CS50x Iran

Working 9 to 5

Problem set

پاییز و زمستان ۱۴۰۴

24-Hour	12-Hour
00:00	12:00 AM
01:00	1:00 AM
02:00	2:00 AM
03:00	3:00 AM
04:00	4:00 AM
05:00	5:00 AM
06:00	6:00 AM
07:00	7:00 AM
08:00	8:00 AM
09:00	9:00 AM
10:00	10:00 AM
11:00	11:00 AM
12:00	12:00 PM
13:00	1:00 PM
14:00	2:00 PM
15:00	3:00 PM
16:00	4:00 PM
17:00	5:00 PM
18:00	6:00 PM
19:00	7:00 PM
20:00	8:00 PM
21:00	9:00 PM
22:00	10:00 PM
23:00	11:00 PM
00:00	12:00 AM

در حالی که در بیشتر کشورها بر اساس قالب زمانی 24 ساعته کار می‌کنند، ایالات متحده تمایل دارد از قالب زمانی 12 ساعته استفاده کند. بر این اساس، به جای «09:00 تا 17:00»، بسیاری از آمریکایی‌ها می‌گویند که «9:00 AM تا 5:00PM» یا همان «9 صبح تا 5 بعد از ظهر» کار می‌کنند که در آن «AM» مخفف کلمه "ante meridiem" و "PM" مخفف "post meridiem" است که در آن "meridiem" به معنای میان روز (یعنی همان ظهر) است.

← جدول تبدیل

همانطور که معادل "12:00 AM" در قالب 12 ساعته "00:00" در قالب 24 ساعته خواهد بود، "12:01 AM" تا "12:59 AM" هم معادل با "00:01" تا "00:59" خواهد بود.

در فایلی به نام working.py، تابعی به نام `convert` را پیاده سازی کنید که یک رشته یا `str` را در یکی از قالب‌های 12 ساعته زیر به عنوان ورودی بگیرد و `str` مربوطه را در قالب 24 ساعته (یعنی 9:00 تا 17:00) برگرداند. انتظار داشته باشید که AM و PM با حروف بزرگ نوشته شوند (بدون نقطه) و قبل از هر کدام یک فاصله وجود داشته باشد. فرض کنید این زمان‌ها معرف زمان‌های واقعی هستند، نه لزوماً ساعت 9 صبح و 5 بعد از ظهر.

- 9:00 AM to 5:00 PM
- 9 AM to 5 PM

اگر ورودی داده شده به تابع `convert`، در هیچ یک از این قالب‌ها نبود یا زمان داده شده نامعتبر بود (مثلاً 12:60 AM، 13:00 PM و غیره) یک ارور `ValueError` را نشان دهید. اما فرض نکنید که ساعات کاری یک نفر حتما قبل از ظهر شروع می‌شود و بعد از آن به پایان می‌رسد. ممکن است شخصی تا دیر وقت و حتی ساعات طولانی کار کند (مثلاً از ساعت 5 بعد از ظهر تا 9 صبح).

ساختار working.py را به صورت زیر بسازید. می‌توانید توابع main را تغییر دهید و یا توابع دیگری را پیاده سازی کنید، اما نمی‌توانید هیچ کتابخانه دیگری را import کنید. با اینکه اجباری نیست ولی میتوانید از کتابخانه های re و یا sys استفاده کنید.

```
import re
import sys

def main():
    print(convert(input("Hours: ")))

def convert(s):
    ...
...

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

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

```
pytest test_working.py
```

- ماژول `re` چندین تابع مانند search 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 اگر `re.search`، از یک الگوی capturing groups (به عنوان مثال، پرانتز) عبور کند، یک object match برمی‌گرداند که در آن matchها عبارتند از 1-indexed، که می‌توانید بصورت جداگانه با group، یا بصورت جمعی با groups نشان دهید.
- یادتان باشد می‌توانید یک `int` را با کدهایی مانند زیر که در آن n یک رقم است و پیشوند آن 0 است فرمت کنید:

```
print(f"{n:02}")
```

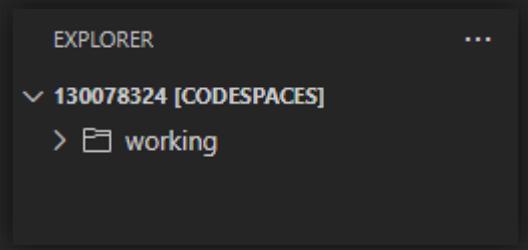
6 قبل از شروع توجه کنید:

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 working در codespace ایجاد شود:

```
$ mkdir working
```



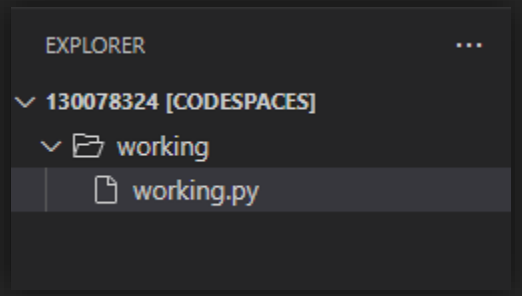
7 قبل از شروع توجه کنید:

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

```
$ cd working
working/ $
```

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

```
working/ $ code working.py
```



اکنون می‌توانید در این فایل برنامه‌ی خود را بنویسید.

اکنون می‌خواهیم برنامه‌ی نوشته شده‌ی خود را تست کنیم:

- دستور `python working.py` را اجرا کنید. مطمئن شوید که برنامه از شما یک زمان ورودی می‌گیرد. 9 AM to 5 PM را تایپ کنید و کلید Enter را فشار دهید. برنامه شما باید 09:00 to 17:00 را به عنوان خروجی برگرداند.

- دستور `python working.py` را اجرا کنید. 10 PM to 8 AM را تایپ کنید و کلید Enter را فشار دهید. برنامه شما باید 22:00 to 08:00 را به عنوان خروجی برگرداند.

- دستور `python working.py` را اجرا کنید. 10:30 PM to 8:50 AM را تایپ کنید و کلید Enter را فشار دهید. برنامه شما باید دوباره 22:30 to 08:50 را به عنوان خروجی برگرداند

اکنون می‌خواهیم برنامه‌ی نوشته شده‌ی خود را تست کنیم:

- دستور `python working.py` را اجرا کنید. سعی کنید عمداً یک `ValueError` را با تایپ 9:60 AM to 5:60 PM و سپس با فشار دادن کلید Enter ایجاد کنید. برنامه شما باید یک `ValueError` نشان دهد.

- دستور `python working.py` را اجرا کنید. سعی کنید عمداً یک `ValueError` را با تایپ 9 AM - 5 PM و سپس با فشار دادن کلید Enter ایجاد کنید. برنامه شما باید یک `ValueError` نشان دهد.

- دستور `python working.py` را اجرا کنید. سعی کنید عمداً یک `ValueError` را با تایپ 09:00 AM - 17:00 PM و سپس با فشار دادن کلید Enter ایجاد کنید. برنامه شما باید یک `ValueError` نشان دهد.

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

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

- تابع `convert` را در نسخه صحیح working.py تغییر دهید. برای مثال برنامه شما ممکن است، نتواند `ValueError` را در زمانی که باید، نشان دهد. تست‌های خود را با `pytest test_working.py` اجرا کنید. pytest باید نشان دهد که حداقل یکی از تست های شما ناموفق بوده است.

- دوباره تابع `convert` را در نسخه صحیح working.py تغییر دهید. برای مثال، ممکن است برنامه شما به اشتباه دقیقه‌ها را حذف کند. تست‌های خود را با `pytest test_working.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/working
```

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/working
```

CS50x Iran

Harvard's Computer Science 50x Iran