[5-4. 자금의 조달 모델링]

In [1]:
```
import pandas as pd
from pandas import DataFrame, Series
import cafle as cf
from cafle import Index, Account
from cafle import Setattr, round_up
```

In [2]:
```
from practice.astn0_overview import overview, idx
```

In [ ]:
```
```

1. Equity 조달 설정

In [3]:
```
equity = Account(idx)
```

In [4]:
```
equity.보통주 = equity.subacc('보통주')
with equity.보통주 as e:
    e.amt = 5_000#백만원
    e.rate = 1.0#지분율 100%
    e.subscd(idx[0], e.amt, note="보통주 출자금")
```

2. Equity 조달 함수

1) Equity 인출

In [5]:
```
@Setattr(equity)
def withdraw_equity_amount(equity, idxno, oprtg):
    amt_wtdrw = 0
    for key, item in equity.dct.items():
        _df = item.jnlscd.loc[item.jnlscd.index == idxno]
        for index, row in _df.iterrows():
            item.send(idxno, row.amt_out, oprtg, "자기자본 납입")
            amt_wtdrw += row.amt_out
    return amt_wtdrw
```

In [ ]:
```
```

In [6]:
```
equity.보통주.dfall
```

Out[6]:

	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	5000.0	5000.0	0.0	0.0	0.0	0.0
2023-02-28	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2023-03-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2023-04-30	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2023-05-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2023-06-30	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2023-07-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2023-08-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2023-09-30	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2023-10-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2023-11-30	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2023-12-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2024-01-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2024-02-29	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2024-03-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2024-04-30	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2024-05-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2024-06-30	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2024-07-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2024-08-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0

2024-09-30	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2024-10-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2024-11-30	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2024-12-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2025-01-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2025-02-28	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2025-03-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2025-04-30	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2025-05-31	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0
2025-06-30	0.0	0.0	0.0	5000.0	0.0	0.0	0.0	0.0

In [7]:
```
oprtg = Account(idx)
```

In [8]:
```
equity.withdraw_equity_amount(idx[0], oprtg)
```

Out[8]:
```
5000
```

In [9]:
```
equity.보통주.dfall
```

Out[9]:

	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	5000.0	5000.0	0.0	0.0	0.0	5000.0
2023-02-28	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2023-03-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2023-04-30	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2023-05-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2023-06-30	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2023-	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0

07-31								
2023-08-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2023-09-30	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2023-10-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2023-11-30	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2023-12-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2024-01-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2024-02-29	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2024-03-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2024-04-30	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2024-05-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2024-06-30	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2024-07-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2024-08-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2024-09-30	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2024-10-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2024-11-30	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2024-12-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2025-01-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2025-02-28	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2025-03-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2025-04-30	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0

2025-05-31	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0
2025-06-30	0.0	0.0	0.0	5000.0	-5000.0	0.0	0.0	0.0

In [10]:
```
oprtg.df
```

Out[10]:

	bal_strt	amt_in	amt_out	bal_end
2023-01-31	0.0	5000.0	0.0	5000.0
2023-02-28	5000.0	0.0	0.0	5000.0
2023-03-31	5000.0	0.0	0.0	5000.0
2023-04-30	5000.0	0.0	0.0	5000.0
2023-05-31	5000.0	0.0	0.0	5000.0
2023-06-30	5000.0	0.0	0.0	5000.0
2023-07-31	5000.0	0.0	0.0	5000.0
2023-08-31	5000.0	0.0	0.0	5000.0
2023-09-30	5000.0	0.0	0.0	5000.0
2023-10-31	5000.0	0.0	0.0	5000.0
2023-11-30	5000.0	0.0	0.0	5000.0
2023-12-31	5000.0	0.0	0.0	5000.0
2024-01-31	5000.0	0.0	0.0	5000.0
2024-02-29	5000.0	0.0	0.0	5000.0
2024-03-31	5000.0	0.0	0.0	5000.0
2024-04-30	5000.0	0.0	0.0	5000.0
2024-05-31	5000.0	0.0	0.0	5000.0
2024-06-30	5000.0	0.0	0.0	5000.0
2024-07-31	5000.0	0.0	0.0	5000.0
2024-08-31	5000.0	0.0	0.0	5000.0
2024-09-30	5000.0	0.0	0.0	5000.0
2024-10-31	5000.0	0.0	0.0	5000.0
2024-11-30	5000.0	0.0	0.0	5000.0
2024-12-31	5000.0	0.0	0.0	5000.0
2025-01-31	5000.0	0.0	0.0	5000.0
2025-02-28	5000.0	0.0	0.0	5000.0
2025-03-31	5000.0	0.0	0.0	5000.0
2025-04-30	5000.0	0.0	0.0	5000.0
2025-05-31	5000.0	0.0	0.0	5000.0
2025-06-30	5000.0	0.0	0.0	5000.0

In [ ]:

2) Equity 상환

In [11]:
```
@Setattr(equity)
def repay_equity_amt(equity, idxno, oprtg):
    if idxno < idx[-1]:
        return 0
    amt_bal = oprtg.bal_end[idxno]
    for key, item in equity.dct.items():
        amt_rpy = item.rate * amt_bal
        oprtg.send(idxno, amt_rpy, item, note=f"자기자본상환({item.name
})")
    return amt_bal
```

In [ ]:
```

```

2. Loan 조달 설정

In [12]:
```
loan = Account(idx)
```

In [13]:
```
with loan as l:
    l.mtrt = idx.mtrt
    l.is_repaid_all = False
```

In [14]:
```
loan.tra = loan.subacc('tra')
with loan.tra as tra:
    tra.rank = 0
    tra.is_wtdrbl = False
    tra.is_repaid = False

    tra.ntnl = tra.subacc('ntnl')
    with tra.ntnl as n:
        n.amt = 40_000#백만원
        n.intlamt = 5_000#백만원
        n.subscd(idx.loan[0], n.amt)
        n.addscd(idx.loan[-1], n.amt)

    tra.IR = tra.subacc('IR')
    with tra.IR as i:
        i.rate = 0.06
        i.cycle = 1#개월
        i.rate_cycle = i.rate / 12 * i.cycle

    tra.fee = tra.subacc('fee')
    with tra.fee as f:
        f.rate = 0.02
```

In [15]:
```
loan.trb = loan.subacc('trb')
with loan.trb as trb:
    trb.rank = 1
    trb.is_wtdrbl = False
    trb.is_repaid = False

    trb.ntnl = trb.subacc('ntnl')
    with trb.ntnl as n:
        n.amt = 5_000#백만원
        n.intlamt = 5_000#백만원
        n.subscd(idx.loan[0], n.amt)
        n.addscd(idx.loan[-1], n.amt)

    trb.IR = trb.subacc('IR')
    with trb.IR as i:
        i.rate = 0.09
        i.cycle = 1#개월
        i.rate_cycle = i.rate / 12 * i.cycle

    trb.fee = trb.subacc('fee')
    with trb.fee as f:
        f.rate = 0.06
```

In [ ]:
```

```

In [16]:
```
loan
```

Out[16]:
```
Account(main, len 30, dct: ['tra', 'trb'])
```

In [17]:
```
loan.vars
```

Out[17]:
```
['index', 'name', 'mtrt', 'is_repaid_all', 'tra', 'trb']
```

In [18]:
```
loan.mtrt
```

Out[18]:
```
26
```

In [19]:
```
loan.is_repaid_all
```

Out[19]:
```
False
```

In [20]:
```
loan.tra
```

Out[20]:
```
Account(tra, len 30, dct: ['ntnl', 'IR', 'fee'])
```

In [21]:
```
loan.tra.ntnl.dfall
```

Out[21]:

	scd_in	scd_in_cum	scd_out	scd_out_cum	bal_strt	amt_in	amt_in_cum	amt_out
2023-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

01-31								
2023-02-28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2023-03-31	0.0	0.0	40000.0	40000.0	0.0	0.0	0.0	0.0
2023-04-30	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2023-05-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2023-06-30	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2023-07-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2023-08-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2023-09-30	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2023-10-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2023-11-30	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2023-12-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2024-01-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2024-02-29	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2024-03-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2024-04-30	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2024-05-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2024-06-30	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2024-07-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2024-08-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2024-09-30	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2024-10-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0

2024-11-30	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2024-12-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2025-01-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2025-02-28	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2025-03-31	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2025-04-30	0.0	0.0	0.0	40000.0	0.0	0.0	0.0	0.0
2025-05-31	40000.0	40000.0	0.0	40000.0	0.0	0.0	0.0	0.0
2025-06-30	0.0	40000.0	0.0	40000.0	0.0	0.0	0.0	0.0

In [ ]:

3. Loan 조달 함수

1) 하위 Loan 추출 함수

In [22]:
```
@Setattr(loan)
def getloan(loan, reverse=False):
    lst = list(loan.dct.values())
    fn = lambda x: x.rank
    lst.sort(key = fn, reverse = reverse)
    for ln in lst:
        yield ln
```

In [23]:
```
list(loan.getloan())
```

Out[23]:
```
[Account(tra, len 30, dct: ['ntnl', 'IR', 'fee']),
 Account(trb, len 30, dct: ['ntnl', 'IR', 'fee'])]
```

In [24]:
```
list(loan.getloan(reverse=True))
```

Out[24]:
```
[Account(trb, len 30, dct: ['ntnl', 'IR', 'fee']),
 Account(tra, len 30, dct: ['ntnl', 'IR', 'fee'])]
```

2) 금융비용 추정 함수

In [25]:
```
@Setattr(loan.dct)
def estimate_fee_amt(ln, idxno):
    if idxno == idx.loan[0]:
        feeamt = ln.ntnl.amt * ln.fee.rate
        ln.fee.addscd(idxno, feeamt, note=f"수수료({ln.name})")
        return feeamt
    return 0

@Setattr(loan.dct)
def estimate_IR_amt(ln, idxno):
    if ln.is_wtdrbl is False:
        return 0
    if ln.is_repaid is True:
        return 0
    ntnlbal = -ln.ntnl.bal_strt[idxno]
    IRamt = ntnlbal * ln.IR.rate_cycle
    if IRamt > 0.0:
        ln.IR.addscd(idxno, IRamt, note=f"이자({ln.name})")
        return IRamt
    return 0
```

3) 대출원금 인출

In [26]:
```
@Setattr(loan.dct)
def set_loan_withdrawable(ln, idxno):
    if idxno == idx.loan[0]:
        ln.is_wtdrbl = True

@Setattr(loan.dct)
def withdraw_ntnl_fixed(ln, acc, idxno):
    if idxno != idx.loan[0]:
        return 0
    amt_wtdrw = ln.ntnl.intlamt
    ln.ntnl.send(idxno, amt_wtdrw, acc, note=f"일시대출금({ln.name})")
    return amt_wtdrw

@Setattr(loan.dct)
def withdraw_ntnl_flexible(ln, acctmp, acc, idxno):
    if idxno < idx.loan[0]:
        return 0
    if ln.is_wtdrbl is False:
        return 0
    if ln.is_repaid is True:
        return 0
    amttopay = acctmp.scd_out[idxno] - acctmp.bal_end[idxno]
    amttopay = max(round_up(amttopay, -2), 0)
    amtscdout = ln.ntnl.rsdl_out_cum[idxno]

    amt_wtdrw = min(amttopay, amtscdout)
    ln.ntnl.send(idxno, amt_wtdrw, acc, note=f"한도대출금({ln.name})")
    return amt_wtdrw
```

4) 금융비용 지출

In [27]:
```
@Setattr(loan.dct)
def pay_fee_amt(ln, acc, idxno):
    feeamt = ln.fee.scd_in[idxno]
    acc.send(idxno, feeamt, ln.fee, note=f"수수료({ln.name})")
    return feeamt

@Setattr(loan.dct)
def pay_IR_amt(ln, acc, idxno):
    IRamt = ln.IR.scd_in[idxno]
    acc.send(idxno, IRamt, ln.IR, note=f"이자({ln.name})")
    return IRamt
```

5) 대출원금 상환

In [28]:
```
@Setattr(loan.dct)
def repay_ntnl_amt(ln, acc, idxno):
    #at maturity
    if idxno >= idx.loan[-1]:
        amt_scd_in = ln.ntnl.rsdl_in_cum[idxno] - ln.ntnl.rsdl_out_
cum[idxno]
        acc.send(idxno, amt_scd_in, ln.ntnl, note=f"대출금상환({ln.nam
e})")
        return amt_scd_in
    return 0

@Setattr(loan.dct)
def setback_loan_unwithdrawable(ln, idxno):
    #at maturity
    if idxno >= idx.loan[-1]:
        ln.is_wtdrbl = False
```

In [ ]:
```

```