|  |  |
| --- | --- |
| 교육 제목 | python 7 |
| 교육 일시 | 2021. 09. 29 |
| 교육 장소 | 영우글로벌러닝 |
| **교육 내용** | |
| 오전 | import pandas as pd  exam\_data = {'이름' : ['서준','우현','인아'],  '수학' : [90, 89, 95],  '영어' : [98, 89, 95],  '음악' : [85, 95, 100],  '체육' : [100, 90, 90]}  df=pd.DataFrame(exam\_data)  # '이름' 열을 새로운 인덱스로 지정하고, df 객체에 변경사항 반영  df.set\_index('이름', inplace=True)  print(df)  ----------------------------------------------------------------------------  # 서준의 수학 점수를 100점으로 변경  df.loc['서준','수학']=100  # 우현과 인아의 수학과 영어점수를 출력하세요 (iloc와 범위, loc와 각각의 자료 배열로 사용)  print(df.iloc[1:, 1:3])  #df.loc['인아',['수학','영어']])  print(df.loc['인아',['수학','영어']])  print("df.iloc : ", df.iloc[[1,2],[0,1]])  print("df.iloc : range ", df.iloc[1:3, 0:1])  print("df.ilod : ", df.loc[['우현','인아'],['수학','영어']])  print("df.loc : range ", df.loc['우현':'인아', '수학':'영어'])  # 인아의 모든 점수를 100으로 수정하세요  df.loc['인아']=100  print(df)  # 서준의 모든 점수 출력  df.loc['서준']  # 수학과 음악 점수만 출력  df[['수학','음악']]  ----------------------------------------------------------------------------------  print(df)  print()  print(df.T)  print()  a=df.T  a  ---------------------------------------------------------------------------------  # reindex : index를 재배치  import pandas as pd  # 딕셔서리를 정의  dict\_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9], 'c3':[10,11,12], 'c4':[13,14,15]}  # 딕셔서리를 데이터프레임으로 변환. 인덱스를 [r0, r1, r2]로 지정  df = pd.DataFrame(dict\_data, index=['r0', 'r1', 'r2'])  print(df)  print('\n')  ---------------------------------------------------------------------------------  # 인덱스를 [r0, r1, r2, r3, r4]로 재지정  new\_index=['r0','r1','r2','r3','r4'] # 추가되는 데이터는 NaN 로 입력됨  ndf=df.reindex(new\_index)  #ndf=df.reindex(new\_index, fill\_value=0) # 추가되는 데이터에서 초기값 설정  print(ndf)  print()  ---------------------------------------------------------------------------------  # index 초기화 : df.reset\_index  print(df.reset\_index())  df=pd.DataFrame(dict\_data, index=['r0','r1','r2'])  print(df)  # sort : 인덱스로 정렬, 특정 컬럼의 값으로 정렬  df.sort\_index(ascending=False) # index로 정렬 ascending=False : 역으로 정렬  df.sort\_values(by=['c0','c1'], ascending=False)  -----------------------------------------------------------------------------------  import pandas as pd  # 딕셔너리 데이터로 판다스 시리즈 만들기  student1=pd.Series({'국어':100, '영어':80, '수학':90, '역사':70})  student2=pd.Series({'국어':90, '영어':85, '수학':95, '과학':100})  print(student1)  print('\n')  print(student1/200, '\n')  print(student1+student2) # series 연산자(+, -, \*, /) series 또는 숫자  print(student1/student2)  print(student1\*student2)  # 내부적으로 인덱스로 정렬한 후 각가의 인덱스로 계산함  # 연산식에 NaN이 포함되면 결과는 NaN로 나타남  print(student1.add(student2, fill\_value=0)) # fill\_value=0 NaN 값을 0으로 채움  # add, sub, mul, div (자료가 없으면 원하는 값으로 대체)  -------------------------------------------------------------------------------------  # dataframe 연산, 기존의 모듈에서 dataset을 불러들임  import pandas as pd  import seaborn as sns  titanic=sns.load\_dataset('titanic') # 모듈에서 제공되는 dataset을 가져오는 매소드  type(titanic)  # df=titanic.head()  df=titanic.loc[ : , ['age','fare']] # 'age'와 'fare' 컬럼의 정보만 가져옴  df=df.iloc[100:201] # 100~200 인덱스 정보만 가져옴  print(df.head(5))  df.add(10, fill\_value=0)  df1=df.head(10)  df2=df.tail(10)  df1.add(df2, fill\_value=0)  -----------------------------------------------------------------------------------  # int(df.age.sum()/df.age.count())  # 평균나이보다 적은 연령의 자료만 출력  avg\_age=int(df.age.sum()/df.age.count())  df.add(0, fill\_value=0)  for idx, age in enumerate(df.age): # enumerate() index와 데이터를 return  if age<avg\_age:  print("{} : {}".format(df.iloc[idx,0], df.iloc[idx,1])) # 원소 검색 : df.iloc 또는 df.loc  #print(df.iloc[idx])  -----------------------------------------------  df\_copy=[]  for idx, item in enumerate(df.age):  if item < avg\_age:  df\_copy.append(df.iloc[idx])  print("{} : {}".format(df.iloc[idx,0], df.iloc[idx,1]))    print()  df\_copy=pd.DataFrame(df\_copy)  df\_copy |
| 오후 | import pandas as pd  file\_path="./read\_csv.csv"  df=pd.read\_csv(file\_path) # read\_csv 컬럼 데이터가 ','로 구분된 자료를 읽어옴  print(df)  df1=pd.read\_csv(file\_path,header=None)  print("df1 == ")  print(df1)  df2=pd.read\_csv(file\_path,index\_col='c0')  print("df2 == ")  print(df2)  ----------------------------------------------------------------------------------  read\_csv(옵션...)  - path : 파일의 위치 포함한 파일명  - sep : 필드를 구분하는 구분 문자 ','  - header : 헤더가 정의되어 있는지 None  - index\_col : 인덱스로 사용될 컬럼명, None 인덱스 없음  - names : 컬럼 이름으로 사용 될 문자열 리스트  - skiprows : 처음 행 부터 skip 하고자 하는 행 수  - skipfooter : 마지막 행 부터 skip 하고자 하는 행 수  - encoding : 텍스트 인코딩 종류를 지정 'ulf\_8'  -------------------------------------------------------------------------------------  save\_file\_path="./sample\_csv.csv"  df.to\_csv(save\_file\_path)  save\_file\_path="./sample\_csv1.csv"  df.to\_csv(save\_file\_path,index=None) # df의 인덱스는 저장하지 않음  # titanic 데이터를 load해서 age가 평균 age보다 적은  # 데이터를 가져와서 save\_titanic.csv 파일에 저장 (인덱스는 빼고)  # titanic.dtypes, titanic.columns  ----------------------------------  import seaborn as sns  titanic=sns.load\_dataset('titanic')  df\_copy=[] # 리스트  avg\_age=titanic.age.sum()/titanic.age.count()  for idx, age in enumerate(titanic.age):  if age<avg\_age:  df\_copy.append(titanic.iloc[idx]) # 이차원 배열의 리스트    pd.DataFrame(df\_copy).to\_csv("./titanic\_save\_file1.csv", index=None)  print(type(df\_copy), " : ", type(pd.DataFrame(df\_copy)))  --------------------------------------------  data\_types=titanic.dtypes  column\_names=titanic.columns  columns=[] # 컬럼명을 리스트로  for idx, dtype in enumerate(data\_types):  if dtype in ['float64','int64']:  columns.append(column\_names[idx]) # 데이터 타입이 float64 또는 int64    titanic\_select=titanic.loc[ : , columns]  titanic\_select.to\_csv("titanic\_save\_file.csv", index=None)  titanic.dtypes  ------------------------------------------------------------------------------------  # import os  # print(os.getcwd()) # 작업하고 있는 경로 확인  df=pd.read\_json("./read\_json\_sample.json") # 상대경로  # df=pd.read\_json("C:/Users/sktu2/Desktop/python\_src/read\_json\_sample.json") # 절대경로  print(df.index)  type(df)  -------------------------------------------------------------------------------------  url="./sample.html"  tables=pd.read\_html(url)  print(tables)  len(tables)  for i in range(len(tables)):  print("tables {}".format(i))  print(tables[i])  print("\n")  df=tables[1]  df.set\_index(['name'], inplace=True) # 'name' 컬럼을 인덱스로 setting  df  df1=tables[0]  df1.set\_index(['c0'], inplace=True)  df1  ---------------------------------------------------------------------------------  !pip install googlemaps  ------------------------------  from bs4 import BeautifulSoup  import requests  import re  import pandas as pd  # 위키피디아 미국 ETF 웹 페이지에서 필요한 정보를 스크래핑하여 딕셔너리 형태로 변수 etfs에 저장  url = "<https://en.wikipedia.org/wiki/List_of_American_exchange-traded_funds>"  resp = requests.get(url)  soup = BeautifulSoup(resp.text, 'lxml')  rows = soup.select('div > ul > li'    etfs = {}  for row in rows:    try:  etf\_name = re.findall('^(.\*) \(NYSE', row.text)  etf\_market = re.findall('\((.\*)\|', row.text)  etf\_ticker = re.findall('NYSE Arca\|(.\*)\)', row.text)    if (len(etf\_ticker) > 0) & (len(etf\_market) > 0) & (len(etf\_name) > 0):  etfs[etf\_ticker[0]] = [etf\_market[0], etf\_name[0]]  except AttributeError as err:  pass  # etfs 딕셔너리 출력  print(etfs)  print('\n')  ---------------------------------------------  ## google 지오코딩 API 통해 위도, 경도 데이터 가져오기  # 라이브러리 가져오기  import googlemaps  import pandas as pd  # my\_key = "----발급받은 API 키를 입력-----"  # 구글맵스 객체 생성하기  maps = googlemaps.Client(key='AIzaSyBhMr1XeliRMaUGATRvlDYDl2A0jcH4b44') # my key값 입력  lat = [] #위도  lng = [] #경도  # 장소(또는 주소) 리스트  places = ["서울시청", "국립국악원", "해운대해수욕장"]  i=0  for place in places:  i = i + 1  try:  print(i, place)  # 지오코딩 API 결과값 호출하여 geo\_location 변수에 저장  geo\_location = maps.geocode(place)[0].get('geometry')  lat.append(geo\_location['location']['lat'])  lng.append(geo\_location['location']['lng'])  except:  lat.append('')  lng.append('')  print(i)  # 데이터프레임으로 변환하기  df = pd.DataFrame({'위도':lat, '경도':lng}, index=places)  print('\n')  print(df)  ---------------------------------------------------------------------------------  파일을 처리하는 방법  읽어오기  - read\_csv, read\_html, read\_excel, read\_json  저장하기  - df.to\_csv(파일명), df.to\_html, df.to\_excel, df.to\_json  json 파일은 구조가 DataFrame과 동일  csv 파일은 2차원 배열로 -> DataFrame으로 변경해서 사용  df=pandas.DataFrame ..  행 df.loc[인덱스명], df.iloc[인덱스첨자] 여러개 start:end, [ , ~~ , ]  열 df.열명 df[열명]  원소 df.loc[[인덱스명],[컬럼명]]  df.iloc[[인덱스첨자],[컬럼의 순서 정수]]  ----------------------------------------------------------------------------------  # seaborn 에서 dataset "iris"를 불러와서  import seaborn as sns  import pandas as pd  iris\_df=sns.load\_dataset('iris')  iris\_df  -----------------------------------------------  # 1. 'species' 컬럼을 인덱스로 설정  iris\_df.set\_index(['species'], inplace=True)  # 2. 나머지 데이터의 합과 평균을 데이터 프레임에 추가  total=[]  avg=[]  for idx in range(len(iris\_df)):  total.append(iris\_df.iloc[idx].sum())  avg.append(iris\_df.iloc[idx].sum()/len(iris\_df.iloc[idx]))  iris\_df['total']=total  iris\_df['avg']=avg  # 3. 변경된 데이터를 파일에 csv 형식으로 저장  iris\_df.to\_csv("./iris\_sample.csv")  # 4. 저장된 파일을 프로그램으로 불러 옴  iris2=pd.read\_csv("./iris\_sample.csv")  # 5. 불러온 데이터를 출력해서 확인  iris2 |