회계가 바로 서야 경제가 바로 섭니다.

제07회 재무빅데이터분석사 자격시험

FDA 1급

Financial big-Data Analyst

▣ 시험시간: 180분
▣ 객관식배점: 문항별 배점 참조
▣ 주관식배점: 문항별 배점 참조

KICPA 한국공인회계사회

이론 객관식

01 다음과 같이 2차원 Array를 만들었다.

```
import numpy as np
arr = np.array([[12,  7,  1,  9],
                [ 5,  2, 10, 16],
                [14,  3, 11,  4],
                [ 8, 13,  6, 15]])
print(arr)
```

```
[[12,  7,  1,  9],
 [ 5,  2, 10, 16],
 [14,  3, 11,  4],
 [ 8, 13,  6, 15]]
```

상기 2차원 Array를 다음과 같이 출력하고 싶다.

```
[[ 6  8 13 15]
 [ 3  4 11 14]
 [ 2  5 10 16]
 [ 1  7  9 12]]
```

상기와 같이 출력하기 위한 코드로 옳은 것을 고르시오. **(3점)**

㉮	arr.sort(axis = 0) arr = arr[::-1, :] print(arr)
㉯	arr2 = np.sort(arr, axis=1) arr2 = np.flip(arr2, axis=1) print(arr2)
㉰	arr.sort(axis=1) arr = np.flip(arr, axis=0) print(arr)
㉱	arr2 = np.sort(arr, axis=1) arr2 = arr2[::-1, :] arr2 = arr2[:, ::-1] print(arr2)
㉲	arr2 = arr[::-1, :] arr2.sort(axis=0) print(arr2)

① ㉮ ② ㉯ ③ ㉰
④ ㉱ ⑤ ㉲

해설

㉰ 순서대로 실행하면, 목표 결과를 얻을 수 있다.

정답 : ③

02 다음과 같은 배열을 만들었다.

```
import numpy as np
np.random.seed(10)
arr = np.random.randint(1, 150, size=(4, 3, 4))
print(arr)
```

```
[[[ 10 126  16  65]
  [114 124 114   9]
  [ 74   1  41 116]]

 [[ 17 101 140  55]
  [ 89 108 123  63]
  [ 34 101  55  78]]

 [[ 14 127 142  93]
  [ 87  31  90 141]
  [ 66  32  58  37]]

 [[ 28  19  94  78]
  [ 95 140  75 138]
  [ 16  19 119  72]]]
```

상기 Array를 다음과 같이 출력하고 싶다.

```
import numpy as np
print(arr[(arr >= ⓐ) & (arr <= ⓑ)].reshape(ⓒ, ⓓ))
```

```
[[65 74 41 55]
 [63 34 55 78]
 [31 66 32 58]
 [37 78 75 72]]
```

ⓐ, ⓑ, ⓒ, ⓓ 에 들어갈 내용으로 옳은 것으로 묶인 것을 고르시오. (3점)

㉮	ⓐ : 30, ⓑ : 80, ⓒ : 4, ⓓ : 4
㉯	ⓐ : 40, ⓑ : 70, ⓒ : 3, ⓓ : 5
㉰	ⓐ : 20, ⓑ : 50, ⓒ : 2, ⓓ : 3
㉱	ⓐ : 30, ⓑ : 80, ⓒ : 2, ⓓ : 8
㉲	ⓐ : 30, ⓑ : 90, ⓒ : 4, ⓓ : 4

① ㉮ ② ㉯ ③ ㉰
④ ㉱ ⑤ ㉲

해설

㉯ ValueError: cannot reshape array of size 7 into shape (3,5)
㉰ [[41 34 31]
  [32 37 28]]
㉱ [[65 74 41 55 63 34 55 78]
  [31 66 32 58 37 78 75 72]]
㉲ ValueError: cannot reshape array of size 19 into shape (4,4)

정답 : ①

03 다음은 판다스 데이터프레임인 df를 출력하면 결과가 다음과 같다.

```
print(df)
   Sales  Cost
0   1000   400
1   1500   700
2   1200   500
3    900   300
4   1300   700
```

df에 새로운 칼럼(열) 'Profit'을 만들어 Sales - Cost를 저장하려는 코드이다.

```
import pandas as pd

for ⓐ, ⓑ in df.ⓒ():
    df.loc[ⓓ, 'Profit'] = ⓔ['Sales'] - ⓔ['Cost']
```

ⓐ ~ ⓔ 에 들어갈 내용으로 옳은 것으로 묶인 것을 고르시오. **(3점)**

㉮	ⓐ : i,	ⓑ : row,	ⓒ : iterrows,	ⓓ : i,	ⓔ : df
㉯	ⓐ : i,	ⓑ : row,	ⓒ : items,	ⓓ : i,	ⓔ : row
㉰	ⓐ : i,	ⓑ : data,	ⓒ : iterrows,	ⓓ : i,	ⓔ : df
㉱	ⓐ : I,	ⓑ : data,	ⓒ : itertuples,	ⓓ : i,	ⓔ : data
㉲	ⓐ : idx,	ⓑ : row,	ⓒ : iterrows,	ⓓ : idx,	ⓔ : row

① ㉮ ② ㉯ ③ ㉰
④ ㉱ ⑤ ㉲

정답 : ⑤

해설

```
for idx, row in df.iterrows():
    df.loc[idx, 'Profit'] = row['Sales'] - row['Cost']
   Sales  Cost  Profit
0   1000   400   600.0
1   1500   700   800.0
2   1200   500   700.0
3    900   300   600.0
4   1300   700   600.0
```

04 다음 코드는 데이터프레임인 df에 있는 Amount 칼럼을 보고,

- Amount가 양수이면 Debit 칼럼에 금액을 저장하고 Credit 칼럼은 0을 저장
- Amount가 음수이면 Credit 칼럼에 절대값으로 금액을 저장하고 Debit 칼럼은 0을 저장 하려고 한다.

```
import pandas as pd

df = pd.DataFrame({'Account' : ['현금','매출', '수입임차료', '기계장치', '이자수익'],
'Amount' : ['500', '-300', '-500', '1000', '200']})

df['Amount'] = df['Amount'].ⓕ(int)

def ⓐ(ⓑ) :
    if ⓒ['Amount'] > 0 :
        ⓒ['Debit'] = ⓒ['Amount']
        ⓒ['Credit'] = 0
    else :
        ⓒ['Debit'] = 0
        ⓒ['Credit'] = abs(ⓒ['Amount'])
    return ⓒ

df = df.apply(ⓓ, axis = ⓔ)
print(df)
```

ⓐ ~ ⓕ 에 들어갈 내용으로 옳은 것으로 묶인 것을 고르시오. **(3점)**

㉮	ⓐ : split_account , ⓓ : split_account ,	ⓑ : df , ⓔ : 0	ⓒ : df , ⓕ : astype
㉯	ⓐ : split_account , ⓓ : split_account ,	ⓑ : row , ⓔ : 1	ⓒ : row , ⓕ : astype
㉰	ⓐ : split_account, ⓓ : split_account,	ⓑ : df, ⓔ : 1	ⓒ : df, ⓕ : int
㉱	ⓐ : split_account, ⓓ : split_account,	ⓑ : row, ⓔ : 0	ⓒ : row, ⓕ : astype
㉲	ⓐ : split_account, ⓓ : r,	ⓑ : r, ⓔ : 1	ⓒ : r, ⓕ : astype

① ㉮ ② ㉯ ③ ㉰
④ ㉱ ⑤ ㉲

정답 : ②

해설

㉮ KeyError: 'Amount'
㉰ AttributeError: 'Series' object has no attribute 'int'
㉱ KeyError: 'Amount'
㉲ NameError: name 'r' is not defined

05 다음은 판다스의 pivot_table() 함수를 사용하여 작성한 코드이다.

```
import pandas as pd

data = {'계정과목': ['매출', '매출', '원가', '원가', '판관비', '판관비'],
        '월': [1, 2, 1, 2, 1, 2],
        '금액': [1000, 1200, 600, 700, 300, 350]}

df = pd.DataFrame(data)

pivot_result = pd.pivot_table(df, index = ⓐ, columns = ⓑ, values = ⓒ,
                              aggfunc = ⓓ)

print(pivot_result)
```

위 코드를 실행했을 때, 월별 계정과목 합계가 아래와 같이 출력되길 원한다.

```
계정과목    매출   원가  판관비
월
1     1000  600  300
2     1200  700  350
```

ⓐ ~ ⓓ 에 들어갈 내용으로 옳은 것으로 묶인 것을 고르시오. **(3점)**

㉮	ⓐ : '계정과목' ⓓ : 'sum'	ⓑ : '월'	ⓒ : '금액'
㉯	ⓐ : '계정과목' ⓓ : 'sum'	ⓑ : '금액'	ⓒ : '월'
㉰	ⓐ : '금액' ⓓ : 'sum'	ⓑ : '계정과목'	ⓒ : '월'
㉱	ⓐ : '금액' ⓓ : 'sum'	ⓑ : '월'	ⓒ : '계정과목'
㉲	ⓐ : '월' ⓓ : 'sum'	ⓑ : '계정과목'	ⓒ : '금액'

① ㉮ ② ㉯ ③ ㉰
④ ㉱ ⑤ ㉲

정답 : ⑤

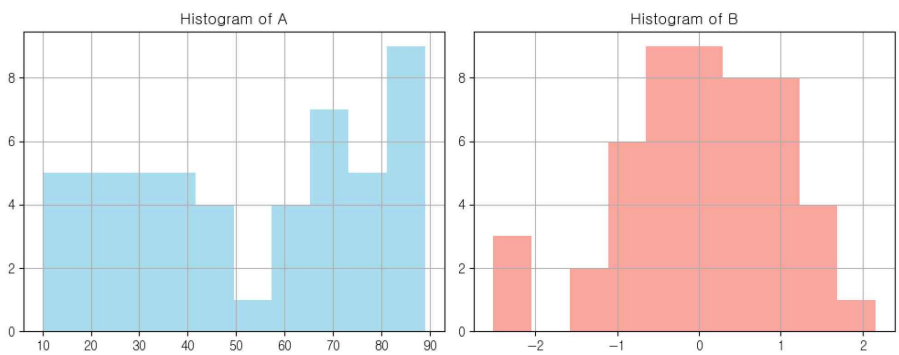
해설

㉲ 결과와 같이 나온다.

06 다음 코드는 데이터프레임 df를 생성하는 코드이다.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# 시드 고정
np.random.seed(0)

df = pd.DataFrame({
    'A': np.random.randint(1, 100, 50),
    'B': np.random.randn(50),
    'C': np.random.choice(['X', 'Y', 'Z'], 50)})
```

상기 데이터프레임을 토대로 다음의 그래프를 생성하였다.



위 그래프를 생성하는 코드로 옳은 것을 고르시오. **(3점)**

㉮	fig, axes = plt.subplots(1, 2, figsize=(10, 4)) df['A'].hist(ax=axes[0], alpha=0.7, color='skyblue') axes[0].set_title("Histogram of A") df['B'].hist(ax=axes[1], alpha=0.7, color='salmon') axes[1].set_title("Histogram of B") plt.tight_layout() plt.show()
㉯	df['C'].value_counts().plot(kind='bar', alpha=0.7, color = 'lightgreen') plt.title("Bar Plot of C (Categorical)") plt.xlabel("Category") plt.ylabel("Count") plt.show()
㉰	df[['A', 'B']].plot(kind='box', figsize=(6, 5)) plt.title("Box Plot of A and B") plt.show()
㉱	plt.figure(figsize=(6, 5)) plt.scatter(df['A'], df['B'], alpha=0.7, color='orchid') plt.title("Scatter Plot of A vs B")

	plt.xlabel("A") plt.ylabel("B") plt.show()
㉮	corr_matrix = df[['A', 'B']].corr() plt.figure(figsize=(4, 3)) sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1) plt.title("Correlation Heatmap (A, B)") plt.show()

① ㉮ ② ㉯ ③ ㉰
④ ㉱ ⑤ ㉮

정답 : ①

해설

㉮ 코드가 그래프 결과와 같이 나온다.

07 다음은 2개의 데이터프레임을 생성하는 코드이다.

```
import pandas as pd

df_sales = pd.DataFrame({
    'account_id': [101, 102, 103, 104],
    'sales': [10000, 20000, 15000, 18000],
    'year': [2021, 2021, 2021, 2021]})

df_expenses = pd.DataFrame({
    'account_id': [101, 102, 105],
    'expenses': [7000, 14000, 5000],
    'year': [2021, 2021, 2021]})
```

상기 2개의 데이터프레임을 조인한 결과가 다음과 같다.

account_id	sales	year_x	expenses	year_y
101	10000	2021	7000	2021
102	20000	2021	14000	2021
103	15000	2021	NaN	NaN
104	18000	2021	NaN	NaN
105	NaN	NaN	5000	2021

이 결과를 만들기 위해 사용된 조인 방식(how)은 무엇인가? **(3점)**

㉮	how = 'left'
㉯	how = 'right'
㉰	how = 'inner'
㉱	how = 'outer'
㉲	how = 'cross'

① ㉮ ② ㉯ ③ ㉰
④ ㉱ ⑤ ㉲

정답 : ④

해설

㉱ df_sales에는 103, 104가 있고 df_expenses에는 105가 있으므로, outer 조인일 경우 103, 104, 105 모두 결과에 나타난다.

08 회계팀에서는 전표번호가 "Invoice #숫자" 형태로 주어진다. 예를 들어 다음과 같다.

- 'Invoice #45'
- 'Invoice #567'
- 'Invoice #2024'

여기서 문자열 전체가 "Invoice #"로 시작하고, 뒤에 1개 이상의 숫자가 따라오며, 그 뒤에는 다른 문자가 없는 경우만 정확히 매칭 하려면 어떤 정규 표현식을 사용해야 하는가? **(3점)**

① '^Invoice #\d+$'
② '^Invoice #\w+$'
③ 'Invoice #[0-9]+'
④ '^\d+Invoice #'
⑤ 'Invoice #\d{2,}'

정답 : ①

해설

① "Invoice #"로 시작하고 뒤에 숫자만 온 뒤 문자열이 끝나야 하므로 '^Invoice #\d+$'가 일치한다.

09 Fraudit 교육용 버전 왼쪽 네비게이터의 Tables 트리의 samples 폴더안에 있는 Bid.tbl을 연다. 그리고 다음과 같이 이를 데이터프레임으로 변환한다.

```
>>> df = Bid.toDF()
>>> df
     입찰번호 프로젝트번호 입찰자번호       입찰일자       입찰금액
0    5001   9001       8048         2009-03-07   188163.44
1    5002   9001       8063         2009-02-03   383786.19
2    5003   9001       8036         2009-02-03   593798.81
3    5004   9001       8059         2009-02-22   796503.19
4    5005   9002       8019         2009-02-11   772554.69
..    ...    ...        ...          ...           ...
274  5275   9063       8058         2009-02-28  1312109.38
275  5276   9064       8006         2009-12-18   619359.56
276  5277   9064       8066         2009-12-18  1199294.32
277  5278   9064       8014         2009-12-18  1797663.21
278  5279   9064       8041         2009-12-18  2419713.88

[279 rows x 5 columns]
```

상기 df에 다음과 같이 칼럼을 삽입하고 프로젝트 칼럼을 기준으로 groupby한 결과값을 다음과 같이 넣으려고 한다.

```
>>> import pandas as pd
>>> gb_1 = df.groupby('프로젝트번호')
>>> df['결과값'] = gb_1['입찰금액'].㉮
>>> df
     입찰번호 프로젝트번호 입찰자번호       입찰일자       입찰금액         결과값
0    5001   9001       8048         2009-03-07   188163.44   796503.19
1    5002   9001       8063         2009-02-03   383786.19   796503.19
2    5003   9001       8036         2009-02-03   593798.81   796503.19
3    5004   9001       8059         2009-02-22   796503.19   796503.19
4    5005   9002       8019         2009-02-11   772554.69  2410526.50
..    ...    ...        ...          ...           ...         ...
274  5275   9063       8058         2009-02-28  1312109.38  1312109.38
275  5276   9064       8006         2009-12-18   619359.56  2419713.88
276  5277   9064       8066         2009-12-18  1199294.32  2419713.88
277  5278   9064       8014         2009-12-18  1797663.21  2419713.88
278  5279   9064       8041         2009-12-18  2419713.88  2419713.88

[279 rows x 6 columns]
```

상기에서 ㉮에 들어갈 내용으로 옳은 것을 고르시오. **(3점)**

① index('max')
② ngroup('min')
③ transform('max')
④ index('min')
⑤ nindex('max')

정답 : ③

해설

③ transform('max')

10 다음 코드를 실행했을 때의 설명으로 옳은 것을 고르시오. **(3점)**

```
import networkx as nx
G = nx.Graph()
G.add_nodes_from([1, 2, 3])
G.add_edge(1, 2)
G.add_edge(2, 3)
```

① G는 방향이 있는 그래프이므로, 엣지의 방향을 고려한다.
② G.add_nodes_from([1, 2, 3])는 노드를 추가하면서 이들 노드 간에 모든 엣지도 자동으로 생성한다.
③ G.add_edge(1, 2)는 1 → 2 방향만 존재하므로, 2에서 1로의 방향은 존재하지 않는다.
④ G는 무방향 그래프(Graph)이므로 1-2, 2-3 엣지는 쌍방향이다.
⑤ G가 가지고 있는 노드 수는 총 2개이다.

정답 : ④

해설

① nx.Graph()로 생성한 그래프는 무방향(undirected) 그래프이다.
② 노드 간 엣지는 자동으로 생성되지 않는다. 엣지를 추가하려면 별도로 add_edge()를 호출해야 한다.
③ 3번의 한 방향만 존재라는 설명은 잘못되었다.
⑤ 노드가 1, 2, 3 총 세 개다.

주관식

11~16

당신은 (주)한공의 내부감사인이다. (주)한공의 회계프로그램에서 2024년 분개장을 입수하여 여러가지 분석을 하려고 한다. 2024년 분개장을 csv로 받은 것이 분개장_07회_1급_1.csv 이며 이는 다음을 누르면 다운로드 받을 수 있다.

분개장_07회_1급_1.csv

분개장_07회_1급_1.csv 을 pandas의 데이터프레임으로 불러온다. Fraudit 테이블로 바로 import를 하면 레코드가 10000개로 제한되므로 pandas를 이용하여야 한다. 데이터프레임으로 불러온 후 구조를 살펴보면 다음과 같다.

칼럼명	데이터 타입	설명
전표일자	object	XXXX-XX-XX 형식으로 되어 있음.
전표번호	object	전표번호
전표라인번호	int64	전표세트내의 레코드 순서 번호
생성부서	object	전표생성부서
생성자	int64	전표생성자
계정코드	int64	계정코드
계정과목	object	계정과목
차변금액	int64	차변금액
대변금액	int64	대변금액

동일한 전표일자에 동일한 전표번호는 하나의 전표세트를 의미한다. 즉, 아래의 네모 선 안에 있는 것과 같이 동일한 전표일자와 동일한 전표번호를 가지는 세트를 하나의 전표세트라고 한다. 아래의 네모 선 안에 있는 하나의 전표세트의 레코드수는 3개이다.

	전표일자	전표번호	전표라인번호	생성부서	생성자	계정코드	계정과목	차변금액	대변금액
0	2024-01-01	2024000110	1	영업부	70035	4020	내수상품매출액	0	9164
1	2024-01-01	2024000110	2	영업부	70035	2110	부가세예수금	0	916
2	2024-01-01	2024000110	3	영업부	70035	1270	내수외상매출금	10080	0
3	2024-01-01	2024000119	1	기획부	120188	1215	보통예금	3800000	0
4	2024-01-01	2024000119	2	기획부	120188	1270	내수외상매출금	0	3800000
5	2024-01-01	2024000216	1	영업부	60009	2087	거래보증금	17719396	0
6	2024-01-01	2024000216	2	영업부	60009	1270	내수외상매출금	0	17719396

11 전표세트에서 차변금액과 대변금액이 모두 0인 계정과목이 포함된 전표세트의 예는 다음과 같다.

전표일자	전표번호	계정코드	계정과목	차변금액	대변금액
2024-01-01	1	XXXX	XXXX	40000	0
2024-01-01	1	XXXX	XXXX	0	0
2024-01-01	1	XXXX	XXXX	0	40000

전표세트에서 차변금액과 대변금액이 모두 0인 계정과목이 포함된 전표세트의 개수를 숫자로만 나타내라. (단, 천단위 콤마는 나타내지 않는다.) **(5점)**

정답 : 4

해설

(1) **New Script** 아이콘을 선택한다.
(2) 아래와 같이 코드를 입력하고 해당 영역을 선택한 후 ⏩를 누른다.

```
# 판다스 라이브러리 불러오기
import pandas as pd

# 파일 임포트
df_merged = pd.read_csv('분개장_07회_1급_1.csv')

# 차대변0빈도 칼럼 삽입
df_merged['차대변0빈도'] = [ 1 if i[0] == 0 and i[1] == 0 else 0
                    for i in zip(df_merged['차변금액'], df_merged['대변금액'])]

# 전표일자와 전표번호 칼럼을 기준으로 groupby 한다.
gb_1 = df_merged.groupby(['전표일자', '전표번호'])
df_요약 = pd.DataFrame({'차대변0빈도':gb_1['차대변0빈도'].sum()}).reset_index()
df_요약_발췌 = df_요약[df_요약['차대변0빈도'] != 0]
print(df_요약_발췌)
```

```
          전표일자         전표번호  차대변0빈도
1027  2024-01-15  2025H000072       8
1958  2024-01-29  2025F000127       2
3400  2024-01-31  2025H000306       2
3412  2024-01-31  2025H000332       3
```

12 다음은 계정과목과 간주계정과목의 관계를 나타낸 표이다.

계정코드	계정과목	간주계정과목	계정코드	계정과목	간주계정과목
2061	선수금	선수금			
4020	내수상품매출액	매출	4045	수출제품매출액	매출

간주계정과목이 '선수금'이면서 차변금액이 0을 초과하거나 대변금액이 0 미만이고, 간주계정과목이 '매출'이면서 대변금액이 0을 초과하거나 차변금액이 0미만인 경우를 포함하는 전표세트를 **선수금매출 전표세트**라고 가정한다. 선수금매출 전표세트의 개수를 숫자로만 나타내라. (단, 천단위 콤마는 나타내지 않는다.) **(5점)**

정답 : 1

해설

(1) **New Script** 아이콘을 선택한다.
(2) 아래와 같이 코드를 입력하고 해당 영역을 선택한 후 ⏩를 누른다.

```
# 판다스 라이브러리 불러오기
import pandas as pd

# 파일 임포트
df = pd.read_csv('분개장_07회_1급_1.csv')

# 선수금 리스트와 매출 리스트 생성
선수금_리스트 = [2061]
매출_리스트 = [4020, 4045]

# df에 현금빈도와 매출빈도 칼럼 삽입
df['선수금_차변_빈도'] = [ 1 if i[0] in 선수금_리스트 and (i[1] > 0 or i[2] < 0) else
0 for i in zip(df['계정코드'], df['차변금액'], df['대변금액'])]
df['매출_대변_빈도'] = [ 1 if i[0] in 매출_리스트 and (i[1] < 0 or i[2] > 0) else 0
for i in zip(df['계정코드'], df['차변금액'], df['대변금액'])]

# 전표일자와 전표번호 칼럼을 기준으로 groupby 한다.
gb_1 = df.groupby(['전표일자', '전표번호'])
# 요약 데이터프레임 작성
df_요약 = pd.DataFrame({
                        '선수금차변빈도':gb_1['선수금_차변_빈도'].sum(),
                        '매출대변빈도' : gb_1['매출_대변_빈도'].sum()
                        }).reset_index()
df_요약_발췌 = df_요약[(df_요약['선수금차변빈도'] >0) & (df_요약['매출대변빈도'] >0)]
print(df_요약_발췌)
```

```
          전표일자         전표번호  선수금차변빈도  매출대변빈도
2463  2024-01-31  2025A002177       1       1
```

13 다음은 계정과목과 계정코드의 관계를 나타낸 표이다.

계정코드	계정과목	간주계정과목	계정코드	계정과목	간주계정과목
4020	내수상품매출액	매출	4045	수출제품매출액	매출
4049	해외내수제품매출액	매출	4050	내수제품매출액	매출

간주계정과목이 '매출'이면서 대변금액이 0을 초과하며 토요일 또는 일요일에 발생한 거래가 포함된 전표세트를 **주말매출 전표세트**라고 가정한다. **주말매출 전표세트**의 개수를 숫자로만 나타내라. (단, 천단위 콤마는 나타내지 않는다.) **(5점)**

정답 : 3

해설

(1) **New Script** 아이콘을 선택한다.
(2) 아래와 같이 코드를 입력하고 해당 영역을 선택한 후 ⏩를 누른다.

```
# 판다스 라이브러리 불러오기
import pandas as pd
# 파일 임포트
df = pd.read_csv('분개장_07회_1급_1.csv')
# 요일 칼럼 삽입
df['전표일자_1'] = pd.to_datetime(df['전표일자'])
df['요일'] = df['전표일자_1'].dt.weekday
# 매출 대변 빈도 칼럼 삽입
매출_리스트 = [4020,4045,4049,4050]
# 주말 매출 레코드 발췌
df_주말매출 = df[
            (df['요일'].isin([5,6]))
            & (df['계정코드'].isin(매출_리스트))
            & (df['대변금액'] > 0) ]
# df_주말매출에 매출_대변_빈도 칼럼 삽입
df_주말매출['매출_대변_빈도'] = [ 1 if i[0] in 매출_리스트 and i[1] > 0 else 0
                    for i in zip(df_주말매출['계정코드'], df_주말매출['대변금액'])]
#전표일자와 전표번호 칼럼을 기준으로 groupby 한다.
gb_1 = df_주말매출.groupby(['전표일자', '전표번호'])
# 요약 데이터프레임 작성
df_요약 = pd.DataFrame({
                    '매출대변빈도' : gb_1['매출_대변_빈도'].sum()
                    }).reset_index()
# 매출대변빈도가 0보다 큰 것 발췌
df_요약_발췌 = df_요약[df_요약['매출대변빈도'] >0]
print(df_요약_발췌)
```

```
        전표일자         전표번호  매출대변빈도
0  2024-01-27  2025A001964       1
1  2024-01-28  2025A001962       1
2  2024-01-28  2025A001972       2
```

14 하나의 전표세트에서 전표라인번호는 1부터 시작해서 1씩 증가한다. 전표라인번호가 1부터 시작하지 않거나 시퀀스가 1씩 증가하지 않은 경우가 포함된 전표세트의 개수를 숫자로만 나타내라. (단, 천단위 콤마는 나타내지 않는다.) **(5점)**

정답 : 1

해설

(1) **New Script** 아이콘을 선택한다.
(2) 아래와 같이 코드를 입력하고 해당 영역을 선택한 후 ⏩를 누른다.

```
# 판다스 라이브러리 불러오기
import pandas as pd
# 파일 임포트
df = pd.read_csv('분개장_07회_1급_1.csv')
# 정렬
df = df.sort_values(by = ['전표일자', '전표번호', '전표라인번호'])
# 전표일자와 전표번호 칼럼을 기준으로 groupby 한다.
gb_1 = df.groupby(['전표일자', '전표번호'])
# "전표라인번호"를 비교하기 위해 한 칸 위(이전 행)의 값을 shift하여 새 칼럼 추가
df['전표라인번호_prev'] = gb_1['전표라인번호'].shift(1).fillna(0).astype(int)
# 전표라인번호 차이 계산
df['라인번호_차이'] = df['전표라인번호'] - df['전표라인번호_prev']
# 전표라인번호가 1보다 큰 구간(즉, 누락 / 점프가 있는 구간) 식별
df_gap = df[df['라인번호_차이'] != 1]
print(df_gap)
```

```
          전표일자       전표번호  전표라인번호 생성부서    생성자 계정코드   계정과목  차변금액  대변금액 \
5437  2024-01-10  2025B000220       2  생산부  100083  1555  부가세대급금  17360     0
     전표라인번호_prev  라인번호_차이
5437           0       2
```

15 하나의 전표세트에 생성자가 2인 이상인 전표세트의 개수를 숫자만 입력하라. **(5점)**

정답 : 22

해설

(1) **New Script** 아이콘을 선택한다.
(2) 아래와 같이 코드를 입력하고 해당 영역을 선택한 후 ⏩를 누른다.

```
# 판다스 라이브러리 불러오기
import pandas as pd
# 파일 임포트
df = pd.read_csv('분개장_07회_1급_1.csv')
# 전표일자와 전표번호 칼럼을 기준으로 groupby 한다.
gb_1 = df.groupby(['전표일자', '전표번호'])
# 요약 데이터프레임 작성
df_요약 = pd.DataFrame({
                        '생성자_빈도' : gb_1['생성자'].nunique()
                        }).reset_index()
# 요약 발췌
df_요약_발췌 = df_요약[df_요약['생성자_빈도'] >1]
# 개수 구하기
print(len(df_요약_발췌))
```

```
22
```

16 전표세트를 가장 많이 생성한 생성자를 구하고자 한다. 만약 하나의 전표세트에 생성자가 2인 이상인 경우에는 각각의 생성자는 그 전표세트를 생성한 것을 본다. 예를 들어, 어떤 전표세트에 생성자가 1, 2가 있는 경우 1도 그 전표세트를 생성한 것이고 2도 그 전표세트를 생성한 것이다. 전표세트를 *가장 많이* 생성한 생성자 번호를 숫자로 입력하라. **(5점)**

정답 : 70035

해설

(1) **New Script** 아이콘을 선택한다.
(2) 아래와 같이 코드를 입력하고 해당 영역을 선택한 후 ⏩를 누른다.

```
# 판다스 라이브러리 불러오기
import pandas as pd
# 파일 임포트
df = pd.read_csv('분개장_07회_1급_1.csv')
# 전표일자와 전표번호, 생성자 칼럼을 기준으로 groupby 한다.
gb_1 = df.groupby(['생성자','전표일자', '전표번호'])
# 요약 데이터프레임 작성
df_요약 = pd.DataFrame({
                        '생성자_빈도' : gb_1['생성자'].nunique()
                        }).reset_index()
# 요약_2 데이터프레임 작성
gb_2 = df_요약.groupby(['생성자'])
df_요약_2 = pd.DataFrame({
                        '생성자빈도' : gb_2['생성자_빈도'].count()
                        }).reset_index()
# 정렬(내림차순)
df_요약_2 = df_요약_2.sort_values(by = ['생성자빈도'],ascending = False)
print(df_요약_2)
```

```
     생성자  생성자빈도
39   70035    388
27   60009    270
93  120008    183
66  100083    183
116 120179    160
..     ...    ...
```

17 당신은 다음과 조건으로 이직 제의를 받았다.

① 제안받은 자리 : 설비 임대 사업을 하는 A회사의 매니저
② 연봉 보수 조건 :
(대안 1) 기본급 연봉 300,000,000원 + 설비 임대사업 총이익의 2%(인센티브)
(대안 2) 고정급 연봉 500,000,000원

당신은 다음과 같이 A회사의 데이터 일부를 입수하였다.

- 총 100개의 임대시설을 보유하고 있음
- 분포 가정

구분	분포	파라미터
단위당 임대료	Beta(alpha, beta, low, high)	alpha = 2, beta = 7, low = 180000000, high = 290000000
임대율	Beta(alpha, beta, low, high)	alpha = 10, beta = 2, low = 0.80, high = 1.0
단위당 변동비	Tri(low, peak, high)	low = 60000000, peak = 80000000, high = 90000000
고정비	Tri(low, peak, high)	low = 1000000000, peak = 1550000000, high = 1600000000

결과 변수의 산식은 다음과 같다.

구분	산식
임대된 시설수	보유한 시설수 × 임대율
총수익	단위당 임대료 × 임대된 시설수
총 변동비	단위당 변동비 × 임대된 시설수
총비용	총 변동비 + 고정비
총이익	총수익 - 총비용
인센티브	총이익 × 0.02

mcerp 라이브러리를 이용하여, 시뮬레이션을 10,000번(기본 디폴트 옵션이므로 mcerp.npts = 10000을 입력하지 않아도 된다) 수행한 결과 (대안 1)이 500,000,000원 이상일 확률을 소수점 2째 자리까지 반올림하여 구하라(예: 0.12345→0.12). 단, 시드는 np.random.seed(0)으로 고정시키고 다음과 같이 라이브러리를 불러온다. **(6점)**

```
import numpy as np
import pandas as pd
from pandas import *
```

```
import mcerp
from mcerp import *
import matplotlib.pyplot as plt
np.random.seed(0)
```

정답 : 0.70

해설

(1) New Script 아이콘을 선택한다.
(2) 아래와 같이 코드를 입력하고 해당 영역을 선택한 후 ⏩를 누른다.

```
import numpy as np
import pandas as pd
from pandas import *
import mcerp
from mcerp import *
import matplotlib.pyplot as plt
# 시드 고정하기
np.random.seed(0) # seed를 0으로 고정시킨다.
# 입력변수의 파라미터 정하기
단위당_임대료 =  Beta(2, 7, 180000000, 290000000)
보유한_시설수 = 100
임대률 = Beta(10, 2, 0.8, 1.0)
임대된_시설수 = 보유한_시설수*임대률
총수익 = 단위당_임대료 * 임대된_시설수
단위당_변동비 = Tri(60000000, 80000000, 90000000)
총_변동비 = 단위당_변동비*임대된_시설수
고정비 = Tri(1000000000, 1550000000, 1600000000)
총비용 = 총_변동비 + 고정비
총이익 = 총수익 - 총비용
인센티브 = 총이익 * 0.02
연봉 = 300000000 + 인센티브
# 급여가 500000000원 이상일 확률 구하기
print(연봉 > 500000000)
```

```
0.7029000000000001  --> 반올림 0.70
```

18~21

투자자 A는 주식의 과거 수익률 자료에 근거하여 5억원의 투자 금액으로 기대수익(수익의 평균)을 최대화 할 수 있는 포트폴리오를 구성하고자 한다. 포트폴리오의 내용은 다음과 같다.

구분	분포	수익률(평균)
혼합형	Normal(0.1251, 0.1431)	13%
인덱스혼합형	Normal(0.1713, 0.2265)	17%
주식형	Normal(0.2193, 0.3004)	23%
채권형	Normal(0.0606, 0.0101)	6%

구분	투자비율	최소 투자비율	최대 투자비율
혼합형	25%	10%	100%
인덱스혼합형	25%	10%	100%
주식형	25%	10%	100%
채권형	25%	10%	100%
합계	100%		

투자상품의 총투자 비율은 100%보다 작거나 같아야 하며, 각각의 투자비율의 범위는 적어도 10% 이상이면서 최대 100%이하이어야 한다. 한편 투자비율은 1% 단위로 변동된다. 투자자 A는 혼합형, 인덱스혼합형, 주식, 채권형 투자상품에 각각 몇 % 정도를 투자하여야 할 것인가? mcerp, geneticalgorithm, matplotlib 라이브러리를 이용하고, 시뮬레이션을 1,000번 수행하고, 시드는 np.random.seed(0)으로 고정시킨다. 따라서 다음과 같이 먼저 라이브러리를 불러오고 설정을 한다.

```
# 관련 라이브러리를 불러온다.
import numpy as np
import mcerp
from mcerp import *
from geneticalgorithm import geneticalgorithm as ga
# 시드 고정하기
np.random.seed(0)
# 시뮬레이션 횟수 정하기
mcerp.npts = 1000
```

18 혼합형(%를 제외한 숫자만 입력한다. 예를 들어, 5%이면 5만 입력한다) **(2점)**

19 인덱스혼합형(%를 제외한 숫자만 입력한다. 예를 들어, 5%이면 5만 입력한다) **(2점)**

20 주식형(%를 제외한 숫자만 입력한다. 예를 들어, 5%이면 5만 입력한다) **(2점)**

21 채권형(%를 제외한 숫자만 입력한다. 예를 들어, 5%이면 5만 입력한다) (2점)

혼합형 : 10
인덱스혼합형 : 11
주식형 : 69
채권형 : 10

해설

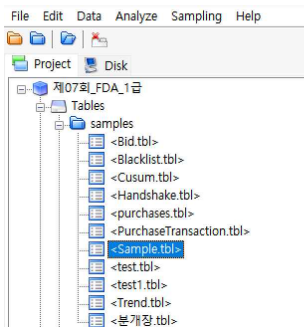
(1) New Script 아이콘을 선택한다.
(2) 아래와 같이 코드를 입력하고 해당 영역을 선택한 후 ⏩를 누른다.

```
import numpy as np
import mcerp
from mcerp import *
from geneticalgorithm import geneticalgorithm as ga
# 시드 고정하기
np.random.seed(0) # seed를 0으로 고정시킨다.
# 시뮬레이션 횟수 정하기
mcerp.npts = 1000
# 입력변수의 분포 반영
입력변수_1 = Normal(0.1251, 0.1431)
입력변수_2 = Normal(0.1713, 0.2265)
입력변수_3 = Normal(0.2193, 0.3004)
입력변수_4 = Normal(0.0606, 0.0101)
# 입력변수의 array 변환
입력변수_1 = 입력변수_1._mcpts
입력변수_2 = 입력변수_2._mcpts
입력변수_3 = 입력변수_3._mcpts
입력변수_4 = 입력변수_4._mcpts

# 목표함수 설정
def f(X) :
    # 전역변수 반영
    global 입력변수_1
    global 입력변수_2
    global 입력변수_3
    global 입력변수_4

    # 결정변수_1,2,3,4(혼합형, 인덱스혼합형, 주식형, 채권형의 투자비율) 설정
    결정변수_1 = X[0]
    결정변수_2 = X[1]
    결정변수_3 = X[2]
    결정변수_4 = X[3]
    # 제약조건 위반시 페널티 반영
    페널티 = 0
    if 결정변수_1 + 결정변수_2 + 결정변수_3 + 결정변수_4 > 100  :
```

```
        페널티 = -1000*(결정변수_1 + 결정변수_2 + 결정변수_3 + 결정변수_4 -100) # ①
    # 0.01을 곱하는 것은 ga() 함수의 파라미터 중 variable_type이 정수이므로 1% 변동 반영
    결과변수 = 입력변수_1*결정변수_1*0.01 \
           + 입력변수_2*결정변수_2*0.01 \
           + 입력변수_3*결정변수_3*0.01 \
           + 입력변수_4*결정변수_4*0.01 \
           + 페널티
    return -np.mean(결과변수)   # 최대화이므로 -를 반영한다.
varbound=np.array([[10,100],
                   [10,100],
                   [10,100],
                   [10,100]])
model = ga(function = f,
           dimension = 4,
           variable_type = 'int',  # 변동단위가 1%이므로 정수처리를 한다.
           variable_boundaries = varbound)
model.run()
```

```
The best solution found:
[10. 11. 69. 10.]

Objective function:
-0.18883368097342096
```

22 당신은 ㈜한공의 세전매출의 적정성을 테스트하기 위하여 MUS 샘플링을 수행하려고 한다. 관련 자료는 Sample.tbl이다. Sample.tbl은 Fraudit 교육용 버전을 설치하면 Project Folder >> samples folder가 생기고 그 안에 있다.



MUS 샘플링을 위한 파라미터 설정은 다음과 같다.

구분	설명
모집단의 화폐금액(Population)	'세전매출' 칼럼의 합계액
허용왜곡표시(Tolerable Error)	5%
예상오차율(Expected Error)	0.5%
Confidence Level	90%

Random starting point는 1000을 입력하고 샘플을 추출한다.

한편, 샘플에 포함된 다음의 송장번호의 세전매출 금액과 입증감사후 세전매출 금액의 내용은 다음과 같다.

송장번호	영업사원번호	CUSTNO	제품코드	세전매출	세전매출_Audit
A900026	118	21254	5	3,722.94	1,000
A900181	105	30608	4	107,589.24	100,000

Fraudit을 이용하여 샘플 추출한 후 이를 분석한 결과가 다음과 같았다.

Reject : the population as materially misstated.

population amount(5,307,370.36) × Tolerable error(5.0%) = 265,369.0
UpperErrorLimit : ㉮
265,369.0 < ㉮

빈칸 ㉮에 들어갈 금액을 입력하라. 단, Fraudit에서 나타내는 방식으로 천단위 콤마와 소수점 2째짜리까지 입력하라. **(6점)**

정답 : 336733.24

23 (주)한공의 분개장에서 제품매출이 발생한 레코드만 발췌한 것이 FDA_07회_1급_제품매출.tbl이다. 이는 아래에서 받을 수 있다.

FDA_07회_1급_제품매출.tbl

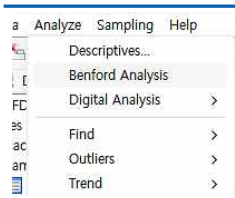
상기 파일을 불러온 후의 테이블 구조는 다음과 같다.

칼럼명	데이터 타입	설명
전표일자	String	전표일자
전표번호	Integer	전표번호
계정코드	Integer	계정코드
계정과목	String	계정과목
차변금액	Integer	차변금액
대변금액	Integer	대변금액
거래처	String	거래처
승인일자	String	승인일자
프로젝트코드	String	프로젝트코드

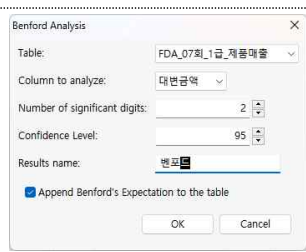
대변금액의 앞의 2단위(예: 대변금액이 12345이면 12를 의미함)에 대해서 벤포드 분석을 하려고 한다. 앞의 2단위의 벤포드 평균이 Expected라고 가정할 경우 실제 발견된 비율(Found Distribution)의 Z값이 2.0을 초과하는 항목의 개수는 몇 개인지 숫자만 입력하라? 예를 들어 앞의 2단위가 10, 11, 12, 13이며 이들의 Z값이 2.0을 초과한다면 4를 입력한다. **(6점)**

정답 : 10

해설



(1) Analyze >> Benford Analysis 메뉴를 누른다.



(2) Tabe : FDA_07회_1급_제품매출 선택
(3) Column to analyze : 대변금액 선택
(4) Number of significant digits : 2 선택
(5) Confidence Level : 95% 선택
(6) Results name : 벤포드 입력
(7) Append Benford's Expectation to the table : 체크
(8) OK 버튼을 누른다.

	First_2_Dig	Counts	Found	Expected	Z_score
0	40	32	.0247	.0107	4.7518
1	20	47	.0363	.0212	3.6777
2	80	16	.0124	.0054	3.2296
3	72	17	.0131	.0060	3.1483
4	22	38	.0293	.0193	2.5244
5	13	26	.0201	.0322	2.3900
6	32	7	.0054	.0134	2.3732
7	10	69	.0533	.0414	2.0781
8	25	12	.0093	.0170	2.0527
9	57	3	.0023	.0076	2.0160
10	28	29	.0224	.0152	1.9880
11	16	46	.0355	.0263	1.9792

(9) Z_score가 2.0을 초과하는 2단위 항목은 총 10개이다.

24 당신은 random_text 칼럼에 포함된 데이터를 분석하여 농축수산물의 항목별 금액 합계를 구하고자 한다. random_text 칼럼에 포함된 데이터만 추출하여 csv 파일로 변환한 것이 『random_data.csv』 파일이며 이는 아래의 링크를 누르면 다운로드 받을 수 있다.

random_data.csv

『random_data.csv』 의 레코드 수는 20000개이며 데이터의 1번째 ~ 9번째 레코드는 다음과 같다.

가지 509원 강릉
오이 9,774원 인천
쥐포 사업자 등록번호 833-93-81426 울산 ₩8,379 전화 010-4611-8359
5,674원 감귤 목포
오이 인천 ₩6,324 전화 010-5333-1711
오이 주식회사 9,144원 사업자 등록번호 전라북도
강원도 ₩3,914 전화 010-8428-6977 오이
울산 사업자 등록번호 ₩1,269 치약 사업자 등록번호 573-58-45382

레코드에 입력된 내용은 항목, 금액, 광역주소, 전화, 사업자 등록번호 등이 랜덤으로 섞여 들어가 있다. 분석하고자 하는 농축수산물 항목을 리스트로 만든 것은 다음과 같다.

```
product_list = ["사과", "배", "오이", "가지", "고추", "감자", "상추",
    "쥐포", "치약", "세발낙지", "소고기 1 킬로", "돼지고기 1근",
    "쌀 1kg", "감귤", "옥수수", "양파", "당근"]
```

『random_data.csv』 를 데이터프레임으로 불러온 후 농축수산물의 항목별 금액 합계를 구한 결과 금액의 합계가 가장 낮은 농축수산물의 항목을 입력하라. **(8점)**

정답 : 배

25 다음은 풋옵션과 관련된 내용이다.

• 아메리칸 풋옵션	• 현재주가 : 200	• 행사가격 : 180
• 무위험이자율 : 5%	• 배당률 : 0%	• 만기 : 3년
• 변동성 : 30%	• 스텝(노드) : 300	• dt = 만기/스텝(노드)

풋옵션 가격을 소수점 2째 자리까지 반올림하여 입력하라.(예 : 0.12345→0.12)
(단, 소수점 처리를 위해서 코드 상단에 np.set_printoptions(precision=4, suppress =True)를 입력해서 처리하라.) **(6점)**

정답 : 20.34

해설

```
>>> import numpy as np
>>> np.set_printoptions(precision = 4, suppress = True)
    # 기초변수 설정(한줄에 여러 변수를 설정하는 경우 세미콜론으로 구분함)
>>> 최초주가 = 200 ; 행사가격 = 180 ; 무위험이자율 = 0.05 ; 변동성 = 0.30
>>> 만기 =3 ; 스텝 =300; dt =만기/스텝 ; 할인계수 = np.exp(-무위험이자율*dt)
    # u와 d계산
>>> U = np.exp(변동성 * np.sqrt(만기/스텝))  # u계산
>>> D = np.exp(-변동성 *np.sqrt(만기/스텝))  # d계산
    # p와 (1-p)계산
>>> P_U = (np.exp(무위험이자율*(만기 /스텝))-D)/(U-D)   # p
>>> P_D = 1 - P_U
    # 주가트리 생성
>>> 주가_트리 = np.zeros([스텝 + 1, 스텝 + 1])
>>> for i in range(스텝 +1):
...     for j in range(i +1):
...         주가_트리[j, i] = 최초주가 * (U ** (i - j)) * (D ** j)
    # 옵션트리 생성
>>> 옵션_트리 = np.zeros([스텝 + 1, 스텝 + 1])
    # 마지막 스텝 : max((행사가격-주가), 0) 계산
>>> 옵션_트리[:,스텝] = np.maximum((행사가격 - 주가_트리[:,스텝]), 0)
>>> print(옵션_트리) # 소수점 2째자리 반올림
>>> for node in range(스텝-1,-1,-1) : #역순으로
...     for time in range(스텝-1,-1,-1) : #역순으로
...         if time >= node :  #대각선을 기준으로 상단만 결과 나오게 함
...             옵션_트리[node, time] = np.maximum(행사가격-주가_트리[node,time],
...             (옵션_트리[node,time+1]*P_U+옵션_트리[node+1,time+1]*P_D)*할인계수)
>>> print(옵션_트리) # 소수점 2째자리 반올림
```

```
[[ 20.3425  18.7441  17.2311 ...   0.      0.      0.    ]
 [  0.     21.9666  20.281  ...   0.      0.      0.    ]
 [  0.      0.      23.6799 ...   0.      0.      0.    ]
 ...
 [  0.      0.      0.     ... 179.9738 179.973  179.9722]
 [  0.      0.      0.     ...   0.     179.9746 179.9738]
 [  0.      0.      0.     ...   0.      0.     179.9753]]
```