소 프로젝트 2

1910896 천옥희(통계학과)

2-1

```python
#mod1
#2-1
#pythonLecture 클래스 정의
class PythonLecture:
    def __init__(self,first,second,third,absence):  #멤버함수 __init__ (1차 시험점수, 2차 시험점수, 3차 시험점수, 결석횟수)
        self.first=first
        self.second=second
        self.third=third
        self.absence=absence

    # 3개의 시험의 평균 점수를 산출하는 함수
    def average(self):
        return (self.first+self.second+self.third)/3

    # 3개의 점수를 20%, 30%, 50% 가중치로 평균을 산출하는 함수
    def weightavg(self):
        return self.first*0.2+self.second*0.3+self.third*0.5
```

1) PythonLecture이라는 클래스를 정의하였다. 클래스 안에서 멤버변수인 1차 시험 점수는 first, 2차 시험 점수는 second, 3차 시험점수는 third, 결석횟수는 absence라는 변수명으로 지정해주었다. 이 변수들을 __init__안에 넣어 객체가 생성될 때 멤버변수들을 받아 클래스 있는 함수를 사용할 때 자동적으로 호출될 수 있도록 하였다.

2) 클래스에 함수명을 averge로 지정해주고, def averge(self): 밑에 1차, 2차, 3차의 시험점수를 모두 더하고 3으로 나눠준 값을 돌려주도록 해주었다. 클래스의 averge함수를 통해 3개의 시험의 평균 점수를 산출할 수 있다.

3) 클래스에 weight average의 줄인 말인 weightavg를 함수명으로 지정해주고. def weightavg: 밑에 1차 시험점수에 0.2를 곱해주고, 2차 시험점수에 0.3을 곱해주고, 3차 시험점수에 0.5를 곱해 모든 수를 더해 주는 값을 돌려주도록 지정해주었다. 클래스의 weightavg함수를 통해 3개의 점수를 20%, 30%, 50% 가중치로 평균을 산출할 수 있다.

출력결과 확인

```python
a=PythonLecture(90,80,70,0)
print(a.average())
print(a.weightavg())
```

```
C:\Users\user\venv\Scripts\python.exe C:/소프로젝트/mod1.py
80.0
77.0

Process finished with exit code 0
```

사진 2-1(1) 사진 2-1(2)

사진2-1(1)은 실행시킬 명령어고, 사진 2-1(2)는 이를 실행시킨 결과다. a라는 객체를 a=PythonLecture (90,80,70,0)으로 만들어 first=90, second=80, third=80, ascend=0이 되도록 해줬다. print(a.average()) 를 해줘 위의 객체 3개 시험의 평균점수를 출력해보았다. 80.0으로 옳게 나왔다다. print(a.weightavg())를 실행시켜 가중치로 평균을 산출한 함수를 출력해보았다. 77.0으로 옳게 나왔으므로 이 클래스의 3개의 시험의 평균을 구하는 averge 함수와 20%,30%,50%의 가중치로 평균을 구하는 weightavg 함수는 잘 구현되었음을 확인할 수 있다

2-2

```
#2-2
    #가중치 평균을 최종성적으로 받아 학점을 산출하는 함수
    def grade(self):
        final=self.weightavg()          #가중치 평균 final에 반환
        if self.absence<5:              #결석이 5회 미만일 때
            if final >= 90:             #가중치 평균이 90점이상~100점일 때 학점 A
                return 'A'
            elif final >= 80:           #가중치 평균이 80점 이상~90점 미만일 때 학점 B
                return 'B'
            elif final >= 70:           #가중치 평균이 70점 이상~80점 미만일 때 학점 C
                return 'C'
            elif final >= 60:           #가중치 평균이 60점 이상~70점 미만일 때 학점 D
                return 'D'
            else:                       #가중치 평균이 60점 미만일 때 학점 F
                return 'F'
        else:                           #결석이 5회 이상일 때 성적 상관없이 학점 F
            return 'F'
```

클래스에 grade라는 함수명을 지정해주고, weightavg의 반환 값을 최종 성적으로 받아오기 위해서 final=self.weightavg()라고 지정해주었다. 여기서 self.을 weightavg함수 앞에 써주는 이유는 self.을 빼버리면 weightavg라는 것은 정의가 되어 있지 않다는 오류가 뜨게 되며, final은 정의되지 않는다. weightavg함수는 PythonLecture 클래스 안에 있는 함수이기 때문에 self.를 앞에 적어 같은 클래스 안에 있다는 것을 의미해주기 위해서다. 결석이 5회 미만이면 이라는 조건문 밑에 final이 90 이상 100점 미만이면 A, 80점 이상 90점 미만이면 B, 70점 이상 80 점 미만이면 C, 60점 이상 70점 미만이면 D를 60점 미만이면 F를 되돌려 줄 수 있도록 if, elif, else 조건문을 써줬다. 그리고 결석이 5회 이상이면 F를 되돌려 준다는 조건문을 써주었다. 이는 클래스 안에서 grade함수는 weightavg함수인 가중치평균의 값을 반환받아 학점을 산출하지만, 결석이 5회 이상이면 F학점을 부여한다.

출력결과 확인

```
a=PythonLecture(90,80,70,0)
b=PythonLecture(90,80,70,5)
c=PythonLecture(90,50,100,0)
d=PythonLecture(90,30,80,3)
print(a.grade())
print(b.grade())
print(c.grade())
print(d.grade())
```

```
C:\Users\user\venv\Scripts\python.exe C:/소프로젝트/mod1.py
C
F
B
D

Process finished with exit code 0
```

사진 2-2(1)　　사진 2-2(2)

사진 2-2(1)은 실행시킬 명령어고, 사진 2-2(2)는 이를 실행시킨 결과이다.
a라는 객체를 a=PythonLecture(90,80,70,0)으로 만들어 학점이 C가 나오나 확인하기 위함이고, b는 학점은 F가 아닌데 결석이 5회 이상이니 F가 나오는지 확인하기 위해 결석수를 5로 했다. 또한, c, d는 학점이 성적에 따라 잘 나눠서 나오는지 확인하기 위해 1,2,3차의 점수에 차이를 주고, 결석 수는 5미만으로 하였다. 이를 확인해 보기 위해 각각에 클래스 뒤에 .grade()를 붙여 출력해주었다. 그 결과 a클래스의 학점은 C, b클래스의 학점은 F, c클래스의 학점은 B, d클래스의 학점은 D로 옳게 나왔다. 따라서 클래스의 grade함수는 결석과 가중치 평균에 따라 학점을 산출하도록 잘 구현되어 있음을 확인할 수 있다.

2-3

```python
#2-3
import mod1            #mod1 모듈(PythonLecture클래스 사용을 위함)
print('='*31)          #'='의 30개 출력
print('Python Lecture Score Calculator')  #'Python Lecture Score Calculator' 출력
print('='*31)          #'='의 30개 출력

for i in range(1,4):                      #1~3까지 차례로 대입
    a=input('Student %d > ' %i).split()    #학생의 1차 2차 3차 시험 점수와 결석횟수 입력받고 공백을 기준으로 문자열 분리
    # i=1일 때, 학생 1이라는 객체를 가짐
    if i== 1:
        # 리스트인 a에서 인덱싱으로 각각의 변수값 가져오고, 정수로 바꾸줌
        student1 = mod1.PythonLecture(int(a[0]),int(a[1]),int(a[2]),int(a[3]))
    # i=2일 때, 학생 2이라는 객체를 가짐
    elif i==2:
        # 리스트인 a에서 인덱싱으로 각각의 변수값 가져오고, 정수로 바꾸줌
        student2 = mod1.PythonLecture(int(a[0]), int(a[1]), int(a[2]), int(a[3]))
    # i=3일 때, 학생 3이라는 객체를 가짐
    else:
        # 리스트인 a에서 인덱싱으로 각각의 변수값 가져오고, 정수로 바꾸줌
        student3 = mod1.PythonLecture(int(a[0]), int(a[1]), int(a[2]), int(a[3]))
#=============Result============== 출력
print('='*12+'Result'+'='*13)
#Num Score Grade 출력
print('Num Score Grade')
#학생의 순서, 최종성적, 학점 순으로 출력
print(' 1  %0.2f   %s' %(student1.weightavg(),  student1.grade()))
print(' 2  %0.2f   %s' %(student2.weightavg(),  student2.grade()))
print(' 3  %0.2f   %s' %(student3.weightavg(),  student3.grade()))
```

사진 2-3(1)

1) 2-1과 2-2에서 정의한 클래스를 사용하기 위해서 import mod1를 통해 클래스를 부른다. 여기서 mod1은 위의 2-1과 2-2의 파일을 mod1로 저장해놨기 때문에 import 뒤에 mod1를 써줬다.
출력함수인 3개의 print를 통해서 ========== 와 같은 모양을 만들었다.

Python Lecture Score Calculator

==========

여기서 '='를 모두 직접 작성하지 않아도 *31이라는 수식을 써서 출력해도 되기 때문에 print('='*31)을 써주었다.

2) student1> student>2 sutdent3>은 student 뒤에 숫자만 다르다 따라서 for문을 이용하여 1~3까지 차례로 대입해주도록 하였다. 학생의 1차 2차 3차 시험 점수와 결석 횟수를 입력받아야 하기 때문에 input 안에 'student %d > '를 적어준다. 여기서 %d를 사용하는 이유는 작은 따옴표('')안에 1~3을 순서대로 대입하는 i를 넣어주면 1~3이 대입 되는게 아니라 i가 그대로 나오기 때문에 정수포맷함수를 사용해 써주었다. input() 뒤에 .split을 써준 이유는 적힌 4개의 숫자를 공백을 기준으로 리스트로 바꿔주기 위해서다. 학생의 점수와 결석횟수를 입력을 받으면 숫자가 아닌, 문자열로 받게 된다. input앞에 int 문장을 쓰면 안 되냐는 의문이 들 수 있지만, int를 통해 정수로 바꿔버리면 숫자를 입력을 했을 때 4개의 숫자를 띄어쓰게 되면 오류가 발생하게 된다. 따라서 split을 이용해 리스트를 만들었고, 리스트는 a라는 변수명으로 지정해주었다. 리스트로 바꾸는 이유는 4개의 숫자가 문자열로 되어 있기 때문에 각각의 변수들에 대입할 때 인덱싱 함수를 사용해서 대입하기 위해서다.

3)여기서 조건문을 사용하는 이유는 클래스 객체가 학생1일 때와 학생2일 때, 학생3일 때 모두 다르기 때문이다. i=1일 때는 학생1을 나타내기 때문에 student1의 객체를 가질 수 있도록 한다. 여기서 mod1.을 사용하는 이유는 PythonLecture의 클래스는 같은 파이썬 파일에 있는 게 아니라 다른 파일에 있기 때문에 클래스가 있는 파일이름인 mod1.을 붙여줘야 한다. 또한, 인덱싱으로 받은 각각의 요소 값은 문자열을 리스트로 바꾼 것을 받았기 때문에 정수 값이 아니다. 따라서 int라는 함수를 각각의 인덱싱 앞에 붙여 정수로 바꿔줘야 한다. 이를 i=2 일 때는 student2로 i=3일 때는 student3의 객체를 가질 수 있도록 해준다.

4) 위에서 작업한 내용(사진 2-3(1)처럼)은 사진 2-3(2)처럼 student 3>의 줄까지 출력해줄 것이다. 위에 3줄은 1)에서 작업한 내용이고, student 1>~student 3>까지의 내용은 2)와 3) 작업할 내용이다. 2)와 3)에서 출력함수를 쓰지 않았는데 출력되는지 의문이 있겠지만, input문장은 사용자의 입력을 받는 함수라서 input안에 사용자에서 물어볼 질문을 써주면, 질문내용이 출력되게 되어있다. 옆에 초록색으로 입력한 값은 사용자가 4개의 수를 입력한 수이다. 여기서는 학생의 1차, 2차, 3차 시험점수와 결석 횟수를 순서대로 입력한 것이며, 앞에서 구현이 잘 되었는지 확인하기 위해 각각의 질문(student1>~student3>)에 임의의 4수를 입력해주었다. 출력함수에 '='*12+ 'Result' + '='*13를 써줘 문자열들을 서로 더해서 =========result==========와 같은 모형이 출력될 수 있도록 하였고, 출력함수를 통해 밑에 내용들도 출력될 수 있도록 하였다. 학생의 순서, 최종성적, 학점 순으로 사진2-3(1)처럼 출력되기 위해서 각각의 위치에 맞게 띄어쓰기를 해주었다. 여기서 최종성적과 학점은 3)에서 만든 객체를 이용해 출력해 주면 된다. 객체들은 2-1과 2-2에서 만든 클래스함수의 객체이므로, 이 클래스 함수를 사용할 수 있다. 따라서 최종성적은 가중치 평균값을 구해주는 weightavg 함수를 사용하고, 학점은 grade함수를 사용해주었다. 이를 포맷함수를 이용하여 출력하도록 하였는데 최종성적은 소수점 2자리가 나올 수 있도록 소수포맷코드인 %0.2f를 사용해주었고, 학점은 문자이기 때문에 문자열 포맷을 사용해 출력해주었다.

```
================================
Python Lecture Score Calculator
================================
Student 1 > 90 80 70 3
Student 2 > 85 90 95 0
Student 3 > 50 90 70 6
============Result=============
Num Score Grade
 1  77.00   C
 2  91.50   A
 3  72.00   F
```

사진 2-3(2)

출력결과 확인

사진 2-3(2)은 출력결과를 보여준다. Student1의 1차 시험점수는 90 2차는 80 3차는 70 결석횟수는 3을 입력하여 Num=1 Socre=이 세 점수의 가중치 평균인 77.00, Grade=가중치 평균이 70점 이상 80점미만이므로 C가 올바르게 모두 출력되었음을 확인할 수 있다. Student2의 1차 시험점수는 85 2차는 90 3차는 95 결석횟수는 0을 입력하여 Num=2 Socre=이 세 점수의 가중치 평균인 91.50, Grade=가중치 평균이 90점 이상 100점미만 이므로 A, Score의 값이 소수점까지 잘 나옴을 확인할 수 있다. Student3의 1차 시험점수는 50 2차는 90 3차는 70 결석횟수는 6을 입력하여 Num=1 Socre=이 세 점수의 가중치 평균인 72.00, Grade=가중치 평균이 70점 이상 80점미만 이지만, 출석일수가 5일 이상이므로 F가 나와야한다. 확인결과 출력모양도 잘 나타나 있음을 볼 수 있으며, 최종점수의 소수점, 결석횟수와 시험점수에 따른 학점이 모두 올바르게 출력됨을 확인할 수 있다.

2-4

```python
# 2-4
import mod1       #mod1 모듈(PythonLecture클래스 사용을 위함)
import openpyxl   #openpyxl모듈(엑셀파일 부르기 위함)

filename = 'score_table.xlsx'        #엑셀파일에 filename이라는 변수명을 지정
wb_obj=openpyxl.load_workbook(filename)    #엑셀 모듈 여는 함수에 xb_obj라는 변수명 지정
sheet_obj = wb_obj.active                  #시트 객체 지정

for i in range(2, sheet_obj.max_row + 1):  #시트의 데이터를 모두 읽기 위한 반복문
    result = []                            #result는 리스트임을 지정
    for j in range(2, sheet_obj.max_column + 1):
        currentCell_obj = sheet_obj.cell(row=i, column=j)     #각 셀의 데이터를 currentCell_obj으로 지정
        result.append(currentCell_obj.value)                  #각 셀들의 값을 리스트에 추가
    a=mod1.PythonLecture(result[0],result[1],result[2],result[3])   #a는 PythonLecture클래스,리스트 인덱싱을 통해 클래스 변수값 추가
    grade=a.grade()                        #a객체의 grade함수를 grade라는 변수명으로 지정
    Score=a.weightavg()                    #a객체의 weightavg함수를 weightavg라는 변수명으로 지정
    newCell_obj = sheet_obj['F%d'%i]       #F2~F11의 셀 위치지정
    newCell_obj.value = Score              #각 셀에 최종성적(가중치평균)추가
    newCell_obj = sheet_obj['G%d'%i]       #G2~G11의 셀 위치지정
    newCell_obj.value = grade              #각 셀에 학점추가

newCell_obj=sheet_obj['F1']                #F1셀 위치지정
newCell_obj.value='Final Score'            #F1셀에 'Final Score' 문자추가
newCell_obj=sheet_obj['G1']                #G1셀 위치지정
newCell_obj.value='Final Grade'            #G1셀에 'Final Grade' 문자추가

wb_obj.save('score_table_final.xlsx')      #추가한 데이터를 포함한 데이터를 'score_table_final.xlsx'라는 이름으로 저장
```

사진2-4 (1)

1) 2-1과 2-2에서 생성한 클래스를 사용하기 위해 import mod1을 써주었다. 또한, 엑셀파일을 부를 수 있는 openpyxl모듈을 사용하기 위해서도 import openpyxl도 써주었다.

2)엑셀파일 이름을 filename이라는 변수명으로 지정해주었다. 지정해주는 이유는 긴 파일 이름을 여러번 쓰는 것보다 짧은 이름으로 바꿔주면 편리하기 때문이다. openpyxl 모듈을 이용해 파일을 열었다. load_workbook은 엑셀 모듈을 여는 함수이며, 이를 wb_obj이라는 변수명으로 지정해주었다. 다음으로 시트 객체를 지정해주었는데 wb_obj와 acrive를 사용해 시트 객체를 지정해주고, sheet_obj이라는 변수명으로 지정해주었다.

	A	B	C	D	E
1		1	2	3	absence
2	1	78	78	79	5
3	2	94	72	55	1
4	3	85	94	62	1
5	4	100	50	61	6
6	5	88	61	79	0
7	6	59	70	66	5
8	7	56	84	89	2
9	8	99	54	93	0
10	9	99	85	74	4
11	10	78	71	64	3

사진 2-4score_table.xlsx(1)

3) 각각의 셀을 지정해 주기 위해서 행과 열이 필요하다. 행이 1일 때 열은 1~j개 까지 있고, 행이 2일 때도 열은 1~j개가 있기 때문에 i개의 행과 j개의 행을 하나하나 지정해주어야 한다. 이를 반복문을 이용해 간결하게 나타낼 수 있다. 행 반복문 밑에 열의 반복문을 써주면, 행이 1~i일 때의 열 1~j을 모두 지정해 줄 수 있다. 따라서 이중 반복문을 사용한다. 행은 i로 지정해주고, 범위는 (2~현재시트의 행 수+1), 열을 j로 지정해주고, 범위는(2~현재시트의 열 수+1) 해주었다. 1부터 하지 않은 이유는 사진 2-4score_table.xlsx(1)를 보면 엑셀파일에서의 변수명을 뺀 데이터의 값들은 (2,2)부터 시작하기 때문이다. 행과 열 두 반복

문 밑에는 cell을 지정해주기 위해 sheet_obj.cell(row=i,column=j)를 써주고, 이를 currentCell_obj 변수명으로 지정해주었다. 그리고 각각의 행에 따른 열의 값들을 리스트로 만들어 주기 위해서 행 반복문 밑에 열 반복문에 들어가기 전에 result는 리스트임을 지정해주었다. 이를 append를 이용하여 각각의 셀의 값들을 result에 추가해 각 행에 대한 열값들이 리스트로 만들어지도록 하였다.

4) 3)에서 만들어준 리스트(result)를 이용해 PythonLecture 클래스의 변수 값으로 지정해줄 것이다. 리스트의 첫 번째 수는 1차 시험점수이기 때문에 result[0]으로 입력해주고, 두 번째 수는 2차 시험점수로 result[1], 세 번째 수는 3차 시험점수로 result[2], 마지막 수는 결석횟수로 result[3]으로 나타내줬다. 이 클래스는 같은 파일이 아닌 mod1의 파일에 있으므로, mod1.PythonLecture(result[0],result[1],result[2],result[3])의 형식으로 써주었으며, 이를 a라고 지정해주었다. 이 a를 이용하여 이제 각각의 행에 대한 최종성적과 학점을 산출해줄 것이다. 최종성적은 가중치 평균이므로, PythonLecture클래스의 weightavg 함수를 이용하고, 학점은 PythonLecture클래스의 grade함수를 이용할 것이다. 따라서 a.grade()에 grade변수명을 지정해주고, a.weightavg에 Score변수명을 지정해주었다. F열 행(2~11)들 각각의 셀에 대한 최종점수를, G열 행(2~11)들 각각의 셀에 대한 학점을 넣어야하기 때문에 행 반복문 밑에 넣었다. newCell_obj=sheet_obj['F%d'%i]를 써 F열들의 셀을 지정할 수 있도록 하였으며, Score값은 각 행에 대한 최종 점수이므로 newCell_obj.value =Score써주었다. newCell_obj=sheet_obj['G%d'%i]를 써 G열들의 셀을 지정할 수 있도록 하였으며, grade값은 각 행에 대한 학점이므로 newCell_obj.value=grade를 써주었다. 이는 F열과 G열에 대한 각각의 행 2~11까지 반복 수행되고, 각 행에 대한 최종 성적과 학점을 엑셀 파일에 추가해준다. 이제 F열과 G열 각각의 행(2~11)에 해당하는 값을 넣어주었으면, F1과 G1셀에 해당하는 변수명을 추가해주어야 한다. F1과 G1 셀에 해당하는 변수명을 직접 추가해 주기위해 반복문 밖으로 나와 위치를 지정하고, 값들에 대해 문자열을 추가해주었다. 마지막으로 score_table_final.xlsx이름으로 생성해주기 위해 wb_obj.save('score_table_final.xlsx')를 입력해 저장해주었다. 이는 위의 사진2-4 score_table.xlsx와 별도로 새로운 score_table_final.xlsx의 이름의 파일이 생성된다.

결과 확인

	A	B	C	D	E
1		1	2	3	absence
2	1	78	78	79	5
3	2	94	72	55	1
4	3	85	94	62	1
5	4	100	50	61	6
6	5	88	61	79	0
7	6	59	70	66	5
8	7	56	84	89	2
9	8	99	54	93	0
10	9	99	85	74	4
11	10	78	71	64	3

사진2-4 score_table.xlsx(2)

	A	B	C	D	E	F	G
1		1	2	3	absence	Final Score	Final Grade
2	1	78	78	79	5	78.5	F
3	2	94	72	55	1	67.9	D
4	3	85	94	62	1	76.2	C
5	4	100	50	61	6	65.5	F
6	5	88	61	79	0	75.4	C
7	6	59	70	66	5	65.8	F
8	7	56	84	89	2	80.9	B
9	8	99	54	93	0	82.5	B
10	9	99	85	74	4	82.3	B
11	10	78	71	64	3	68.9	D

사진2-4 score_table_final.xlsx

사진2-4 score_table.xlsx(2)와 사진2-4 score_table_final.xlsx은 위에서 작업한 내용(사진 2-4(1))을 실행시킨 후의 엑셀 파일이다. 사진2-4 score_table.xlsx(1)의 score_table.xlsx 파일은 사진2-4 score_table.xlsx(2)에서 봤듯이 그대로 남아 있고, 사진2-4 score_table_final.xlsx의 사진처럼 F열과 G열에 최종성적과 학점이 추가되어 score_table_final.xlsx 파일이 새로 생성됨을 볼 수 있다. 사진2-4 score_table_final.xlsx을 통해 F1에 'Final Score'의 변수명과 G1에 'Final Grade'라는 변수명이 올바르게 추가되었음을 확인할 수 있다. 또한, F(2~11)과 G(2~11)의 값들이 오류 없이 올바르게 추가되었음을 확인할 수 있으므로, 엑셀로 정리된 성적파일 score_table.xlsx로부터 데이터를 읽어와 최종성적과 학점에 해당하는 열을 각각 추가하여 score_table_final.xlsx를 생성하는 프로그램이 잘 구현되었음을 알 수 있다.