

[5-6. 구현된 현금흐름의 엑셀 출력]

```
In [1]: from datetime import date
import pandas as pd
from pandas import DataFrame, Series
from cafile import Write, extnddct
```

```
In [2]: from practice.cashflow import overview, idx, acc, sales, cost, equity, loan
```

```
In [ ]:
```

1. 사업개요 출력 연습

1) 출력할 데이터 확인

```
In [3]: overview
```

```
Out[3]: {'business': {'사업명': ['○○동 □□오피스텔 개발사업'],
'위치': ['서울시 ㄷㄷ구 ○○동 123번지 일원'],
'지역지구': ['일반상업지구, 지구단위계획구역'],
'주용도': ['업무시설(오피스텔), 판매시설'],
'규모': ['지상15층 / 지하3층'],
'주구조': ['철근콘크리트구조'],
'주차대수': ['법정 161대', '계획 161대']},
'area':      대지면적   건축면적   연면적   용적률연면적
m2   1500    750  13000    9500
평     454    227   3933    2874}
```

```
In [4]: #dictionary
overview['business']
```

```
Out[4]: {'사업명': ['○○동 □□오피스텔 개발사업'],
'위치': ['서울시 ㄷㄷ구 ○○동 123번지 일원'],
'지역지구': ['일반상업지구, 지구단위계획구역'],
'주용도': ['업무시설(오피스텔), 판매시설'],
'규모': ['지상15층 / 지하3층'],
'주구조': ['철근콘크리트구조'],
'주차대수': ['법정 161대', '계획 161대']}
```

```
In [5]: #DataFrame
overview['area']
```

```
Out[5]:
```

| | 대지면적 | 건축면적 | 연면적 | 용적률연면적 |
|----|------|------|-------|--------|
| m2 | 1500 | 750 | 13000 | 9500 |
| 평 | 454 | 227 | 3933 | 2874 |

2) 엑셀파일 및 sheet 생성

```
In [6]: #Excel Workbook 생성
wb = Write("exercise/Exercise_overview.xlsx")
```

```
In [7]: #Excel Sheet 생성
ws = wb.add_ws('overview')
wb.ws['overview'].set_column("A:X", 12)
```

```
Out[7]: 0
```

3) 사업개요 및 사업면적 출력(세로출력)

```
In [8]: ws("[사업개요]", wb.bold)
ws(overview['business'], fmtkey=wb.bold, fmt=wb.nml, valdrtn='col',
drtn='col')
ws.nextcell(1)
```

```
Out[8]: Cell(9, 0)
```

```
In [9]: ws("[사업면적]", wb.bold)
ws(overview['area'], fmtkey=wb.bold, fmt=wb.nml, valdrtn='col', drtn='col')
```

```
Out[9]: Cell(15, 0)
```

4) 엑셀 sheet2 생성

```
In [10]: #Excel Sheet 생성
ws = wb.add_ws('overview2')
wb.ws['overview2'].set_column("A:K", 12)
```

```
Out[10]: 0
```

5) 사업개요 및 사업면적 출력(가로출력)

```
In [11]: ws("[사업개요]", wb.bold)
         cell = ws(overview['business'], fmtkey=wb.bold, fmt=wb.nml, valdrtn
         = 'col', drtn='row')
```

```
In [12]: ws.setcell(0, cell.col + 1)
         ws("[사업면적]", wb.bold)
         ws(overview['area'], fmtkey=wb.bold, fmt=wb.nml, valdrtn='col', drt
         n='row')
```

```
Out[12]: Cell(1, 7)
```

6) 엑셀워크북 클로징

```
In [13]: wb.close()
```

```
Out[13]: True
```

=> "Exercise_overview.xlsx" 파일 생성 확인

```
In [ ]:
```

2. 사업개요 및 Assumption 데이터 출력

- 일반적으로 사업개요와 Assumption 데이터를 한 sheet에 출력함.
- 앞에서 한 사업개요와 함께 Assumption 데이터를 출력하고자 함.

1) 엑셀 워크북 및 sheet 생성

```
In [14]: #Excel Workbook 생성
         wb = Write("result/Result_Cashflow.xlsx")
```

```
In [15]: #Excel Sheet 생성
         ws = wb.add_ws("assumption")
         wb.ws["assumption"].set_column("A:K", 12)
```

```
Out[15]: 0
```

2) 머릿글 출력

```
In [16]: ws("ASSUMPTION", wb.bold)           #sheet 제목 출력
ws("Written at: " + wb.now)                  #작성 일시 출력
ws.nextcell(2)                               #두칸 띄우기
```

Out[16]: Cell(4, 0)

3) 사업개요 출력

```
In [17]: #사업개요 출력
ws("[Business Overview]", wb.bold)
ws(overview["business"], fmtkey=wb.bold, fmt=wb.nml,
      valdrtn='col', drtn='col')
ws.nextcell(1)
```

Out[17]: Cell(13, 0)

```
In [18]: #사업면적 출력
ws("[Business Area]", wb.bold)
ws(overview["area"], fmtkey=wb.bold, fmt=wb.nml,
      valdrtn='col', drtn='col')
ws.nextcell(1)
```

Out[18]: Cell(20, 0)

```
In [19]: #Index 출력
ws("[Index]", wb.bold)
fmt = [wb.bold, wb.month, wb.date, wb.date]
ws(["", "개월수", "시작일", "종료일"])
ws(["사업기간", len(idx) - 1, idx[0], idx[-1]], fmt)
ws(["매출기간", len(idx.sales), idx.sales[0], idx.sales[-1]], fmt)
ws(["대출기간", idx.mtrt, idx.loan[0], idx.loan[-1]], fmt)
ws(["건축기간", len(idx.cstrn), idx.cstrn[0], idx.cstrn[-1]], fmt)
ws.nextcell(1)
```

Out[19]: Cell(27, 0)

```
In [20]: #Equity 출력
ws("[Equity]", wb.bold)
ws({"구분": [_.name for _ in equity.dct.values()]},
    fmtkey=wb.bold, fmt=wb.nml, valdrtn='col')
    #=> key값은 bold체로, 데이터값은 일반글씨체로, 데이터값은 칼럼 방향으로 출력
ws({"출자금액": [_.amt for _ in equity.dct.values()]},
    fmtkey=wb.bold, fmt=wb.num, valdrtn='col')
ws.nextcell(1)
```

Out[20]: Cell(31, 0)

```
In [21]: #Loan 출력
ws("[Loan]", wb.bold)
ws({"구분": [_.name for _ in loan.dct.values()]},
    fmtkey=wb.bold, fmt=wb.nml, valdrtn='col')
ws({"순위": [_.rank for _ in loan.dct.values()]},
    fmtkey=wb.bold, fmt=wb.num, valdrtn='col')
ws({"대출금액": [_.ntnl.amt for _ in loan.dct.values()]},
    fmtkey=wb.bold, fmt=wb.num, valdrtn='col')
ws({"최초인출": [_.ntnl.intlamt for _ in loan.dct.values()]},
    fmtkey=wb.bold, fmt=wb.num, valdrtn='col')
ws({"한도인출": [_.ntnl.amt - _.ntnl.intlamt for _ in loan.dct.values()]},
    fmtkey=wb.bold, fmt=wb.num, valdrtn='col')
ws({"수수료율": [_.fee.rate for _ in loan.dct.values()]},
    fmtkey=wb.bold, fmt=wb.pct, valdrtn='col')
ws({"금리": [_.IR.rate for _ in loan.dct.values()]},
    fmtkey=wb.bold, fmt=wb.pct, valdrtn='col')
ws.nextcell(1)

ws(["만기", idx.mtrt], [wb.bold, wb.month])
ws(["총 대출금액", sum([ln.ntnl.amt for ln in loan.dct.values()])],
    [wb.bold, wb.num])
ws.nextcell(1)
```

Out[21]: Cell(43, 0)

4) 분양매출 가정 출력

```
In [22]: ws.setcell(4, 6)
```

```
In [23]: ws("[분양매출가정_오피스텔]", wb.bold)
slstbl = sales.분양테이블['오피']
slssum = DataFrame(slstbl.sum(), columns=['합계']).T
slstbl = pd.concat([slstbl, slssum])
ws(slstbl, fmtkey=wb.bold, fmt=wb.num, valdrtn='row', drtn='col')
ws.nextcell(1)
```

Out[23]: Cell(12, 6)

```
In [24]: ws("[분양매출가정_근생시설]", wb.bold)
slstbl = sales.분양테이블['근생']
slssum = DataFrame(slstbl.sum(), columns=['합계']).T
slstbl = pd.concat([slstbl, slssum])
ws(slstbl, fmtkey=wb.bold, fmt=wb.num, valdrtn='row', drtn='col')
ws.nextcell(1)
```

Out[24]: Cell(17, 6)

```
In [25]: ws("[분양매출금액]", wb.bold)
ws(sales.분양매출, fmtkey=wb.bold, fmt=wb.num, valdrtn='row', drtn='col')
ws.nextcell(1)
```

Out[25]: Cell(21, 6)

```
In [26]: ws("[분양대금 납입일정]", wb.bold)
tmpfmt = {'구분':wb.nml, '오피':wb.pct, '근생':wb.pct, '납입오피':wb.num,
'납입근생':wb.num, '납입소계':wb.num}
slstbl = sales.대금납입일정
slssum = DataFrame(slstbl.sum(), columns=['합계']).T
slstbl = pd.concat([slstbl, slssum])
ws(slstbl, fmtidx=wb.date, fmtkey=wb.bold, fmt=tmpfmt,
valdrtn='row', drtn='col')
ws.nextcell(1)
```

Out[26]: Cell(31, 6)

```
In [27]: ws("[분양률 가정]", wb.bold)
tmpfmt = {'오피':wb.pct, '근생':wb.pct, '계약오피':wb.num, '계약근생':wb.num,
'계약소계':wb.num}
slstbl = sales.분양률가정
slssum = DataFrame(slstbl.sum(), columns=['합계']).T
slstbl = pd.concat([slstbl, slssum])
ws(slstbl, fmtidx=wb.date, fmtkey=wb.bold, fmt=tmpfmt,
valdrtn='row', drtn='col')
```

Out[27]: Cell(39, 6)

In []:

3. Cashflow 출력

1) sheet 생성

```
In [28]: #Excel Sheet 생성
ws = wb.add_ws("cashflow")
wb.ws["cashflow"].set_column("A:A", 12)
```

Out[28]: 0

2) 머릿글 출력

```
In [29]: ws("CASH FLOW", wb.bold)
ws("Written at: " + wb.now)
ws.nextcell(2)
```

Out[29]: Cell(4, 0)

3) Cashflow data 출력

(1) Index 출력

```
In [30]: cell = ws("[Cashflow]", wb.bold)
ws.nextcell(2)
ws(idx, wb.date, valdrtn='col', drtn='col')
ws("합계", wb.nml)

ws.setcell(cell)
ws.nextcell(1, drtn='col')
```

Out[30]: Cell(5, 1)

(2) 출력하고자 하는 딕셔너리 데이터들을 묶어서 하나의 딕셔너리로 통합

```
In [31]: #사업비의 경우 3개의 층으로 데이터가 구성되어 있으므로, 미리 데이터 형태를 2개 층으로
#조정함.
사업비 = {key: {key2: list(item2.df.amt_in) for key2, item2 in item.d
ct.items()}} for key, item in cost.dct.items()}}
```

```
In [32]: cfdct = {
    '운영_기초': extnddct(
        {'기초잔액': acc.oprtg.df.bal_strt},
    ),
    '자금조달': extnddct(
        {'Eqt_'+key: list(item.amt_out) for key, item in equity.dct.items()},
        {'Loan_'+key: list(item.ntnl.df.amt_out) for key, item in loan.dct.items()}
    ),
    '분양매출': extnddct(
        {'분양_'+key: list(item.amt_out) for key, item in sales.dct.items()}
    ),
    '운영_유입': extnddct(
        {'현금유입': list(acc.oprtg.df.amt_in)},
    ),
    '금융비용': extnddct(
        {'Fee_'+key: list(item.fee.df.amt_in) for key, item in loan.dct.items()},
        {'IR_'+key: list(item.IR.df.amt_in) for key, item in loan.dct.items()}
    ),
    '**사업비',
    '상환_loan': extnddct(
        {'상환'+key: list(item.ntnl.df.amt_in) for key, item in loan.dct.items()}
    ),
    '상환_Eqt': extnddct(
        {'Eqt_'+key: list(item.df.amt_in) for key, item in equity.dct.items()}
    ),
    '운영_유출': extnddct(
        {'현금유출': acc.oprtg.df.amt_out},
    ),
    '운영_기말': extnddct(
        {'기말잔액': acc.oprtg.df.bal_end}
    ),
}
```

(3) 데이터 합계 계산


```
In [33]: dctsum = {}
         for key, dct in cfdct.items():
             dctsum[key] = {}
             for key2, srs in dct.items():
                 if key2 in ['기초잔액', '기말잔액']:
                     tmpval = '-'
                 else:
                     tmpval = sum(srs)
             dctsum[key][key2] = pd.concat([Series(srs), Series([tmpval])], index=['합계']))
```

(4) cashflow 데이터 출력

```
In [34]: ws(dctsum, fmtkey=wb.bold, fmt=wb.num, valdrtn='row', drtn='col')
         cell = ws.nextcell(2)
```

4) 요약 Cashflow 출력

(1) Index 출력

```
In [35]: ws.setcell(cell.row, 0)
         cell = ws("[요약 Cashflow]", wb.bold)
         ws.nextcell(2)
         ws(idx, wb.date, valdrtn='col', drtn='col')
         ws("합계", wb.nml)
         ws.setcell(cell)
         ws.nextcell(1, drtn='col')
```

```
Out[35]: Cell(41, 1)
```

(2) 출력하고자 하는 딕셔너리 데이터들을 묶어서 하나의 딕셔너리로 통합

```
In [36]: cfdct = {
    '운영_기초': extnoddct(
        {'기초잔액': acc.oprtg.df.bal_strt},
    ),
    '자금조달': extnoddct(
        {'Eqt_'+key: list(item.amt_out) for key, item in equity.dct.items()},
        {'Loan_'+key: list(item.ntnl.df.amt_out) for key, item in loan.dct.items()}
    ),
    '분양매출': extnoddct(
        {'분양_'+key: list(item.amt_out) for key, item in sales.dct.items()}
    ),
    '운영_유입': extnoddct(
        {'현금유입': list(acc.oprtg.df.amt_in)},
    ),
    '금융비용': extnoddct(
        {'Fee_'+key: list(item.fee.df.amt_in) for key, item in loan.dct.items()},
        {'IR_'+key: list(item.IR.df.amt_in) for key, item in loan.dct.items()}
    ),
    '사업비': extnoddct(
        {key: list(item.mrg.df.amt_in) for key, item in cost.dct.items()}
    ),
    '상환_loan': extnoddct(
        {'상환'+key: list(item.ntnl.df.amt_in) for key, item in loan.dct.items()}
    ),
    '상환_Eqt': extnoddct(
        {'Eqt_'+key: list(item.df.amt_in) for key, item in equity.dct.items()}
    ),
    '운영_유출': extnoddct(
        {'현금유출': acc.oprtg.df.amt_out},
    ),
    '운영_기말': extnoddct(
        {'기말잔액': acc.oprtg.df.bal_end}
    ),
}
```

(3) 데이터 합계 계산

```
In [37]: dctsum = {}
         for key, dct in cfdct.items():
             dctsum[key] = {}
             for key2, srs in dct.items():
                 if key2 in ['기초잔액', '기말잔액']:
                     tmpval = '-'
                 else:
                     tmpval = sum(srs)
             dctsum[key][key2] = pd.concat([Series(srs), Series([tmpval])], index=['합계'])
```

(4) cashflow 데이터 출력

```
In [38]: ws(dctsum, fmtkey=wb.bold, fmt=wb.num, valdrtn='row', drtn='col')
```

```
Out[38]: Cell(74, 1)
```

```
In [ ]:
```

4. Financing data 출력

1) sheet 생성

```
In [39]: #Excel Sheet 생성
         ws = wb.add_ws("financing")
         wb.ws["financing"].set_column("A:A", 12)
```

```
Out[39]: 0
```

2) 머릿글 출력

```
In [40]: ws("FINANCING", wb.bold)
         ws("Written at: " + wb.now)
         cell = ws.nextcell(2)
```

3) Index 출력

```
In [41]: ws.nextcell(3)
ws(idx, wb.date, valdrtn='col', drtn='col')
ws("합계", wb.nml)

ws.setcell(cell)
ws.nextcell(1, drtn='col')
```

```
Out[41]: Cell(4, 1)
```

4) Financing data 출력

(1) 빈 딕셔너리 생성

```
In [42]: fncdct = {}
```

(2) fncdct에 Loan 데이터 입력

```
In [43]: for key, ln in loan.dct.items():
    fncdct['Loan_'+key] = wb.dctprt_loan(ln)
    #dctprt_loan : 딕셔너리 형태로 Loan데이터를 출력하는 함수
```

(3) fncdct에 Equity 데이터 입력

```
In [44]: #Equity데이터를 딕셔너리 형태로 변환
eqtdct = {}
for key, eqt in equity.dct.items():
    eqtdct['Eqt_'+key] = {
        '인출한도': eqt._df.scd_out,
        '상환예정': eqt._df.scd_in,
        '인출금액': eqt._df.amt_out,
        '상환금액': eqt._df.amt_in,
        '대출잔액': eqt._df.bal_end,
    }
fncdct['Equity'] = eqtdct
```

(4) 합계 데이터 입력

```
In [45]: ttldct = {}
for key, dct in fncdct.items():
    ttldct[key] = {}
    for key2, dct2 in dct.items():
        ttldct[key][key2] = {}
        for key3, srs in dct2.items():
            if key3 in ['대출잔액', '누적이자', '누적수수료', '누적미인출']:
                tmpval = '-'
            else:
                tmpval = sum(srs)
            ttldct[key][key2][key3] = pd.concat([srs, Series([tmpval], index=['합계'])])
```

(5) 최종 딕셔너리 데이터 출력

```
In [46]: ws(ttldct, fmtkey=wb.bold, fmt=wb.num, valdrtn='row', drtn='col')
```

```
Out[46]: Cell(38, 1)
```

```
In [ ]:
```

5. Balance Sheet 출력

1) sheet 생성

```
In [47]: ws = wb.add_ws("balancesheet")
wb.ws["balancesheet"].set_column("A:X", 12)

ws("BALANCE SHEET", wb.bold)
ws("Written at: " + wb.now)
cell = ws.nextcell(2)
```

2) 분양매출 출력

```

In [48]: ws(" [[분양매출]]", wb.bold)
ws("오피스텔", wb.bold, drtn='row')
slstbl = sales.분양테이블['오피'][['호실면적', '호실수', '평단가', '매출소계']]
slssum = DataFrame(slstbl.sum(), columns=['합계']).T
slstbl = pd.concat([slstbl, slssum])
ws(slstbl, fmtkey=wb.bold, fmt=wb.num, valdrtn='row', drtn='col')
ws.nextcell(-1, drtn='col')

ws("근생시설", wb.bold, drtn='row')
slstbl = sales.분양테이블['근생'][['면적', '평단가', '면적m2', '매출소계']]
slssum = DataFrame(slstbl.sum(), columns=['합계']).T
slstbl = pd.concat([slstbl, slssum])
ws(slstbl, fmtkey=wb.bold, fmt=wb.num, valdrtn='row', drtn='col')
ws.nextcell(-1, drtn='col')

ttl_sales = sales.분양매출['합계']
ws(["분양매출 합계", "", "", "", "", "", ttl_sales], [wb.bold, wb.nml, wb.nml, wb.nml, wb.nml, wb.nml, wb.numb])
ws.nextcell(2)

```

Out[48]: Cell(17, 0)

3) 사업비 출력

```

In [49]: ws("[사업비]", wb.bold)
ws(["구분1", "구분2", "", "", "", "금액"], wb.bold)
ttl_costs = 0
#Write operating costs
for key1, item1 in cost.dct.items():
    ws(key1, wb.bold, drtn='row')
    subttl = 0
    for key2, item2 in item1.dct.items():
        amt = item2.bal_end[-1]
        ws([key2, "", "", "", amt], [wb.nml, wb.nml, wb.nml, wb.nml,
wb.num])
        subttl += amt
    ws(["소계", "", "", "", subttl], [wb.bold, wb.nml, wb.nml, wb.nml,
wb.num])
    ws.nextcell(-1, drtn='col')
    ttl_costs += subttl
#Write financing costs
for key, item in loan.dct.items():
    ws("대출"+key, wb.bold, drtn='row')
    subttl = 0
    amt_fee = item.fee.bal_end[-1]
    ws(["Fee_"+key, "", "", "", amt_fee], [wb.nml, wb.nml, wb.nml,
wb.nml, wb.num])
    amt_IR = item.IR.bal_end[-1]
    ws(["IR_"+key, "", "", "", amt_IR], [wb.nml, wb.nml, wb.nml, wb
.nml, wb.num])
    subttl = amt_fee + amt_IR

    ws(["소계", "", "", "", subttl], [wb.bold, wb.nml, wb.nml, wb.nml,
wb.num])
    ws.nextcell(-1, drtn='col')
    ttl_costs += subttl
ws(["총사업비 합계", "", "", "", "", ttl_costs], [wb.bold, wb.nml, wb.nml,
wb.nml, wb.nml, wb.nml, wb.num])
ws.nextcell(2)

```

Out[49]: Cell(53, 0)

4) 사업이익 출력

```

In [50]: ws("[사업이익]", wb.bold)
ttl_profit = ttl_sales - ttl_costs
ws(["총 사업이익", "", "", "", "", ttl_profit], [wb.bold, wb.nml, wb.nml,
wb.nml, wb.nml, wb.num])
ws(["매출액 대비 이익률", "", "", "", "", ttl_profit / ttl_sales], [wb.nml,
wb.nml, wb.nml, wb.nml, wb.nml, wb.pct])
ws(["총사업비 대비 이익률", "", "", "", "", ttl_profit / ttl_costs], [wb.nml,
wb.nml, wb.nml, wb.nml, wb.nml, wb.pct])

```

Out[50]: Cell(57, 0)

In []:

6. 엑셀 워크북 클로징

In [51]: `wb.close()`

Out[51]: `True`

In []: