به نام خدا

اعضای تیم : امیر محمد محمدی / سپهر ترابی / سارا عظیمی

تابع زیر (که در تصویر مشخص است) وظیفه ساخت یه رشته رندوم را دارد :

```
def my_string(set_len):
```

یک کلاس main داریم که دارای متد های test_int و test_string و test_float و create_test_file و run_test است.

Test_int: وظیفه ی این متد ساختن تعدادی test case به صورت اعداد صحیح random است و testcase های ساخته شده را در یک فایل test_int.txt ذخیره می نماید.

```
def test_int(self,length,cls):
```

Test_string: وظیفه ی این متد ساختن تعدادی test case به صورت رشته‌های random است و testcase های ساخته شده را در یک فایل test_string.txt ذخیره می نماید.

```
def test_string(self,length,cls):
```

Test_float: وظیفه ی این متد ساختن تعدادی test case به صورت اعداد اعشاری random است و testcase های ساخته شده را در یک فایل test_float.txt ذخیره می نماید.

```
def test_float(self,length,cls):
```

Create_test_file: test case های تولید شده را در یک فایل به نام test.py قرار می‌دهد.

```
def create_test_file(self,class_name):
```

Run_test: فایل test.py را اجرا کرده و خروجی آن را در فایل output.txt ذخیره میکند

```
def run_test(self):
```

یک فایل stack.py داریم که برای مثال در آن یک کلاس Stack میباشد . (ولی شما میتوانید هرکلاس دلخواهی را در آنجا قرار دهید).

بخش طراحی و ساخت تست کیس تصادفی و تست و اجرا را (امیر محمد محمدی / سپهر ترابی) زده ایم و بخش coverage test بعهده خانم عظیمی بود.

برای انجام coverage testing، تابع هایی را در فایل coverage_test.py تعریف کردیم که به شرح زیر است :

- تابع run_test(file_path) که به عنوان ورودی یک فایل می گیرد، فایل داده شده را با استفاده از coverage.py که از library های پایتون است، اجرا می کند. به این صورت که:

```
def run_test(file_path):
    call("./venv/bin/coverage run %s" % file_path, shell=True)
    call("./venv/bin/coverage html", shell=True)
```

با استفاده از اولین call، coverage test اجرا می شود و با استفاده از دومین call، فایل html آن تولید می شود.

- تابع hello() که این تابع نیز برای اجرای coverage test است اما دیگر فایل را به عنوان ورودی نمی گیرد، بلکه از روی url وارد شده می تواند مکان فایل را تشخیص دهد، و در نهایت تابع run_test را با استفاده از file path به دست آمده صدا می زند.
- تابع run_server(file_path) که این تابع نیز آدرس فایل را به عنوان ورودی می گیرد و آن را روی سرور اجرا می کند (host='0.0.0.0',port=8080).
- تابع output(file_path) مانند همان تابع قبل است، با این تفاوت که نتیجه را داخل ترمینال نمایش می دهد.

برای اجرای برنامه، command زیر را اجرا می کنیم:

Python coverage_test run_server test.py

که coverage_test نام فایلی است که توضیح داده شد، run_server یکی از توابع داخل آن است (که می توان به جای آن از output نیز استفاده کرد) و test.py نام فایلی است که تست کیس ها در آن وجود دارد. Command می تواند به صورت زیر نیز باشد:

Python coverage_test output test.py

که در این صورت، نتیجه داخل ترمینال چاپ می شود.

نمونه ای از برنامهٔ اجرا شده بدین صورت است:

Coverage report: 66%

Module ↓	statements	missing	excluded	coverage
mymath.py	12	6	0	50%
mymath_test.py	17	4	0	76%
Total	**29**	**10**	**0**	**66%**

coverage.py v4.5.1, created at 2018-07-10 07:55