



Lines of Code

Problem set



یکی از راه‌های اندازه‌گیری پیچیدگی یک برنامه، شمارش تعداد خطوط کد آن (LOC)، به استثنای خطوط خالی و کامنت است. به عنوان مثال، برنامه ای مانند زیر:

```
# Say hello

name = input("What's your name? ")
print(f"hello, {name}")
```

این قطعه کد فقط دارای 2 خط هست و نه 4 خط. چون خط اول کامنت و خط بعدی فقط یک خط خالی هست و هیچ کدام حساب نمی‌شوند.

البته که صرفاً، تعداد خط‌های یک برنامه نمیتواند تنها پارامتر برای تشخیص پیچیدگی یک برنامه باشد.

برای مثال تابع زیر را در نظر بگیرید.

```
def is_even(n):  
    if n % 2 == 0:  
        return True  
    else:  
        return False
```

با وجود اینکه تعداد خط تابع قبل دو برابر تابع زیر است اما پیچیدگی تابع قبل دو برابر پیچیدگی این تابع نیست :

```
def is_even(n):  
    return n % 2 == 0
```

درواقع تابع اول، حتی قابل درک تر هم هست.

بنابراین بهتر است تعداد خطوط یک برنامه را طبق قاعده [grain of salt](#) در نظر بگیریم.

با این حال در فایلی به نام `lines.py` برنامه ای بنویسید که تنها و دقیقا یک پارامتر `command-line` دریافت کند که آدرس یک فایل پایتون باشد. سپس تعداد خط های آن فایل پایتون رو بدون در نظر گرفتن کامنت ها و خط های خالی به عنوان خروجی برگرداند.

اگر کاربر پارامتری وارد نکرد یا آدرس فایلی را وارد کرد که فایل پایتون نبود و یا آدرس فایلی را وارد کرد که وجود نداشت، با `sys.exit` برنامه خارج شود.

- این طور در نظر بگیرید که یک کامنت با `#` شروع میشود و یا ممکن است کامنت از نوع `docstring` نیز باشد که باید به عنوان یک کامنت در نظر گرفته شود.
- همچنین خط هایی که خالی هستند نباید به عنوان یک خط شمرده شوند.

▼ نکته

- همانطور که اطلاع دارید نوع داده `str` شامل چندین متد مانند `lstrip` و `startswith` میباشد که میتواند در روند نوشتن برنامه شما مفید واقع شود.
- برای اطلاع بیشتر از این متد ها میتوانید به این [آدرس](#) مراجعه کنید.
- به این نکته دقت کنید که تابع `open` میتواند اروری با عنوان `FileNotFoundError` برگرداند.
- برای مطالعه بیشتر در مورد این ارور به این [آدرس](#) مراجعه کنید .
- همچنین شما میتوانید برنامه خود را با استفاده از سورس کد [هفته 6](#) تست کنید.

```
$ python lines.py
Too few command-line arguments
$ python lines.py foo
Not a Python file
$ python lines.py foo bar
Too many command-line arguments
$ python lines.py foo.py
File does not exist
$ python lines.py hello.py
2
$
```

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 `lines` در `codespace` ایجاد شود:

```
$ mkdir lines
```

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

```
$ cd lines  
lines/ $
```

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

```
lines/ $ code lines.py
```

اکنون می‌توانید در این فایل برنامه‌ی خود را بنویسید.

برای تست کد خود مراحل زیر را انجام دهید:

- برنامه خود را با `python lines.py` اجرا کنید. برنامه شما باید با `sys.exit` و با متن ارور `Too few command-line arguments` خارج شود.

```
Too few command-line arguments
```

- دو برنامه پایتون با نام های `hello.py` و `goodbye.py` ایجاد کنید. برنامه را با `python lines.py` اجرا کنید. برنامه شما باید با `sys.exit` همراه با متن ارور `Too many command-line arguments` خارج شود.

```
Too many command-line arguments
```

- یک فایل تکست با نام `invalid_extention.txt` ایجاد کنید. سپس برنامه خود را با دستور `python lines.py invalid_extension.txt` اجرا کنید. برنامه شما باید با `sys.exit` و متن ارور `Not a Python file` خارج شود.

```
Not a Python file
```

- برنامه خود را با دستور `python lines.py non_existent_file.py` با فرض اینکه این فایل وجود ندارد اجرا کنید. برنامه شما باید با `sys.exit` و متن ارور `File does not exist` از برنامه خارج شود.

```
File does not exist
```

- چند فایل پایتون که شامل کامنت‌های معمولی، کامنت از نوع `docstring`، خط‌های خالی و یکسری خط کد باشد ایجاد کنید و آن‌ها را به این شکل اجرا کنید. `python lines.py FILENAME` که `FILENAME` اسم فایل شما خواهد بود. برنامه شما در خروجی باید تعداد خط‌های برنامه را برگرداند.

❖ 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/lines
```

لبخند های سبز :) به این معنی هستند که برنامه‌ی شما در تست **قبول شده** و اخم های قرمز :) یعنی برنامه‌ی شما **دارای ایراد** هست، همچنین پیام های زرد ! بدین معنی است که برای اینکه check50 بتواند این بخش برنامه شما را بررسی کند باید ابتدا ارورهایی که با اخم های قرمز :) جلوتر آمده اند را برطرف کنید.

. برای دیدن جزئیات بیشتر درباره این تست که توسط check50 صورت گرفت، وارد لینکی که به شما خروجی داده است شوید.

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

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

توجه کنید هرچند بار که لازم باشد میتوانید کد خود را اصلاح و ارسال کنید ، نمره آخرین پاسخ شما در نظر گرفته میشود .

CS50x Iran

Harvard's Computer Science 50x Iran

