CS50x Iran

Regular, um, Expressions

Problem set

پاییز و زمستان ۱۴۰۴

حداقل در زبان انگلیسی، گفتن " um " برای مواقعی که تلاش می‌کنیم کلمه ای را به یاد بیاوریم، غیر عادی نیست. هرچه بیشتر این کار را انجام دهید، بیشتر جلب توجه می‌کنید!

در فایلی به نام um.py، تابعی به نام `count` را پیاده سازی کنید که یک خط متن از نوع `str` را به عنوان ورودی دریافت کند و تعداد دفعاتی را که " um " در آن متن ظاهر می‌شود، بدون حساسیت به حروف کوچک و بزرگ، به صورت `int` برگرداند. کلمه را به خودی خود، نه به عنوان زیر رشته‌ای از یک کلمه دیگر، در نظر بگیرید. به عنوان مثال، در متنی مانند hello, um, world ، تابع باید 1 برگرداند. با متنی مانند yummy، تابع باید 0 برگرداند.

ساختار um.py را به صورت زیر بسازید. می‌توانید توابع اصلی آن را تغییر دهید و یا آنطور که صلاح می‌دانید اجرا کنید، اما نمی‌توانید هیچ کتابخانه دیگری را import کنید. با اینکه اجباری نیست ولی می‌توانید از re و یا sys استفاده کنید.

```
import re
import sys

def main():
    print(count(input("Text: ")))

def count(s):
    ...
...

if __name__ == "__main__":
    main()
```

قبل یا بعد از پیاده‌سازی `count` در um.py، در یک فایل دیگر به نام test_um.py سه یا چند تابع را پیاده‌سازی کنید که مجموعاً اجرای `count` شما را بطور کامل آزمایش کنند، نام هر کدام باید با `test_` شروع شود تا بتوانید تابع خود را با دستور مشابه زیر اجرا کنید:

```
pytest test_um.py
```

▼ نکته

- یادتان باشد که ماژول `re` توابعی مانند `findall` دارد که می‌توانید در این لینک اطلاعات بیشتری در مورد آن کسب کنید.
- می‌توانید در Regular Expressions (یا regex) از کاراکتر های ویژه ای استفاده کنید. این لینک را مطالعه کنید.
- چون کاراکتر \ در regex می‌تواند با escape sequences (برای مثال \ که اگر در متن استفاده شود، حرف بعد از آن به عنوان یک فرمان در نظر گرفته می‌شود. برای مثال \n که حرف n بعد از \ به معنای new line بوده و هر جای متن استفاده شود، به معنای رفتن به خط بعد و شکستن متن است) اشتباه گرفته شوند، بهتر است از نماد رشته خام پایتون برای الگوهای regex استفاده کنید. مانند Format strings که با پیشوند f مشخص می‌شوند، رشته‌های خام نیز یک پیشوند r دارند. مثلا به جای `"harvard\.edu"` از `r"harvard\.edu"` استفاده کنید.
- توجه داشته باشید که \b به عنوان مرزی بین \w (یعنی کاراکترهای کلمه‌ای) و یک \W (کاراکترهای غیر کلمه‌ای مانند ویرگول، فاصله و ...) و برعکس؛ و یا بین یک \w (مثلا کلمات) در ابتدا یا انتهای رشته‌ی مدنظر بکار می‌رود. اطلاعات بیشتر را در این لینک مشاهده کنید.
- استفاده از سایت regexr.com می‌تواند برای تست کردن Regular Expressions و مشاهده گرافیکی نتیجه عبارتی که نوشته‌اید، بسیار کمک کننده باشند. همچین از سایت thefreedictionary مثالهایی از کلمات دارای "um" را مشاهده کنید.

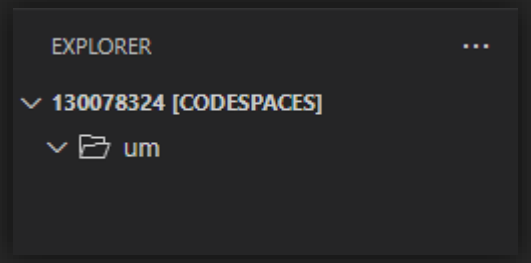
5 قبل از شروع توجه کنید:

وارد code.cs50.io شوید، سپس بر روی پنجره‌ی Terminal کلیک کنید. توجه داشته باشید که دستور پنجره‌ی Terminal شما باید به صورت زیر باشد:

```
$
```

سپس کد زیر را اجرا کنید تا یک پوشه به نام um در codespace ایجاد شود:

```
$ mkdir um
```



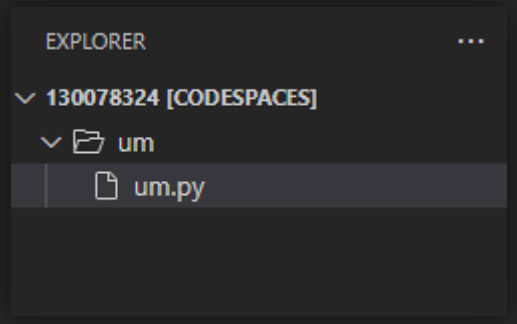
6 قبل از شروع توجه کنید:

سپس `cd um` را اجرا کنید تا به پوشه‌ی um منتقل شوید. اکنون شما باید آدرس `um/ $` را در Terminal مشاهده کنید:

```
$ cd um
um/ $
```

حالا می‌توانید دستور `code um.py` را اجرا کنید تا فایلی با نام um.py ایجاد شود:

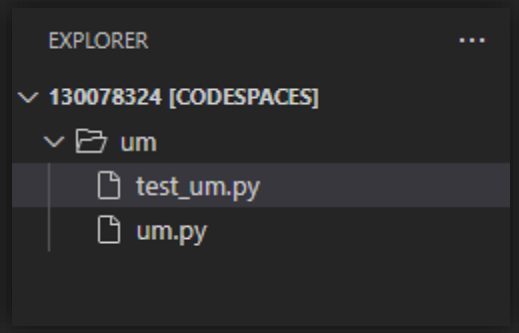
```
um/ $ code um.py
```



اکنون می‌توانید در این فایل برنامه‌ی خود را بنویسید.

یادتان باشد که دستور `code test_um.py` را نیز اجرا کنید تا فایلی با نام test_um.py ایجاد کنید:

```
um/ $ code test_um.py
```



اکنون می‌توانید در این فایل برنامه‌ی خود را بنویسید.

اکنون می‌خواهیم برنامه‌ی نوشته شده‌ی خود را تست کنیم:

- دستور `python um.py` را اجرا کنید. مطمئن شوید که برنامه از شما یک ورودی می‌گیرد. um را تایپ کنید و کلید Enter را فشار دهید. تابع `count` شما باید 1 را به عنوان خروجی برگرداند.

- دستور `python um.py` را اجرا کنید. um? را تایپ کنید و کلید Enter را فشار دهید. تابع `count` شما باید 1 را به عنوان خروجی برگرداند.

- دستور `python um.py` را اجرا کنید. Um, thanks for the album. را تایپ کنید و کلید Enter را فشار دهید. تابع `count` شما باید 1 را به عنوان خروجی برگرداند.

- دستور `python um.py` را اجرا کنید. Um, thanks, um... را تایپ کنید و کلید Enter را فشار دهید. تابع `count` شما باید 2 را به عنوان خروجی برگرداند.

اکنون می‌خواهیم test_um.py نوشته شده‌ی خود را تست کنیم:

- برای آزمایش تست های خود، `pytest test_um.py` را اجرا کنید. سعی کنید از نسخه های درست و نادرست um.py برای تعیین میزان تشخیص خطاها توسط تست ها استفاده کنید.
- همچنین مطمئن شوید که نسخه صحیح um.py را دارید. تست های خود را با `pytest test_um.py` اجرا کنید. pytest باید نشان دهد که تمام آزمایشات شما قبول شده است.

- تابع `count` را در نسخه صحیح um.py تغییر دهید. برای مثال `count` ممکن است، به اشتباه هر "um" را که بخشی از یک کلمه است، حساب کند. تست‌های خود را با `pytest test_um.py` اجرا کنید. pytest باید نشان دهد که حداقل یکی از تست های شما ناموفق بوده است.

- دوباره تابع `count` را در نسخه صحیح um.py تغییر دهید. برای مثال، تعداد ممکن است به اشتباه فقط با یک "um" مطابقت داشته باشد که از دو طرف با یک فاصله احاطه شده است. تست‌های خود را با `pytest test_um.py` اجرا کنید. pytest باید نشان دهد که حداقل یکی از تست‌های شما ناموفق بوده است.

❖ شما می‌توانید از آدرس زیر برای بررسی کردن کد خود استفاده کنید. CS50 از این برنامه برای آزمایش کد شما استفاده می‌کند. از این دستور استفاده کنید تا کدهایتان را امتحان کنید.

```
$ check50 cs50/problems/2022/python/um
```

لبخند های سبز به این معنی هستند که برنامه‌ی شما در تست قبول شده و اخم های قرمز یعنی برنامه‌ی شما دارای ایراد هست. با مراجعه به check50 URL می‌توانید خروجی مورد انتظار و خروجی برنامه‌ی خود را بررسی کنید.

با اجرا کردن صورت زیر در Terminal پاسخ خود را ارسال کنید.

```
$ submit50 cs50/problems/2022/python/um
```

CS50x Iran

Harvard's Computer Science 50x Iran