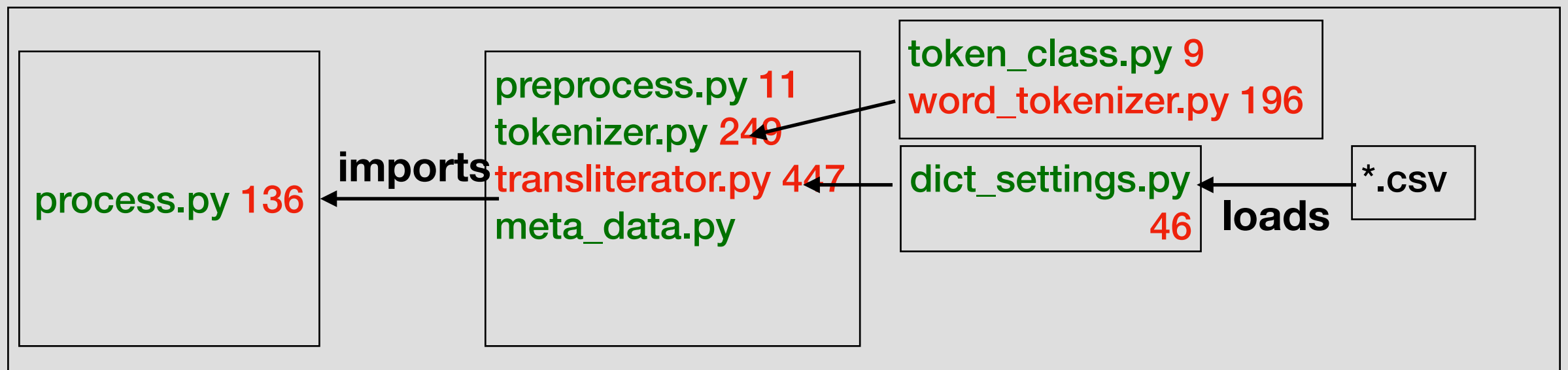


API — все программы-клиенты импортируют класс **Processor**

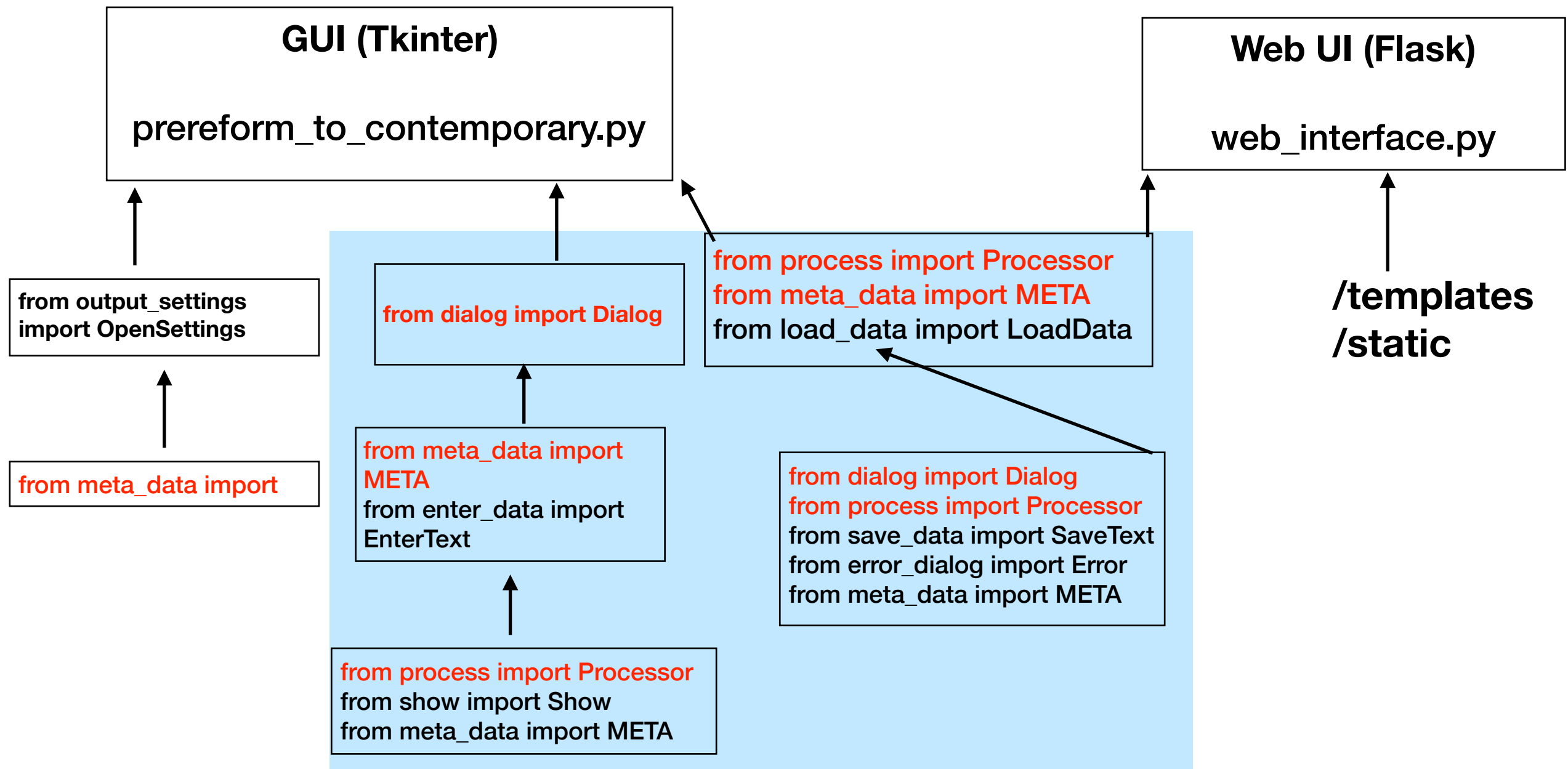
`Processor.process_text()`

params: text, show, delimiters, check_brackets

returns: text, changes, wrong_edits, str_json

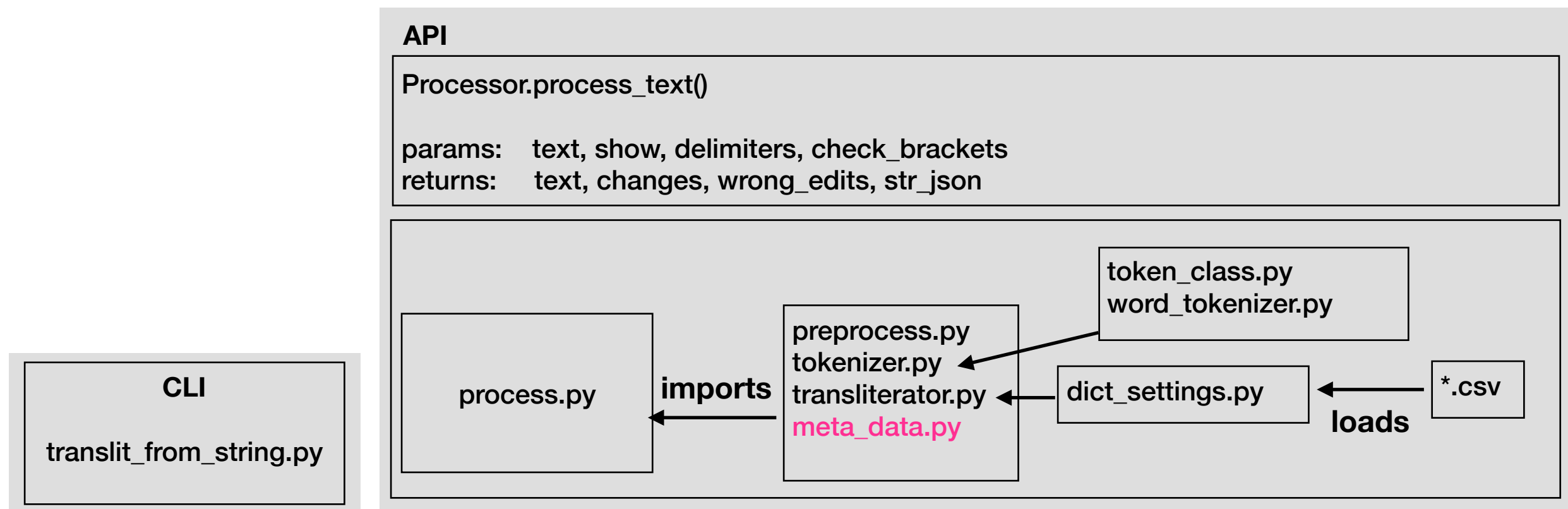


Текущая структура проекта (стр. 2 из 2)



Поток работ 1: консольный интерфейс (стр. 1 из 3)

1. Определить все возможные входящие параметры для `Processor.process_text()`
2. Написать тесты (`unittest` / `pytest`) для `Processor.process_text()` — чтобы ничего не поломать при переходе с `Python 2` на `Python 3`, а также в качестве документации, чтобы сторонние люди могли посмотреть эти тесты, и из них понять, что именно подается в основные функции программы и что именно эти функции возвращают
3. Перенести на `Python 3` (просто запустить в `Python 3` и смотреть, что сломалось, и менять код до тех пор, пока все тесты не будут проходить)
4. Преобразовать в пакет и сделать его инсталлируемым — чтобы потом можно было сделать так: `$ pip install translit-from-string`, а в программах-клиентах писать: `from translit_from_string import Processor`. После этого код этой программы можно будет убрать из репозитория, в котором будет находиться код для `GUI` и/или `Web UI`
5. Опционально: Придумать красивое название для этого пакета для размещения в `PyPi.org` (вместо `translit-from-string`)
6. Опционально: Можно было бы добавить обработку целого файла и директории с файлами из командной строки



Поток работ 1: консольный интерфейс (стр. 2 из 3)

- Кто-то один создает репозиторий в гитхабе, где будет мастер-версия кода консольной программы
- В репозитории будет мастер-бранч с кодом программы под Python 2 и отдельный второй бранч (py3-branch) для кода под Python 3. Тогда в процессе работы по переносу кода на Python 3, если что-то не будет получаться, то можно будет возвращаться в мастер-бранч, посмотреть, как это все работало изначально и, возможно, дописывать дополнительные тесты и пытаться их пройти в py3-branch
- Все остальные делают себе fork из этого репозитория и клонируют себе на диск. Потом создают отдельный бранч из мастера или из py3-branch, пишут код, commit-ят и push-ат в свои репозитории на гитхабе. Когда какой-то разумный блок кода готов, каждый делает merge этой ветви в свой мастер-бранч или py3-branch, делает commit и push, а потом создают Pull Request. Пока владелец основного репозитория рассматривает PR, каждый может создать у себя другой бранч и продолжать работу — делать commit-ы в и push-ить их в свои репозитории в гитхабе, чтобы не модифицировать первоначальный PR

Поток работ 1: консольный интерфейс (стр. 3 из 3)

- Предлагаемая файловая структура репозитория:

Этап 1 (master-branch)
Просто скопировать
файлы из существующего
репо)

/translit_from_string/
translit_from_string.py

process.py

meta_data.py
tokenizer.py
transliterater.py
preprocess.py

dict_settings.py

token_class.py
word_tokenize.py

*.cvcs

Этап 2 (master-branch и py3-branch)
Реорганизовать файлы и поменять
import-ы, так как cvs-файлы будут в
отдельной папке

/translit_from_string/
readme.md
.gitignore
/tests/
 __init__.py # пустой
 /system/
 __init__.py
 test_translit_from_string.py
 /integration/
 __init__.py
 test_process.py
 /unit/
 __init__.py
 __test_что_то.py
/translit_from_string/
 __init__.py # (пока) пустой
 *.py # все py-файлы
/data/

Этап 3 (py3-branch) Плюс ко всему,
следующие файлы, нужные для того,
чтобы сделать пакет
инсталлируемым:

/translit_from_string/
...
requirements.txt # может и не надо
MANIFEST.in
setup.py

Поток работ 2: графический интерфейс (стр. 1 из 1)

- Здесь пока только очень общие вещи:
- Если не хватает какого-то функционала, то есть два пути — либо сначала добавить его, а потом переносить на Python 3, либо сначала перенести на Python 3, а потом добавлять функционал. Мне кажется, второй путь перспективней, так как за последние годы интерфейс у Tkinter вроде бы сильно поменялся, и обновить старый интерфейс до нового и уже работать в нем мне кажется намного проще, чем изучить старый интерфейс и что-то реализовать в нем. Вообще Tkinter — это отдельная большая тема. Чтобы в нем с нуля разобраться для реализации нового функционала, нужно много времени, как мне кажется
- В целом, здесь я предложил бы сделать, как и для потока работ 1, отдельный репозиторий. Потом реорганизовать файловую структуру, чтобы в одной папке был код, относящийся к графическому интерфейсу, в другой папке — код, относящийся к потоку работ 1 (когда будет завершен поток 1, эту папку можно будет удалить), а код, относящийся к веб-интерфесу, отсюда выкинуть
- После того, как код будет обновлен до Python 3 и, при необходимости, добавлен новый функционал, можно использовать инструмент вроде PyInstaller, который позволяет из питоновского пакета сделать exe-программу. Такую программу можно запускать под Windows, и не будет требоваться, чтобы у пользователя был установленный Питон