회계가 바로 서야 경제가 바로 섭니다.

제06회 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
>>> 배열 = np.array([[3, 4, 7],
...                  [13, 5, 6],
...                  [10, 11, 9]])
>>> print(배열)
[[ 3  4  7]
 [13  5  6]
 [10 11  9]]
```

상기 2차원 Array를 다음과 같이 출력하고 싶다.

```
[[ 3  4  6]
 [10  5  7]
 [13 11  9]]
```

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

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

① ㉮ ② ㉯ ③ ㉰
④ ㉱ ⑤ ㉲

해설

㉮와 ㉯ 모두 출력 결과처럼 나온다.

정답 : ①, ②

02 다음과 같이 2차원 Array인 배열을 만들었다.

```
>>> import numpy as np
>>> 배열 =  np.array([[ 10 , 12 , 13 ],
...                   [ 45 , 32 , 16 ],
...                   [ 45 , 32 , 16 ]])
```

상기 2차원 Array를 다음과 같이 출력하고 싶다.

```
[[32 16]
 [32 16]]
```

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

㉮	print(배열[1:, 1:])
㉯	print(배열[1:3, 1:3])
㉰	print(배열[-2:, -2:])
㉱	print(배열[-2:, 1:])
㉲	print(배열[1:, -2:])

① ㉮ ② ㉯ ③ ㉰
④ ㉱ ⑤ ㉳

해설

㉮, ㉯, ㉰, ㉱, ㉳ 모두 출력 결과처럼 나온다.

정답 : ①, ②, ③, ④, ⑤

03 다음과 같이 3차원 Array인 배열을 만들었다.

```
>>> import numpy as np
>>> np.random.seed(50)
>>> 배열 = np.random.randint( 1, 100, size = (3,2,3))
>>> print(배열)
[[[49 97 12]
  [34 95  5]]

 [[71 71 23]
  [ 6  3 96]]

 [[72 69 79]
  [36 93 92]]]
```

상기 3차원 Array를 다음과 같이 출력하고 싶다.

```
>>> import numpy as np
>>> print(배열[np.where(배열 > ㉮)].reshape(㉯,㉰))
[[97 95 71 71 96]
 [72 69 79 93 92]]
```

㉮, ㉯, ㉰ 에 들어갈 내용으로 옳은 것으로 묶인 것을 모두 고르시오. **(3점)**

① ㉮ : 50, ㉯ : 2, ㉰ : 5
② ㉮ : 60, ㉯ : 5, ㉰ : 2
③ ㉮ : 48, ㉯ : 2, ㉰ : 5
④ ㉮ : 68, ㉯ : 5, ㉰ : 2
⑤ ㉮ : 68, ㉯ : 2, ㉰ : 5

해설

㉮는 49~68 중에 하나이어야 하며 ㉯는 2, ㉰는 5이어야 한다.

정답 : ①, ⑤

04 다음과 같이 코드를 작성하였다.

```
>>> import numpy as np
>>> np.random.seed(0)
>>> 배열 = np.random.randint( 1, 100, size = (3,2,3))
>>> print(배열)
[[[45 48 65]
  [68 68 10]]

 [[84 22 37]
  [88 71 89]]

 [[89 13 59]
  [66 40 88]]]
>>> 배열_1 = np.㉮(배열)
>>> print(배열_1)
[[[88 40 66]
  [59 13 89]]

 [[89 71 88]
  [37 22 84]]

 [[10 68 68]
  [65 48 45]]]
```

상기에서 ㉮ 에 들어갈 내용으로 옳은 것은? **(3점)**

① flipud
② fliplr
③ sort
④ flip
⑤ arrange

정답 : ④

05 다음 중 결과가 *다른* 것을 고르시오. **(2점)**

㉮	`import numpy as np` `print(np.array([[0],[1],[2]]) + np.arange(3))`
㉯	`import numpy as np` `print(np.arange(3) + np.array([[0],[1],[2]]))`
㉰	`import numpy as np` `print(np.array([0, 1, 2]).reshape(-1, 1) + np.arange(3))`
㉱	`import numpy as np` `print(np.add(np.array([[0], [1], [2]]), np.arange(3)))`
㉲	`import numpy as np` `print(np.array([0,1,2]) + np.arange(3))`

① ㉮ ② ㉯ ③ ㉰
④ ㉱ ⑤ ㉲

해설

㉮~㉱는 결과가 다음과 같다.

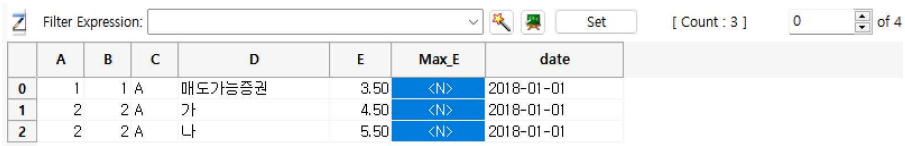
```
[[0 1 2]
 [1 2 3]
 [2 3 4]]
```

㉲ 는 결과가 다음과 같다.

```
[0 2 4]
```

정답 : ⑤

06 다음 Fraudit 테이블 명은 res1이며 sub table이 4개로 구성되어 있다.


Filter Expression: Set [Count : 3] 0 of 4

	A	B	C	D	E	Max_E	date
0	1	1	A	매도가능증권	3.50	<N>	2018-01-01
1	2	2	A	가	4.50	<N>	2018-01-01
2	2	2	A	나	5.50	<N>	2018-01-01

Max_E 칼럼의 값이 E열의 값 중 최대값을 나타내는 코드를 아래와 같이 작성하고자 한다. ㉮ ~ ㉰ 에 들어갈 내용으로 *옳지 않은* 것은? **(3점)**

```
>>> for i in range(len( ㉮ )) :
...     for ㉯ in range( len( ㉭ )) :
...         res1[㉱]['Max_E'][j] = ㉰(res1[i]['E'])
```

㉮ : res1	㉭ : res1	㉯ : j
㉱ : i	㉰ : max	

① ㉮ ② ㉭ ③ ㉯
④ ㉱ ⑤ ㉰

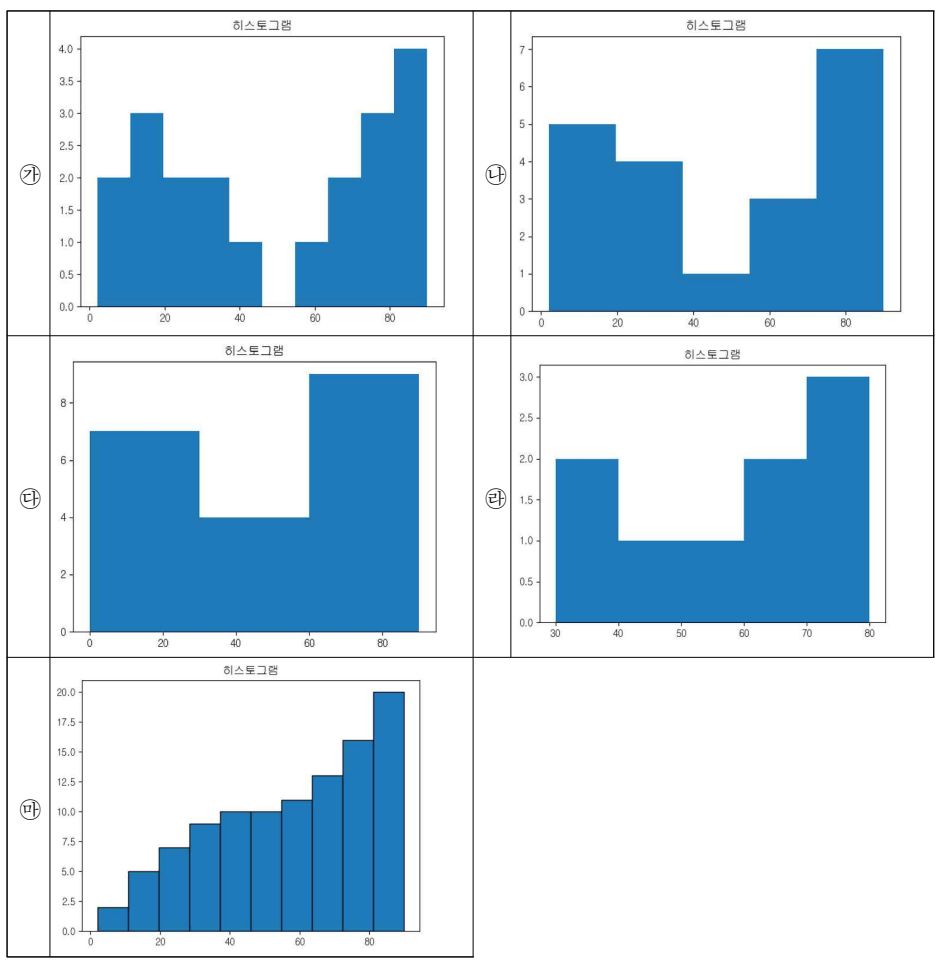
해설
㉭ res1[i]

정답 : ②

07 다음은 히스토그램 그래프를 보여주는 코드이다.

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> a = np.array([89, 34, 56, 87, 90, 23, 45, 12, 65, 78,
...               9, 34, 12, 11, 2, 65, 78, 82, 28, 78])
>>> plt.hist(a)
>>> plt.title("히스토그램")
>>> plt.show()
```

상기 코드 결과 그래프와 가장 유사한 그래프는 무엇인가? **(2점)**



① ㉮ ② ㉯ ③ ㉰
④ ㉱ ⑤ ㉲

정답 : ①

08 아래의 결과처럼 나타내고 싶다면 ㉮에 들어갈 코드로 옳은 것은? 단, print 함수를 사용하지 않는다. **(3점)**

```
>>> data = [
...   {'code': 1, '계정과목': '매출채권', '금액':1500, '재무제표': '재무상태표'},
...   {'code': 2, '계정과목': '매출원가', '금액':15000, '재무제표': '손익계산서'},
...   {'code': 3, '계정과목': '현금', '금액':1000, '재무제표': '현금흐름표'},
...   {'code': 4, '계정과목': '매출채권', '금액':500, '재무제표': '자본변동표'},
...         ]
>>> df = pd.DataFrame.from_records(data)
>>> df ㉮
0    재무상태표
1    손익계산서
2    현금흐름표
3    자본변동표
Name: 재무제표, dtype: object
```

① ['계정과목']　② ['재무제표']　③ ['code']
④ ['금액']　⑤ ['index']

정답 : ②

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

```
>>> df = purchases.toDF()
>>> df
    송장번호     일자     거래처  거래처코드 제품코드    금액
0        2 2009-01-01  Vijay  OJQ26  GNR  85351.0
1        2 2009-01-01  Vijay  HLP66  PYX  76560.0
2        2 2009-01-01  Vijay  JMW26  UMA  34024.0
3        3 2009-01-04  Vijay  QSG52  COK   1215.0
4        3 2009-01-04  Vijay  OTF93  JRH  46297.0
   ...        ...    ...    ...  ...      ...
8365  4994 2009-01-22  Vijay  ZXF02  PYX  55403.0
8366  4997 2009-01-15   Adam  RHA00  UMA   8343.0
8367  4998 2009-01-18  Vijay  NXH96  GLH  58731.0
8368  4998 2009-01-18  Vijay  RHA00  HVM  80754.0
8369  4999 2009-01-03  Suzie  YGX43  GNR  59376.0

[8370 rows x 6 columns]
```

상기 df를 가지고 피봇테이블을 만들고자 한다.

```
>>> import pandas as pd
>>> pd.pivot_table(df, index = ㉮,
...                    values = ㉯,
...                    columns = ㉰,
...                    aggfunc = ㉱,
...                    margins = ㉲)
제품코드         COK        CQZ        DEG  ...        ZIE        ZVX         All
거래처                                      ...
Adam    6851529.0  8527454.0  7292323.0  ...  7206651.0  8270612.0  116971024.0
Suzie   6824714.0  6551426.0  7865266.0  ...  7443609.0  7517765.0  104528687.0
Vijay  13439025.0 13586371.0 12802266.0  ... 12780077.0 13546846.0  203789118.0
All    27115268.0 28665251.0 27959855.0  ... 27430337.0 29335223.0  425288829.0
```

상기에서 ㉮, ㉯, ㉰, ㉱, ㉲에 들어갈 내용으로 *옳지 않은* 것을 고르시오. **(3점)**

㉮ : '거래처'	㉯ : '송장번호'	㉰ : '제품코드'
㉱ : 'sum'	㉲ : True	

① ㉮ ② ㉯ ③ ㉰
④ ㉱ ⑤ ㉲

해설

㉯ '금액'

정답 : ②

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

```
>>> df = purchases.toDF()
>>> df
    송장번호    일자     거래처  거래처코드 제품코드   금액
0       2 2009-01-01  Vijay  OJQ26  GNR  85351.0
1       2 2009-01-01  Vijay  HLP66  PYX  76560.0
2       2 2009-01-01  Vijay  JMW26  UMA  34024.0
3       3 2009-01-04  Vijay  QSG52  COK   1215.0
4       3 2009-01-04  Vijay  OTF93  JRH  46297.0
   ...       ...    ...    ...  ...      ...
8365  4994 2009-01-22  Vijay  ZXF02  PYX  55403.0
8366  4997 2009-01-15   Adam  RHA00  UMA   8343.0
8367  4998 2009-01-18  Vijay  NXH96  GLH  58731.0
8368  4998 2009-01-18  Vijay  RHA00  HVM  80754.0
8369  4999 2009-01-03  Suzie  YGX43  GNR  59376.0

[8370 rows x 6 columns]
```

상기 df에 다음과 같이 인덱스 칼럼을 삽입하고 거래처 칼럼을 기준으로 groupby 한 인덱스를 넣으려고 한다.

```
>>> import pandas as pd
>>> gb_1 = df.groupby('거래처')
>>> df['인덱스'] = gb_1.㉮
>>> df
```

	송장번호	일자	거래처	거래처코드	제품코드	금액	인덱스
0	2	2009-01-01	Vijay	OJQ26	GNR	85351.0	2
1	2	2009-01-01	Vijay	HLP66	PYX	76560.0	2
2	2	2009-01-01	Vijay	JMW26	UMA	34024.0	2
3	3	2009-01-04	Vijay	QSG52	COK	1215.0	2
4	3	2009-01-04	Vijay	OTF93	JRH	46297.0	2
...	...	...	...	...	...	...	...
8365	4994	2009-01-22	Vijay	ZXF02	PYX	55403.0	2
8366	4997	2009-01-15	Adam	RHA00	UMA	8343.0	0
8367	4998	2009-01-18	Vijay	NXH96	GLH	58731.0	2
8368	4998	2009-01-18	Vijay	RHA00	HVM	80754.0	2
8369	4999	2009-01-03	Suzie	YGX43	GNR	59376.0	1

[8370 rows x 7 columns]

상기에서 ㉮에 들어갈 내용으로 옳은 것을 고르시오. **(3점)**

① ngroup()　② group()　③ transform()
④ index()　⑤ nindex()

해설

ngroup()이 들어가야 한다.

정답 : ①

11 networkX 라이브러리에서 어떤 함수를 사용하면 두 노드 사이의 최단 경로를 찾을 수 있는가? **(2점)**

① path_short()
② shortest_distance()
③ find_path()
④ shortest_path()
⑤ path_finder()

해설

두 노드 사이의 최단 경로를 찾는 함수는 shortest_path()이다.

정답 : ④

주관식

12~16

당신은 (주)한공의 내부감사인이다. (주)한공의 회계프로그램에서 2020년 분개장을 입수하여 여러가지 분석을 하려고 한다. 2020년 분개장을 csv로 받은 것이 분개장_06회_1급_2.csv[1] 이며 이는 다음을 누르면 다운로드 받을 수 있다.

분개장_06회_1급_2.csv

분개장_06회_1급_2.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
전표	object	01-2019.12.19-0008-0001 형식으로 되어 있음. 여기서 2019.12.19은 전표일자를 의미하며, 0008은 전표번호를 의미한다.
계정코드	int64	계정코드
계정명	object	계정과목명(한글)
차변금액	int64	차변금액
대변금액	int64	대변금액
거래처코드	float64	거래처코드
승인일자	object	승인일자
발의부서	object	발의부서

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 2개이다.

	전표	계정코드	계정명	차변금액	대변금액	거래처코드	승인일자	발의부서
10	05-2019.12.18-0010-0001	4310504	복리후생비(관,경조금)	300000	0	nan	2020.01.01	기획총무팀
11	05-2019.12.18-0010-0002	2110801	미지급비용	0	300000	621.0	2020.01.01	기획총무팀
12	02-2019.12.18-0030-0001	1112301	선급금	0	200000	1017.0	2020.01.15	기획총무팀
13	02-2019.12.18-0030-0002	1110101	현금	200000	0	nan	2020.01.15	기획총무팀

1) 재무빅데이터분석사 홈페이지(https://fda.kicpa.or.kr) >> 고객센터 >> 자료실에서 다운로드 한다.

12 전표칼럼에서 전표일자와 전표번호를 발췌하여 각각 전표일자 칼럼과 전표번호 칼럼에 넣는다. 전표세트 중 차변합계와 대변합계가 일치하지 않은 전표세트의 갯수를 숫자만 입력하라. 단, 없으면 0을 입력한다. **(7점)**

정답 : 0

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_06회_1급_2.csv')
# 전표칼럼을 전표일자와 전표번호로 구분한다.
df.insert(1, '전표일자' , df['전표'].str[3:13])
df.insert(2, '전표번호' , df['전표'].str[14:18])
# 분개장 전표일자와 전표번호로 오름차순 만들기
df = df.sort_values(['전표일자', '전표번호'])
# 요약
gb_1 = df.groupby(['전표일자', '전표번호'])
df_차대 = pd.DataFrame({
                        '차변합계':gb_1['차변금액'].sum(),
                        '대변합계' : gb_1['대변금액'].sum()
                        }).reset_index()
df_차대['차이'] = df_차대['차변합계'] - df_차대['대변합계']
df_차이_발췌 = df_차대[df_차대['차이'] != 0]
print(df_차이_발췌)
```

```
Empty DataFrame
Columns: [전표일자, 전표번호, 차변합계, 대변합계, 차이]
Index: []
```

13 다음은 계정과목과 간주계정과목의 관계를 나타낸 표이다.

계정코드	계정과목	간주계정과목	계정코드	계정과목	간주계정과목
1110101	현금	**현금**	1110302	보통예금	**보통예금**
1110303	외화보통예금	**보통예금**	1110702	외상매출금	**매출채권**
1110704	외상매출금(Direct)	**매출채권**	1110712	받을어음	**매출채권**
1111701	미수금	**매출채권**	2110601	선수금(일반)	**매출채권**
4111101	제품매출	**매출**	4111301	제품수출매출(Direct)	**매출**
4111302	제품수출매출(Local)	**매출**	4111501	제품매출(반품)	**매출**
4130101	상품매출(일반)	**매출**	4130105	상품수출매출(Local)	**매출**
4130501	상품매출(반품)	**매출**			

간주계정과목이 '보통예금'이면서 차변금액이 0을 초과하고, 간주계정과목이 '매출'이면서 대변금액이 0을 초과하는 경우를 포함하는 전표세트를 보통예금매출 전표세트라고 가정한다. 보통예금매출 전표세트의 개수를 숫자로만 나타내라. 단, 천단위 콤마는 나타내지 않는다. **(7점)**

정답 : 137

해설

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

```
# 판다스 라이브러리 불러오기
import pandas as pd
# 보통예금과 매출 계정코드 리스트 작성
보통예금_리스트 = [1110302, 1110303]
매출_리스트 = [4111101,4111301,4111302,4111501, 4130101, 4130105, 4130501]
# df에 현금빈도와 매출빈도 칼럼 삽입
df['보통예금_차변_빈도'] = ((df['계정코드'].isin(보통예금_리스트)) & (df['차변금액']
>0)).astype(int)
df['매출_대변_빈도']   =   ((df['계정코드'].isin(매출_리스트))   &   (df['대변금액']
>0)).astype(int)
# 그룹바이 : 전표일자, 전표번호
gb_1 = df.groupby(['전표일자', '전표번호'])
# df에 그룹바이 인덱스를 전표_인덱스 칼럼에 삽입
df['전표_인덱스'] = gb_1.ngroup()
# 요약 데이터프레임 작성
df_요약 = pd.DataFrame({
                        '보통예금차변빈도':gb_1['보통예금_차변_빈도'].sum(),
                        '매출대변빈도' : gb_1['매출_대변_빈도'].sum()
                        }).reset_index().reset_index()
# 요약 데이터프레임의 index 칼럼명을 전표_인덱스로 수정
```

```python
df_요약.rename(columns = {'index': '전표_인덱스'},inplace =True)
# 보통예금차변빈도 > 0 & 매출대변빈도 > 0인 항목 발췌
df_요약_발췌 = df_요약[(df_요약['보통예금차변빈도'] >0) & (df_요약['매출대변빈도'] >0)]
# 결과 출력
print(df_요약_발췌)
```

```
      전표_인덱스        전표일자  전표번호  보통예금차변빈도  매출대변빈도
19         19  2020.01.02  0006         1       1
73         73  2020.01.03  0003         1       1
125       125  2020.01.06  0002         1       1
127       127  2020.01.06  0004         1       1
195       195  2020.01.08  0007         1       1
...       ...         ...   ...       ...     ...
8834     8834  2020.09.24  0011         1       1
8836     8836  2020.09.24  0013         6       1
8843     8843  2020.09.24  0020         6       1
9000     9000  2020.09.29  0022         2       1
9127     9127  2020.09.30  0029         1       1
[137 rows x 5 columns]
```

14 다음은 계정과목과 계정코드의 관계를 나타낸 표이다.

계정코드	계정과목	계정코드	계정과목
1110101	현금	1110302	보통예금
1110303	외화보통예금	1110702	외상매출금
1110704	외상매출금(Direct)	1110712	받을어음
1111701	미수금	2110601	선수금(일반)
4111101	제품매출	4111301	제품수출매출(Direct)
4111302	제품수출매출(Local)	4111501	제품매출(반품)
4130101	상품매출(일반)	4130105	상품수출매출(Local)
4130501	상품매출(반품)	2112001	반품충당부채(환불부채)

정상적인 매출거래 전표세트의 정의는 다음과 같다.

① 대변 계정과목이 ['제품매출', '제품수출매출(Direct)','제품수출매출(Local)', '제품매출(반품)', '상품매출(일반)','상품수출매출(Local)', '상품매출(반품)'] 중 하나 이며, 해당 대변 계정과목의 대변금액이 0을 초과하거나 차변금액이 0 미만인 경우가 포함된다.
② 차변 계정과목이 ['현금','보통예금', '외화보통예금' ,'외상매출금', '외상매출금(Direct)','받을어음', '미수금', '선수금(일반)', '반품충당부채(환불부채)'] 중 하나 이며, 해당 차변계정과목의 차변금액이 0을 초과하거나 대변금액이 0미만 경우가 포함된다.

따라서, 정상적인 매출거래가 아닌 전표세트의 정의는 다음과 같다.

① 대변 계정과목이 ['제품매출', '제품수출매출(Direct)','제품수출매출(Local)', '제품매출(반품)', '상품매출(일반)','상품수출매출(Local)', '상품매출(반품)'] 중 하나 이며, 해당 대변 계정과목의 대변금액이 0을 초과하거나 차변금액이 0 미만인 경우가 포함된다.
② 차변 계정과목이 ['현금','보통예금', '외화보통예금' ,'외상매출금', '외상매출금(Direct)','받을어음', '미수금', '선수금(일반)','반품충당부채(환불부채)'] 이 아닌 항목들로만 구성된 경우이다.

정상적인 매출거래가 아닌 전표세트의 개수를 숫자로만 나타내라. 단, 천단위 콤마는 나타내지 않는다. **(7점)**

정답 : 5

해설

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

```
# 판다스 라이브러리 불러오기
import pandas as pd
#매출 리스트
매출_리스트 = [4111101,4111301,4111302,4111501, 4130101, 4130105, 4130501]
#매출상대_관련계정코드
매출상대_리스트 = [1110101, 1110302,1110303, 1110702, 1110704,1110712,
                1111701,2110601,2112001]
# df에 매출빈도, 매출상대_차변_빈도 칼럼 삽입
df['매출_대변_빈도'] = ((df['계정코드'].isin(매출_리스트)) &
                    ((df['대변금액'] >0)|(df['차변금액'] <0))).astype(int)
df['매출상대_차변_빈도'] = ((df['계정코드'].isin(매출상대_리스트)) &
                    ((df['차변금액'] >0)|(df['대변금액'] <0))).astype(int)
#groupby : 전표번호, 전표일자
gb_전표_인덱스 = df.groupby('전표_인덱스')
#summarize 테이블
df_매출_관련계정_총괄 = pd.DataFrame({
                '매출상대_차변빈도' : gb_전표_인덱스['매출상대_차변_빈도'].sum(),
                '매출_대변빈도' : gb_전표_인덱스['매출_대변_빈도'].sum(),
                '차변합계' : gb_전표_인덱스['차변금액'].sum(),
                '대변합계' : gb_전표_인덱스['대변금액'].sum()
                 }).reset_index()
df_매출_관련계정_총괄=df_매출_관련계정_총괄[(df_매출_관련계정_총괄['매출상대_차변빈도']==0)
                    & (df_매출_관련계정_총괄['매출_대변빈도'] >0) ]
#상세테이블 작성
상세분석_리스트 = df_매출_관련계정_총괄['전표_인덱스'].tolist()
#상세테이블 작성
매출_관련계정_상세 = TableArray()
for i in 상세분석_리스트:
    매출_관련계정_상세 = 매출_관련계정_상세 + TableArray(
                    Table(data = df[df['전표_인덱스']==i]))
if len(매출_관련계정_상세) ==0 :
    del 매출_관련계정_상세
매출_관련계정_상세.set_format('차변금액', "#,###")
매출_관련계정_상세.set_format('대변금액', "#,###")
print(len(매출_관련계정_상세))
```

```
5
```

15 다음은 계정과목과 간주계정과목의 관계를 나타낸 표이다.

계정코드	계정과목	간주계정과목	계정코드	계정과목	간주계정과목
4111101	제품매출	**매출**	4111301	제품수출매출(Direct)	**매출**
4111302	제품수출매출(Local)	**매출**	4111501	제품매출(반품)	**매출**
4130101	상품매출(일반)	**매출**	4130105	상품수출매출(Local)	**매출**
4130501	상품매출(반품)	**매출**	1110702	외상매출금	**매출채권**
1110704	외상매출금(Direct)	**매출채권**	1110712	받을어음	**매출채권**
1110762	구매카드채권	**매출채권**	2110101	외상매입금	**매입채무**
2110102	구매카드채무	**매입채무**	2110113	구매자금채무	**매입채무**

매출거래와 매입거래의 논리적 정의(알고리즘)는 다음과 같다

(1) 매출거래는 다음의 조건을 모두 충족하는 레코드이다.
  ① 간주계정과목이 매출이거나 매출채권이다.
  ② 차변금액이 0이 아니거나, 대변금액이 0이 아니다.
(2) 매입거래는 다음의 조건을 모두 충족하는 레코드이다.
  ① 간주계정과목이 매입채무이다.
  ② 차변금액이 0이 아니거나, 대변금액이 0이 아니다.

따라서, 거래처코드로 groupby를 한 후 매출거래의 빈도(count)와 매입거래의 빈도(count)가 모두 1 이상이면 이는 매출거래와 매입거래가 모두 발생한 거래처라고 정의한다. (단, 거래처코드가 NaN인 경우 fillna(0)을 이용하여 0으로 처리하고 그 거래처는 매출거래와 매입거래가 모두 발생한 거래처에서 제외한다.)

매출거래와 매입거래가 모두 발생한 거래처의 개수를 숫자로만 나타내라. 단, 천 단위 콤마는 나타내지 않는다. **(7점)**

정답 : 87

해설

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

```
# 판다스 라이브러리 불러오기
import pandas as pd
#매출 리스트
매출_계정코드 = [4111101, 4111301, 4111302, 4111501, 4130101, 4130105, 4130501]
#매출채권 리스트
매출채권_계정코드 = [1110702, 1110704, 1110712, 1110762]
#매입채무 리스트
매입채무_계정코드 = [2110101, 2110102, 2110113]
# 매출통합계정 리스트
매출_통합계정 = 매출_계정코드 + 매출채권_계정코드
# 매입통합계정 리스트
매입_통합계정 = 매입채무_계정코드
# df에 매출빈도, 매입빈도 칼럼 삽입
df['매출_빈도'] = ((df['계정코드'].isin(매출_통합계정)) &
              ((df['대변금액'] != 0)|(df['차변금액'] != 0))).astype(int)
df['매입_빈도'] = ((df['계정코드'].isin(매입_통합계정)) &
              ((df['대변금액'] != 0)|(df['차변금액'] != 0))).astype(int)
# 거래처 NaN을 0처리
df['거래처코드'].fillna(0, inplace = True)
# groupby : 거래처 코드
gb_거래처 = df.groupby('거래처코드')
# 매출_매입_총괄 요약
df_매출_매입_요약 =  pd.DataFrame(
                  {'매출_빈도' : gb_거래처['매출_빈도'].sum(),
                  '매입_빈도' : gb_거래처['매입_빈도'].sum(),
                  '전표_인덱스_빈도' : gb_거래처['전표_인덱스'].nunique(),
                  '전표_인덱스' : gb_거래처['전표_인덱스'].unique(),
                  '차변금액' : gb_거래처['차변금액'].sum(),
                  '대변금액' : gb_거래처['대변금액'].sum(),
                  }).reset_index()
# df_매출_매입_요약에서 매출_빈도 > 0 & 매입_빈도 > 0 인 경우 발췌
df_매출_매입_발췌 =  df_매출_매입_요약[(df_매출_매입_요약['매출_빈도'] > 0) &
                  (df_매출_매입_요약['매입_빈도'] > 0)].reset_index()
# df_매출_매입_발췌의 개수
print(len(df_매출_매입_발췌))
```

```
87
```

16 거래처 중 가장 많은 계정과목(계정코드)의 개수를 가지는 거래처에 대해서는 면밀한 검토가 필요하다고 보았다. 하나의 거래처가 중복되는 계정과목(계정코드)를 가지는 경우 중복을 제외한 계정과목(계정코드)의 개수가 가장 많은 거래처가 검토 대상이다. 단, 거래처코드가 NaN인 경우 fillna(0)을 이용하여 0으로 처리하고 그 거래처는 제외한다. 중복을 제외한 계정과목(계정코드)의 개수가 가장 많은 거래처코드(0을 제외한다)의 중복을 제외한 계정과목(계정코드)의 개수를 숫자로만 나타내라. 즉, 거래처코드가 아니라 중복을 제외한 계정과목(계정코드)의 개수를 나타내라. 단, 천단위 콤마는 나타내지 않는다. **(8점)**

정답 : 15

해설

```
# 판다스 라이브러리 불러오기
import pandas as pd
# groupby : 거래처 코드
gb_거래처 = df.groupby('거래처코드')
# 거래처_계정과목 요약
df_거래처_계정과목_요약 = pd.DataFrame(
                  {'계정과목_빈도' : gb_거래처['계정명'].nunique(),
                  '계정코드_빈도' : gb_거래처['계정코드'].nunique(),
                  '전표_인덱스_빈도' : gb_거래처['전표_인덱스'].nunique(),
                  '차변금액' : gb_거래처['차변금액'].sum(),
                  '대변금액' : gb_거래처['대변금액'].sum(),
                  }).reset_index().reset_index()
# sorting
print(df_거래처_계정과목_요약.sort_values('계정코드_빈도', ascending = False))
```

```
     상세분석  거래처_인덱스   거래처코드  계정과목_빈도  계정코드_빈도  \
0       0       0     0.0      202      202
612     0     612   612.0       15       15
846     0     846   846.0       14       14
1245    0    1245  1245.0       11       11
945     0     945   945.0       10       10
...   ...     ...     ...      ...      ...
99      0      99    99.0        1        1
529     0     529   529.0        1        1
530     0     530   530.0        1        1
531     0     531   531.0        1        1
1338    0    1338  1338.0        1        1
```

17 (주)한공은 신제품을 출시할 계획이다. 판매량, 판매단가, 단위당 재료비는 불확실성이 존재하는 요소이며, 이에 대해서는 다음과 같은 분포라고 가정한다.

구분	분포	파라미터
판매량	PERT(low, peak, high)	low = 6000, peak = 12000, high = 18000
판매단가	Normal(mu, sigma) or N(mu, sigma)	mu(평균) = 20(유로) sigma(표준편차) = 1.00(유로)
단위당 재료비	Normal(mu, sigma) or N(mu, sigma)	mu(평균) = 13(유로) sigma(표준편차) = 0.7(유로)

기타 단위당 원가는 3.0(유로)로 고정된다고 가정한다. 결과 변수의 산식은 다음과 같다.

구분	산식
수익	판매량 × 판매단가
총원가	판매량 × (단위당 재료비 + 기타 단위당 원가)
총이익	수익 - 총원가
이익률	총이익 / 수익

mcerp 라이브러리를 이용하여, 시뮬레이션을 10,000번 수행한 결과 총이익이 35,000유로 이상일 확률을 소수점 셋째 자리에서 반올림하여 구하라(예: 0.123→0.12). 단, 시드는 np.random.seed(0)으로 고정시키고 다음과 같이 라이브러리를 불러온다. **(5점)**

```
import numpy as np
import pandas as pd
from pandas import *
import mcerp
from mcerp import *
import matplotlib.pyplot as plt
```

정답 : 0.76

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
# 입력변수의 파라미터 정하기
min_판매량, mode_판매량, max_판매량 =6000.0, 12000.0, 18000.0
lmb_판매 = 4.0
평균_판매단가, 표준편차_판매단가 = 20.0, 1.0
평균_단위당재료비, 표준편차_단위당재료비 = 13.0, 0.7
기타단위당원가 = 3.0
# 시드 고정하기
np.random.seed(0) # seed를 0으로 고정시킨다.
# 입력변수의 분포 설정하기
판매량_i = PERT(min_판매량, mode_판매량, max_판매량)
판매단가_i = N(평균_판매단가, 표준편차_판매단가)
단위당재료비_i = N(평균_단위당재료비, 표준편차_단위당재료비)
기타단위당원가_i = 기타단위당원가
# 결과변수의 산식 설정하기
수익_o = 판매량_i * 판매단가_i
총원가_o = 판매량_i * (단위당재료비_i + 기타단위당원가_i)
총이익_o = 수익_o - 총원가_o
이익률_o = 총이익_o/수익_o
# 결과변수의 분포 출력 및 threshold 분석
총이익_o.plot()
plt.axvline(x =35000, color ='red')
plt.show()
print(총이익_o > 35000)
```

18 A회사는 미국으로부터 항생제를 수입하여 국내에 판매하고 있다. 이 항생제는 유통기한이 짧아 부패하기 쉬운 제품이다. A회사는 3년 동안의 판매 실적 데이타를 잘 관리해 오고 있다. A회사의 관리부 김확실 부장은 수입을 많이 해서 재고로 발생하는 폐기 비용과 수입한 물량보다 수요가 많이 발생해서 재수입하는 추가 운송비용을 최소화 할 수 있는 월 적정 수입 물량을 알고 싶어한다. 3년치의 월 평균 판매량과 손실비용, 운송비용에 대한 내역은 다음과 같다.

- 3년치의 월 평균 판매량 : 5,000개
- 수입이 수요보다 많아 월말 폐기해야 하는 손실 비용 : $50/개
- 수입이 수요보다 부족해 추가 급주문으로 인한 운송 비용 : $10/개

월 판매량은 불확실성이 존재하는 요소이며, 이에 대해서는 다음과 같은 분포라고 가정한다.

구분	분포	파라미터
월 판매량	Normal(평균, 표준편차)	평균 = 5000, 표준편차 = 2000

한편, 제약조건인 수입량의 범위는 다음과 같다.

구분	범위(단위 : 개)	변동폭(단위 : 개)
월 수입량	2,000~7,000	10(즉, 물류를 고려하여 10개 단위로만 수입 가능)

결과 변수의 산식은 다음과 같다.

구분	산식(논리를 설명한 것이므로, 파이썬 코드 자체를 의미하는 것이 아님. 따라서 파이썬 코드를 아래와 동일하게 작성하면 에러가 날 수 있음)
폐기 비용	단위당 폐기비용 = 50

	if 월 수입량 >= 월 판매량 :
	  폐기 비용 = (월 수입량 – 월 판매량) * 단위당 폐기비용
	else :
	  폐기 비용 = 0

운송 비용	단위당 운송비용 = 10

	if 월 수입량 <= 월 판매량 :
	  운송 비용 = (월 판매량 – 월 수입량) * 단위당 운송비용
	else :
	  운송 비용 = 0

총 비용	폐기 비용 + 운송 비용

총 비용의 평균을 최소화하는 월 수입량을 숫자만 입력하고 천단위 콤마는 제외한다. 변동폭 10개를 고려해야 하므로 수입량은 10개씩 변동가능하다(예: 최적해가 516이라고 하면 최적해는 510 또는 520 중 더 최적인 것을 선택한다). mcerp, geneticalgorithm, matplotlib 라이브러리를 이용하고, 시뮬레이션을 1,000번 수행하고, 시드는 np.random.seed(0)으로 고정시킨다. 따라서 다음과 같이 먼저

라이브러리를 불러오고 설정을 한다. **(6점)**

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

정답 : 3060

해설

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

```
# 관련 라이브러리를 불러온다.
import numpy as np
import mcerp
from mcerp import *
from geneticalgorithm import geneticalgorithm as ga
import matplotlib.pyplot as plt
# 시드 고정하기
np.random.seed(0)
# 시뮬레이션 횟수 정하기
mcerp.npts = 1000
# 입력변수(월 판매량)의 분포 반영
입력변수 = Normal(5000, 2000)
# 입력변수(월 판매량)의 array 변환
입력변수_1 = 입력변수._mcpts
# 목표함수 설정 : 사용자 함수
def f(X):
    global 입력변수_1
    결정변수_1 = X[0]
    단위당_폐기비용 = 50
    단위당_운송비용 = 10
    폐기비용 = np.where(결정변수_1 >= 입력변수_1,
                    (결정변수_1 - 입력변수_1) * 단위당_폐기비용, 0)
    운송비용 = np.where(결정변수_1 <= 입력변수_1,
                    (입력변수_1 - 결정변수_1) * 단위당_운송비용, 0)
    결과변수 = 폐기비용 + 운송비용
    return np.mean(결과변수)
# 입력변수 범위
```

```
varbound = np.array([[2000,7000]])
# 최적화 함수(ga) 파라미터 입력
model = ga(function = f,
           dimension =1,
           variable_type = 'int',
           variable_boundaries = varbound)
# 최적화 함수(ga) 실행
model.run()
```

19 당신은 ㈜한공의 세전매출의 적정성을 테스트하기 위하여 MUS 샘플링을 수행하려고 한다. 관련 자료는 Sample.tbl이다. Sample.tbl은 Fraudit 교육용 버전 왼쪽 네비게이터의 Tables 트리의 samples 폴더안에 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	95%

Random starting point는 10000을 입력하고 샘플을 추출한다.

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

송장번호	영업사원번호	CUSTNO	제품코드	세전매출	세전매출_Audit
A900056	102	30608	5	4,617.34	3,000
A900059	102	21256	4	188,381.16	185,000

Fraudit을 이용하여 샘플 추출한 후 이를 분석한 결과가 다음과 같았다.

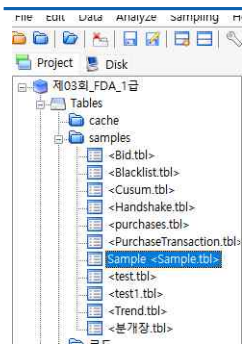
Conclusion : Accept the population as fairly stated..

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)**

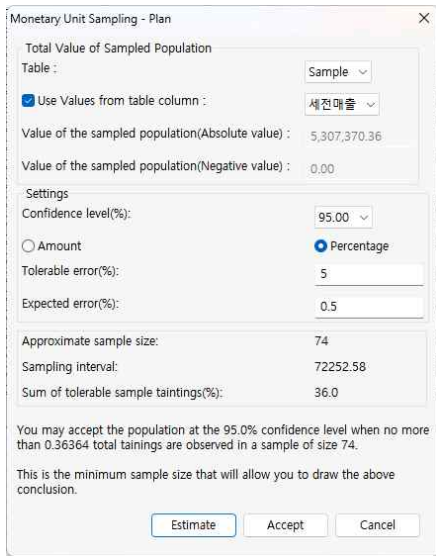
정답 : 262,508.86

해설



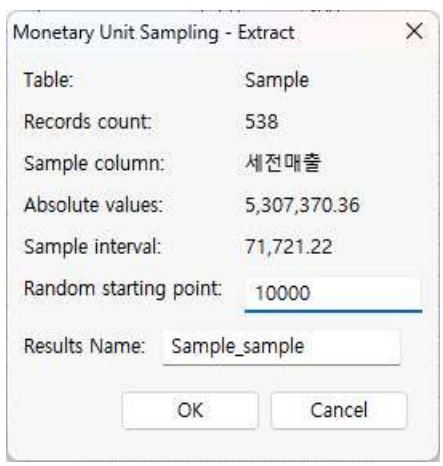
(1) Sample 테이블을 연다.

(2) Sampling >> MUS(PPS) >> Plan, Extract 메뉴를 선택한다.



(3) Table : **Sample** 선택
- Use Values from table column : **세전매출**
- Confidence level(%) : 95.00
- Percentage : 체크
- Tolerable error(%) : 5
- Expected error(%) : 0.5

(4) Estimate 버튼을 누른다.
(5) Accept 버튼을 누른다.



(6) Random starting point : 10000 입력
(7) Results Name : **Sample_sample** 입력
(8) OK 버튼을 누른다.

	Row	송장번호	송장일자	영업사원번호	CUSTNO	제품코드	단가	수량	세전매출	세전매출_Audit	매출세	세후매출
0	1	A900027	2014-03-15 00:00:00	101	21256	3	34.99	1836	64,241.64	64,241.64	9636.25	73877.89
1	4	A900030	2014-03-16 00:00:00	102	21285	1	28.5	1116	31,806.0	31,806.0	4770.9	36576.9
2	5	A900032	2014-03-02 00:00:00	101	21330	2	23.15	1248	28,891.2	28,891.2	4333.68	33224.88
3	15	A900041	2014-06-22 00:00:00	102	21425	5	5.59	222	1,240.98	1,240.98	186.15	1427.13
4	17	A900043	2014-06-17 00:00:00	102	21450	3	34.99	587	20,539.13	20,539.13	3080.87	23620.0
5	30	A900056	2014-04-06 00:00:00	102	30608	5	5.59	826	4,617.34	31.000	692.6	5309.94
6	33	A900059	2014-03-26 00:00:00	102	21256	4	99.99	1884	188,381.16	185.000	28257.17	216638.33
7	33	A900059	2014-03-26 00:00:00	102	21256	4	99.99	1884	188,381.16	185.000	28257.17	216638.33

(9) 세전매출_Audit 칼럼에서 상기 음영 부분과 같이 숫자를 입력한다.

(10) Sampling >> MUS(PPS) >> Evaluation 메뉴를 선택한다.

(11) 왼쪽의 결과를 확인한다.

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

제품매출

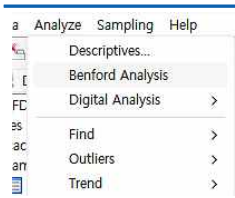
상기 파일을 불러온 후의 테이블 구조는 다음과 같다.

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

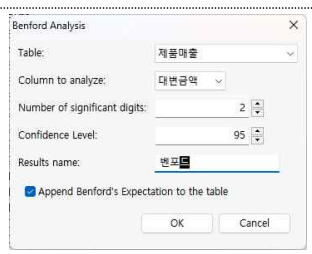
대변금액의 앞의 2단위(예: 대변금액이 12345이면 12를 의미함)에 대해서 벤포드 분석을 하려고 한다. 앞의 2단위의 벤포드 평균이 Expected라고 가정할 경우 실제 발견된 비율(Found Distribution)의 Z값이 2.5를 초과하는 항목의 개수는 몇 개인지 숫자만 입력하라. 예를 들어 앞의 2단위가 10, 11, 12, 13이며 이들의 Z값이 2.5를 초과한다면 4를 입력한다. **(7점)**

정답 : 3

해설



(1) Analyze >> Benford Analysis 메뉴를 누른다.



(2) Tabe : **제품매출** 선택
(3) Column to analyze : **대변금액** 칼럼 선택
(4) Number of significant digits : 2 선택
(5) Confidencee 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	80	27	.0208	.0054	7.4025
1	14	55	.0425	.0300	2.5586
2	20	41	.0317	.0212	2.5200
3	16	20	.0154	.0263	2.3597
4	10	70	.0541	.0414	2.2176

(9) Z_score가 2.5을 초과하는 2단위 항목은 80, 14, 20 총 3개이다.

21 당신은 적요란에 포함된 데이터를 분석하여 지역별, 연령별 연봉금액을 비교하려고 한다. 적요란에 포함된 데이터만 추출하여 csv 파일로 변환한 것이 『적요.csv』 파일이며 이는 아래의 링크를 누르면 다운로드 받을 수 있다.

『적요.csv』

『적요.csv』 의 레코드 수는 100000개이며 데이터의 첫 번째와 두 번째 레코드는 다음과 같다.

첫 번째 레코드	yejun83@naver.com, 920524-8131112, 052-038-7998, 세종특별자치시 노원구 삼성922로, 연봉 111736000원, 강성민
두 번째 레코드	연봉 136778000원, juweon50@naver.com, 세종특별자치시 용산구 가락3로, 이현주, 740921-6402472, 031-680-3538

레코드에 입력된 내용은 이름, 주소, 주민등록번호, 전화번호, 이메일, 연봉이며 이는 랜덤으로 섞여 들어가 있다. 각 항목에 대한 설명은 다음과 같다.

연봉	연봉 XXXXX원 또는 연봉 ₩XXXXX 형식으로 입력
주소	광역시도의 명칭은 다음과 같다. '서울특별시', '부산광역시', '대구광역시', '인천광역시', '광주광역시', '대전광역시', '울산광역시', '세종특별자치시', '경기도', '강원도', '충청북도', '충청남도', '전라북도', '전라남도', '경상북도', '경상남도', '제주특별자치도'

『적요.csv』 를 데이터프레임으로 불러온 후 광역시도별 연봉의 합계를 구한 결과 연봉의 합계가 가장 낮은 광역시도의 명칭을 입력하라. **(5점)**

정답 : 강원도

해설

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

```
# 관련 라이브러리를 불러온다.
import pandas as pd
import re

df = pd.read_csv('적요.csv')
def extract_sido(text):
    pattern = r'(경기도|강원도|충청북도|충청남도|전라북도|전라남도|경상북도|경상남도|제주특별자치도|\w*특별시|\w*광역시|\w*자치시)'
    match = re.search(pattern, text)
    if match:
        return match.group(1)
    return None
```

```
def extract_salary(text):
    pattern = r'연봉\s*(\d+[,\d]*원|₩\d+[,\d]*)'
    match = re.search(pattern, text)
    if match:
        return match.group(1).replace('₩', '').replace('원', '').replace(',', '')
    return None

df['시도명'] = df['적요'].apply(extract_sido)
df['연봉'] = df['적요'].apply(extract_salary).astype('int64')

gb_시도명 = df.groupby(['시도명'])
df_요약_시도명 = pd.DataFrame({'연봉합계' : gb_시도명['연봉'].sum()}).reset_index()
print(df_요약_시도명.sort_values('연봉합계'))
```

	시도명	연봉합계
0	강원도	685107434000
3	경상북도	690868608000
15	충청남도	692795639000
16	충청북도	693682537000
1	경기도	697448157000
13	전라북도	697876556000
14	제주특별자치도	699714762000
12	전라남도	713500738000
2	경상남도	713548688000
7	부산광역시	766729030000
8	서울특별시	775067522000
11	인천광역시	775666394000
5	대구광역시	777259854000
10	울산광역시	779778736000
9	세종특별자치시	780056307000
6	대전광역시	786589901000
4	광주광역시	786857182000

22 다음은 콜옵션과 관련된 내용이다.

• 아메리칸 콜옵션	• 현재주가 : 100	• 행사가격 : 120
• 무위험이자율 : 6%	• 배당률 : 0%	• 만기 : 3년
• 변동성 : 20%	• 스텝(노드) : 300	• dt = 만기/스텝(노드)

콜옵션 가격을 소수점 첫째 자리에서 반올림하여 입력하라.(예 : 1.5 → 2)

단, 소수점 처리를 위해서 코드 상단에 np.set_printoptions(precision=4, suppress =True)를 입력해서 처리하라. **(5점)**

정답 : 14

해설

```
>>> import numpy as np
>>> np.set_printoptions(precision = 4, suppress = True)
    # 기초변수 설정(한줄에 여러 변수를 설정하는 경우 세미콜론으로 구분함)
>>> 최초주가 = 100 ; 행사가격 = 120 ; 무위험이자율 = 0.06 ; 변동성 = 0.20
>>> 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((주가 - 행사가격), 0) 계산
>>> 옵션_트리[:,스텝] = np.maximum((주가_트리[:,스텝] - 행사가격), 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(주가_트리[node,time]- 행사가격,
...                 (옵션_트리[node,time+1]*P_U+옵션_트리[node+1,time+1]*P_D)*할인계수)
>>> print(옵션_트리) # 소수점 2째자리 반올림
[[   13.6558    14.7728    15.9566 ... 38641.1563 39424.1088 40222.8793]
 [    0.        12.5099    13.5588 ... 37121.3153 37873.5649 38641.0124]
 [    0.         0.        11.4335 ... 35661.0681 36383.8188 37121.1714]
```

```
...
[   0.       0.       0.    ...    0.       0.       0.   ]
[   0.       0.       0.    ...    0.       0.       0.   ]
[   0.       0.       0.    ...    0.       0.       0.   ]]
```