|  |  |
| --- | --- |
| 교육 제목 | python 6 |
| 교육 일시 | 2021. 09. 28 |
| 교육 장소 | 영우글로벌러닝 |
| **교육 내용** | |
| 오전 | # 학생 리스트를 선언합니다.  students = [  { "name": "윤인성", "korean": 87, "math": 98, "english": 88, "science": 95 },  { "name": "연하진", "korean": 92, "math": 98, "english": 96, "science": 98 },  { "name": "구지연", "korean": 76, "math": 96, "english": 94, "science": 90 },  { "name": "나선주", "korean": 98, "math": 92, "english": 96, "science": 92 },  { "name": "윤아린", "korean": 95, "math": 98, "english": 98, "science": 98 },  { "name": "윤명월", "korean": 64, "math": 88, "english": 92, "science": 92 }  ]  print("이름", "총점", "평균", sep="\t")  for student in students:  # 점수의 총합과 평균을 구합니다.  score\_sum=student["korean"]+student["math"]+student["english"]+student["science"]  score\_average=score\_sum/4  #출력합니다  print(student["name"], score\_sum, score\_average, sep="\t")  ---------------------------------------------------------------------------  # 딕셔너리를 리턴하는 함수를 선언  def create\_student(name, korean, math, english, science):  return {  "name": name,  "korean": korean,  "math": math,  "english": english,  "science": science  }  students = [  create\_student("윤인성", 87, 98, 88, 95),  create\_student("연하진", 92, 98, 96, 98),  create\_student("구지연", 76, 96, 94, 90),  create\_student("나선주", 98, 92, 96, 92),  create\_student("윤아린", 95, 98, 98, 98),  create\_student("윤명월", 64, 88, 92, 92)  ]  # 학생 점수의 합을 구하는 함수  def student\_get\_sum(student):  return student["korean"]+student["math"]+student["english"]+student["science"]  # 평균을 구하는 함수  def student\_get\_avg(student):  return student\_get\_sum(student)/4  # 출력하는 함수  def student\_to\_string(student):  return "{}:\t{}\t{}".format(student["name"],student\_get\_sum(student), student\_get\_avg(student))  # 학생을 한명씩 반복합니다.  print("이름", "총점", "평균", sep="\t")  for student in students:  print(student\_to\_string(student))  # 점수의 총합과 평균을 구합니다.  # score\_sum=student["korean"]+student["math"]+student["english"]+student["science"]  # score\_average=score\_sum/4  #출력합니다  # print(student["name"], score\_sum, score\_average, sep="\t")  --------------------------------------------------------------------------------------  # class로 생성하여 object를 관리  class Student: #class 클래스이름:  def \_\_init\_\_(self, name, korean, math, english, science):  self.name=name  self.korean=korean  self.math=math  self.english=english  self.science=science    # 학생 점수의 합을 구하는 함수  def get\_sum(self):  return self.korean+self.math+self.english+self.science  # 평균을 구하는 함수  def get\_avg(self):  return self.get\_sum()/4  # 출력하는 함수  def to\_string(self):  return "{}:\t{}\t{}".format(self.name, self.get\_sum(), self.get\_avg())    def \_\_str\_\_(self):  return "{}:\t{}\t{}".format(self.name, self.get\_sum(), self.get\_avg())    def \_\_eq\_\_(self, value):  return self.get\_sum() == value.get\_sum()  def \_\_ne\_\_(self, value):  return self.get\_sum() != value.get\_sum()  def \_\_gt\_\_(self, value):  return self.get\_sum() > value.get\_sum()  def \_\_ge\_\_(self, value):  return self.get\_sum() >= value.get\_sum()  def \_\_lt\_\_(self, value):  return self.get\_sum() < value.get\_sum()  def \_\_le\_\_\_\_(self, value):  return self.get\_sum() <= value.get\_sum()  a\_class=Student("윤인성", 87, 98, 88, 95)  # Student 클래스 a\_class 인스턴스, 데이터  print(a\_class.name, a\_class.math, a\_class.english)  students = [  Student("윤인성", 87, 98, 88, 95),  Student("연하진", 92, 98, 96, 98),  Student("구지연", 76, 96, 94, 90),  Student("나선주", 98, 92, 96, 92),  Student("윤아린", 95, 98, 98, 98),  Student("윤명월", 64, 88, 92, 92)  ]  print("studants의 데이터 갯수 : ", len(students))  students[1].name  print("이름", "총점", "평균", sep="\t")  for student in students:  print(student.to\_string())  ----------------------------------------------------------------------------  # isinstance(인스턴스, 클래스명)  class Human:  def \_\_init\_\_(self):  pass    class Student(Human):  def \_\_init\_\_(self):  pass    student=Student()  # instance 확인  print("isinstance(student, Human) : ", isinstance(student, Human))  print("type(student)==Human ", type(student)==Human)  ---------------------------------------------------------------------------  class Student: #class 클래스이름:  count=0    def \_\_init\_\_(self, name, korean, math, english, science):  self.name=name  self.korean=korean  self.math=math  self.english=english  self.science=science    #클래스 변수 설정  Student.count+=1  print("{}번째 학생이 생성되었습니다.".format(Student.count))    # 학생 점수의 합을 구하는 함수  def get\_sum(self):  return self.korean+self.math+self.english+self.science  # 평균을 구하는 함수  def get\_avg(self):  return self.get\_sum()/4  # 출력하는 함수  def to\_string(self):  return "{}:\t{}\t{}".format(self.name, self.get\_sum(), self.get\_avg())    def \_\_str\_\_(self):  return "{}:\t{}\t{}".format(self.name, self.get\_sum(), self.get\_avg())    def \_\_eq\_\_(self, value):  return self.get\_sum() == value.get\_sum()  def \_\_ne\_\_(self, value):  return self.get\_sum() != value.get\_sum()  def \_\_gt\_\_(self, value):  return self.get\_sum() > value.get\_sum()  def \_\_ge\_\_(self, value):  return self.get\_sum() >= value.get\_sum()  def \_\_lt\_\_(self, value):  return self.get\_sum() < value.get\_sum()  def \_\_le\_\_\_\_(self, value):  return self.get\_sum() <= value.get\_sum()  ---------------------------------------------------------------------------------  students = [  Student("윤인성", 87, 98, 88, 95),  Student("연하진", 92, 98, 96, 98),  Student("구지연", 76, 96, 94, 90),  Student("나선주", 98, 92, 96, 92),  Student("윤아린", 95, 98, 98, 98),  Student("윤명월", 64, 88, 92, 92)  ]  print("이름", "총점", "평균", sep="\t")  for student in students:  print(str(student)) #student의 \_\_str\_\_() 함수가 실행 됨    print("students[0]==students[1] = ", students[0] != students[1])  -------------------------------------------------------------------------  # 클래스를 선언합니다.  class Student:  # 클래스 변수  count = 0  students = []  # 클래스 함수  @classmethod  def print(cls):  print("------ 학생 목록 ------")  print("이름\t총점\t평균")  for student in cls.students:  print(str(student))  print("------- ------- -------")  # 인스턴스 함수  def \_\_init\_\_(self, name, korean, math, english, science):  # 인스턴스 변수 초기화  self.name = name  self.korean = korean  self.math = math  self.english = english  self.science = science  Student.count += 1  Student.students.append(self)  def get\_sum(self):  return self.korean + self.math +\  self.english + self.science  def get\_average(self):  return self.get\_sum() / 4  def \_\_str\_\_(self):  return "{}\t{}\t{}".format(\  self.name,\  self.get\_sum(),\  self.get\_average())  #학생 리스트를 선언합니다.  Student("윤인성", 87, 98, 88, 95),  Student("연하진", 92, 98, 96, 98),  Student("구지연", 76, 96, 94, 90),  Student("나선주", 98, 92, 96, 92),  Student("윤아린", 95, 98, 98, 98),  Student("윤명월", 64, 88, 92, 92)  #출력합니다.  Student.print()  ----------------------------------------------------------------------------  # 모듈을 가져옵니다.  import math  # 클래스를 선언합니다.  class Circle:  def \_\_init\_\_(self, radius):  self.\_\_radius = radius  def get\_circumference(self):  return 2 \* math.pi \* self.\_\_radius  def get\_area(self):  return math.pi \* (self.\_\_radius \*\* 2)    def get\_radius(self): # getter  return self.\_\_radius  def set\_radius(self, value): # setter  self.\_\_radius=value  # 원의 둘레와 넓이를 구합니다.  circle = Circle(10)  print("# 원의 둘레와 넓이를 구합니다.")  print("원의 둘레:", circle.get\_circumference())  print("원의 넓이:", circle.get\_area())  print()  # \_\_radius에 접근합니다.  print("# \_\_radius에 접근합니다.")  print(circle.get\_radius())  #print(circle.\_\_radius)  print()  circle.set\_radius(3)  print("# 원의 둘레와 넓이를 구합니다.")  print("원의 둘레:", circle.get\_circumference())  print("원의 넓이:", circle.get\_area())  print()  --------------------------------------------------------------------------  class Circle:  def \_\_init\_\_(self, radius):  self.\_\_radius = radius  def get\_circumference(self):  return 2 \* math.pi \* self.\_\_radius  def get\_area(self):  return math.pi \* (self.\_\_radius \*\* 2)    @property  def radius(self): # getter  return self.\_\_radius  @radius.setter  def radius(self, value): # setter  self.\_\_radius=value  -------------------------------------------------------  # inheritance 상속  # 부모 클래스를 선언합니다.  class Parent:  def \_\_init\_\_(self):  self.value="테스트"  print("Parent 클래스의 \_\_init\_\_ 메소드가 호출되었습니다.")  def test(self):  print("Parent 클래스의 test() 메소드입니다.")    # 자식 클래스를 선언합니다.  class Child(Parent):  def \_\_init\_\_(self):  Parent.\_\_init\_\_(self)  print("Child 클래스의 \_\_init()\_\_ 메소드가 호출되었습니다.")    # 자식 클래스의 인스턴스를 생성하고 부모의 메소드를 호출합니다.  child=Child()  child.test()  print(child.value) |
| 오후 | # pandas를 활용하여 데이터 처리  import pandas as pd  dict\_data={'a':1, 'b':2, 'c':3}  ar=pd.Series(dict\_data)  print(type(dict\_data))  print(type(ar))  print(dict\_data)  print(ar)  print(dict\_data['a'], " : ", ar['a'], " : ", ar[0]) # ar에서 'a' 인덱스명  print(ar[['a','c']])  ar[1:2] # 인덱스 첨자로 접근하면 마지막 첨자 위치 데이터 불포함  ar['b':'c'] # 인덱스명으로 접근하면 마지막 데이터 포함  # 딕셔너리를 시리즈로 변경 : 딕셔너리의 키가 시리즈의 인덱스명이 됨  # 접근은 인덱스명 또는 인덱스 첨자로 접근 가능  --------------------------------------------------------------------------------  dict\_data={'a':[1,2,3], 'b':[4,5,6], 'c':[7,8,9]} # 'a'는 key 값, 데이터프레임의 컬럼명  df=pd.DataFrame(dict\_data)  print(df)  #index명을 부여  df1=pd.DataFrame(dict\_data, index=["idx1", 'idx2', 'idx3'])  print(df1)  a\_var=[[1,2,3,],[4,5,6],[7,8,9]]  df2=pd.DataFrame(a\_var, index=["idx1",'idx2','idx3'], columns=['a1','b1','c1'])  df2  print(df2.rename(index={'idx1': 'ix1', 'idx2':'ix2', 'idx3':'ix3'}, inplace=True)) # df2의 값을 바꿈 inplace=True 를 넣어야 바뀜  df2  ---------------------------------------------------------------------------  # 행인덱스명 변경 : object.index = 새로운 행 인덱스 배열  # 컬럼 인덱스명 변경 : object.columns = 새로운 컬럼명 배열  a\_var=[[1,2,3],[4,5,6],[7,8,9]]  df=pd.DataFrame(a\_var)  df  df.index=['a','b','c']  df.columns=['c1','c2','c3']  df1=df  print("df1 : ", df1)  df1.rename(index={'a':'ida','b':'idb'}, inplace=True)  print("df1 : ")  print(df1)  # 행 삭제 axis=0, 열 삭제 axis=1  df2=df1  print(df1.drop('ida', axis=0, inplace=True)) # axis=0 가로축부터 0로 시작  df1  df2.drop('c1',axis=1)  -------------------------------------------------------------------------------  a\_var=[[1,2,3],[4,5,6],[7,8,9]]  df=pd.DataFrame(a\_var)  df1=df # df의 주소를 복사, df의 값이 변하면 df1도 변함  df2=df[:] # df의 값을 복사, df와 df2가 각기 다른 자료료 처리 가능  df1.drop(0, axis=0, inplace=True)  df1  ---------------------------------------------------------------------  import pandas as pd  # DataFrame() 함수로 데이터프레임 변환, 변수 df에 저장  exam\_data = {'수학' : [90,80,70], '영어' : [98,89,95],  '음악' : [85,95,100], '체육' : [100,90,90]}  df=pd.DataFrame(exam\_data, index=['서준', '우현', '인아'])  print(df)  print('\n')  # 데이터프레임 df를 복제하여 변수 df2에 저장. df2의 1개 행(row) 삭제  #df2=df[:]  df2=df.copy()  df2.drop('우현', inplace=True)  print(df2)  print('\n')  # 데이터프레임 df를 복제하여 변수 df3에 저장. df3의 2개 행(row) 삭제  #df3=df[:]  df3=df.copy()  df3.drop(['우현','인아'], axis=0, inplace=True)  print(df3)  ----------------------------------------------------------------------------  exam\_data = {'수학' : [90,80,70], '영어' : [98,89,95],  '음악' : [85,95,100], '체육' : [100,90,90]}  df=pd.DataFrame(exam\_data, index=['서준', '우현', '인아'])  print(df.loc['서준'])  print(df.iloc[0])  print(df.loc['서준':'인아']) # ['인아']의 자료도 포함  print(df.iloc[0:2]) # 첨자 인덱스 2의 자료는 포함하지 않음  print(df['수학']) # 컬럼 인덱스로 검색  print(df.수학)  print(df[['수학','영어']]) # 컬럼 인덱스로 검색  df.loc["과학"]=[100,90,80,88] # row로 데이터를 추가하는 경우 df.loc[인덱스명]=[값, ...]  df  ---------------------------------------------------------------------------------  # 키보드에서 이름과 국어, 영어, 수학 점수를 입력받아 이름을 인덱스로 저장하는 score detaframe을 생성  # 결과를 화면에 출력, d\이름에 'quit' 입력되면 종료  # 성적 입력 프로그램 작성  import pandas as pd  def input\_score():  scores=input("국어 영어 수학 점수 입력 > ").split()  for i, score in enumerate(scores):  scores[i]=int(score) # 데이터 타입 변환  return scores  df=pd.DataFrame(columns=["국어","영어","수학"]) # 컬럼명만 부여한 데이터프레임 생성  while True:  name= input("이름 입력/quit 종료 > ")  if name=="quit":  break    score=input\_score()  df.loc[name]=score # 입력된 name 인덱스로 점수 추가    df # 데이터 출력  # 컬럼을 접근하는 방법 : df.컬럼명 또는 df[컬럼명]  # row를 접근하는 방법 : df.iloc[인덱스], df.iloc[1,3,5], df.iloc[0:3]  # df.loc['aa':'cc'], df.loc[['aa','cc']], df.iloc[시작: 끝: 간격]  # df.loc[인덱스명, 컬럼명], df.iloc[index 번호, col 열번호]  ------------------------------------------------------------------------------------  search\_name=input("검색할 이름 입력 : ")  for name in list(df.index):  if search\_name==name:  kor=df.loc[search\_name,'국어']  eng=df.loc[search\_name,'영어']  math=df.loc[search\_name,'수학']  print("{} : {}, {}, {}, total = {}".format(search\_name,kor,eng,math,kor+eng+math))  ------------------------------------------------------------------------------------  import pandas as pd  # DataFrame() 함수로 데이터프레임 변환. 변수 df에 저장  exam\_data = {'이름' : [ '서준', '우현', '인아'],  '수학' : [ 90, 80, 70],  '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)  print('\n')  # 데이터프레임 df의 특정 원소 1개 선택 ('서준'의 '음악' 점수)  a = df.loc['서준', '음악']  print(a)  b = df.iloc[0, 2]  print(b)  print('\n')  # 데이터프레임 df의 특정 원소 2개 이상 선택 ('서준'의 '음악', '체육' 점수)  c = df.loc['서준', ['음악', '체육']]  print(c)  d = df.iloc[0, [2, 3]]  print(d)  e = df.loc['서준', '음악':'체육']  print(e)  f = df.iloc[0, 2:]  print(f)  print('\n')  # df의 2개 이상의 행과 열로부터 원소 선택 ('서준', '우현'의 '음악', '체육' 점수)  g = df.loc[['서준', '우현'], ['음악', '체육']]  print(g)  h = df.iloc[[0, 1], [2, 3]]  print(h)  i = df.loc['서준':'우현', '음악':'체육']  print(i)  j = df.iloc[0:2, 2:]  print(j)  ------------------------------------------------------------------------------  # 데이터프레임 df의 특정 원소 2개 이상 선택('서준'의 '음악', '체육' wjatn)  c=df.loc['서준',['음악','체육']]  print(c)  d=df.iloc[0, [2,3]]  print(d)  e=df.iloc[0,2:]  print(e)  f=df.loc['서준','음악':'체육'] # 음악부터 체육까지  print(f)  print('\n')  # 국어 점수를 모두 80점으로 추가  df["국어"] = 80  print(df)  df.loc["행추가"]=0  df  -----------------------------------------------------------------------------  def input\_dataframe(df):  while True:  name = input("이름 입력 quit 종료 > ")  if name == "quit":  break  data = input\_data()  df.loc[name] = data # 입력된 name 인덱스로 데이터 추가  def input\_data():  while True:  values = input("키와 몸무게 ex) 170 51.2 > ").split()  if len(values) == 2:  break  else:  print("키와 몸무게를 형식에 맞게 입력하세요 ")    for i, value in enumerate(values):  values[i] = float(value)  return values  def search\_data(name, df):  for i, idx\_name in enumerate(list(df.index)):  if name == idx\_name:  return i  def print\_data(idx, df):  print("{} : {}, {}".format(df.iloc[idx], df.iloc[idx,0], df.iloc[idx,1]))    import pandas as pd  df = pd.DataFrame(columns=["키","몸무게"])  input\_dataframe(df) # 자료 입력  print("1 : ")  print(df)  # 검색하고자 하는 이름 입력  search\_name = input("검색할 이름 입력 : ")  idx = search\_data(search\_name, df)  if idx < len(df):  print\_data(idx, df)  value = input\_data() # 키와 몸무게 입력  df.iloc[idx] = value  print("2 : ")  df  #평균키보다 검적은 키를 가진 사람의 이름과 키를 출력  avg\_height = sum(df.키)/len(df.키)  print(avg\_height)  for i, height in enumerate(df.키):  if height < avg\_height:  print("3 => {} : {}".format(df.iloc[i], df.iloc[i,0])) |