TeamLab/lab_for_gachon_cs50

Lab #2 - 숙제 제출 연습하기

Copyright 2015 © document created by TeamLab.Gachon@gmail.com

Introduction

Gachon CS50 강의에서는 모든 숙제를 TeamLab에서 개발된 자동 채점 시스템(Gachon Autograder)를 통해 제출하게 된다. 본 실습에서는 자동 채점 시스템을 사용하여 숙제를 제출하는 과정을 연습한다. 아직 파이썬의 주요 내용을 배우지 않은 상태이기 때문에 매우 간단한 사칙연산 코드를 작성한다.

숙제 제출을 위한 submit_assignment.py 다운로드

첫 번째 할 일을 숙제 제출용 프로그램인 "submit_assignment.py"을 다운로드하는 것이다. Mac OS 또는 CS50 서버를 사용중인 수강생은 아래 코드를 서버에 로그인 한 후 입력하면 숙제 파일이 자동으로 다운로드 된다. 주소가 너무 길다고 걱정할 필요가 없다. 마우스를 사용하여 copy&paste를 하면 console[1]창에 붙여넣기가 가능하다[2].

```
wget https://raw.githubusercontent.com/TeamLab/lab_for_gachon_cs50/master/submit_assignment.py
```

다운로드 후 `ls submit_assignment.py` 이라고 입력하면 `submit_assignment.py`이라고 출력될 것이다.

숙제 template 파일 다운로드

두 번째로 할일은 `submit_assignment.py` 파일로 우리가 숙제를 수행할 template 파일을 다운로드 하는 것이다. template 파일은 숙제 수행시 기본적으로 제공되는 코드 초안을 의미한다. 본 수업에서 사용되는 "Gachon Autograder" 프로그램은 template 파일에서 수강생이 핵심이 되는 부분만 수정하여 제출하면, 자동으로 제출 유무의 확인과 오류를 점검해준다. 이를 사용하기 위해서는 먼저 숙제 template 파일을 다운로드해야 한다. 숙제 template 파일을 다운로드 하는 명령어는 `python3.4 submit_assignment.py -get <lab_assignment_name>` 이다. 이미 익숙해졌겠지만 `python3.4 submit_assignment.py`은 `submit_assignment.py` 프로그램을 파이썬3.4 인터프리터로 실행시키는 것을 의미하며, `-get`은 숙제 template 파일 다운로드를, `<lab_assignment_name>`는 다운로드 대상이 되는 숙제 이름으로 각 숙제마다 바뀌게 된다. 우리는 첫 번째 테스트 숙제로 `arithmetic_function.py` template 파일을 다운로드하여 수정하도록 하겠다. 이를 위한 명령어는 아래와 같다.

```
python3.4 submit_assignment.py -get test
```

위 명령어를 입력 하면, 아래와 같은 내용이 뜨면서 Login ID와 Password를 물어보게 될 것이다. cs50.gachon.ac.kr 웹 페이지에 가입시 사용했던 email주소와 비밀번호를 입력하면. `arithmetic_function.py` 파일이 성공적으로 다운로드 됐다는 메세지를 확인하게 될 것이다.

```
== Getting templates | test
Login ID:
Password :
arithmetic_function.py file is created for your test assignment
Thank you for using the program. Enjoy Your Assignment - From TeamLab
```

arithmetic_function.py 내용 보기

다음으로 다운로드 된 `arithmetic_function.py` 파일의 구조를 파악해 보자. 구조 파악을 위해서는 vi 에디터로 해당 파일을 열어야 하는데 콘솔창에서 `vi arithmetic_function.py`라고 입력하면 수정할 수 있게 된다. 모든 숙제 template 파일은 동일하게 아래 세 가지 함수[4]로 나뉘게 된다.

분류	의미
test 함수	숙제를 실제 수행해야 할 함수로, 함수내에 `Modify codes below` 라는 문구가 작성되어 있음, 사용자는 해당 문구내 부분을 수정하여야 함
helper 함수	숙제 진행을 원활히 돕기 위해 출제자가 작성한 함수, 따로 수정이 필요없는 함수로 수정할 경우, 숙제 제출이 불가능할 수도 있음
main 함수	template 제일 하단에 있는 숙제 test용 코드로 `def main():` 안에 작성됨, 해당 코드는 실제 코드가 바르게 작성되었을 경우, 실행 결과에 대해서 상세히 기술되어 있음

`arithmetic_function.py` 의 경우 아래 4가지의 test 함수와 하나의 main 함수로 구성되어 있다.

- addition(a, b) - a,b 두 개의 숫자가 입력될 경우 두 수의 합을 반환함
- minus(a, b) - a,b 두 개의 숫자가 입력될 경우 a에서 b를 뺀 값을 반환함
- multiplication(a, b) - a,b 두 개의 숫자가 입력될 경우 두 수의 곱을 반환함
- division(a, b) - a,b 두 개의 숫자가 입력될 경우 b로 a를 나눈 값을 반환함

addition 함수의 경우, 아래와 같이 작성되어 있다.

```
def addition(a, b):
    # '''
    # Input:
    #   -a: 실수 값 (Integer of float)
    #   -b: 실수 값 (Integer of float)
    # Output:
    #   -두 값의 합
    # Examples:
    #   >>> addition(3,5)
    #   8
    #   >>> addition(3,2)
    #   5
    # '''
    # pass
    # ===Modify codes below=============
```

```
    result = None

    # ================================

    return result
```

위 함수에서 `# ===Modify codes below==========` 윗 부분까지가 함수에 대한 간략한 설명이다. `# Input:` 부분은 이 함수에 입력되는 값의 유형에 대해 설명하고, `# Output:` 부분은 이 함수의 결과값에 대해 서술한다. 중요한 부분은 `# Examples:` 이다. 해당 부분 하단에는 이 함수를 실제 수행하였을 때 나올 수 있는 결과값에 대해서 작성되어 있다. 이를 실제 확인하게 위해서는 파이썬 쉘을 실행 시켜야 한다. 파이썬 쉘은 `python3.4` 라고 치면 나오는 프로그래밍 환경을 의미한다. 파이썬 쉘을 실행후 아래와 `>>>` 이후에 있는 코드를 입력해 보자

```
YOUR_ID@cs50:~$ python3.4
Python 3.4.0 (default, Jun 19 2015, 14:20:21)
[GCC 4.8.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import arithmetic_function as af
>>> result = af.addition(10,5)
>>> print (result)
None
```

첫 번째 `import` 문은 작성된 프로그램을 호출하는 명령어이다. 우리가 현재 작성한 프로그램 파일이름은 "arithmetic_function.py"로 여기서 확장자인 .py를 제외하고 입력한다. 뒤에 붙은 `as af` 의미는 `arithmetic_function`을 `af`라는 이름으로 부르겠다는 의미이다. 두 번째 코드는 `arithmetic_function` 코드에 있는 `addition` 이라는 함수를 부른다는 의미이다. 해당 함수는 입력 값으로 a와 b를 받아야 되는데 여기서는 a와 b과 각각 10과 5로 매칭된다. `result = af.addition(10,5)`함수의 결과값이 `result`라는 이름의 변수로 저장된 다는 의미이다. 마지막 코드는 `result`의 결과를 출력하라는 뜻이다. 현재까지 코드를 수정된 것이 없기 때문에 현재 결과 값은 `None`으로 설정되어 있다.

arithmetic_function.py 수정 하기

실제 수강자가 고쳐야할 부분은 아래 코드 부분이다. 본 함수의 목적이 입력된 두 값의 덧셈이므로 `result = None`을 `result = a+b`로 수정해 주면 된다. 나머지 함수들도 각 함수의 목적에 맞게 `result = None`을 부분을 수정해주자.

```
   # ===Modify codes below=============

    result = None

    # ================================
```

제대로 수정되었는지를 확인하기 위해서는 수정된 프로그램을 실행시켜 보자. `python3.4 arithmetic_function.py`입력해주면 프로그램이 실행된다. 이때 실행되는 코드는 `main()` 함수안에 있는 아래

코드이다. `print` 문은 괄호 안에 있는 값을 화면에 출력하는 명령어로 아래 코드는 먼저 `Addition Test` 라는 문장을 출력하고 다음으로 `addition(3,5)`의 결과를 출력한다. 각각의 출력되는 기대 값은 `#` 기호 옆 주석[5]으로 작성되어 있다. 숙제 진행시 수강자가 수정한 프로그램을 실행하면서 나오는 결과값들이 `main()` 함수 내의 주석 처리된 기댓값들과 같은지 확인하면 된다. `main()` 함수는 숙제 제출에 영향을 주지 않는 함수로 사용자가 원할 경우 자유로운 수정이 가능하다.

```
def main():
    print ("Addition Test")
    print (addition(3,5)) # Expected Result: 8
    print (addition(10,5) == 15) # Expected Result: True
    print ("Addition Test Closed ₩n")


    print ("Minus Test")
    print (minus(3,5)) # Expected Result: -2
    print (minus(10,5) == 5) # Expected Result: True
    print (minus(10,15) == 5) # Expected Result: False
    print ("Minus Test Closed ₩n")

    print ("Multiplication Test")
    print (multiplication(3,5)) # Expected Result: 15
    print (multiplication(10,5) == 50) # Expected Result: True
    print (multiplication(10,-3) == 20) # Expected Result: False
    print ("Multiplication Test Closed ₩n")

    print ("division Test")
    print (division(5,5)) # Expected Result: 1
    print (division(5,4)) # Expected Result: 1.25
    print (division(10,5) == 2) # Expected Result: True
    print (division(10,-3) == 0.33333) # Expected Result: False
    print ("division Test Closed ₩n")
```

arithmetic_function.py 제출 하기

수정이 완료되었을 경우 숙제를 다시 서버에 제출해야 한다. 숙제 제출을 위한 명령어는 아래와 같다.

```
python3.4 submit_assignment.py -submit arithmetic_function.py
```

본 명령을 실행하여 프로그램의 문법상 에러가 없을 경우, 아래와 같은 형태로 숙제 제출 확인 메세지를 받게 된다.

```
-------------------- | ---------- | --------------------
       Function Name |    Passed? |             Feedback
-------------------- | ---------- | --------------------
            division |       PASS |             Good Job
               minus |       PASS |             Good Job
```

```
      multiplication |       PASS |             Good Job
            addition |       PASS |             Good Job
-------------------- | ---------- | --------------------
```

첫 번째 Lab Assignment인 vimrc 제출을 위해서도 아래와 같이 명령어를 입력하자. vimrc 숙제는 .vimrc의 생성과 필요한 설정들이 제대로 입력되었는지 보기 때문에 특별히 다운로드된 "vimrc_test.py" 파일을 수정할 필요는 없다. 만약 숙제 제출에 오류가 발생했다면, 첫 번째 Lab Assignment 문서를 **정독**[6] 하면서 빼먹은 것이 없는지 확인해 보기 바란다. 참고로 에러가 나왔을 경우 반드시 `vi ~/.vimrc` 명령으로 .vimrc 파일을 수정한 후, `python3.4 vimrc_test.py`와 `python3.4 submit_assignment.py -submit vimrc_test.py` 명령을 순차적으로 입력해야 한다.

```
python3.4 submit_assignment.py -get vimrc
python3.4 vimrc_test.py
python3.4 submit_assignment.py -submit vimrc_test.py
```

Next Work

두 번째 Lab Assignment를 무사히 마친 것을 축하한다. 아마 지금쯤이면 수업시간에 욕은 못하겠고[7] 상당히 진이 빠져 있는 상태일 것이다. 아직 우리에게는 12척의 배 대신에 한 개의 assignment가 더 남았다. 물론 집에가도 상관없지만, 지금 집에 간다면 drop을 더 권장한다. 첫 번째 강의 마지막 Assignment로 이동하길 바란다. 할 수 있다. 걱정말자.

> **Human knowledge belongs to the world** - from movie 'Password' -

Footnotes

1: 서버접속후 나오는 검은색 화면을 콘솔창이라고 부른다.↩

2: wget 명령은 파일을 인터넷을 통해 다운로드 하는 프로그램이다. 때에 따라 자신의 컴퓨터에 설치안되어 있을 수도 있으니, 설치하도록 하자. 구글에서 "Mac wget 설치" 또는 "리눅스 wget 설치"로 검색하면 레퍼런스가 많을 것이다. ↩

3: 키보드 왼쪽 하단 alt와 ctrl 사이에 있는 윈도우 로고모양 키를 의미한다.↩

4: 함수란 프로그래밍에서 일반적으로 사용하는 코드 묶음을 의미한다. 현 시점에서 자세히 설명하지는 않으니, 넘어가길 바란다.↩

5: 주석은 프로그램 작성자가 다른 개발자 또는 사용자에게 해당 프로그램을 설명하기 위해 적은 메세지를 의미한다. 주석은 컴퓨터가 해석하여 실행하지 않기 때문에 프로그램 실행에 영향을 주지 않는다.↩

6: 경험상 이해될 때까지 읽다보면 언젠가 이해된다. 모르면 능력 충만한 TA들에게 물어보도록 하자↩

7: 실제 2014년 수강생들은 "I See~" 정도의 욕을 수업시간에 육성으로 터트렸다. 정상이다 걱정말자.↩