## Задание 1.Отчет.

Проверьте все страницы на наличие ссылок. Если есть страницы без ссылок,

укажите их номера. Опишите выбранный вами способ проверки по шагам, чтобы

любой пользователь смог воспроизвести ваши действия.

Задачи:

1)Определить количество ссылок. В моем случае их было 9999.

2)Скачать с git мою программу для проверки ссылок

[**https://github.com/ItraRomal/testing0.git**](https://github.com/ItraRomal/testing0.git)

3)Запустить консоль с помощью комбинации клавиш **win + r** и введения в открывшееся поля **cmd**

Предполагается, что на этом компьютере установлен python.

4)Открыть расположение программу с помощью команды

**cd адрес\_папки\_в\_которой\_хранится\_программа**

5)Запустить программу следующим образом:

**python shironin\_link\_checker.py номер\_первой\_проверяемой\_страницы номер\_последней\_страницы**

6)По завершении работы программа оповестит Вас и предложит создать файл с отчетом. В случае отказа отчет будет выведен в терминале

(Более подробное описание работы программы описано в файле **README.txt**)

## Задание 1. Подробный комментарий.

1)Определить количество ссылок.

1.1.Проверить наличие 10, 100, 1000 и т.д. страницы  
1.2.При обнаружении интервала - постепенно сократить его до обнаружения последней страницы. В данном случае интервал был 1000 : 10000. Так что последняя страница была найдена очень легко.

2)Исходя из числа ссылок выбрать два инструмента для проверки

2.1.Т.к. было обнаружено 9999 страниц, то метод поиска пустых страниц вручную малоэффективен. Логично начать искать или написать программу. Быстрее всего будет использовать 2 разные сторонние программы ( две - для контроля неэффективности незнакомого ПО)

2.2. Мною были выбраны программы **Xenu’s Link Sleuth** и **LinkChecker**

3)Дважды использовать обе программы. (опять же, все для избежания ошибки). При наличии явных проблем - найти решение и повторить пункт.

3.1.**LinkChecker**

Эта программа работает довольно медленно. К тому же после завершения проверки выяснилось, что одновременно она проверяет 8543 ссылку. Почему - не знаю.

3.2. **Xenu’s Link Sleuth**

Эта программа работает очень быстро, но проверяет лишь по 999 ссылок. В связи с этим пришлось делать дважды по 10 тестов.

Вторая проблема - даже при отсутствии ссылки страница не считалась дефектной, а значит тесты не выдали ошибок. Так что я прогнал все ссылки еще раз, но в этот по завершении проверки фильтровал результаты по наличию текста. Так очень легко были найдены страницы без ссылок, с похвалой упорству, славой великому зверю и прочими прелестями.

К тому же была обнаружена особенность - в репорте текст страницы соответствует предыдущей странице. Т.е. на 1-ой странице пол репорту не было ничего, что ложно.

4)Сравнение результатов  
Xenu за 20 минут работы дважды прогнал все ссылки и выдал две пустые:

<https://s3.eu-central-1.amazonaws.com/qa-web-test-task/638.html>

<https://s3.eu-central-1.amazonaws.com/qa-web-test-task/3395.html>  
LinkChecker все еще работал на тот момент. А после выдал, что битых ссылок нет, что, конечно же, не показывает наличие или отсутствие пустых страниц.

5)Вручную была обнаружена совершенно пустая страница

[https://s3.eu-central-1.amazonaws.com/qa-web-test-task/8543.html](https://s3.eu-central-1.amazonaws.com/qa-web-test-task/3395.html)

Так как ни один из выбранных инструментов не смог ее определить был сделан вывод, что проще и полезней самому написать программу.

6)Под это дело был установлен git, pycharm и, собственно python.

7)После многих часов гугления была создана программа, довольно быстро проверяющая ссылки и автоматически создающая репорт

https://github.com/ItraRomal/testing0.git

# Задание 2

## Анализ программы

**Предполагается, что тестируемая версия программы - финальная и в ней до релиза больше не будут вноситься изменения, исключая баг-фиксы.**

В первую очередь для проведения тестирования я соберу информацию по:

1.Изменениям, внесенным в программу со времени предыдущего релиза (улучшения, новые фичи).

2.Имеющимся баг-репортам (как открытым, так и закрытым), накопленным за период между предыдущим и нынешним релизами.

3.Имеющимся тест-кейсам.

## План тестирования

1. Функционала, описанного в приложенном мануале (который могу считать списком требований к программе).

1.1.Работу интерфейса программы (графические элементы, вид программы при смене языка) .

1.2.Создание и редактирование файлов.

1.3.Работу программы со сканером.

1.4.Работу программы с данными полученными со сканера.

2. Установку и удаление программы.

3. Проверить скорость работы основных процессов на разном железе, сравнить с производительностью предыдущей релизной версии.

## Тестирование

1.Составить тест-кейсы под каждый вышеописанный пункт.

2.Провести тесты. Желательно разными методами (чтобы проверить эффективность и качество выбранного теста) и с повторами (чтобы проверить стабильность результата теста) для контроля качества тестирования.

## Финальный этап (обработка результатов)

1.По каждому из проведенных этапов тестирования составить и оформить отдельный список баг-репортов. Кроме того сравнить их со списками багов, оставшихся с предыдущей релизной версии.

2.При обнаружении ошибок и неточностей по результатам проведения тест-кейсов - составить список комментариев к данным тест-кейсам и отдельным тестам.

3.Оценить критичность выявленных багов для функционирования программы.

4.Составить предварительный отчет о готовности программы к релизу.