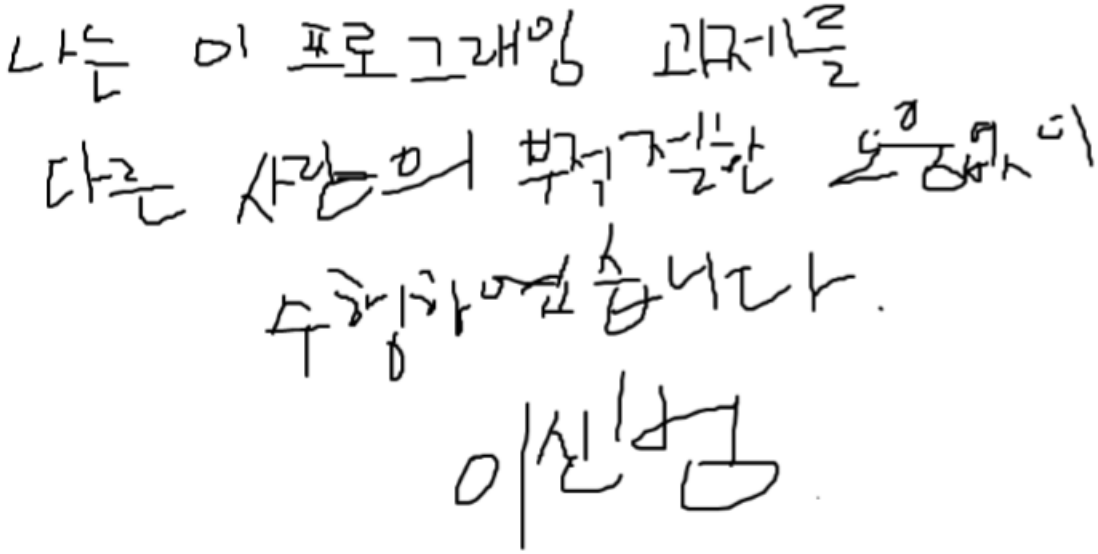
객체 지향 프로그래밍 ASSN5

20180285 이신범

무은재학부

nm160

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.



(1) 개요

어싸인 5는 python의 기본 개념을 잘 이해하고 있는지를 알아보는 문제였다.

Magic method를 이용해 operator를 overloading 하고, python에 존재하는 기초 문법들을 응용하여 계산기를 만드는 문제였다.

1. 어싸인 5를 수행하기 위해 알아야 했던 사항은 다음과 같다.
2. Magic method를 통해 operator를 overloading 할 수 있는가?
3. 예외 처리 : raise Nameerror를 사용할 수 있는가?
4. For문, if문 등 python을 이루는 기초 문법을 다룰 수 있는가?

(2) 사전 지식

1. Magic method는 __init__ 과 같이, underline 2개가 양쪽에 존재하는 method를 말하는데, 파이썬 자체에 이미 특정 행동을 할 수 있도록 구현해놓은 method를 응용하는 것을 말한다. 이를 이용하면 더 효율적인 코딩을 할 수 있는데, 예를 들어

   Person 클래스가 존재한다고 가정한다면,

   Class Person:

   Name = ""

   Age = -1

   Def __init-__(self,name,age):

    Self.name = name

    Self.age = age

   라고 했을 때, __init__을 이용하면 객체를 만들면서 이름과 나이를 설정할 수 있다.

   A = Person("철수", 23)

   하지만, __init__ 이라는 magic method를 사용하지 않는다면,

   Class Person:

   Name = ""

Age = -1

Def set_name(self, name):

Self.name = name

Def set_age(self, age):

Self.age = age

q = Person

q.name = "철수"

q.age = 23 과 같이 설정을 해줄 것이다.

__init__의 magic method를 이용하면 훨씬 코드가 간편해진다.

2. Python의 여러 특징들

파이썬은 변수를 선언할 때, 변수의 타입을 미리 정해주지 않아도 된다는 간편성이 있다. 또한 for문이나 if문을 이용할 때, 들여쓰기를 통해 문단을 구분한다.

(3) Class 구성

1. myMatrix

myMatrix는 row, col, val (list 이용) 를 변수로 갖는 클래스로 만들었다.

addScalar , subScalar, mulScalar, divScalar 의 함수를 통해 scalar 연산을 구현했고,

add , sub, mulMatrix 함수를 통해 Matrix 간의 연산을 구현했다.

이 때, Matrix의 경우 dimension이 맞지 않으면 Nameerror가 출력된다.

또한, mulMatrix의 경우 행렬의 계산을 코드로 직접 만들어줘야 하는데, for 문을 3개 사용함으로써 이를 구현할 수 있었다.

2. myScalar

myScalar는 float형 val를 변수로 갖는 클래스로 만들었다.

addStr의 함수를 통해 Scalar + String을 했을 때, Scalar를 str로 만들어 문자열 덧셈을 overloading 해주었다.

나머지 연산에 대해서는 magic method를 통해, op의 type에 따라 각각에 맞는 operator를 overloading 할 수 있었다.

3. myString

   myString은 str형 val를 변수로 갖는 클래스로 만들었다.

   addScalar, mulScalar 함수를 통해 Scalar와 String간의 연산을 구현했다.

   addScalar의 경우, Scalar + String과 구분하여 String + Scalar의 연산을 할 수 있도록, Scalar 값을 str 타입으로 바꾸어 줌으로써 문자열 덧셈을 구현했다.

   mulScalar의 경우, op.val < 0 인 경우에는 Scalar 값이 음수이므로, 문자열을 reverse 해주었다. (rev = self.val[::-1])

(4) 구현에 대한 사진 첨부

다음의 test.py 코드를 통해 assn5에서 제시한 문제가 구현이 되었음을 확인했다.

```
a = myScalar(3.0)
q = myScalar(-4.0) #
aa = myScalar(7.2)
b = myMatrix(2,3,[1.0,2.0,3.0,4.0,5.0,6.0])
g = myMatrix(2,3,[6.0, 5.0, 4.0, 3.0, 2.0, 1.0])
h = myMatrix(3,2,[1.0, 2.0, 3.0, 4.0, 5.0, 6.0])
c = myString('abc')


print(b)
d = a + b
print(d, end='WnWn')
#
#
#
#
print('myScalar check')
print(a+q, end='WnWn')
print(a-q, end='WnWn')
print(a*q, end='WnWn')
print(a/q, end='WnWn')

print(a+b, end='WnWn')
print(a-b, end='WnWn')
print(a*b, end='WnWn')

print(a+c, end='WnWn')
print(a*c, end='WnWn')
#print(a/b, end='WnWn') # myScalar / myMatrix 했을 때 에러출력
#print(a-c, 'dd') # myScalar - myString 했을 때 에러출력
#print(a/c, 'dd') # myScalar / myString 했을 때 에러출력
#
print('myMatrix check')
```

```
print(b+a, end='WnWn')
print(b-a, end='WnWn')
print(b*a, end='WnWn')
print(b/a, end='WnWn')

print(b+g, end='WnWn')
print(b-g, end='WnWn')
print(b*h, end='WnWn')

# 아래는 myMatrix 연산에서 not defined! 를 출력하는 연산의 경우이다.
# print(b/g)
# print(b+c)
# print(b-c)
# print(b*c)
# print(b/c)
# print(b*g)
print('myString check')
print(c+a, end='WnWn')
print(c*a, end='WnWn')
print(c+q, end='WnWn')
print(c*q, end='WnWn')
print(c+c, end='WnWn')


# print(c/b, end='WnWn')
# myString 연산에서 위의 경우를 제외하고는 모두 not defined가 뜬다.
#
#
#
```

이를 출력하였을 때,

[[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]]

[[4.0, 5.0, 6.0], [7.0, 8.0, 9.0]]

myScalar check

-1.0

7.0

-12.0

-0.75

[[4.0, 5.0, 6.0], [7.0, 8.0, 9.0]]

[[-2.0, -1.0, 0.0], [1.0, 2.0, 3.0]]

[[3.0, 6.0, 9.0], [12.0, 15.0, 18.0]]

3.0abc

abcabcabc

myMatrix check

[[4.0, 5.0, 6.0], [7.0, 8.0, 9.0]]

[[-2.0, -1.0, 0.0], [1.0, 2.0, 3.0]]

[[3.0, 6.0, 9.0], [12.0, 15.0, 18.0]]

[[0.0, 0.0, 1.0], [1.0, 1.0, 2.0]]

[[7.0, 7.0, 7.0], [7.0, 7.0, 7.0]]

[[-5.0, -3.0, -1.0], [1.0, 3.0, 5.0]]

[[22.0, 28.0], [49.0, 64.0]]

myString check

abc3.0

abcabcabc

abc-4.0

cbacbacbacba

abcabc

3.0abc

10.2

Process finished with exit code 0

와 같이, 올바른 결과가 나왔다. 또한 예외적인 경우에도 에러가 올바르게 출력되는지 확인을 하고 #을 통해 주석 처리를 해두었다.

(5) 토론

1. 파일을 처음 실행할 때, Tab 오류가 뜨는 것을 해결하는 것에서 잠깐 어려웠다. 하지만, 검색을 통해 python 코드의 경우 tab이 tab으로 이루어진 것과, spacebar로 tab 역할을 하는 것이 섞여있다면 코드가 올바르게 작동하지 않는다는 것을 보았다.

   조교님이 작성해주신 코드에서 tab은 tab자체로 작동했지만, 이후 내가 tab을 입력해줄 경우 spacebar로 구성된 tab이 입력되는 것을 확인했고, 조교님이 작성해주신 tab을 지운 뒤 내가 다시 tab키를 눌러줌으로써 해결할 수 있었다.

2. Mystring을 제작할 때, op.val < 0 이라는 연산자를 오버로딩 할 수 있도록, myScalar에

__lt__(self,op):

Return self < op 를 오버로딩 해주었다. 또한 문자열을 reverse하고, 특정 횟수(int)만큼 반복시켜 주는 것을 생각하는데 시간이 좀 걸렸다. Reverse의 경우 python 코드의 장점을 이용해 rev = self.val[::-1] 이라는 1줄로 해결할 수 있었다.

(6) 결론

1. 파이썬은 편하다. 확실히 c나 c++보다 직관적이며, 사람들이 미리 작성해놓은 코드들을 incluse 함으로써 다양한 것을 할 수 있다고 들었다. 이번에 사용한 magic method 역시 처음에는 어떻게 사용하는 것인지 몰라 다소 헤맸지만, 사용법을 조금 알고나니 이번 어싸인을 작성하는데 굉장히 많은 도움이 되는 method 라는 것을 깨달았다.