Lab #2 - basic_operations 간단한 연산

Introduction

축하합니다! 2주차임에도 불구하고 이 글을 읽고 있다는 것은 여러분들이 포기하지 않고 수업을 들을 의지를 가지고 있다는 것입니다. 그 의지를 마지막까지 유지하시면 좋겠습니다.

이번 Lab은 python의 간단한 연산 문제들을 수행하는 것입니다. 연산이라고 해서 덧셈, 뺄셈과 같은 산수만 생각할 수 있으나, 문자열을 포함하여 파이썬에서 기초 데이터형을 어떻게 다루는지까지 연습하게 됩니다.
이번 차시 부터 Lab Assignment의 설명을 자세하게 하지 않을 것입니다. 이것은 이 수업의 전반적인 추세로 시간이 지나감에 따라 상세한 설명과 가이드는 없어지고 스스로 하는 부분들이 많아집니다. 막상 해보면 할 수 있으니 스스로 해보시길 바랍니다.

backend.ai 설치

숙제를 제출하기 앞서, 레블업의 backend.ai를 여러분의 파이썬에 설치하셔야 합니다. 설치하는 과정은 매우 쉽습니다. 아래처럼 터미널 또는 cmd 창에서 입력을 하시면 됩니다.

```
pip install backend.ai-client
```

숙제 파일(lab_2.zip) 다운로드

먼저 해야 할 일은 숙제 파일을 다운로드 받는 것 입니다. 이미 해보았기 때문에 어렵지 않을 것입니다. Chrome 또는 익스플로러와 같은 웹 브라우저 주소창에 아래 주소를 입력합니다.

https://github.com/TeamLab/Gachon_CS50_Python_KMOOC/blob/master/lab_assignment/lab_2/lab_2.zip

다운로드를 위해 `View Raw` 또는 `Download` 버튼을 클릭합니다. 또는 아래 다운로드 링크를 클릭하면 자동으로 다운로드가 됩니다. Lab 2 - 다운로드 다운로드 된 lab_2.zip 파일을 작업 폴더로 이동한 후 압축해제 후 작업하길 바랍니다. 압축해제 하면 폴더가 `linux_mac`과 `windows`로 나눠져 있습니다. 자신에게 맞는 폴더로 이동해서 코드를 수정해 주시기 바랍니다.

참고로 이제는 작업환경에 대한 용어에 익숙해 지셔야 합니다. 아래는 우리가 주로 사용하는 작업 환경에 대한 설명입니다.

분류	설명
Console	3주차에서 배우게 되겠지만 윈도우에서는 실행 → "cmd"로 들어가는 검은 화면, 우분투와 리눅스에서는 터미널 환경을 말합니다.
Python Shell	Console에서 `python`를 입력했을 나오는 환경으로 파이썬의 다양한 명령어를 실행시킬 수 있습니다
Atom editor	(아톰 에디터) 본 강의의 파이썬 코드 파일은 Atom editor에서 수정되는 것을 기본으로 합니다

수정 해야할 함수 종류들

압축해제 된 `basic_operations.py` 숙제 파일을 살펴봅시다. 코드가 좀 길긴 하지만 기본적인 형태는 lab 1의 `arithmetic_function.py`와 다르지 않습니다. 본 lab에서 수정해야 할 함수들은 아래와 같습니다.

함수	설명
str_to_int	문자열 값을 입력받아 정수형으로 바꾸는 함수

str_to_float	문자열 값을 입력받아 실수형으로 바꾸는 함수
number_to_str	정수형 또는 실수형의 값을 입력받아 문자열 값으로 바꾸는 함수
add_string_number	문자열 값과 숫자형 값을 입력받아, 두 값을 문자열 값으로 연결하는 함수
add_string_string	문자열 값과 문자열 값을 입력받아, 두 값을 문자열 값으로 연결하는 함수
associative_law_add	(a + b) + c 와 같이 숫자형 값을 입력받아 결합 법칙을 이용한 덧셈 결과 값을 반환해주는 함수
associative_law_mutiple	(a *b)* c 와 같이 숫자형 값을 입력받아 결합 법칙을 이용한 곱셈 결과 값을 반환해주는 함수
distributive_law	a * (b + c) 와 같이 숫자형 값을 입력받아 분배 법칙을 이용한 결과 값을 반환해주는 함수
exponent	밑과 승수 값을 입력 받아 지수 계산 결과 값을 반환해주는 함수

하나의 함수를 예로 들면 모든 함수는 아래와 같은 구조로 쓰여져 있습니다.

```
# 데이터 형변환================================================
def str_to_int(string_number):
    # """
    #   Input:
    #   -string_number: 숫자형태의 문자열
    # Output:
    #   -integer: 정수 값
    # Examples:
    #   >>> str_to_int("3")
    #   3
    #   >>> str_to_int("135")
    #   135
    # """
    # ===Modify codes below=================
    result = None

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

    return result
```

함수 제일 상단에 `def str_to_int(string_number)` 에 `def` 는 함수를 선언하는 예약어 이고, `str_to_int` 는 본 함수의 이름입니다. `string_number` 는 본 함수에 입력되는 변수의 이름입니다. 마지막에 붙어 있는 `:` 는 파이썬의 기본 문법으로 `:` 이하로 들여쓰기가 있는 줄까지 본 함수의 영역입니다. 그 아래부터 `# ===Modify codes below=================` 전 까지는 본 함수에 대한 설명으로 Input 변수, Output 변수, 결과 확인을 위한 예시들로 구성되어 있습니다. 예시의 경우 간단히 표시하기 위해 함수명과 Input 값만 적었지만 실제로 `python shell` 에서 실행 시키기 위해서는 아래와 같이 입력해야 합니다. 혹시 헷갈릴까봐 하는 말이지만 `python shell` 로 들어가기 위해서는 `cmd` 에서 `python` 라는 명령어를 입력해 주어야 합니다. 실제 현재 코드를 본 Lab에 목적에 맞게 수정한 후 `python shell` 에서 아래와 같이 입력해봅시다.

```
>>> import basic_operations as bo
>>> bo.str_to_int("3")
3
```

위 코드에서 첫 번째 줄 `import basic_operations as bo` 은 `basic_operations` 이라는 모듈을 부르는 명령어로 `import basic_operations` 모듈을 `bo` 라는 이름으로 사용하겠다는 뜻입니다. 모듈에 대한 설명은 이후에 하겠습니다. 지금은 단지 모듈의 이름은 우리가 작성한 파일의 이름과 동일하다고만 이해하고 있으면 됩니다. 아래줄에 `bo.str_to_int("3")` 는 우리가 작성한 프로그램의 함수를 실제로 실행 시키는 명령입니다. 우리가 `basic_operations` 의 이름을 축약하여 `bo` 로 바꾸었기 때문에 `bo` 라는 이름으로 함수를 실행할 수 있습니다. 만약 첫번째 줄의 명령어에 `as bo` 가 빠져 있다면 `basic_operations.str_to_int("3")` 로 실행할 수 있습니다. 이미 알고 있겠지만 `str_to_int` 는 함수 이름이고, 함수에 정의한대로 입력 값을 넣어야 합니다. 우리가 함수를 선언 할 때, `str_to_int(string_number)` 에서 보듯이 `string_number` 라고 하는 하나의 값만 입력할 수 있도록 지정했기 때문에 테스트 예제에서도 `("3")` 으로 하나의 값만 입력하였습니다. 만약 `("3","4")` 와 같이 다른 값들을 입력하게 되면 에러 메세지를 보게 될 것입니다. 각 함수에 목적에 맞게 9개의 함수를 수정해봅시다.

수정후 테스트 하기

본 lab 숙제를 맞게 작성했는지 확인해보도록 합시다. 참고로 모든 함수를 다 수정한 후 테스트 할 필요는 없습니다. 함수를 수정 때마다 테스트는 가능합니다. 테스트를 위한 기본코드는 `main`함수에 아래와 같이 들어가 있습니다. 상당히 많은 내용이기 때문에 상단에 테스트 코드만 보도록 합시다.

```
def main():
    print("Str_to_int Test")
    print(str_to_int("56"))  # Expected Result: 56
    print("====> ", str_to_int("27") == 27)  # Expected Result: True
    print("Str_to_int Test Closed \n")

    print("str_to_float Test")
    print(str_to_float("8.4501"))  # Expected Result: 8.4501
    print("====> ", str_to_float("3.4") == 3.4)  # Expected Result: True
    print(str_to_float("6.74") == 9.8)  # Expected Result: False
    print("Str_to_float Test Closed \n")
    # 이하 생략
```

각 함수의 테스트 코드 들은 Test 시작과 끝을 알리는 `print` 문 사이에 작성되어 있습니다. 각 테스트에는 `print`문을 사용하여 함수의 결과를 화면에 출력해보게 하고 `====>`뒤에 `True`의 결과값이 나오는지 확인하게 합니다. 아무런 수정없이 코드를 `python basic_operations.py` 명령어로 실행 시키면 아래와 같은 결과 값이 출력될 것입니다. 참고로 실행은 당연히 `console 환경`상에서 해야 합니다.

```
Str_to_int Test
None
====>  False
Str_to_int Test Closed

str_to_float Test
None
====>  False
False
Str_to_float Test Closed
# 이하 생략
```

각 함수를 수정 후 `python basic_operations.py` 명령어로 다시 실행 시키면 아래와 같은 결과를 볼 수 있을 것입니다.

```
Str_to_int Test
56
====>  True
Str_to_int Test Closed

str_to_float Test
8.4501
====>  True
False
Str_to_float Test Closed
# 이하 생략
```

`main` 함수 내에 코드들은 사용자가 마음대로 수정이 가능합니다. 테스트하고 싶은 숫자로 새로 넣거나 필요없는 테스트는 지워도 숙제 제출에는 영향을 주지 않으니 참고하시길 바랍니다. 참고로 해당 코드를 실행 시키지 않기 위해서는 아래와 같이 코드 앞에 `#`을 붙여주시면 됩니다.

```
def main():
    #print("Str_to_int Test")
    #print(str_to_int("56"))  # Expected Result: 56
    #print("====> ", str_to_int("27") == 27)  # Expected Result: True
    #print("Str_to_int Test Closed \n")
```

당연히 `#`을 제거 하면 다시 코드는 정상적으로 실행이 됩니다. `#`은 해당 코드를 실행 시키지 않고 주석(Comment)를 달아 설명해줄 수 있게 하는 예약어입니다. 원래는 코드에 대한 설명을 개발자가 달아줄 때 사용되나 테스트 할 경우, 실행하지 않을 코드에 달아서 테스트를 조정할 수 있습니다. 모든 코드를 다 수정한 후 `python basic_operations.py` 을 입력하면, 총 9번의 `====> True` 가 표시될 것입니다. 하나라도 `====> False`가 있다면 제대로 수정되지 않은 것이니 다시 확인하시길 바랍니다.

숙제 제출하기

아마 첫 번째 차시에서 고생을 하신 분이라면 이번 숙제는 매우 쉽게 수행하였을 것 입니다. 그러나 숙제 제출 방법을 상세히 설명해주는 것은 이번이 마지막입니다. 윈도우 기준으로 숙제 제출을 위해서 하셔야 할 일은 아래와 같습니다.

- `windows` + `r` 를 누르고 cmd 입력 후 확인을 클릭합니다.
- 작업폴더로 경로를 이동합니다.
- cmd 창에서 아래의 코드를 입력합니다.

```
python submit.py
```

위 명령어를 입력 하면, 아래와 같은 내용이 띄면서 Login ID와 Password를 물어보게 될 것입니다. http://theteamlab.io 웹 페이지에 가입시 사용했던 Login ID와 비밀번호를 입력합니다.

```
== Submmting solutions | arithmetic_function.py
Login ID:
Password :
```

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

```
-------------------- | ---------- | --------------------
       Function Name |    Passed? |             Feedback
-------------------- | ---------- | --------------------
   add_string_number |       PASS |             Good Job
   add_string_string |       PASS |             Good Job
associative_law_mutiple |       PASS |             Good Job
       number_to_str |       PASS |             Good Job
          str_to_int |       PASS |             Good Job
            exponent |       PASS |             Good Job
        str_to_float |       PASS |             Good Job
 associative_law_add |       PASS |             Good Job
    distributive_law |       PASS |             Good Job
-------------------- | ---------- | --------------------
```

Next Work

사람에 따라서는 이번 숙제를 정말 쉽게 한 사람도 있을 것입니다. 사실 Lab 1과 기본적인 숙제하는 방식은 다르지 않습니다. 그러나 앞으로 진행될 Lab들은 상당히 어려울 것입니다. 그러나 우리에겐 slack과 TA들이 있습니다.

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