BigData

- 실습코드 소스 코드 참고

<넘파이(NumPy)>

NumPy는 파이썬에서 다차원 배열 및 행렬을 처리하는 라이브러리입니다. "Numerical Python"의 약어로, 과학적 및 수학적 계산을 위한 강력한 도구를 제공합니다. NumPy는 데이터 분석, 머신 러닝, 과학 연구, 엔지니어링 등 다양한 분야에서 사용되며, 파이썬의 기본 리스트보다 더 빠르고 메모리 효율적인 다차원 배열을 제공합니다.

NumPy의 주요 특징과 기능은 다음과 같습니다:

다차원 배열(N-dimensional array): NumPy의 핵심 데이터 구조인 NumPy 배열(ndarray)은 동일한 데이터 타입을 가지는 다차원 배열을 생성하고 조작할 수 있습니다. 이러한 배열은 효율적인 메모리 사용과 빠른 연산을 제공합니다.

브로드캐스팅(Broadcasting): NumPy는 서로 다른 크기의 배열 간에 연산을 수행할 수 있도록 브로드캐스팅 기능을 제공합니다. 이를 통해 배열 간의 연산이 간단하게 수행될 수 있습니다.

벡터화 연산(Vectorized operations): NumPy는 배열에 대한 연산을 원소별로 수행하는 벡터화 연산을 지원합니다. 이로 인해 반복문을 작성하지 않고도 배열 연산을 수행할 수 있습니다.

선형 대수 및 통계 함수: NumPy는 선형 대수(행렬 곱셈, 역행렬 계산 등)와 통계(평균, 분산, 표준편차 등) 관련 함수를 포함하고 있어 데이터 분석 및 과학적 연구에 유용합니다.

데이터 입출력: NumPy는 다양한 파일 형식(텍스트, 바이너리)의 데이터를 읽고 쓸 수 있는 기능을 제공합니다.

C/C++와의 통합: NumPy는 C/C++ 코드와 쉽게 통합할 수 있는 인터페이스를 제공하므로 고성능 수치 계산을 위해 C/C++ 라이브러리와 함께 사용할 수 있습니다.

NumPy는 데이터 과학 및 과학 연구 분야에서 표준 라이브러리로 자리 잡고 있으며, 다른 데이터 처리 및 시각화 라이브러리와 통합하여 파이썬을 데이터 분석 및 과학 연구용으로 강력한 도구로 만들어 주고 있습니다.

```
import random
## 파이썬 2차원 리스트 생성
SIZE = 5
pythonList = [ [random.randint(0,255) for _ in range(SIZE)] for _ in range(SIZE)]

## 리스트를 출력하기
for i in range(SIZE) :
    for k in range(SIZE) :
        print("%3d" % pythonList[i][k], end=' ')
    print()
print()

## 리스트에 100을 더하기.
for i in range(SIZE) :
```

```
    for k in range(SIZE) :
        pythonList[i][k] += 100

## 리스트를 출력하기
for i in range(SIZE) :
    for k in range(SIZE) :
        print("%3d" % pythonList[i][k], end=' ')
    print()
```

```
171  97  60 138 211
  3  70 127 245  56
207 137  50  12 143
 35  75 176  10 174
121 106  78 244 191

271 197 160 238 311
103 170 227 345 156
307 237 150 112 243
135 175 276 110 274
221 206 178 344 291
```

와 같이 무작위의 수가 2차원 리스트로 나옴.(Java Math.random 사용이랑 비슷한 것 같음)

pip3 install numpy - Package 설치

<넘파이 2차원 배열 생성>

```
import numpy as np
## 넘파이 2차원 배열 생성
SIZE = 5
numpyAry = np.random.randint(0, 255, size=(SIZE, SIZE))

## 배열을 출력하기
print(numpyAry)
print()

## 배열에 100을 더하기.
numpyAry += 100

## 배열을 출력하기
print(numpyAry)
```

```
[[238 132  90 171 207]
 [ 92 223 163 105  60]
 [119 146 104  84  42]
 [ 55 108 188 100 246]
 [ 40  17 251  62  59]]

[[338 232 190 271 307]
 [192 323 263 205 160]
 [219 246 204 184 142]
```

 [155 208 288 200 346]
 [140 117 351 162 159]] - 이와 같이 출력됨
<NumPy 배열 사용 방법>

import numpy as np
#(2,3)은 행렬을 의미
ary = np.random.randit(0, 255, size=(2,3))
ary

<BroadCasting>
NumPy에서 배열 간 연산을 효율적으로 수행하기 위한 방법으로 사용
Tensorflow - Matrix BroadCasting에서 사용, DL에서 주로 사용

<첨자와 슬라이싱>
NumPy 배열 - List와 같이 index를 통해서 처리 할 수 있다.

<기타 NumPy 기능>
조건식 표현(Where), 통계 함수(sum, mean 등), 정렬(Sort), Unique(중복값 제거),
intersect1d(공통값 추출)

<비전처리>
비전처리(Vision Preprocessing)는 컴퓨터 비전(Computer Vision) 분야에서 이미지나 비디오 데이터를 분석하기 전에 데이터를 사전 처리하거나 준비하는 과정을 의미합니다. 비전처리는 이미지나 비디오에서 유용한 정보를 추출하거나 노이즈를 제거하고, 데이터를 향상시키기 위해 다양한 기술을 사용합니다. 주요 목적은 컴퓨터 비전 작업(객체 감지, 패턴 인식, 분할 등)을 수행하는 데 필요한 데이터 품질을 향상시키는 것입니다.

tkinter .jpg 확장자 .jpeg 확장자 사용 안 됨

<Pandas와 MatPlotlib>
Pandas는 엑셀이 없어도 엑셀의 기능을 파이썬에서 사용할 수 있도록 도와주는 라이브러리
파이썬으로 시각화 하는데 사용

MatPlotlib 그래프를 쉽고 다양하게 표현하는 라이브러리.

<MongoDB>
[MongoDB 켜기]
C:\Users\admin>mongod --version
db version v7.0.0
Build Info: {
    "version": "7.0.0",
    "gitVersion": "37d84072b5c5b9fd723db5fa133fb202ad2317f1",
    "modules": [],
    "allocator": "tcmalloc",
    "environment": {
        "distmod": "windows",
        "distarch": "x86_64",
        "target_arch": "x86_64"

```
    }
}
```
Linux, Mac에서의 설치 및 실행은 책 참고

MongoDB 구조
- Document, Collection, DataBase

한 행의 데이터 = Document
Document에 객체를 넣을 수 있음.
BSON(Binary JSON)의 형태로 저장
필드에는 문제열만 들어가지만 값에 들어갈 수 있는 형식에는 배열, 숫자, 3차원 위치 좌표로부터 필드 값을 가지는 또 다른 오브젝트도 값으로 가질 수 있음.
BSON 구조는 거의 모든 정보를 표현 할 수 있도록 다양한 값의 형식을 지원한다.

Collection과 DataBase
<MongoDB 사용 및 Collection과 DataBase 조회>
```
Mongo
use testDB
switched to db testDB
db.myCollection.insertOne({x:1})
db
testDB
show dbs
show collections
```

<도큐먼트 삽입, 수정, 삭제>
<MongoDB 게시판 작성하기>

<데이터 입력>

<한개의 데이터 입력>
```
예> db.article.insertOne({
  board_id: "secretboard_id",
  title: 'my Secret Title',
  content: 'hi, hello1',
  author: 'noname'
})
```

<게시판 도큐먼트 생성하기>
```
use board
freeboard_result = db.board.insertOne({name: "자유게시판"})
freeboard_id = freeboard_result.insertedId
```

<게시판 데이터 입력>
```
db.article.insertMany([
  {
    board_id: "freeboard_id",
    title: 'hello',
    content: 'hi, hello1',
```

```
    author: 'Karoid'
  },
  {
    board_id: "freeboard_id",
    title: 'hi',
    content: 'hi, hello2',
    author: 'Jeong'
  },
  {
    board_id: "freeboard_id",
    title: 'hi',
    content: 'hi, hello3',
    author: 'Hong',
    comments: [
      {
        author: 'karoid',
        content: 'hello Hong!'
      }
    ]
  }
])


<비밀게시판 만들기>
secretboard_id = db.board.insertOne({name: '비밀게시판'}).insertedId
db.article.insertOne({
  board_id: "secretboard_id",
  title: 'my Secret Title',
  content: 'hi, hello1',
  author: 'noname'
})


<글을 불러오기 - 조회>
use board // Document 조회
freeboard_id = db.board.find({name: '자유게시판'}).toArray()[0]._id

db.article.find({board_id: "freeboard_id"},{
_id: false, board_id: false, author: false, comments: false})

<글 수정하기>
use board // Document 조회
db.article.updateMany({},{$set: {upvote: 0}})

<글 추천수 증가>
secretboard_id = db.board.find({name: "비밀게시판"}).toArray()[0]._id
db.article.updateMany({board_id: secretboard_id}, {$inc: {upvote: 1}})

<댓글이 달린 자유게시글의 id 저장>
freeboard_id = db.board.find({name: "자유게시판"}).toArray()[0]._id
```

```
doc_id = db.article.find({board_id: "freeboard_id", author: "Hong"}).toArray()[0]._id

<Content 값 수정>
db.articel.updateOne({_id: 'doc_id'}, {$set: {content: 'updated'}})

<comments 필드의 요소 추가>
db.article.updateOne(
  { _id: 'doc_id' },  {
    $push: {comments: {author: "Quote", content: "reply"}}
})

db.article.updateOne(
  { _id: 'doc_id' },  {
    $set: { "comments.$[karoidcomment].upvote": 0 }
  },
  {
    arrayFilters: [{ "karoidcomment.author": "Quote" }]
  }
)

<Document 삭제>
db.article.deleteMany({})  // Collection 내의 모든 Document 삭제
db.article.drop()       // Collection 삭제(article 컬렉션)
db.board.drop()     // Collection 삭제(board 컬렉션)
db.dropDatabase() // 선택된 DB 삭제

순차적으로 하고 show collections 하면 비어있는 것 확인

<Web 환경으로 연결>
Python - Django(장고)
Node.js - Express(익스프레스) 등이 있음.

Spring Boot로도 연결할 수 있음.
```