Документация проекта Parser

Введение

Данный проект представляет из себя утилиту командной строки, в которой в качестве параметра указывается произвольный URL. Она извлекает по этому URL страницу, обрабатывает ее и формирует текстовый файл с текстом статьи, представленной на данной странице.
Программа состоит из двух файлов: **setting.py** и **main.py**

setting.py

В данном файле содержится класс **Tag**, который представляет из себя настраиваемые параметры конкретно взятого тега. Он хранит в себе три параметра для редактирования текста внутри конкретно взятого тега: **before**, **after**, **delete**.

before - это текст который мы можем подставить перед содержимым тега. По умолчанию данный параметр не содержит текста.

after - это текст который мы можем подставить после содержимого тега.

delete - это параметр отвечает за то будет ли удален текст внутри тега. По умолчанию он задан как **False**

Также в файле содержится словарь **option** в котором, в качестве ключей, содержатся названия тегов, содержимое которых требуется изменить в процессе извлечения страницы, а в значениях экземпляры класса **Tag**

main.py

В данный файл мы добавляем сразу несколько библиотек:

requests для того чтобы выполнить запрос к требуемой странице;

BeautifulSoup для сохранения и редактирования содержимого тегов страницы;

textwrap для редактирования максимальной длины текста в одной строке внутри текстового редактора;

os для открытия, создания и сохранения статьи в отдельный файл;

также импортируем **option** из **setting.py**

В качестве константы в начале файла указан **HEADERS** в котором содержатся необходимые данные для запроса к странице

Функция **get_html()** предназначена для создания сессии запроса. Входный параметром данной функции является url адрес веб страницы

Функция **save_file()** отвечает сохранения статьи в текстовый документ, также в данной функции осуществляется следующее правило форматирования ширина строки не больше 80 символов (если больше, переносим по словам). В качестве входных параметров выступают **items** - весь нужный текст полученный в результате извлечения полезной информации из статьи и **path** - путь к файлу

Функция **reformat()** в данной функции происходит редактирование текста внутри тега. Входными параметрами являются **soup**- тег страницы, **after,before, delete** параметры для редактирования

Функция **reformat_link()** в данной функции происходит редактирование текста внутри тега, у которого есть ссылка
Входными параметрами являются **soup**- тег страницы, **after,before, delete** параметры для редактирования. **link_after,link_before, link_delete** параметры для редактирования ссылки

Функция **get_content()** в данной функции мы создаем **soup** который извлекает требуемую страницу
извлекаем название статьи **title** и редактируем текст в соответствии с извлеченными параметрами **heading** из **option**
ищем родительский тег абзацев для сохранения содержимого статьи **items**
извлекаем параметры **link** из **option** для ссылок
в цикле вызываем функцию **reformat()** либо **reformat_link()** в зависимости от тега, параметры для них извлекаем из словаря **option**
весь контент сохраняем в список **article**

Функция **parse()** в данной функции мы вызывает ввод **URL** страницы, создаем путь к файлу **path** обрабатывая **URL**
создаем папки согласно пути **path,** если они еще не созданы
присваиваем имя файла **FILE,** полученное из **URL**
вызываем функцию **get_html()** и сохраняем в переменную **html**

В случае положительного статуса запроса страницы
вызываем функцию **get_content()**
вызываем функцию **save_file()** входными параметрами являются результат функции **get_content()** и **path+FILE**
иначе вывести ошибку

Результат

Программа создаст и откроет текстовый документ с отредактированным текстом, с соответствующими параметрами, указанными настройками. Были протестированы сайты gazeta.ru,lenta.ru,meduza.io и приложены файлы полученные в результате выполнения программы.В дальнейшем в проекте можно реализовать запись в базу данных, и считывать не одну страницу сайта, а сразу много

Исходный код setting.py

```
class Tag:
  def __init__(self, after, before='', delete=False):
      self.before = before
      self.after = after
      self.delete = delete


option = {
  'h1': Tag('\n\n', '\n\n'),
  'h2': Tag('\n\n', '\n\n'),
  'h3': Tag('\n\n', '\n\n'),
  'p': Tag('\n\n'),
  'a_link': Tag('] ', ' ['),
  'a': Tag(''),
  'blockquote': Tag('\n\n')
}
```

Исходный код main.py

```
import requests
from bs4 import BeautifulSoup
import textwrap
import os
from setting import option
```

```
HEADERS = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101
Firefox/69.0', 'accept': '*/*'}


def get_html(url):
  session = requests.Session()
  r = session.get(url, headers=HEADERS)
  return r


def save_file(items, path):
  with open(path, 'w', encoding='utf-8', newline='\n') as file:
    for item in items:
      item = item.replace('\n', '`~')
      item = textwrap.fill(item, 80)
      item = item.replace('`~', '\n')
      file.write(item)


def reformat(soup, after, before, delete=False):
  tmp = ''
  for j in soup:
    if not delete:
      tmp = before+j.get_text()+after
    j.string = tmp
    j.name = 'content'
  return soup


def reformat_link(soup, after, before, link_before, link_after, link_delete=False, delete=False):
  tmp = ''
  link = ''
  for j in soup:
    if not link_delete:
      link = link_before + j.get('href') + link_after
    if not delete:
      tmp = before+j.get_text()+after+link
    j.string = tmp
    j.name = 'content'
  return soup
```

```
def get_content(html):
    soup = BeautifulSoup(html, 'html.parser')
    heading = option['h1']
    items = soup.find('p').parent
    link = option['a_link']
    title = ''
    if not heading.delete:
        title = soup.find('h1').get_text()+heading.after
    for key, value in option.items():
        tag = items.find_all(key)
        if key == 'a':
            reformat_link(tag, value.after, value.before, link.before, link.after, link.delete,
value.delete)
        else:
            reformat(tag, value.after, value.before, value.delete)

    main = items.find_all('content')
    article = [title]
    for k in main:
        article.append(k.get_text())
    return article


def parse():
    URL = input('Введите URL: ')
    URL = URL.strip()
    end = URL.rfind('/')
    path = os.path.join(os.getcwd(), URL[8:end])
    if not os.path.exists(path):
        os.makedirs(path)
    FILE = URL[end:]+'.txt'
    html = get_html(URL)
    if html.status_code == 200:
        article = get_content(html.text)
        save_file(article, path+FILE)

        os.startfile(path+FILE)
    else:
        print('Error')


if __name__ == '__main__':
    parse()
```