과정명		모듈명		회차명	

◈ 학습내용

- 판다스가 제공하는 데이터셋(Series, DataFrame, Panel)
- 판다스가 제공하는 데이터셋의 연산

◈ 학습목표

- 판다스 라이브러리의 필요성을 이해한다
- 판다스 라이브러리를 이용해 데이터셋을 만들 수 있다
- 파이썬에 제공하는 기본타입들을 판다스 데이터 셋으로 전환하고 원하는 방식으로 데이터를 제어할 수 있다

내레이션

(기입하지 않습니다.)

과정명		모듈명		회차명	

간지 부분

◈ 판다스란

1. 판다스 자료구조
2. 판다스를 이용한 연산

내레이션

(기입하지 않습니다.)

◈ 데이터셋

- 앞선 과정에서 우린 다양한 소스들로부터 데이터를 수집하는 방법을 배웠습니다.
- 수집된 데이터들은 각자 동일하거나 다른 형태의 데이터들로 되어 있습니다.
- 데이터셋은 다향한 형태의 데이터들을 하나의 구조로 만들어서 관리나 분석을 용이하게 할 수 있습니다.
- 파이썬에서는 데이터셋을 관리하기 위해 pandas라는 라이브러리를 제공합니다.
- 데이터 분석을 위해서는 파이썬에서 기존에 사용하던 타입인 list, dict 과 같은 것 들을 그냥 사용할 수는 없습니다.
- 머신러닝이나 딥러닝에서는 numpy나 pandas 라이브러를 이용해서 파이썬이 제공하는 기본 타입들을 데이터 분석에 사용하기 쉽도록 제공하고 있습니다.

내레이션

◈ pandas library

- pandas는 데이타 분석(Data Analysis)을 위해 널리 사용되는 파이썬 라이브러리 패키지입니다.
- 클래스를 별도로 만들지 않고도 pandas 라이브러릴 이용하여 데이터를 읽고 쓸 수 있습니다.
- 파이썬에 제공하는 통계관련 패키지들에서도 자주 사용하는 라이브러리 입니다.
- 아나콘다(Anaconda)를 이용해 파이썬을 설치하셨다면 이미 pandas 패키지는 설치되어 있습니다.
- 만일 별도로 파이썬만 설치되어 있다면, pandas 를 별도로 설치해야 합니다.

내레이션

◈ pandas library

- 콘솔창에서 pip install pandas 라고 하면 설치가 됩니다. 혹시 설치가 되지 않는다면, 파이썬의 path 설정이 잘못 되어 있거나, 권한이 부족한 경우입니다. pip 명령이 작동 되지 않는다면 환경변수에 경로를 지정해야 하고, 권한이 부족하여 설치가 되지 않는다면 콘솔창의 마우스 오른쪽을 눌러서 관리자 권한으로 실행하면 됩니다. 이 과정에서는 처음부터 아나콘다 패키지를 이용해 파이썬을 설치했으므로 그냥 사용하시면 됩니다.

내레이션

◈ pandas library의 구성및 내용

- pandas 라이브러리는 클래스와 여러가지의 내장함수로 구성되어있습니다.
- 제공하는 데이터 타입은 Series(1차원), Dataframe(2차원), Panel(3차원)입니다
- 판다스를 이용해 list나 dict 타입등을 판다스에서 사용하는 데이터 구조로 변환할 수 있습니다.
- 데이터 분석에서 사용하는 데이터 타입은 파이썬에 제공하는 list나 dict 타입을 그냥 사용하기에는 무리가 있습니다.
- pandas 는 csv파일을 읽고 쓰거나, excel 파일을 읽고 쓰기 쉽도록 구성되어 있습니다.

내레이션

◈ pandas 사용법

- pandas를 사용하려면 pandas 라이브러리를 import 하여야 합니다.
- import 시 별명을 붙여서 사용하는 경우가 많습니다.
- import pandas as pd
- pandas 에서 제공하는 데이터 타입은 1차원 구조인 Series, 2차원 구조인 DataFrame, 3차원 구조인 Panel이 있습니다. (3차원은 현재 사용 못함)
- 기존의 언어와 달리 각 차원별로 별도의 이름을 붙입니다.

내레이션

◈ Series

- pandas가 제공하는 1차원 구조의 타입입니다
- Series 구조는 배열/리스트와 같은 일련의 시퀀스(연속된) 데이터를 저장합니다.
- list 타입을 Series 구조로 바꿀 수 있습니다. list 타입을 Series 로 바꿀때는 데이터를 식별하기 위해 인덱스가 지정됩니다.
- 별도의 레이블 인덱스를 지정하지 않으면 0부터 시작하는 인덱스가 자동으로 부여됩니다.
- 시리즈는 인덱스와 값 쌍으로 구성되기 때문에 dict 타입과 유사합니다.
- 파이썬이 제공하는 dict 타입을 이용해서 인덱스를 강제로 부여하는 방법으로 시리즈를 만들 수도 있습니다.

내레이션

◈ 인덱스 구조

- 인덱스는 자기와 짝을 이루는 데이터의 값과 주소를 저장합니다.
- 인덱스는 데이터 값의 검색이나 정렬, 선택, 결합 등을 할때 쉽게 할 수 있도록 도와주는 구조입니다
- 데이터를 빠르게 식별하도록 도와주는 구조입니다 .

index	value
0	10
1	20
2	30
3	40

Series 구조의 데이터에 접근하기

변수명[인덱스] = 값

내레이션

◈ Series 예제

파일명 : exam9_1.py

```
import pandas as pd
#pandas 라이브러리를 사용하기 위해 import 를 한다. i
#import 시 pd 라는  별도의 이름을  부여하였다.

#파이썬의 list를 사용하여 Series 만들기
# 자동으로 인덱스가 부여된다
data = [10,20,30,40,50]
series = pd.Series(data)
print(type(series))
print(series)


#직접 index 를 부여해보기 (dict타입 사용)
data2 = {'a':1, 'b':2, 'c':3, 'd':4, 'e':5}
series2 = pd.Series(data2)
print(series2)
```

```
<class 'pandas.core.series.Series'>
0    10
1    20
2    30
3    40
4    50
dtype: int64
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

내레이션

-

◈ DataFrame

- pandas가 제공하는 2차원 구조의 타입입니다
- DataFrame 은 데이터를 행과 열로 이루어진 테이블의 형태로 저장합니다.
- pandas 의 DataFrame은 R이라는 통계언어에서 비롯되었습니다. 다른 언어의 기능이 파이썬으로 구현되어서 R언어보다는 기능이 좀 떨어지지만, 데이터를 다루기 충분한 요소들을 가지고 있습니다.
- DataFrame은 열과 행으로 구성되는데 열은 각자 다른 타입으로 구성될 수 있습니다.
- 데이터를 저장할때, 전통적인 방법이 테이블 형태로 데이터를 저장하는 것입니다. 관계형 데이터베이스라던가 Excel 등의 자료 그리고 csv형식의 자료들이 테이블의 구조형태입니다.
- DataFrame은 이러한 유형의 데이터를 다루는데 최적화된 도구입니다.

내레이션

-

◈ DataFrame 예제

파일명 : exam9_2.py

```
import pandas as pd

data = {
        'name':['홍길동', '임꺽정', '장길산', '홍경래'],
        'kor':[90, 80, 70, 70],
        'eng':[99, 98, 97, 46],
        'mat':[90, 70, 70, 60],
}

df = pd.DataFrame(data)
print("타입 : ", type(df))
print(df)
```

dict 타입의 데이터를 정의한다
dict타입은 키와 값 쌍으로 이루어진다.
특정키값에 데이터를 여러개 넣을 경우에 키:리스트 형태로 저장할 수 있다

```
타입 :  <class 'pandas.core.frame.DataFrame'>
  name  kor  eng  mat
0  홍길동   90   99   90
1  임꺽정   80   98   70
2  장길산   70   97   70
3  홍경래   70   46   60
```

내레이션

◈ Panel

- Panel은 3차원 자료구조이다
- Panel은 Axis 0 (items), Axis 1 (major_axis), Axis 2 (minor_axis) 등 3개의 축을 가지고 있다
- Axis 0은 그 한 요소가 2차원의 DataFrame 에 해당된다.
- Axis 1은 DataFrame의 행(row)에 해당되고, Axis 2는 DataFrame의 열(column)에 해당된다.
- 파이썬의 3차원배열을 이용해 Panel 객체를 만들 수 있습니다

내레이션

◈ Panel 예

파일명 : exam9_3.py

```
import pandas as pd

#3차원 배열을 만든다.  – list of list o flist  [ [ [ ], [], [ ]. ].

array = [ [[1,2,3], [4,5,6]],
          [[7,8,9], [10,11,12]],
          [[13,14,15], [16,17,18]]]

array= pd.Panel(array)
print("타입 : ", type(array))
print(array)
```

실행하면 경고 메시지가 나옵니다.
앞으로는 Panel 함수가 사라진다는
의미입니다

```
타입 :  <class 'pandas.core.panel.Panel'>
<class 'pandas.core.panel.Panel'>
Dimensions: 3 (items) x 2 (major_axis) x 3 (minor_axis)
Items axis: 0 to 2
Major_axis axis: 0 to 1
Minor_axis axis: 0 to 2
```

내레이션

◈ Series 데이터 엑세스 방법

- Series 타입의 데이터를 접근 하는 방법은 인덱스를 활용합니다. 각각의 요소에 대한 접근 방법은 변수명[인덱스] 입니다
  - 예)
  - print(mydata1[0]) 값을 읽어서 출력하기
  - mydata1[0]=100 해당 인덱스의 값을 변경하기

- 슬라이싱을 이용해 출력 할 수 도 있습니다
  - 예)
  - print(mydata1[0: 3]) 0번방부터 3번방 이전까지 마지막 방에 있는 데이터는 제외됩니다.
  - print(mydata1[:3]) 0번방부터 3번방 이전까지 출력됩니다
  - print(mydata1[3:]) 3번방 이후로 마지막까지 출력됩니다.

- 인덱스가 숫자가 아니고 lable이 있을 경우 label 을 이용해 접근할 수 있습니다
  - 예)
  - print (mydata2['one'])

내레이션

◈ Series 데이터 엑세스

파일명 : exam9_4.py

```
import pandas as pd
data = [10,20,30,40,50,60]
s = pd.Series(data)
print(s[0]) #0번째 해당 데이터 출력하기
s[1]=200 #1번째 데이터 200으로 수정
print(s)
print(s[:5]) #5번째 데이터부터 마지막까지 출력
print(s[2:5]) #2번데이터부터 4번 데이터까지 출력
print(s[3:]) #3번 이후로 출력

data = {'one':'일', 'two':'이', 'three':'삼', 'four':'사', 'five':'오'}
series = pd.Series(data)
print(series)
print(series['one'])
print(series[0:3])
print(series['one':'three'])
print(series['two':])
```

```
one      일
two      이
three    삼
four     사
five     오
dtype: object
```

내레이션

◈ DataFrame 데이터 엑세스 방법

- DataFrame은 pandas 라이브러리에서 가장 빈번하게 사용되는 데이터 타입입니다.
- 행과 열로 구성되어 있고, iloc, loc 라는 두개의 함수를 이용해 데이터에 접근할 수 있습니다.
- iloc 함수는 행과 열의 인덱스를 이용해 접근합니다. loc 함수는 행의 인덱스와 열의 컬럼명(필드명)을 통해 데이터를 접근합니다.

예) 데이터가 다음과 같을 때

index	name	kor	eng
0	홍길동	90	85
1	장길산	80	75
2	임꺽정	70	65
3	홍경래	60	55

print(mydata.iloc[0, 0])
print(mydata.loc[0, 'name'])

print(mydata.iloc[1, 2])
print(mydata.loc[1, 'kor'])

내레이션

◈ DataFrame 데이터 엑세스 예제

파일명 : exam9_5.py

```
import pandas as pd
data = {
 ' name ' :[ ' 홍길동 ' ,  ' 임꺽정 ' ,  ' 장길산 ' ,  ' 홍경래 ' ,  ' 이상민 ' ,  ' 김수경 ' ],
 ' kor ' :[90, 80, 70, 70, 60, 70],
 ' eng ' :[99, 98, 97, 46, 77, 56],
 ' mat ' :[90, 70, 70, 60, 88, 99],
}
df = pd.DataFrame(data)

print( df.head() ) #앞의 다섯명에 대한 데이터만 나온다.

print( " 지정한 열만 출력 " )
print( df[ ' name ' ])
print( df[ ' kor ' ])
print( df.columns) #컬럼이름만 출력
.....이어서
```

내레이션

◈ DataFrame 데이터 엑세스 예제

파일명 : exam9_5.py

```
#iloc 함수 : 배열에서의 위치값으로 데이터를 접근 할 수 있다.
print("iloc 함수 사용 ----------")
print( df.iloc[0,0]) #0,0번에 해당하는 데이터 출력하기
print( df.iloc[3,2]) #3번째 행의 2번째 열
print( df.iloc[2:4,2]) #2번행의 2번열, 3번행의 2번열

print( df.iloc[2:4,2:4]) #2번행의 2,3번열, 3번행의 2,3번열

print("loc함수 사용 ----------")
#loc 함수는 필드명으로 데이터를 출력할 수 있다 .
print( df.loc[0, 'name']) #0번째 행의 name필드 값 출력
print( df.loc[3, 'eng']) #0번째 행의 name필드 값 출력
print( df.loc[:, 'name':'eng']) #슬라이싱 모든행의 name열부터 kor열까지 출력하라
```

내레이션

◈ Panel 데이터 접근방법

- Panel은 3개의 축을 가지고 있다
- , Panel 의 데이터 구조를 Dict 타입으로 다음처럼 나타낼 수 있다

```
data = {
        'class1':{
                'name': ['박동석', '권다윤', '김민제'],
                'height':[173, 177, 183]
        },
        'class2':{
                'name': ['김민수', '이용희', '임미영'],
                'height':[179, 167, 180]
        }
}

panel = pd.Panel( data )
class1 = panel['class1']
print( class1.iloc[0,0] )
```

class2

class1

index	name	height
0	박동석	173
1	권다윤	177
2	김민제	183

내레이션

◈ Panel 예제

파일명 : exam9_6.py

```
import pandas as pd

data = {
        'class1':{
                'name': ['박동석', '권다윤', '김민제'],
                'height':[173, 177, 183]
        },
        'class2':{
                'name': ['김민수', '이용희', '임미영'],
                'height':[179, 167, 180]
        }
}

panel = pd.Panel(data)
print(panel)
```

…. 이어서

내레이션

◈ Panel 예제

파일명 : exam9_6.py

```
df = panel['class1']
print( df.iloc[0,0])
print( df.columns)

print("--- for ---")
for key in panel:
    item = panel[key]
    for col in item.columns:
        for row in item[col]:
            print(row, end=' ')
        print()
```

```
Dimensions: 2 (items) x 3 (major_axis) x 2 (minor_axis)
Items axis: class1 to class2
Major_axis axis: 0 to 2
Minor_axis axis: name to height
  name height
0  박동석    173
1  권다윤    177
2  김민제    183
  name height
0  김민수    179
1  이용희    167
2  임미영    180
박동석
Index(['name', 'height'], dtype='object')
--- for ---
박동석 권다윤 김민제
173 177 183
김민수 이용희 임미영
179 167 180
```

내레이션

간지 부분

◈ 판다스란

1. 판다스 자료구조
2. 판다스를 이용한 연산

내레이션

(기입하지 않습니다.)

◈ Series 연산

- pandas 객체의 산술연산은 3단계의 프로세스를 거친다

  - 1. 행, 열 인덱스를 기준으로 모든 원소를 정렬한다
  - 2. 동일한 위치에 있는 원소끼리 일대일로 대응시킨다
  - 3. 일대일로 대응되어 있는 원소끼리 연산을 처리한다.
  - (단, 대응되는 원소가 없을 경우 NaN으로 처리한다

- pandas 는 연산시 벡터 연산을 수행한다.
- Series 객체에 어떤 숫자를 더하거나 빼는 등의 연산을 진행하면 모든 원소에 대응된다.

내레이션

◈ Series 연산 1

파일염 : exam9_7.py

```
import pandas as pd

data1 = {'kor':90, 'eng':70, 'mat':80}
data2 = {'kor':90, 'eng':70, 'mat':80}
data3 = {'kor':90, 'eng':70, 'mat':80}
data4 = {'eng':90, 'mat':70, 'kor':80}          #인덱스의 순서가 바뀌어도 정렬을 진행하기때문에 알아서 대응된다.
series1 = pd.Series( data1 )
series2 = pd.Series( data2 )
series3 = pd.Series( data3 )
series4 = pd.Series( data4 )

result1 = series1 + series2 + series3 + series4
result2 = result1/4

print("총점 ----------")
print( result1 )
print("평균 ----------")
print( result2 )
```

```
총점 ----------
eng    300
kor    350
mat    310
dtype: int64
평균 ----------
eng    75.0
kor    87.5
mat    77.5
dtype: float64
```

내레이션

◈ Series 연산 2

파일명 : exam9_8.py

```
import pandas as pd


#인덱스가 없을 경우 값이 NaN으로 들어가서 연산이 되지 않는다

data1 = {'kor':90, 'mat':80}
data2 = {'kor':90, 'eng':70}
data3 = {'kor':90, 'eng':70, 'mat':80}

series1 = pd.Series( data1 )
series2 = pd.Series( data2 )
series3 = pd.Series( data3 )
result1 = series1 + series2 + series3

print(result1)
```

```
eng      NaN
kor    270.0
mat      NaN
dtype: float64
```

내레이션

◈ Series 연산 3

파일명 : exam9_9.py

```
import pandas as pd

#인덱스가 없을 경우 값이 NaN으로 들어가서 연산이 되지 않는다
#add 연산시 fill_value=0 옵션을 주면 대응되는 셀에 값이 없을 경우 fill_value에서
#지정한 값으로 대신해준다


data1 = {'kor':90, 'mat':80}
data2 = {'kor':90, 'eng':70}
data3 = {'kor':90, 'eng':70, 'mat':80}

series1 = pd.Series( data1 )
series2 = pd.Series( data2 )
series3 = pd.Series( data3 )
result1 = series1.add(series2, fill_value=0).add(series3,fill_value=0)

print(result1)
```

```
eng     140.0
kor     270.0
mat     160.0
dtype: float64
```

내레이션

◈ DataFrame 연산

- DataFrame은 여러개의 Series의 결합으로 볼 수 있습니다. 그래서 연산 방법도 Series의 내용이 적용됩니다.
- 산술연산이 모두 가능합니다.

  - 1. 행, 열 인덱스를 기준으로 모든 원소를 정렬한다
  - 2. 동일한 위치에 있는 원소끼리 일대일로 대응시킨다
  - 3. 일대일로 대응되어 있는 원소끼리 연산을 처리한다.
  - (단, 대응되는 원소가 없을 경우 NaN으로 처리한다

- 새로운 행을 추가하거나 삭제가 가능합니다.
- 주의사항은 새로운 행을 추가하거나 삭제시 결과 값을 반환받아야 합니다. 결과값을 반환받지 않으면 추가되거나 삭제된 행은 원래대로 보입니다
- 데이터 프레임에 새로운 필드를 손쉽게 붙일 수 있습니다

내레이션

◈ Dataframe 연산

파일명 : exam9_9.py

```
#시리즈를 결합하여  DataFrame 을 만듭니다
import pandas as pd

data1 = { ' kor ' :90,  ' eng ' :70,  ' mat ' :80}
data2 = { ' kor ' :90,  ' eng ' :70,  ' mat ' :80}
data3 = { ' kor ' :90,  ' eng ' :70,  ' mat ' :80}

data = pd.DataFrame()
data = data.append( data1 , ignore_index=True)
data = data.append( data2 , ignore_index=True)
data = data.append( data3 , ignore_index=True)

#각 필드값들을 더하여 새로 작성한 필드에 저장함 이미 있던 필드의 경우
#data['kor'] 대신에 data.kor로 접근  가능합니다.

data['total'] = data.kor + data.eng + data.mat
data['avg'] = data.total/3

print(data)
```

내레이션

◈ DataFrame 데이터 엑세스 예제

파일명 : exam9_11.py
dict 타입을 데이터 프레임으로 전환후 새로운 필드를 추가하기

파일명 : exam9_6.py

```
import pandas as pd
data = {
        'name':['홍길동', '임꺽정', '장길산', '홍경래', '이상민', '김수경'],
        'kor':[90, 80, 70, 70, 60, 70],
        'eng':[99, 98, 97, 46, 77, 56],
        'mat':[90, 70, 70, 60, 88, 99],
}
df = pd.DataFrame(data)

#새로운 필드 추가하기
df['total'] = df.kor + df.eng + df.mat
df['avg'] = df.total/3

print(df)
```

내레이션

◈ DataFrame 연산예제

파일명 : exam9_12.py

```
import pandas as pd
data = {
'fruits':['망고', '딸기', '수박', '파인애플'],
'price':[2500, 5000,10000, 7000],
'count':[5, 2, 2, 4],
}
df = pd.DataFrame(data)

#새로운 행 추가하기 , 주의 : 반드시 반환되는 값을 받아야 추가가 된다
#append 함수는 데이터가 추가된 객체를 반환한다 .
df.append({'fruits':'사과', 'price':3500, 'count':10} , ignore_index=True)
print(df)

#ignore_index=True 옵션 : 기존의 인덱스를 무시하라는 옵션인데 붙여야 한다
print("데이터가 추가된 객체를 받을 경우 ---------")
df = df.append({'fruits':'사과', 'price':'3500', 'count':10}, ignore_index=True )
print(df)
```

내레이션

◈ DataFrame

```
#특정 필드 삭제 : axis는 축을 말하는데 axis값이 0이면 가로, axis 값이 1이면 세로를 말함
df2 = df.drop("price", axis=1) #필드가 하나 삭제된 객체를 반환
print(df2)
print("원본")
print(df)

#특정행을 삭제
df3 = df.drop(0, axis=0) #행의 인덱스를 이용해서 삭제한다
print(df3)
```

-

```
  fruits  price  count
0     망고   2500      5
1     딸기   5000      2
2     수박  10000      2
3   파인애플   7000      4
데이터가 추가된 객체를 받을 경우 -----
  fruits  price  count
0     망고   2500      5
1     딸기   5000      2
2     수박  10000      2
3   파인애플   7000      4
4     사과   3500     10
  fruits  count
0     망고      5
1     딸기      2
2     수박      2
3   파인애플      4
4     사과     10
원본
  fruits  price  count
0     망고   2500      5
1     딸기   5000      2
2     수박  10000      2
3   파인애플   7000      4
4     사과   3500     10
  fruits  price  count
1     딸기   5000      2
2     수박  10000      2
3   파인애플   7000      4
4     사과   3500     10
```

내레이션

◈ 적용하기

다음 테이블 데이터를 데이터프레임 객체로 만들어 봅시다.

X1	X2	X3	X4
2.9	9.2	13.2	2
2.4	8.7	11.5	3
2	7.2	10.8	4
2.3	8.5	12.3	2
3.2	9.6	12.6	3

위 테이블에 마지막 요소로 다음 데이터를 추가하세요

(10, 20, 30, 40)

위 테이블에 total 필드를 만들고 각 필드들의 합을 구하여 total 필드에 적용하세요

내레이션

(기입하지 않습니다.)

◈ 적용하기 -> 풀이

파일명 : homework9.py

```
import pandas as pd
data = {
   'X1':[2.9, 2.4, 2, 2.3, 3.2],
   'X2':[9.2, 8.7, 7.2, 8.5, 9.6],
   'X3':[13.2, 11.5, 10.8, 12.3, 12.6],
   'X4':[2, 3, 4, 3, 2]
}
df = pd.DataFrame(data)
print(df)
df = df.append( {'X1':10, 'X2':20, 'X3':30, 'X4':40}, ignore_index=True)

df['total'] = df.X1 + df.X2 + df.X3 + df.X4
print(df)
```

dict 타입을 이용해 데이터를 작성 후 DataFrame을 적용한다

추가 데이터는 Series 객체를 만들어야 한다.
ignore_index=True 속성을 주어야 한다

내레이션

(기입하지 않습니다.)

◈ 문제풀기

문제 1) 다음은 Pandas library 에 대한 설명글이다. 옳지 않을 것을 고르시오.

① Pandas는 pip 나 conda 로 설치해야 하는 파이썬의 추가 API 이다.
② Pandas 는 통계 패키지에 많이 사용하는 데이터 타입이다.
③ Anaconda 사용시에도 별도의 설치를 하여야 한다
④ Pandas library 는 Series, DataFrame, Panel 등의 데이터 셋을 지원한다

정답) 3
해설) Anaconda 패키지는 이미 설치되어 있습니다
난이도) 4(1:아주어려움, 2:어려움 3: 보통 4: 쉬움 5: 아주 쉬움)
관련페이지번호) 7

내레이션

(기입하지 않습니다.)

번호	다음에 제시된 파서중 추가로 설치하지 않아도 사용 가능한 기본 파서는?	정답	난이도	해설	관련학습보기
2	**다음과 같이 Serirs 객체가 있을 경우 데이터 출력을 하려면 어떻게 해야야 하나?** **series = pd.Series(['사과', '망고', '바나나', '수박', '딸기 '])** **print ( )** **결과 : 바나나 수박 딸기**	series[2:	4	Series 객체도 슬라이싱이 적용됩니니다 2,3,4 세개의 데이터가 출력되어야 하므로 2번방부터 마지막까지 적용됩니다.	32
3	**다음과 같이 Serirs 객체가 있을 경우 데이터 출력을 하려면 어떻게 해야야 하나?** **series2 = pd.Series({'red':'빨간색', 'green':'초록색', 'blue':'파란색', 'black':'검은색'})** **print ( )** **빨간색, 초록색, 파란색**	series2['red':'blue']	4	인덱스가 문자열이므로 문자열로 주면 된다. series2['red':'blue']	29
4	**data = {** **'name':['Tom', 'Brown', 'Jane', 'Stepany', 'John'],** **'month1':[1200, 2000, 3000, 1800, 4500],** **'month2':[2400, 2300, 4300, 2800, 4200],** **'month3':[1800, 2700, 3500, 3800, 4300],** **}** **data = pd.DataFrame(data)** **위의 데이터는 ¼ 분이 영업자들의 실적이다. 이 실적을 새로운 필드를 추가하여 평균을 구하는 코드를 작성하시오**	정답은 해설참조	4	**data['quater1'] = (data['month1'] + data['month2'] + data['month3'])/3** 또는 **data['quater1'] = (data.month1+ data.month2+ data.month3)/3**	6
5	위 6번의 데이터에 새로운 사람의 자료를 추가하는 코드를 작성하세요 name='Bread', month1=3400, month2 = 4000, month3=5000	정답은 해설참조	5	**data = data.append({'name':'Bread', 'month1':3400, 'month2':4000, 'month3':5000}, ignore_index=True)**	32

번호	문제	정답	난이도	해설	관련학습보기
6	위 4번의 데이터중에서 Stepany 의 데이터중 3월 데이터를 보고 싶다면 어떻게 해야 하는지 코드를 작성하시오(iloc 함수 사용하기)	정답은 해설참조	4	**data.iloc[3, 3]**	31
7	위 4번의 데이터중에서 john 의 데이터중 2월 데이터를 보고 싶다면 어떻게 해야 하는지 코드를 작성하시오(loc 함수 사용하기)	정답은 해설참조	4	**data.loc[4, 'month2']**	33
8	**4번의 데이터에서 , month3 필드를 삭제하는 코드를 작성하시오**	정답은 해설참조	4	**data = data.drop('month2', axis=1)**	28
9	**4번의 데이터에서 jane 의 데이터를 삭제하는 코드를 작성하시오**		5	**data = data.drop(2, axis=0)**	34
10	**다음 설명중 잘못 된것은?.** **① Series 객체는 list를 통해서만 만들 수 있다** **② Series 객체는 dict 타입을 통해 직접 index 를 부여할 수 있다** **③ DataFrame은 행과 열로 구성된 테이블 형태 구조체이다** **④ Panel 은 3차원 데이터 구조이다**	1	2	list, dict 타입 둘다 가능합니다	30

과정명		모듈명		회차명	

◈ 핵심요약

▶ Series

- Series 타입은 판다스에서 1차원 데이터를 지원하기 위해 만든 데이터 타입입니다.
- 데이터를 접근 하는 방법은 인덱스를 활용합니다, 인덱스에 조건을 부여할 수 도 있습니다. data[조건식] 조건식을 만족하는 결과만 출력
- 인덱스는 자동으로 부여됩니다.
- 사용자가 dict 타입을 이용해 별도의 인덱스를 부여할 수 도 있습니다.

▶ DataFrame

- DataFrame 은 판다스에서 2차원 데이터를 지원하기 위해 만든 데이터 타입입니다
- 테이블의 구조이고 행과 열이 있습니다 .가장 많이 사용하는 데이터 타입입니다.
- 각 열과 행은 하나의 Series 타입으로 볼 수 도 있습니다.
- iloc함수는 열번호와 행번호를 이용해 데이터프레임 요소에 접근할 수 있고 , loc 함수는 필드에 colum명이나 행에 row명이 붙었을땡 직접 행이나 열명으로 접근이 가능합나다.
- csv파일이나 excel 파일을 다루기 용이합니다.

▶ Panel

- DataFrame 은 판다스에서 3차원 데이터를 지원하기 위해 만든 데이터 타입입니다

내레이션

(기입하지 않습니다.)