تولیدکننده‌ی شطرنجی

می‌خواهیم تابعی بنویسیم که بر روی تولیدکننده‌ها (`generator`) پیمایش کند و به آن‌ها اطلاعاتی ارسال کند.

متاسفانه کریم اینترنتش قطع شده و با میوه‌هایش هم نتوانست مهره‌های شطرنج را بسازد و تصمیم گرفت بخوابد تا شاید زمان بگذرد. کریم خواب دید که مسئول یک مسابقه‌ی برنامه‌نویسی شده است.

او از شرکت‌کنندگان خواسته یک تولیدکننده یا `generator` بنویسند که عدد مطلوبی را حدس بزند؛ این تولیدکننده وضعیت (بزرگ‌تر، کوچک‌تر یا مساوی بودن) عدد حدسی، نسبت به عدد مطلوب را از پیمایش‌کننده خود دریافت می‌کند.

کریم به عنوان مسئول مسابقه باید این تولیدکننده‌ها را بررسی کند، اما چون توانایی برنامه‌نویسی در کریم مشاهده نمی‌شود، از شما کمک می‌خواهد که تابعی به عنوان پیمایش‌کننده برای بررسی این تولیدکننده‌ها بنویسید.

پروژه اولیه

پروژه اولیه را از اینجا دانلود کنید.

ساختار فایل‌های این پروژه به صورت زیر است.

```
guess-generator
├── generators.py
└── source.py
```

جزئیات

تولیدکننده‌ها انتظار دارند مقادیر زیر را از پیمایش‌کننده خود دریافت کنند:

- `G` : عدد حدسی از عدد مطلوب بزرگ‌تر باشد.
- `L` : عدد حدسی از عدد مطلوب کوچک‌تر باشد.
- `E` : عدد حدسی با عدد مطلوب برابر باشد.

یک نمونه از تولیدکننده مورد نظر:

```python
import random

def guess_generator(min_value, max_value):
    num = random.randint(min_value, max_value + 1)
    resp = (yield num)
    while resp != 'E':  # Equal
        if resp == 'G':  # Greater
            max_value = num - 1
        elif resp == 'L':  # Less
            min_value = num + 1
```

```
        num = random.randint(min_value, max_value + 1)
        resp = (yield num)
```

شما باید تابعی بنویسید که پارامترهای زیر را دریافت کند:

- `guess_generator` : تولیدکننده‌ای که باید پیمایش شود.
- `min_value` : حداقل مقدار برای حدس
- `max_value` : حداکثر مقدار برای حدس
- `assumed_number` : عدد فرضی برای حدس

و در خروجی یک لیست با شرایط زیر برگرداند:

- لیست شامل تمام اعداد حدس زده شده توسط تولیدکننده به ترتیب پیمایش باشد.
- بعد از هر حدس عجیب یک عضو `'!'` باشد.

یک حدس عجیب است اگر یکی از شرایط زیر را داشته باشد:

- اگر بعد از دریافت `G` از طرف پیمایش‌گر عددی بزرگتر یا مساوی حدس قبلی حدس زده شود، این یک حدس عجیب است!
- اگر بعد از دریافت `L` از طرف پیمایش‌گر عددی کوچکتر یا مساوی حدس قبلی حدس زده شود، این یک حدس عجیب است!
- اگر بعد از دریافت `E` از طرف پیمایش‌گر حدس زدن عدد ادامه پیدا کند، این یک حدس عجیب است!
- اگر عدد حدسی کمتر از `min_value` یا بیشتر از `max_value` باشد، این یک حدس عجیب است!
- اگر در کل فرایند پیمایش سه مرتبه حدس عجیب اتفاق افتاد، یک عضو `'!!!'` به لیست خروجی اضافه شود و پیمایش متوقف شود.

برای مثال:

```
import generators
from source import guess_generator_iterator

gen = generators.guess_generator_1_correct
min_value, max_value, num = 1, 100, 50
lst = guess_generator_iterator(gen, min_value, max_value, num)
print(lst)

gen = generators.guess_generator_2_lazy
min_value, max_value, num = 1, 10, 7
lst = guess_generator_iterator(gen, min_value, max_value, num)
print(lst)

gen = generators.guess_generator_3_careless
min_value, max_value, num = 1, 30, 15
lst = guess_generator_iterator(gen, min_value, max_value, num)
print(lst)

gen = generators.guess_generator_4_stupid
min_value, max_value, num = 1, 10, 5
```

```
lst = guess_generator_iterator(gen, min_value, max_value, num)
print(lst)

```

خروجی نمونه بالا:

```
[5, 15, 74, 71, 55, 49, 50]
[1, 2, 3, 4, 5, 6, 7]
[22, 2, 8, 9, 20, 18, 11, 15, 16, '!', 14, '!', 14, '!', '!!!']
[4, 3, '!', 2, '!', 1, '!', '!!!']
```

در این مثال تولیدکننده‌های اول و دوم می‌توانند به درستی عدد خواسته شده را حدس بزنند و هیچ حدس عجیبی هم نداریم؛ بنابراین حدس‌های تولیدکننده به ترتیب داخل یک لیست خروجی داده می‌شود. در دو مثال بعدی هر کدام از تولیدکننده‌ها سه حدس عجیب دارند که بعد از هر کدام از آن حدس‌ها یک `'!'` به لیست اضافه می‌شود و در آخر هم یک `'!!!'` به لیست اضافه می‌شود و حاصل برگردانده می‌شود.

نکات

- شما تنها مجاز به تغییر فایل `source.py` و تکمیل تابع `guess_generator_iterator` هستید.
- برای فرستادن کد صرفا فایل `source.py` را با فرمت `zip` فشرده کرده و بفرستید.
- می‌توانید کد تست نمونه را با استفاده از این لینک دانلود کنید.

قسمت آموزشی

در این قسمت راهنمایی‌های سوال به ترتیب در روزهای شنبه، دوشنبه و چهارشنبه ساعت ۱۸ اضافه می‌شود. مشکلات‌تان در راستای حل سوال را می‌توانید از بخش "سوال بپرسید" مطرح کنید.

▼ راهنمایی ۱

در ابتدا برای حل این سوال شرط‌های گفته شده در سوال برای «حدس فرضی» داده شده رو بررسی کنید و حدس‌های بد و خوب رو مشخص و از هم جدا کنید.

▼ راهنمایی ۲

نتیجه پیمایش تولیدکننده ورودی را درون یک متغیر قرار دهید. خروجی `yield` در تولیدکننده برابر مقداری است که شما از طریق تابع `send` متغیر ایجاد شده به تولیدکننده می‌فرستید. همین‌طور با استفاده از `try` و `except` می‌توانید پایان یافتن پیمایش تولیدکننده را مدیریت کنید.

در کل این سئوال به توانایی خواندن داکیومنت پایتون هم ربط دارد، پیشنهاد می‌کنیم برای یادگیری بهتر حتما لینک اول سوال را بخوانید.

▼ راهنمایی ۳

```python
numebr = 1

def myGenerator():
    global numebr
    while True:
        status = (yield numebr)
        if status == 'double':
            numebr *= 2
        else:
            numebr += 1

def myFunction():
    myGen = myGenerator()
    current = next(myGen)
    while True:
        try:
            print(current)
            if current > 100:
                myGen.close()
                break
            elif current%10 == 0:
                current = myGen.send('double')
            else:
                current = myGen.send('single')
        except:
            break

myFunction()
```

خروجی کد بالا به صورت زیر می‌باشد:

```
1
2
3
4
5
6
7
8
9
10
20
40
80
160
```