cafle 임포트 하기

```
In [1]: import cafle
```

```
In [2]: idx = cafle.Index('2023.01', 12)
        idx
```

Out[2]: DateIndex(['2023.01.31', '2023.02.28', '2023.03.31', '2023.04.30',
        '2023.05.31', '2023.06.30', '2023.07.31', '2023.08.31', '2023.09.3
        0', '2023.10.31', '2023.11.30', '2023.12.31'])

```
In [3]: acc = cafle.Account(idx)
        acc
```

Out[3]: Account(main, len 12)

```
In [4]: acc.df
```

Out[4]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	0.0	0.0	0.0
2023-02-28	0.0	0.0	0.0	0.0
2023-03-31	0.0	0.0	0.0	0.0
2023-04-30	0.0	0.0	0.0	0.0
2023-05-31	0.0	0.0	0.0	0.0
2023-06-30	0.0	0.0	0.0	0.0
2023-07-31	0.0	0.0	0.0	0.0
2023-08-31	0.0	0.0	0.0	0.0
2023-09-30	0.0	0.0	0.0	0.0
2023-10-31	0.0	0.0	0.0	0.0
2023-11-30	0.0	0.0	0.0	0.0
2023-12-31	0.0	0.0	0.0	0.0

```
In [5]: type(acc.df)
```

Out[5]: pandas.core.frame.DataFrame

```
In [ ]:
```

In [6]:
```python
import cafle as cf
```

In [7]:
```python
idx = cf.Index("2023.01", 12)
idx
```

Out[7]: DateIndex(['2023.01.31', '2023.02.28', '2023.03.31', '2023.04.30',
'2023.05.31', '2023.06.30', '2023.07.31', '2023.08.31', '2023.09.3
0', '2023.10.31', '2023.11.30', '2023.12.31'])

In [8]:
```python
acc = cf.Account(idx)
acc
```

Out[8]: Account(main, len 12)

In [9]:
```python
acc.df
```

Out[9]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	0.0	0.0	0.0
2023-02-28	0.0	0.0	0.0	0.0
2023-03-31	0.0	0.0	0.0	0.0
2023-04-30	0.0	0.0	0.0	0.0
2023-05-31	0.0	0.0	0.0	0.0
2023-06-30	0.0	0.0	0.0	0.0
2023-07-31	0.0	0.0	0.0	0.0
2023-08-31	0.0	0.0	0.0	0.0
2023-09-30	0.0	0.0	0.0	0.0
2023-10-31	0.0	0.0	0.0	0.0
2023-11-30	0.0	0.0	0.0	0.0
2023-12-31	0.0	0.0	0.0	0.0

In [ ]:

In [10]:
```python
from cafle import Index, Account
```

In [11]:
```python
idx = Index('2023.01', 12)
idx
```

Out[11]: DateIndex(['2023.01.31', '2023.02.28', '2023.03.31', '2023.04.30',
'2023.05.31', '2023.06.30', '2023.07.31', '2023.08.31', '2023.09.3
0', '2023.10.31', '2023.11.30', '2023.12.31'])

In [12]:
```python
acc = Account(idx)
acc
```

Out[12]: Account(main, len 12)

In [13]:
```python
acc.df
```

Out[13]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	0.0	0.0	0.0
2023-02-28	0.0	0.0	0.0	0.0
2023-03-31	0.0	0.0	0.0	0.0
2023-04-30	0.0	0.0	0.0	0.0
2023-05-31	0.0	0.0	0.0	0.0
2023-06-30	0.0	0.0	0.0	0.0
2023-07-31	0.0	0.0	0.0	0.0
2023-08-31	0.0	0.0	0.0	0.0
2023-09-30	0.0	0.0	0.0	0.0
2023-10-31	0.0	0.0	0.0	0.0
2023-11-30	0.0	0.0	0.0	0.0
2023-12-31	0.0	0.0	0.0	0.0

In [14]:
```python
acc.addamt('2023.01', 30_000)
acc.subamt('2023.04', 10_000)
acc.subamt('2023.05', 10_000)
```

In [15]: `acc.df`

Out[15]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	30000.0	0.0	30000.0
2023-02-28	30000.0	0.0	0.0	30000.0
2023-03-31	30000.0	0.0	0.0	30000.0
2023-04-30	30000.0	0.0	10000.0	20000.0
2023-05-31	20000.0	0.0	10000.0	10000.0
2023-06-30	10000.0	0.0	0.0	10000.0
2023-07-31	10000.0	0.0	0.0	10000.0
2023-08-31	10000.0	0.0	0.0	10000.0
2023-09-30	10000.0	0.0	0.0	10000.0
2023-10-31	10000.0	0.0	0.0	10000.0
2023-11-30	10000.0	0.0	0.0	10000.0
2023-12-31	10000.0	0.0	0.0	10000.0

In [ ]:

주요 모듈 소개

1. Index 모듈

정수 인덱스 생성

- Index(마지막 숫자) : from 0 to 마지막 숫자의 RangeIndex 생성
- Index(시작 숫자, 마지막 숫자) : from 시작 숫자 to 마지막 숫자의 RangeIndex 생성
- Index(시작 숫자, 마지막 숫지, 숫자 간격) : from 시작 숫자 to 마지막 숫자를 숫자 간격에 따라 띄워서 RangeIndex 생성

In [26]:
```
idx = Index(12)
idx
```

Out[26]: `RangeIndex(range(0, 12))`

```
In [27]: idx[0]
```

```
Out[27]: 0
```

```
In [29]: idx[11]
```

```
Out[29]: 11
```

```
In [30]: idx[-1]
```

```
Out[30]: 11
```

```
In [32]: # range() 함수와 같이 마지막 숫자는 제외됨.
         idx[12]
```

```
---------------------------------------------------------------------
---------
IndexError                                Traceback (most recent c
all last)
/var/folders/01/c04y5bhn61l8kzg4jnpy_8dw0000gp/T/ipykernel_45374/2
658602549.py in <module>
      1 # range() 함수와 같이 마지막 숫자는 제외됨.
----> 2 idx[12]

~/opt/anaconda3/envs/FastCampus2/lib/python3.7/site-packages/cafle
/index.py in __getitem__(self, key)
    263
    264             if is_scalar(key):
--> 265                 return getitem(key)
    266
    267             if isinstance(key, slice):

IndexError: range object index out of range
```

```
In [35]: idx = Index(10, 24)
         idx
```

```
Out[35]: RangeIndex(range(10, 24))
```

```
In [36]: idx[0]
```

```
Out[36]: 10
```

```
In [37]: idx[-1]
```

```
Out[37]: 23
```

```
In [39]:  # 슬라이싱 적용 가능
          idx[0:3]
```

```
Out[39]:  RangeIndex(range(10, 13))
```

```
In [40]:  idx = Index(0, 12, 2)
          idx
```

```
Out[40]:  RangeIndex(range(0, 12, 2))
```

```
In [41]:  idx[0]
```

```
Out[41]:  0
```

```
In [42]:  idx[1]
```

```
Out[42]:  2
```

```
In [ ]:
```

월별 인덱스 생성

- Index(시작월, 생성 갯수) : 시작월의 마지막 날에서 시작하여 총 생성 갯수의 날짜 Index 생성
- Index(시작월, 마지막월) : 시작월의 마지막 날에서 시작하여 마지막월 마지막 날까지의 월별 날짜 Index 생성

```
In [45]:  idx = Index('2023.01', 12)
          idx
```

```
Out[45]:  DateIndex(['2023.01.31', '2023.02.28', '2023.03.31', '2023.04.30',
          '2023.05.31', '2023.06.30', '2023.07.31', '2023.08.31', '2023.09.3
          0', '2023.10.31', '2023.11.30', '2023.12.31'])
```

```
In [46]:  idx[3]
```

```
Out[46]:  datetime.date(2023, 4, 30)
```

```
In [47]:  idx[-1]
```

```
Out[47]:  datetime.date(2023, 12, 31)
```

```
In [ ]:
```

```
In [48]: idx = Index('2023.01', '2023.12')
         idx
```

```
Out[48]: DateIndex(['2023.01.31', '2023.02.28', '2023.03.31', '2023.04.30',
         '2023.05.31', '2023.06.30', '2023.07.31', '2023.08.31', '2023.09.3
         0', '2023.10.31', '2023.11.30', '2023.12.31'])
```

```
In [49]: idx[0]
```

```
Out[49]: datetime.date(2023, 1, 31)
```

```
In [50]: idx[-1]
```

```
Out[50]: datetime.date(2023, 12, 31)
```

```
In [ ]:
```

개별 인덱스 생성

- 개별적인 날짜 값을 리스트로 묶어서 Index를 생성할 수 있음.

```
In [51]: from datetime import date as D
```

```
In [52]: D(2023, 1, 31)
```

```
Out[52]: datetime.date(2023, 1, 31)
```

```
In [53]: idx = Index([D(2023, 1, 31), D(2023, 2, 28), D(2023, 3, 31)])
         idx
```

```
Out[53]: DateIndex(['2023.01.31', '2023.02.28', '2023.03.31'])
```

```
In [54]: idx[0]
```

```
Out[54]: datetime.date(2023, 1, 31)
```

```
In [55]: idx[-1]
```

```
Out[55]: datetime.date(2023, 3, 31)
```

```
In [ ]:
```

```
In [56]: idx = Index([D(2023, 1, 10), D(2023, 2, 10), D(2023, 3, 10)])
         idx
```

```
Out[56]: DateIndex(['2023.01.10', '2023.02.10', '2023.03.10'])
```

In [ ]:

실제 모델 상 설정 예

In [57]:
```python
idx = Index('2023.01', 30)              # 총 사업기간
idx.mtrt = 26 #maturity                 # 대출 만기
idx.loan = Index('2023.03', idx.mtrt + 1) # 대출 기간
idx.cstrn = Index('2023.04', 24)        # 공사기간
```

In [58]:
```python
idx
```

Out[58]:
```
DateIndex(['2023.01.31', '2023.02.28', '2023.03.31', '2023.04.30',
'2023.05.31', '2023.06.30', '2023.07.31', '2023.08.31', '2023.09.3
0', '2023.10.31', '2023.11.30', '2023.12.31', '2024.01.31', '2024.
02.29', '2024.03.31', '2024.04.30', '2024.05.31', '2024.06.30', '2
024.07.31', '2024.08.31', '2024.09.30', '2024.10.31', '2024.11.30'
, '2024.12.31', '2025.01.31', '2025.02.28', '2025.03.31', '2025.04
.30', '2025.05.31', '2025.06.30'])
```

In [59]:
```python
idx.loan
```

Out[59]:
```
DateIndex(['2023.03.31', '2023.04.30', '2023.05.31', '2023.06.30',
'2023.07.31', '2023.08.31', '2023.09.30', '2023.10.31', '2023.11.3
0', '2023.12.31', '2024.01.31', '2024.02.29', '2024.03.31', '2024.
04.30', '2024.05.31', '2024.06.30', '2024.07.31', '2024.08.31', '2
024.09.30', '2024.10.31', '2024.11.30', '2024.12.31', '2025.01.31'
, '2025.02.28', '2025.03.31', '2025.04.30', '2025.05.31'])
```

In [60]:
```python
idx.cstrn
```

Out[60]:
```
DateIndex(['2023.04.30', '2023.05.31', '2023.06.30', '2023.07.31',
'2023.08.31', '2023.09.30', '2023.10.31', '2023.11.30', '2023.12.3
1', '2024.01.31', '2024.02.29', '2024.03.31', '2024.04.30', '2024.
05.31', '2024.06.30', '2024.07.31', '2024.08.31', '2024.09.30', '2
024.10.31', '2024.11.30', '2024.12.31', '2025.01.31', '2025.02.28'
, '2025.03.31'])
```

In [ ]:

2. Account 모듈

간략 사용 예

```
In [68]:  공사비 = Account(idx)
          공사비
```

```
Out[68]:  Account(main, len 30)
```

```
In [70]:  공사비.df
```

Out[70]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	0.0	0.0	0.0
2023-02-28	0.0	0.0	0.0	0.0
2023-03-31	0.0	0.0	0.0	0.0
2023-04-30	0.0	0.0	0.0	0.0
2023-05-31	0.0	0.0	0.0	0.0
2023-06-30	0.0	0.0	0.0	0.0
2023-07-31	0.0	0.0	0.0	0.0
2023-08-31	0.0	0.0	0.0	0.0
2023-09-30	0.0	0.0	0.0	0.0
2023-10-31	0.0	0.0	0.0	0.0
2023-11-30	0.0	0.0	0.0	0.0
2023-12-31	0.0	0.0	0.0	0.0
2024-01-31	0.0	0.0	0.0	0.0
2024-02-29	0.0	0.0	0.0	0.0
2024-03-31	0.0	0.0	0.0	0.0
2024-04-30	0.0	0.0	0.0	0.0
2024-05-31	0.0	0.0	0.0	0.0
2024-06-30	0.0	0.0	0.0	0.0
2024-07-31	0.0	0.0	0.0	0.0
2024-08-31	0.0	0.0	0.0	0.0
2024-09-30	0.0	0.0	0.0	0.0
2024-10-31	0.0	0.0	0.0	0.0
2024-11-30	0.0	0.0	0.0	0.0
2024-12-31	0.0	0.0	0.0	0.0
2025-01-31	0.0	0.0	0.0	0.0
2025-02-28	0.0	0.0	0.0	0.0
2025-03-31	0.0	0.0	0.0	0.0
2025-04-30	0.0	0.0	0.0	0.0
2025-05-31	0.0	0.0	0.0	0.0
2025-06-30	0.0	0.0	0.0	0.0

In [ ]:

In [69]:
```python
공사비.총공사비 = 30_000
```

In [63]:
```python
공사비.공사기간 = len(idx.cstrn)
공사비.공사기간
```

Out[63]: 24

In [64]:
```python
공사비.월별공사비 = 공사비.총공사비 / 공사비.공사기간
공사비.월별공사비
```

Out[64]: 1250.0

In [65]:
```python
공사비리스트 = [공사비.월별공사비] * 공사비.공사기간
공사비리스트
```

Out[65]:
```
[1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0,
 1250.0]
```

In [66]:
```python
공사비.addamt(idx.cstrn, 공사비리스트)
```

In [67]:
```python
공사비.df
```

Out[67]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	0.0	0.0	0.0
2023-02-28	0.0	0.0	0.0	0.0
2023-03-31	0.0	0.0	0.0	0.0
2023-04-30	0.0	1250.0	0.0	1250.0
2023-05-31	1250.0	1250.0	0.0	2500.0
2023-06-30	2500.0	1250.0	0.0	3750.0
2023-07-31	3750.0	1250.0	0.0	5000.0
2023-08-31	5000.0	1250.0	0.0	6250.0
2023-09-30	6250.0	1250.0	0.0	7500.0
2023-10-31	7500.0	1250.0	0.0	8750.0
2023-11-30	8750.0	1250.0	0.0	10000.0
2023-12-31	10000.0	1250.0	0.0	11250.0
2024-01-31	11250.0	1250.0	0.0	12500.0
2024-02-29	12500.0	1250.0	0.0	13750.0
2024-03-31	13750.0	1250.0	0.0	15000.0
2024-04-30	15000.0	1250.0	0.0	16250.0
2024-05-31	16250.0	1250.0	0.0	17500.0
2024-06-30	17500.0	1250.0	0.0	18750.0
2024-07-31	18750.0	1250.0	0.0	20000.0
2024-08-31	20000.0	1250.0	0.0	21250.0
2024-09-30	21250.0	1250.0	0.0	22500.0
2024-10-31	22500.0	1250.0	0.0	23750.0
2024-11-30	23750.0	1250.0	0.0	25000.0
2024-12-31	25000.0	1250.0	0.0	26250.0
2025-01-31	26250.0	1250.0	0.0	27500.0
2025-02-28	27500.0	1250.0	0.0	28750.0
2025-03-31	28750.0	1250.0	0.0	30000.0
2025-04-30	30000.0	0.0	0.0	30000.0
2025-05-31	30000.0	0.0	0.0	30000.0
2025-06-30	30000.0	0.0	0.0	30000.0

In [ ]:

실제 모델 상 사용 예

```
In [56]:  공사비 = Account(idx)
          with 공사비 as c:
              c.amt = 30_000 #총공사비
              c.prd = len(idx.cstrn) #공사기간
              c.amtunt = c.amt / c.prd #월별 공사비
              c.untlst = [c.amtunt] * c.prd #월별 공사비의 리스트
              c.addamt(idx.cstrn, c.untlst) #cstrn index에 공사비 리스트를 매칭시켜
          입력
```

```
In [57]:  공사비.df
```

실제 모델 상 사용 예

Out[57]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	0.0	0.0	0.0
2023-02-28	0.0	0.0	0.0	0.0
2023-03-31	0.0	0.0	0.0	0.0
2023-04-30	0.0	1250.0	0.0	1250.0
2023-05-31	1250.0	1250.0	0.0	2500.0
2023-06-30	2500.0	1250.0	0.0	3750.0
2023-07-31	3750.0	1250.0	0.0	5000.0
2023-08-31	5000.0	1250.0	0.0	6250.0
2023-09-30	6250.0	1250.0	0.0	7500.0
2023-10-31	7500.0	1250.0	0.0	8750.0
2023-11-30	8750.0	1250.0	0.0	10000.0
2023-12-31	10000.0	1250.0	0.0	11250.0
2024-01-31	11250.0	1250.0	0.0	12500.0
2024-02-29	12500.0	1250.0	0.0	13750.0
2024-03-31	13750.0	1250.0	0.0	15000.0
2024-04-30	15000.0	1250.0	0.0	16250.0
2024-05-31	16250.0	1250.0	0.0	17500.0
2024-06-30	17500.0	1250.0	0.0	18750.0
2024-07-31	18750.0	1250.0	0.0	20000.0
2024-08-31	20000.0	1250.0	0.0	21250.0
2024-09-30	21250.0	1250.0	0.0	22500.0
2024-10-31	22500.0	1250.0	0.0	23750.0
2024-11-30	23750.0	1250.0	0.0	25000.0
2024-12-31	25000.0	1250.0	0.0	26250.0
2025-01-31	26250.0	1250.0	0.0	27500.0
2025-02-28	27500.0	1250.0	0.0	28750.0
2025-03-31	28750.0	1250.0	0.0	30000.0
2025-04-30	30000.0	0.0	0.0	30000.0
2025-05-31	30000.0	0.0	0.0	30000.0
2025-06-30	30000.0	0.0	0.0	30000.0

```
In [68]: print("총공사비 : ", 공사비.amt)
         print("공사기간 : ", 공사비.prd)
         print("월공사비 : ", 공사비.amtunt)
```

```
총공사비 :  30000
공사기간 :  24
월공사비 :  1250.0
```

In [ ]:

Account 모듈 개요

- 재무모델에서 사용되는 현금흐름의 유출입을 기록 및 관리 할 목적으로 작성된 모듈
- 생성과 동시에 "df"라는 이름(DataFrame)의 데이터프레임과 "jnl" 이름(Journal)의 데이터프레임 두개를 생성함

- **"df" :** 현금흐름의 유출입 내용을 정리하여 매기별 기초 현금과 현금의 유출입, 기말 현금을 기록함. 예상 현금흐름의 유출입도 기록할 수 있음.
  - column의 내용 :
    - 'scd_in' : 계획된 현금 유입액
    - 'scd_in_cum' : 계획된 현금 누적유입액
    - 'scd_out' : 계획된 현금 유출액
    - 'scd_out_cum' : 계획된 현금 유출액
    - 'bal_strt' : 기초 현금
    - 'amt_in' : 실제 현금 유입액
    - 'amt_in_cum' : 실제 현금 누적유입액
    - 'amt_out' : 실제 현금 유출액
    - 'amt_out_cum' : 실제 현금 누적유출액
    - 'bal_end' : 기말 현금
    - 'rsdl_in_cum' : 계획된 현금 누적유입액 - 실제 현금 누적유입액
    - 'rsdl_out_cum' : 계획된 현금 누적유출액 - 실제 현금 누적유출액
  - "df"를 통해 출력시 "['bal_strt', 'amt_in', 'amt_out', 'bal_end']"의 4개 컬럼만 출력함.
  - "dfall"을 통해 출력시 전체 컬럼 출력

- **"jnl" :** 분개장에 기록하듯 개별 현금흐름의 유출입을 건건이 기록함.
  - columns의 내용 :
    - 'amt_in' : 현금 유입액
    - 'amt_out' : 현금 유출액
    - 'rcvfrm' : receive from, 어떠한 객체로 부터 현금이 유입되었는지
    - 'payto' : pay to, 어떠한 객체에게 현금을 유출하였는지
    - 'note' : 비고 사항
  - "jnlscd" : 계획된 현금 유출입의 기록

- 데이터 입력 메서드
  - addamt(*index*, *amount*) : _index_에 맞춰서 현금(*amount*)의 유입을 기록
  - subamt(*index*, *amount*) : _index_에 맞춰서 현금(*amount*)의 유출을 기록
  - addscd(*index*, *amount*) : _index_에 맞춰서 현금(*amount*)의 유입 계획을 기록
  - subscd(*index*, *amount*) : _index_에 맞춰서 현금(*amount*)의 유출 계획을 기록

- Account 객체 간 현금 유출입 기록
  - Account1.**send**(*index*, *amount*, *Account2*) : *index* 시점에 맞춰 *Account1* 객체에서 *Account2* 객체로 _amount_의 현금을 이동하는 것으로 기록
    - *Account1* : 현금(*amount*)의 유출을 기록
    - *Account2* : 현금(*amount*)의 유입을 기록

Account 모듈 예시

1) 사업기간 설정

```
In [113]: idx = Index('2023.01', 6)
          idx
```

```
Out[113]: DateIndex(['2023.01.31', '2023.02.28', '2023.03.31', '2023.04.30',
          '2023.05.31', '2023.06.30'])
```

2) 자금의 조달 설정

```
In [114]: eqt = Account(idx)
          eqt.amt = 30_000
          eqt.addamt(idx[0], eqt.amt) #최초 보유 현금 기록
          eqt.subscd(idx[1], 15_000) #첫번째 인출 계획 기록
          eqt.subscd(idx[3], 15_000) #두번째 인출 계획 기록
```

In [115]:
```
eqt.df
```

Out[115]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	30000.0	0.0	30000.0
2023-02-28	30000.0	0.0	0.0	30000.0
2023-03-31	30000.0	0.0	0.0	30000.0
2023-04-30	30000.0	0.0	0.0	30000.0
2023-05-31	30000.0	0.0	0.0	30000.0
2023-06-30	30000.0	0.0	0.0	30000.0

In [116]:
```
eqt.dfall
```

Out[116]:

	scd_in	scd_in_cum	scd_out	scd_out_cum	bal_strt	amt_in	amt_in_cum	amt_out
2023-01-31	0.0	0.0	0.0	0.0	0.0	30000.0	30000.0	0.0
2023-02-28	0.0	0.0	15000.0	15000.0	30000.0	0.0	30000.0	0.0
2023-03-31	0.0	0.0	0.0	15000.0	30000.0	0.0	30000.0	0.0
2023-04-30	0.0	0.0	15000.0	30000.0	30000.0	0.0	30000.0	0.0
2023-05-31	0.0	0.0	0.0	30000.0	30000.0	0.0	30000.0	0.0
2023-06-30	0.0	0.0	0.0	30000.0	30000.0	0.0	30000.0	0.0

3) 사업비 지출 계획 설정

In [117]:
```
cst = Account(idx)
cst.amt = 28_000 #총 사업비 금액
cst.addscd(idx[1], 8_000)
cst.addscd(idx[2], 7_000)
cst.addscd(idx[3], 7_000)
cst.addscd(idx[4], 6_000)
```

In [118]: `cst.df`

Out[118]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	0.0	0.0	0.0
2023-02-28	0.0	0.0	0.0	0.0
2023-03-31	0.0	0.0	0.0	0.0
2023-04-30	0.0	0.0	0.0	0.0
2023-05-31	0.0	0.0	0.0	0.0
2023-06-30	0.0	0.0	0.0	0.0

In [119]: `cst.dfall`

Out[119]:

	scd_in	scd_in_cum	scd_out	scd_out_cum	bal_strt	amt_in	amt_in_cum	amt_out
2023-01-31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2023-02-28	8000.0	8000.0	0.0	0.0	0.0	0.0	0.0	0.0
2023-03-31	7000.0	15000.0	0.0	0.0	0.0	0.0	0.0	0.0
2023-04-30	7000.0	22000.0	0.0	0.0	0.0	0.0	0.0	0.0
2023-05-31	6000.0	28000.0	0.0	0.0	0.0	0.0	0.0	0.0
2023-06-30	0.0	28000.0	0.0	0.0	0.0	0.0	0.0	0.0

4) 운영계좌 설정

In [120]: `oprtg = Account(idx)`

```
In [121]:  oprtg.df
```

Out[121]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	0.0	0.0	0.0
2023-02-28	0.0	0.0	0.0	0.0
2023-03-31	0.0	0.0	0.0	0.0
2023-04-30	0.0	0.0	0.0	0.0
2023-05-31	0.0	0.0	0.0	0.0
2023-06-30	0.0	0.0	0.0	0.0

5) 현금흐름 추정

```
In [122]:  # idx[0] : Nothing
```

```
In [123]:  # idx[1]
           eqt.send(idx[1], eqt.scd_out[idx[1]], oprtg, "자기자본 인출")
           oprtg.send(idx[1], cst.scd_in[idx[1]], cst, "사업비 지급")
```

```
In [124]:  # idx[2]
           eqt.send(idx[2], eqt.scd_out[idx[2]], oprtg, "자기자본 인출")
           oprtg.send(idx[2], cst.scd_in[idx[2]], cst, "사업비 지급")
```

```
In [125]:  # idx[3]
           eqt.send(idx[3], eqt.scd_out[idx[3]], oprtg, "자기자본 인출")
           oprtg.send(idx[3], cst.scd_in[idx[3]], cst, "사업비 지급")
```

```
In [126]:  # idx[4]
           eqt.send(idx[4], eqt.scd_out[idx[4]], oprtg, "자기자본 인출")
           oprtg.send(idx[4], cst.scd_in[idx[4]], cst, "사업비 지급")
```

```
In [127]:  # idx[5]
           eqt.send(idx[5], eqt.scd_out[idx[5]], oprtg, "자기자본 인출")
           oprtg.send(idx[5], cst.scd_in[idx[5]], cst, "사업비 지급")
```

In [128]: `oprtg.jnl`

Out[128]:

	amt_in	amt_out	rcvfrm	payto	note
2023-02-28	15000.0	0	None	None	자기자본 인출
2023-02-28	0.0	8000.0	None	None	사업비 지급
2023-03-31	0.0	7000.0	None	None	사업비 지급
2023-04-30	15000.0	0	None	None	자기자본 인출
2023-04-30	0.0	7000.0	None	None	사업비 지급
2023-05-31	0.0	6000.0	None	None	사업비 지급

In [129]: `eqt.df`

Out[129]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	30000.0	0.0	30000.0
2023-02-28	30000.0	0.0	15000.0	15000.0
2023-03-31	15000.0	0.0	0.0	15000.0
2023-04-30	15000.0	0.0	15000.0	0.0
2023-05-31	0.0	0.0	0.0	0.0
2023-06-30	0.0	0.0	0.0	0.0

In [130]: `eqt.jnl`

Out[130]:

	amt_in	amt_out	rcvfrm	payto	note
2023-01-31	30000	0	None	None	add_amt
2023-02-28	0	15000.0	None	None	자기자본 인출
2023-04-30	0	15000.0	None	None	자기자본 인출

In [131]:
```
cst.df
```

Out[131]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	0.0	0.0	0.0
2023-02-28	0.0	8000.0	0.0	8000.0
2023-03-31	8000.0	7000.0	0.0	15000.0
2023-04-30	15000.0	7000.0	0.0	22000.0
2023-05-31	22000.0	6000.0	0.0	28000.0
2023-06-30	28000.0	0.0	0.0	28000.0

In [132]:
```
cst.jnl
```

Out[132]:

	amt_in	amt_out	rcvfrm	payto	note
2023-02-28	8000.0	0	None	None	사업비 지급
2023-03-31	7000.0	0	None	None	사업비 지급
2023-04-30	7000.0	0	None	None	사업비 지급
2023-05-31	6000.0	0	None	None	사업비 지급

6) 현금흐름 추정(for 문 사용)

In [135]:
```python
# 현금흐름 조건 재설정
idx = Index('2023.01', 6)

eqt = Account(idx)
eqt.amt = 30_000
eqt.addamt(idx[0], eqt.amt) #최초 보유 현금 기록
eqt.subscd(idx[1], 15_000) #첫번째 인출 계획 기록
eqt.subscd(idx[3], 15_000) #두번째 인출 계획 기록

cst = Account(idx)
cst.amt = 28_000
cst.addscd(idx[1], 8_000)
cst.addscd(idx[2], 7_000)
cst.addscd(idx[3], 7_000)
cst.addscd(idx[4], 6_000)

oprtg = Account(idx)
```

In [136]:
```python
for i in range(0, 6):
    eqt.send(idx[i], eqt.scd_out[idx[i]], oprtg, "자기자본 인출")
    oprtg.send(idx[i], cst.scd_in[idx[i]], cst, "사업비 지급")
```

In [138]: `oprtg.df`

Out[138]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	0.0	0.0	0.0
2023-02-28	0.0	15000.0	8000.0	7000.0
2023-03-31	7000.0	0.0	7000.0	0.0
2023-04-30	0.0	15000.0	7000.0	8000.0
2023-05-31	8000.0	0.0	6000.0	2000.0
2023-06-30	2000.0	0.0	0.0	2000.0

In [139]: `oprtg.jnl`

Out[139]:

	amt_in	amt_out	rcvfrm	payto	note
2023-02-28	15000.0	0	None	None	자기자본 인출
2023-02-28	0.0	8000.0	None	None	사업비 지급
2023-03-31	0.0	7000.0	None	None	사업비 지급
2023-04-30	15000.0	0	None	None	자기자본 인출
2023-04-30	0.0	7000.0	None	None	사업비 지급
2023-05-31	0.0	6000.0	None	None	사업비 지급

In [141]: `cst.dfall`

Out[141]:

	scd_in	scd_in_cum	scd_out	scd_out_cum	bal_strt	amt_in	amt_in_cum	amt_out
2023-01-31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2023-02-28	8000.0	8000.0	0.0	0.0	0.0	8000.0	8000.0	0.0
2023-03-31	7000.0	15000.0	0.0	0.0	8000.0	7000.0	15000.0	0.0
2023-04-30	7000.0	22000.0	0.0	0.0	15000.0	7000.0	22000.0	0.0
2023-05-31	6000.0	28000.0	0.0	0.0	22000.0	6000.0	28000.0	0.0
2023-06-30	0.0	28000.0	0.0	0.0	28000.0	0.0	28000.0	0.0

7) 현금흐름 추정(for 문 사용 2)

```
In [142]:    # 현금흐름 조건 재설정
             idx = Index('2023.01', 6)

             eqt = Account(idx)
             eqt.amt = 30_000
             eqt.addamt(idx[0], eqt.amt)  #최초 보유 현금 기록
             eqt.subscd(idx[1], 15_000)  #첫번째 인출 계획 기록
             eqt.subscd(idx[3], 15_000)  #두번째 인출 계획 기록

             cst = Account(idx)
             cst.amt = 28_000
             cst.addscd(idx[1], 8_000)
             cst.addscd(idx[2], 7_000)
             cst.addscd(idx[3], 7_000)
             cst.addscd(idx[4], 6_000)

             oprtg = Account(idx)
```

```
In [143]:    for i in idx:
                 eqt.send(i, eqt.scd_out[i], oprtg, "자기자본 인출")
                 oprtg.send(i, cst.scd_in[i], cst, "사업비 지급")
```

```
In [144]:    oprtg.df
```

Out[144]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	0.0	0.0	0.0
2023-02-28	0.0	15000.0	8000.0	7000.0
2023-03-31	7000.0	0.0	7000.0	0.0
2023-04-30	0.0	15000.0	7000.0	8000.0
2023-05-31	8000.0	0.0	6000.0	2000.0
2023-06-30	2000.0	0.0	0.0	2000.0

```
In [ ]:
```