이익 나침반

학번 : 2318071

이름 : 손준화

Github address : https://github.com/SONJUNHWA/Homework1/blob/main/Homework1.py

1. 계산기의 목적

배당금 계산기 프로그램의 목적은 주식 투자자들이 배당률, 보유 주식 수, 주식 가격을 입력하여 예상 배당금 수익을 간편하게 계산할 수 있도록 돕는 것이다. 이를 통해 투자자들은 예상 수익을 쉽게 예측하고 투자 결정을 내릴 수 있다.

2. 계산기의 네이밍의 의미

투자를 할 때 확실한 방향을 제시해 준다는 의미로, 주식 투자에 있어 중요한 배당 수익을 정확하고 손쉽게 계산하여, 투자자가 더 나은 결정을 내릴 수 있도록 도와준다.

3. 계산기 개발 계획

입력 변수

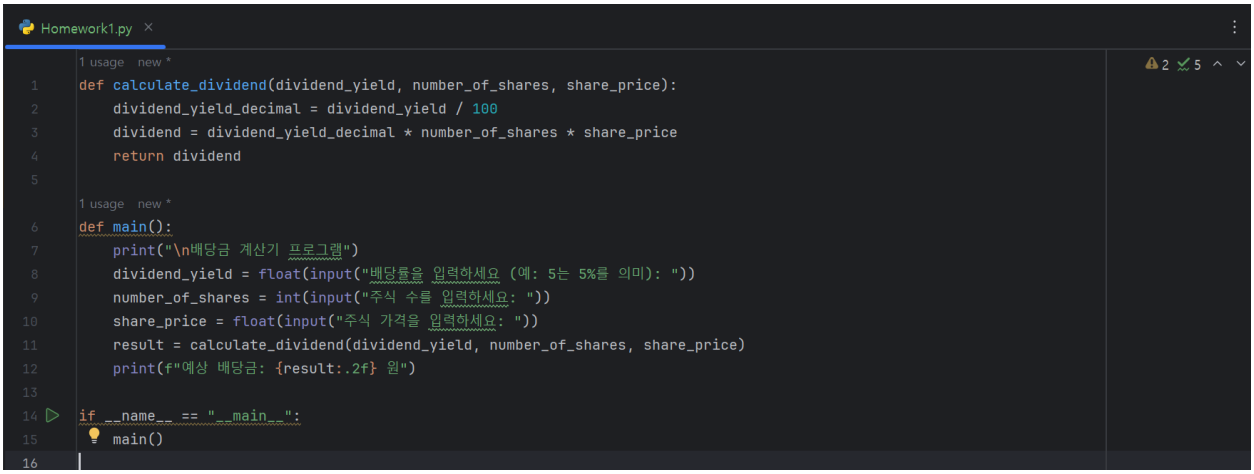
- 배당률 (dividend_yield) : 주식이 지급하는 배당률 (예: 5 는 5%를 의미).
- 주식 수 (number_of_shares) : 사용자가 보유한 주식의 수.
- 주식 가격 (share_price) : 주식의 현재 가격.

연산 과정 설계

- 조건문
    - 사용자가 입력한 값이 유효한지(음수 또는 비수치 값이 아닌지) 확인합니다.
    - 배당률, 주식 수, 주식 가격이 모두 양수인지 검사합니다.
    - 입력 값에 문제가 있을 경우 오류 메시지를 출력하고 다시 입력을 받습니다.
- 반복문
    - 프로그램이 종료될 때까지 사용자가 계산을 반복적으로 수행할 수 있도록 합니다.
    - 사용자가 계속해서 배당금 계산을 수행하거나 프로그램을 종료할 수 있도록 선택할 수 있게 합니다.

4. 계산기 개발 과정

1 단계 : 초기 개발



```python
def calculate_dividend(dividend_yield, number_of_shares, share_price):
    dividend_yield_decimal = dividend_yield / 100
    dividend = dividend_yield_decimal * number_of_shares * share_price
    return dividend


def main():
    print("\n배당금 계산기 프로그램")
    dividend_yield = float(input("배당률을 입력하세요 (예: 5는 5%를 의미): "))
    number_of_shares = int(input("주식 수를 입력하세요: "))
    share_price = float(input("주식 가격을 입력하세요: "))
    result = calculate_dividend(dividend_yield, number_of_shares, share_price)
    print(f"예상 배당금: {result:.2f} 원")

if __name__ == "__main__":
    main()
```

우선, 입력을 받아 간단히 배당금을 계산하는 코드를 작성한다.

- ‘calculate_dividend’ 함수는 배당률, 주식 수, 주식 가격을 입력받아 배당금을 계산한다.
- ‘main’ 함수는 사용자로부터 입력을 받고 결과를 출력한다.

2 단계 : 에러 처리 추가

- 사용자가 잘못된 입력을 했을 때 발생하는 에러를 처리하는 코드를 추가한다.

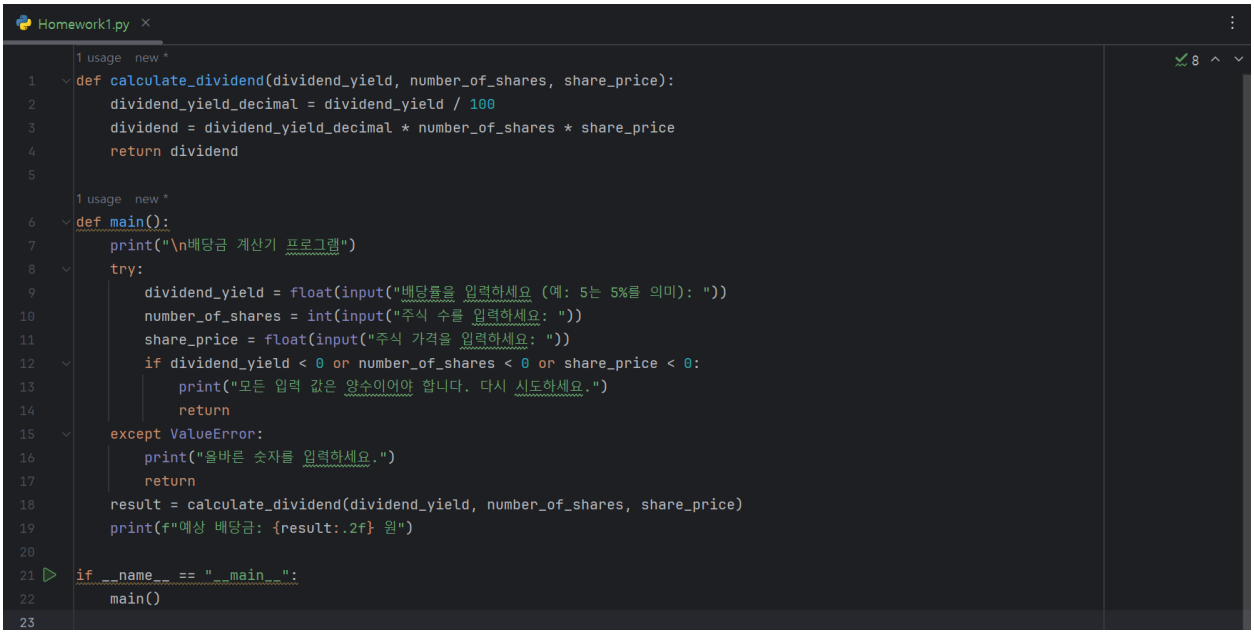
```python
def calculate_dividend(dividend_yield, number_of_shares, share_price):
    dividend_yield_decimal = dividend_yield / 100
    dividend = dividend_yield_decimal * number_of_shares * share_price
    return dividend


def main():
    print("\n배당금 계산기 프로그램")
    try:
        dividend_yield = float(input("배당률을 입력하세요 (예: 5는 5%를 의미): "))
        number_of_shares = int(input("주식 수를 입력하세요: "))
        share_price = float(input("주식 가격을 입력하세요: "))
    except ValueError:
        print("올바른 숫자를 입력하세요.")
        return
    result = calculate_dividend(dividend_yield, number_of_shares, share_price)
    print(f"예상 배당금: {result:.2f} 원")

if __name__ == "__main__":
    main()
```

- ‘try-except ‘ 블록을 사용하여 숫자가 아닌 값이 입력될 경우 ValueError 를 처리한다.

- 사용자가 잘못된 입력을 할 경우 에러 메시지를 출력하고 프로그램을 종료한다.

3 단계 : 입력값 유효성 검사 추가

- 모든 입력 값이 양수인지 확인하는 조건문을 추가한다.



Homework1.py

```
1 usage  new *
def calculate_dividend(dividend_yield, number_of_shares, share_price):
    dividend_yield_decimal = dividend_yield / 100
    dividend = dividend_yield_decimal * number_of_shares * share_price
    return dividend

1 usage  new *
def main():
    print("\n배당금 계산기 프로그램")
    try:
        dividend_yield = float(input("배당률을 입력하세요 (예: 5는 5%를 의미): "))
        number_of_shares = int(input("주식 수를 입력하세요: "))
        share_price = float(input("주식 가격을 입력하세요: "))
        if dividend_yield < 0 or number_of_shares < 0 or share_price < 0:
            print("모든 입력 값은 양수이어야 합니다. 다시 시도하세요.")
            return
    except ValueError:
        print("올바른 숫자를 입력하세요.")
        return
    result = calculate_dividend(dividend_yield, number_of_shares, share_price)
    print(f"예상 배당금: {result:.2f} 원")

if __name__ == "__main__":
    main()
```

- 입력 값이 음수인지 확인하는 조건문을 추가하여 음수 값 입력 시 오류 메시지를 출력한다.

4 단계 : 반복 기능 추가

Homework1.py

```python
1 usage  new *
def calculate_dividend(dividend_yield, number_of_shares, share_price):
    dividend_yield_decimal = dividend_yield / 100
    dividend = dividend_yield_decimal * number_of_shares * share_price
    return dividend


1 usage  new *
def main():
    while True:
        print("\n배당금 계산기 프로그램")
        print("1. 배당금 계산")
        print("2. 종료")

        choice = input("원하는 기능을 선택하세요 (1/2): ")

        if choice == '1':
            try:
                dividend_yield = float(input("배당률을 입력하세요 (예: 5는 5%를 의미): "))
                number_of_shares = int(input("주식 수를 입력하세요: "))
                share_price = float(input("주식 가격을 입력하세요: "))
                if dividend_yield < 0 or number_of_shares < 0 or share_price < 0:
                    print("모든 입력 값은 양수이어야 합니다. 다시 시도하세요.")
                    continue
                result = calculate_dividend(dividend_yield, number_of_shares, share_price)
                print(f"예상 배당금: {result:.2f} 원")
            except ValueError:
                print("올바른 숫자를 입력하세요.")

        elif choice == '2':
            print("프로그램을 종료합니다.")
            break

        else:
            print("잘못된 입력입니다. 다시 선택하세요.")


if __name__ == "__main__":
    main()
```

- 사용자가 계산을 반복적으로 수행할 수 있도록 반복문을 추가한다.
- 'while True' 반복문을 사용하여 사용자가 프로그램을 종료할 때까지 반복적으로 계산할 수 있도록 한다.
- 사용자 선택에 따라 배당금 계산 또는 프로그램 종료를 처리한다.

5. 계산기 개발

* 개발 후 느낀 점
  - 입력 유효성 검사와 예외 처리가 중요함을 깨달았다. 이를 통해 프로그램의 안정성과 사용자 경험을 향상시킬 수 있었다.
  - 반복문을 사용하여 프로그램을 지속적으로 사용할 수 있게 함으로써 사용자 편의성이 크게 개선되었다.
  - 단순한 계산기 프로그램이지만, 실제 사용자 상황을 고려하여 예외 처리를 추가함으로써 실용적인 도구로 만들 수 있음을 배웠다.

* 계산기의 효과

- 사용자가 배당금 계산을 쉽게 수행할 수 있어 투자 결정에 도움을 준다.
- 입력 유효성 검사와 예외 처리를 통해 프로그램의 안정성과 신뢰성을 높였다.
- 반복 기능을 통해 사용자가 다양한 시나리오를 시험할 수 있다.