Homework Assignment 4

보고서 및 논문 윤리 서약

1. 나는 보고서 및 논문의 내용을 조작하지 않겠습니다.
2. 나는 다른 사람의 보고서 및 논문의 내용을 내 것처럼 무단으로 복사하지 않겠습니다.
3. 나는 다른 사람의 보고서 및 논문의 내용을 참고하거나 인용할 시 참고 및 인용 형식을 갖추고 출처를 반드시 밝히겠습니다.
4. 나는 보고서 및 논문을 대신하여 작성하도록 청탁하지도 청탁받지도 않겠습니다.

나는 보고서 및 논문 작성 시 위법 행위를 하지 않고, 명지인으로서 또한 공학인으로서 나의 양심과 명예를 지킬 것을 약속합니다.

학　　과 : 융합소프트웨어학부 데이터사이언스전공
과　　목 : 기초프로그래밍1
담당교수 : 김상균 교수님
학　　번 : 60241996
이　　름 : 김민준 (서명)

김민준.

실습4

```python
def my_plus_ans (x,y):
    return x + y
1개의 사용 위치  신규 *
def my_minus_ans (x,y):
    return x - y
1개의 사용 위치  신규 *
def my_multiply_ans (x,y):
    return x * y
1개의 사용 위치  신규 *
def my_divide_ans (x,y):
    return x / y
1개의 사용 위치  신규 *
def my_calculate(x, y, opcode="+"):
    if opcode =="+":
        return my_plus_ans(x, y)
    elif opcode == "-":
        return my_minus_ans(x, y)
    elif opcode == "*":
        return my_multiply_ans(x, y)
    elif opcode == "/":
        return my_divide_ans(x,y)
```

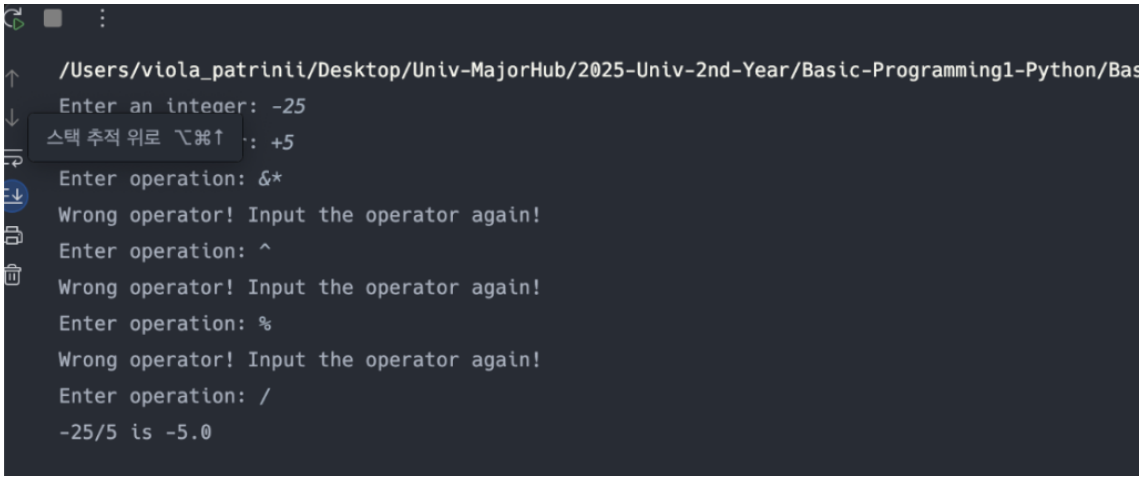
```python
2 개의 사용 위치  신규 *
def InputInt(s):
    while True:
        number = input("Enter an integer: ")
        if isInt(str(number)):
            return int(number)
        else:
            print("Wrong integer! Input the " + s + " integer again!")


1개의 사용 위치  신규 *
def isInt(x):
    if x[0] == "+" or x[0] == "-":
        if len(x) == 1:
            return False
        start_index = 1
    else:
        start_index = 0

    for i in range(start_index, len(x)):
        if x[i] not in "0123456789":
            return False
    return True
```

```
1개의 사용 위치  신규 *
def InputOp():
    while True:
        opcode = input("Enter operation: ")
        if isOp(opcode):
            return opcode
        else:
            print("Wrong operator! Input the operator again!")
1개의 사용 위치  신규 *
def isOp(x):
    if x in "+-*/":
        return True
    else:
        return False


x = InputInt("first")
y = InputInt("second")
Op = InputOp()
print(str(x) + Op + str(y), "is", my_calculate(x,y,Op))
```



실습 4번 코드입니다.

실습3번 코드 위주로 리팩토링 하였으며, isInt(x) 구현 시 x[0] 인덱스가 + - 인지 판별하고, 해당 여부에 따라 start_index 를 0 으로 시작할지 1로 시작할지 나눠 구현하였습니다.

또한 + 혹은 - 만 integer 에 기입하여 len(x) == 1 인 경우도 return False 로 예외 처리 했습니다.

나머지는 실습 지시사항 대로 구현하였습니다.

과제4
problem 4-1

```python

# 빚의 잔액
balance = int(input("Enter the outstanding balance on your credit card: "))
# 연 이자율(고정)
annual_interest_rate = float(input("Enter the annual credit card interest rate as a decimal: "))
# 최소 납입 금액 비율(고정)
monthly_pay_rate = float(input("Enter the minimum monthly payment rate as a decimal: "))
# 월 이자율 (고정)
mon_interest_rate = annual_interest_rate / 12.0
total_paid = 0.0
print(mon_interest_rate)

for i in range(1, 13):
    # 최소 월 납부 금액
    min_month_paid = round(monthly_pay_rate * balance, 2)
    # 월 이자
    interest_paid = round(mon_interest_rate * balance, 2)
    # 실제 상환 금액
    principal_paid = round(min_month_paid - interest_paid, 2)
    # 남은 빚 잔액
    balance = round(balance - principal_paid, 2)
    # 총 납입액 누적
    total_paid = round(total_paid + min_month_paid, 2)

    print("Month: " + str(i))
    print("Minimum monthly payment: $" + str(min_month_paid))
    print("Principle paid: $" + str(principal_paid))
    print("Remaining balance: $" + str(balance))

print("RESULT")
print("Total amount paid: $" + str(total_paid))
print("Remaining balance: $" + str(balance))

```

코드 사진입니다.

아래 사진은 test case1,2 4800, .2 .02, .04 대입시 나오는 결과값 입니다.

매달 최소 납입 금액에서 이자를 제외한 금액만큼 원금을 상환하고, 이를 12개월 동안 반복하여 총 납입액과 최종 잔액을 계산하는 코드입니다.

```
Principle paid: $15.79
Remaining balance: $4720.53
Month: 6
Minimum monthly payment: $94.41
Principle paid: $15.73
Remaining balance: $4704.8
Month: 7
Minimum monthly payment: $94.1
Principle paid: $15.69
Remaining balance: $4689.11
Month: 8
Minimum monthly payment: $93.78
Principle paid: $15.63
Remaining balance: $4673.48
Month: 9
Minimum monthly payment: $93.47
Principle paid: $15.58
Remaining balance: $4657.9
Month: 10
Minimum monthly payment: $93.16
Principle paid: $15.53
Remaining balance: $4642.37
Month: 11
Minimum monthly payment: $92.85
Principle paid: $15.48
Remaining balance: $4626.89
Month: 12
Minimum monthly payment: $92.54
Principle paid: $15.43
Remaining balance: $4611.46
RESULT
Total amount paid: $1131.12
Remaining balance: $4611.46
```


로젝트
assignmenttemp/과제4_602419
행
과제4_60241996_김민준
더보기
Prin id: $101.91
Rema lance: $4265.53
Month: 6
Minimum monthly payment: $170.62
Principle paid: $99.53
Remaining balance: $4166.0
Month: 7
Minimum monthly payment: $166.64
Principle paid: $97.21
Remaining balance: $4068.79
Month: 8
Minimum monthly payment: $162.75
Principle paid: $94.94
Remaining balance: $3973.85
Month: 9
Minimum monthly payment: $158.95
Principle paid: $92.72
Remaining balance: $3881.13
Month: 10
Minimum monthly payment: $155.25
Principle paid: $90.56
Remaining balance: $3790.57
Month: 11
Minimum monthly payment: $151.62
Principle paid: $88.44
Remaining balance: $3702.13
Month: 12
Minimum monthly payment: $148.09
Principle paid: $86.39
Remaining balance: $3615.74
RESULT
Total amount paid: $2030.15
Remaining balance: $3615.74

problem 4-2

```python
# Problem 2

# 빚의 잔액
balance = (int(input("Enter the outstanding balance on your credit card: ")))
# 연 이자율(고정)
annual_interest_rate = float(input("Enter the annual credit card interest rate as a decimal: "))
# 월 이자율 (고정)
mon_interest_rate = annual_interest_rate / 12.0
min_month_payment = 10

check_bool = True
while check_bool:
    temp_balance = balance

    for i in range(12):
        temp_balance = temp_balance * (1 + mon_interest_rate) - min_month_payment

        if temp_balance <= 0:
            print("RESULT")
            print("Monthly payment to pay off debt in 1 year: " + str(min_month_payment))
            print("Number of months needed: " + str(i+1))
            print("Balance: " + str(round(temp_balance,2)))
            check_bool = False
            break
    min_month_payment += 10
```

```
Remaining balance: $4611.46
Enter the outstanding balance on your credit card: 1200
Enter the annual credit card interest rate as a decimal: .18
RESULT
Monthly payment to pay off debt in 1 year: 120
Number of months needed: 11
Balance: -10.05
```

```
Enter the outstanding balance on your credit card: 32000
Enter the annual credit card interest rate as a decimal: .2
RESULT
Monthly payment to pay off debt in 1 year: 2970
Number of months needed: 12
Balance: -74.98
Enter the outstanding balance on your credit card:
```

저는 min_month_payment를 $10씩 증가시키며 1년 안에 빚을 갚을 수 있는 최소 고정 납입금을 찾는 로직을 while 루프로 구현했습니다.
또한, temp_balance변수를 따로 사용하여 원래 잔액은 보존하면서 월별 상환을 구현 했습니다.

또한 check_bool변수를 사용하여, 목표 조건(12개월 내 상환 성공)이 충족되면 while 루프를

종료하도록 제어했습니다.

problem 4-3

```python
# Problem 3 Requirement
# Using Binary(Bisection) Search to Make the Program Run Faster
balance = int(input("Enter the outstanding balance on your credit card: "))
annual_interest_rate = float(input("Enter the annual credit card interest rate as a decimal: "))

# 월 이율 = 연 이율 / 12
monthly_interest_rate = annual_interest_rate / 12.0
lower_bound = balance / 12.0
upper_bound = (balance * (1 + monthly_interest_rate) ** 12.0) / 12.0

epsilon = 0.01  # 오차범위

while (upper_bound - lower_bound) >= epsilon:
    test_balance = balance
    min_monthly_payment = (lower_bound + upper_bound) / 2.0

    for month in range(1,13):
        test_balance = test_balance * (1 + monthly_interest_rate) - min_monthly_payment
        if test_balance <= 0:
            final_month = month
            break

    if test_balance > 0:
        lower_bound = min_monthly_payment
    else:
        upper_bound = min_monthly_payment

print("RESULT")
print("Monthly payment to pay off debt in 1 year:", str(round(min_monthly_payment, 2)))
print("Number of months needed:", final_month)
print("Balance:", str(round(test_balance, 2)))
```

```
Balance: -6.25
Enter the outstanding balance on your credit card: 999999
Enter the annual credit card interest rate as a decimal: .18
RESULT
Monthly payment to pay off debt in 1 year: 91679.91
Number of months needed: 12
Balance: -0.06
```

```
Enter the outstanding balance on your credit card: 320000
Enter the annual credit card interest rate as a decimal: .2
RESULT
Monthly payment to pay off debt in 1 year: 29643.04
Number of months needed: 12
Balance: 0.04
```

저는 **이분 탐색 알고리즘**을 사용해 lower_bound와 upper_bound사이에서 매달 납부해야 할 최소 금액을 0.01오차 내에서 빠르게 찾아내는 로직으로 구현했습니다.

또한, test_balance변수를 따로 사용하여 원래 잔액(balance)을 보존하면서 매달 이자를 계산해 남은 잔액을 추적했고, 조건을 만족하는 순간의 final_month를 추적하여 결과에 출력하도록 했습니다.

test case 에서 요구하는 값에 초점을 맞춰 코드를 구현한 뒤, 보편적인 값을 출력해낼 수 있는 코드로 발전 시켰습니다.

감사합니다.