Pytuto (Python Tutorial) 7th assignment.

Hyungwon Yang
Hyung8758@gmail.com
EMCS Lab

This task was designed for those who just started python. I wish them try to complete the task and improve their python skill. Please search as many solutions as you can through the internet and ask questions to me and your teammates to tackle down the tasks. However, DO NOT share codes. Good luck.

Introduction

Debugging

Have you ever counted how many times you ran the script before you finish it? The longer the script, the more time you would spend for running it for sure. However, the length of the script is not important, but reducing the time of checking your script in the middle of your scripting is the main concern. Why? Because checking less means 1) you save your time, 2) you prove yourself that you are good at scripting, and finally 3) you are adept in debugging. To be honest, saving time is dependent on reason 2 and 3 because it is obviously consequential. When it comes to good at scripting, it can be achieved by many ways such as searching proper scripts and combining them with a little manipulative skill. Furthermore, if you have been doing one or two computer languages, no matter what languages they might be, you must be good at scripting or coding already. Therefore, two reasons (you may say that the first reason is not even a reason) above are not that critical for reducing the times of checking the script. However, debugging the script is not similar to the previous reasons because it requires a lot of scripting experiences, acute skills to find even very minor problems that may cause a whole script unworkable. In this reason, equipped proficiency in debugging is the most important qualification for programmers who can incredibly decrease the amount of time for checking scripts.

Q. What is debugging exactly?

Debugging is the procedure of detecting the problematic lines in the scripts that cause the errors or unintended results. There are a number of errors in the programming languages, but usually syntax-related errors can be easily captured by IDE debugging process. However, result-related errors are very tricky and demanding to figure out because they are undetectable by debugging program.

Q. What IDE provides with good debugging process?

It strongly depends on the users' preference. I believe each of python IDE contains powerful debugging tools in order to improve the users' scripting performance.

However, I recommend Pycharm which is python IDE providing many advanced features because it supports the state-of-the-art debugging tool. (it is similar to Matlab's tool) It allows users to break the lines to access the procedure of the script process. Variable values are presented in parallel with the variable names in the same lines. Moreover, in the middle of the debugging procedure, users can pause the process for calculating attained values or visualizing them.
Recommend Video: https://www.youtube.com/watch?v=QJtWxm12Eo0

Task

A) Python Debugging: syntax-related

A-1) Sean: Hi programmer, I made some python scripts and it caused some error messages. I tried my best to figure out what the problem was. Unfortunately, all my efforts were turned out to be nothing. Would you run the script and find out the problem for me? Thanks in advance.

Request) Check this script: 'Sean_script_1.py' and solve the problem and fix it. You should comment out the line that causes the error and insert new lines below it. Please leave comments inside the script. (Compare the result with the answer result: 'Answer_Sean.png')

A-2) Jack: I just heard that Sean asked you for help related to python script errors. Could you check my one too? I desperately need your help! Thanks millions.

Request) Check this script: 'Jack_script_1.py' and solve the problem and fix it. You should comment out the line that causes the error and insert new lines below it. Please leave comments inside the script. (Compare the result with the answer result: 'Answer_Jack.png')

B) Python Debugging: result-related

B-1) Yvonne: Hello there, I ran the script and it didn't print any error message. So I thought that my script was written correctly. However, when I checked the result by running the script, I realized that I did something wrong in the script because result wasn't what I intended. Would you look through the script for me? I would be really appreciated if you could help me. Thanks!

Request) Check this script: 'Yvonne_script_1.py' and solve the problem and fix it. You should comment out the line that causes the error and insert new lines below it. Please leave comments inside the script. (Compare the result with the answer result: 'Answer_Yvonne.png')

If you finish your assignment, please send the code to 'hyung8758@gmail.com'. The script name should be '?th_Assignment_YourName.py'. In the code script, the assignment number (e.g, 1st assignment), your name and email address should be written in the first line. Ask questions and give comments to me. It is always welcome.

Weekly Tips

* 예외처리 : 개발자의 실수 혹은 사용자의 실수등 여러가지 예상치 못한 에러가 발생하게 되는 데, 이때 이러한 에러에 대응하여 프로그램의 자연스러운 흐름을 유도하는 것이 필요하다.

- 구문에러(syntax error) : 개발자가 새로운 언어를 배울 때 가장 많이 일으키는 실수이다. if 혹은 for 문의 끝에 :를 생략했다던가 혹은 print("~~~) 처럼 마지막 글자에 "를 생략하는 경우가 대표적이다.

- try 를 통한 예외 처리

```
def divide(a,b):
    return a/b
try:

    c = divide(5,2)
except ZeroDivisionError<예외종류>:
    print("It should not be 0")<예외 처리 문장>
except TypeError<예외종류>:
    print("it should not be a string")<예외 처리 문장>
else: <- 예외가 발생하지 않는 경우
    print("Result: {0}".format(c))
finally: <- 예외와 상관없이 최종적으로 수행
    print("The process has been finished")
```

- 다음과 같은 방법으로도 except 를 사용할 수 있다.

except TypeError as e 타입에러를 e 로 받아서 사용한다.

except (ZeroDivisionError,OverflowError) 다수의 에러를 묶어서 모두 처리하게끔 할 수 있다.

- raise 구문 : 프로그래머가 의도적으로 예외를 발생시켜야 할 경우도 있다. 이때 raise 를 사용한다. 아래 예문은 양수로만 나누기를 하는 펑션이며 사용자가 음수값을 입력할 경우 자동적으로 에러를 일으키도록 raise 구문을 사용한 예이다.

```
def PositiveDivide(a,b):
    if b < 0:
        raise NegativeDivisionError(b)
    return a / b
```