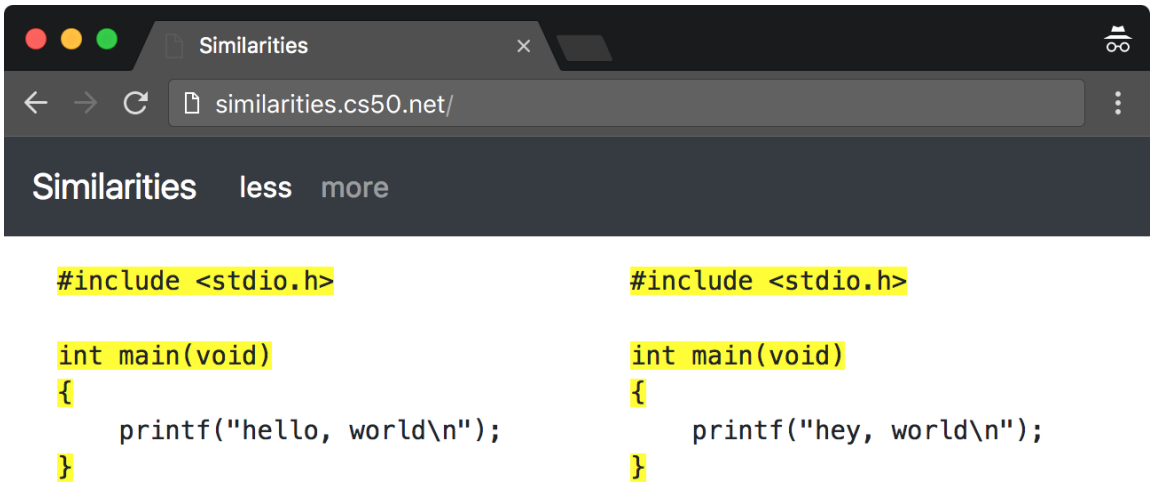
Similarities

tl;dr

1. Створіть програму, яка шукає подібності у двох файлах.

2. Створіть веб-сайт, на якому виділяються подібності у двох файлах, як на рисунку нижче.



Вступ

Визначення того, чи два файли є ідентичними - тривіальна задача (відносно!): пройдіться по символах у кожному із файлів і порівняйте їх між собою. Але визначення того, чи два файли є схожими не є тривіальною задачею. Врешті-решт,

що значить «бути подібним»? Можливо те, що файли мають спільні рядки. Можливо те, що файли мають спільні речення. Можливо те, що файли мають спільні підрядки.

Гадаємо, ви вже зрозуміли, що ваша задача - визначити, чи два файли є схожими!

Перед початком

Ось так потрібно завантажити матеріали завдання до вашого середовища CS50 IDE. Увійдіть до CS50 IDE, а потім у вікні терміналу виконайте команди, наведені далі.

1. Виконайте `update50` щоб переконатись, що ваше середовище IDE оновлене. На виконання цієї команди може піти кілька хвилин.

2. Виконайте `cd` аби впевнитись, що ви знаходитесь у `~/workspace/` (тобто у теці під назвою `workspace`, яка знаходиться у вашому домашньому каталозі `~`).

3. Виконайте `mkdir pset7`, щоб створити теку під назвою `pset7` у вашій теці `workspace`, якщо такої теки у вас ще немає.

4. Виконайте `cd pset7` для переходу до (відкриття) цієї теки.

5. Виконайте `wget http://cdn.cs50.net/2018/fall/psets/7/similarities/similarities.zip` для завантаження заархівованого ZIP-файлу із вихідними матеріалами задачі.

6. Виконайте `unzip similarities.zip` для розархівування файлу.

7. Виконайте `rm similarities.zip` і вкажіть `yes` чи `y` щоб видалити ZIP-файл.

8. Виконайте `ls`. Ви маєте побачити теку `similarities`, яка знаходилась всередині ZIP-файлу.

9. Виконайте `cd similarities`, щоб перейти в цю теку.

10. Виконайте `ls`. Ви маєте побачити матеріали задачі всередині теки.

```
$ cd
$ mkdir pset7
$ cd pset7
$ wget
http://cdn.cs50.net/2018/fall/psets/7/similarities/similarities.zip
$ unzip similarities.zip
$ rm similarities.zip
$ cd similarities
$ chmod a+x compare
$ ls
application.py  compare*  helpers.py  inputs/  requirements.txt
static/  templates/
```

Розуміння

`inputs/`

Всередині цієї теки купа вхідних даних для прикладу, які ви можете порівнювати, зокрема, для пошуку подібностей!

`compare`

Відкрийте файл `compare`. Легко побачити, що ім'я файлу не закінчується на `.py`, хоча файл містить програму, написану на Python. Але це нормально! Зверніть увагу на «шебанг» (символ #!) згори файлу:

```
#!/usr/bin/env python3
```

Цей рядок говорить комп'ютеру інтерпретувати (тобто виконувати) цю програму із використанням `python3` (він же - `python` у CS50 IDE), інтерпретатора, що розуміє Python 3.

Зверніть увагу, як цей файл визначає функцію `main` і викликає у нижній частині файлу. Визначати `main` не є обов'язковим у Python, але визначення функції до того, як ви будете її викликати, є обов'язковим. Відповідно, оскільки `main` викликає

функцію із назвою `positive`, і тому, що ми хотіли тримати головну частину цієї програми у верхній частині файлу, був сенс реалізувати `main` як функцію. Таким чином, `main` вперше викликається у нижній частині файлу (після того, як було визначено `positive`), попри те, що `main` реалізовано у верхній частині файлу.

Немає потреби розуміти кожен рядок коду у `compare`, але зверніть увагу, відповідно до коментарів, що загалом робить цей код: він «парсить» надані аргументи командного рядка, зчитує два файли у змінні як рядки, порівнює ці два рядки і потім виводить список подібностей. Рядки порівнюються між собою одним із трьох способів, що визначається аргументом командного рядка: рядок до рядка, речення до речення чи підрядок до підрядка.

`helpers.py`

Відкрийте файл `helpers.py`. Ох, знайоме нам `TODO`. У цьому файлі визначено три функції, кожна з яких повинна реалізовувати різний алгоритм порівняння: для `рядків`, `речень` та `підрядків`. Зараз кожна з них повертає порожній список. Але це не надовго!

`application.py`

Відкрийте файл `application.py`. Цей файл реалізує веб-додаток, який, врешті-решт, дозволить вам виконувати будь-який із цих трьох алгоритмів на будь-яких двох текстових файлах. Немає потреби повністю розуміти цей файл, особливо `highlight` і `errorhandler`. Але знайте, що `highlight`, за умови, що йому надали рядок `s` та список інших рядків `strings`, підсвічує (шляхом обернення їх у HTML-тег `span`) всі входження цього рядка до списку. А `errorhandler` забезпечує те, що будь-які HTTP-помилки будуть відображені на власній сторінці.

Обов'язково прочитайте функції `index` та `compare`, остання з яких обробляє відправлення форми.

`templates/layout.html`

Відкрийте файл `templates/layout.html`. У цьому файлі знаходиться шаблон загального вигляду для веб-додатку. Мабуть, ви впізнаєте деякі HTML-теги та побачите нові. Зверніть увагу, як шаблон використовує популярну бібліотеку Bootstrap. Насправді, ми взяли за основу шаблон з цієї бібліотеки starter template.

`templates/index.html`

Відкрийте файл `templates/index.html`. Ох, останнє `TODO`. Зверніть увагу, як цей шаблон «розширює» (extends) `layout.html`, що говорить нам про те, що `layout.html` є основою, з якої буде зроблено `index.html`. `block`, визначений у `index.html` буде вставлено у відповідне місце для `block` у `layout.html`.

Врешті-решт, цей файл міститиме форму, за допомогою якої користувач зможе завантажити два файли до вашого веб-додатка для порівняння одним із ваших трьох алгоритмів

`templates/compare.html`

Відкрийте файл `templates/compare.html`. Ми реалізували цей файл для вас. Завдяки його використанню CSS (особливо класу із назвою `col-6`), він забезпечує, що файли користувача, щойно вони будуть завантажені і підсвічені, будуть подані один біля одного.

`templates/error.html`

Відкрийте файл `templates/error.html`. У цьому файлі знаходиться шаблон, за допомогою якого будуть показані усі HTTP помилки. Для цього він використовує компонент Bootstrap під назвою Jumbotron.

`static/styles.css`

Відкрийте файл `static/styles.css`. У цьому файлі знаходяться деякі CSS властивості, які разом реалізують інтерфейс користувача вашого веб-додатку. Насправді, вони модифікують деякі значення Bootstrap за замовчуванням.

`requirements.txt`

Відкрийте файл `requirements.txt` (не змінюючи його, хоча в майбутньому ви можете це зробити, якщо захочете). Цей файл визначає бібліотеки, по одній в рядок, від яких залежить вся ця функціональність.

Специфікація

`helpers.py`

`lines`

Реалізуйте функцію `lines` таким чином, що якщо їй передати дві рядкові змінні (string), `a` і `b`, вона поверне список `list` рядків, які присутні і у `a` і у `b`. У списку не повинно бути дублікатів. Вважайте, що рядки у `a` і `b` будуть розділені символом `\n`, але рядки у списку `list`, що повертається, не повинні закінчуватись на `\n`. Якщо і `a`, і `b` містять один або більше пустих рядків (тобто `\n` перед яким немає ніяких інших символів), список `list`, що буде повернено, повинен містити пустий рядок (тобто `""`).

`sentences`

Реалізуйте функцію `sentences` таким чином, що якщо їй передати дві рядкові змінні (string), `a` і `b`, вона повертає список `list` *унікальних* речень англійською мовою, які присутні і у `a`, і у `b`. У списку `list` не повинно бути дублікатів. Використовуйте `sent_tokenize` із Natural Language Toolkit для того, щоб розбити кожен рядок на список речень ("токенізувати"). Його можна імпортувати за допомогою:

```
from nltk.tokenize import sent_tokenize
```

Відповідно до його документації, `sent_tokenize`, за умови, що йому надали `str` як вхідні дані, повертає список речень англійською мовою, що присутні у цьому рядку. Ця функція вважає, що її вхідні дані насправді є текстом англійською мовою (а не, наприклад, кодом, який випадково також може містити крапки).

`substrings`

Реалізуйте функцію `substrings` таким чином, що якщо їй передати дві рядкові змінні (string), `a` та `b`, і ціле число `n`, вона повертає список `list` усіх підрядків довжиною `n` які присутні і у `a`, і у `b`. У списку не повинно бути дублікатів.

Пригадайте, що підрядком довжини `n` деякого рядка є лише послідовністю із `n` символів цього рядка. Наприклад, якщо `n` дорівнює `2`, а рядком є `Yale`, існує три можливих підрядки довжини `2`: `Ya`, `al`, `le`. Водночас, якщо `n` дорівнює `1` а рядком є `Harvard`, існує сім можливих підрядків довжини `1`: `H`, `a`, `r`, `v`, `a`, `r` і `d`. Але після того, як ми видалимо всі дублікати, існує лише п'ять унікальних підрядків: `H`, `a`, `r`, `v`, and `d`.

`templates/index.html`

Реалізуйте `templates/index.html` таким чином, щоб у ньому була HTML-форма, за допомогою якої користувач може відправляти:

- файл з назвою `file1`
- файл з назвою `file2`
- значення `lines`, `sentences` чи `substrings` для поля `input`, що називається `algorithm`
- число з назвою `length`

Ви можете дивитись на HTML із розв'язку команди за потреби, але спочатку спробуйте створити правильний синтаксис самостійно, використовуючи https://www.google.com/search?q=html+forms!

Тестування

Для того, щоб перевірити вашу реалізацію `lines`, `sentences` і/або `substrings` за допомогою командного рядка, виконайте `compare` наступним чином, де `FILE1` та `FILE2` - два будь-які текстові файли (наприклад, ті, що всередині теки `inputs/`):

```
./compare --lines FILE1 FILE2
./compare --sentences FILE1 FILE2
./compare --substrings 1 FILE1 FILE2
./compare --substrings 2 FILE1 FILE2
...
```

Для того, щоб перевірити вашу реалізацію через веб-додаток, виконайте

```
flask run
```

і перейдіть по URL, яке було виведено на екран.

Обов’язково перевірте ваш варіант з файлами з теки `inputs/` (також доступні за посиланням: у браузері) та на своїх власних файлах!

check50

```
check50 cs50/2018/fall/similarities
```

style50

```
style50 helpers.py
```

Розв’язок команди курсу

CLI

```
~cs50/pset6/less/compare
```

Web

http://similarities.cs50.net/less