Python을 활용한 데이터 분석 강의

파일 처리

HYUNDAI MOBIS

파일 생성하기

- open 함수

  파일 객체 = open(파일 이름, 파일 열기 모드)

```
f = open('new.txt', 'w')
f.close()
```

```
f = open('C:/Python/new.txt', 'w')
f.close()
```

  - 파일 열기 모드
    - r : 읽기 모드 - 파일을 읽기만 할 때 사용
    - w : 쓰기 모드 - 파일에 내용을 쓸 때 사용
    - a : 추가 모드 - 파일의 마지막에 새로운 내용을 추가할 때 사용

- .close() 필수!

파일 읽고 쓰기

- 파일을 쓰기 모드로 열어 출력값 적기

```
f = open('new.txt', 'w')

for i in range(1, 11):
    data = '%d번째 줄입니다.\n' %i
    f.write(data)
f.close()
```

파일 읽고 쓰기

- read() 함수

```
f = open('new.txt', 'r')
data = f.read()
print(type(data))
print(data)
f.close()
```

```
<class 'str'>
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
6번째 줄입니다.
7번째 줄입니다.
8번째 줄입니다.
9번째 줄입니다.
10번째 줄입니다.
```

파일 읽고 쓰기

- readline() 함수

```
f = open('new.txt', 'r')
line = f.readline()
print(line)
f.close()
```

```
1번째 줄입니다.
```

```
f = open('new.txt', 'r')
while True:
    line = f.readline()
    if not line:
        break
    print(line)
f.close()
```

```
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
6번째 줄입니다.
7번째 줄입니다.
8번째 줄입니다.
9번째 줄입니다.
10번째 줄입니다.
```

파일 읽고 쓰기

- readline() 함수

```
f = open('new.txt', 'r')
while True:
    line = f.readline()
    if not line:
        break
    print(line[:-1])
f.close()
```

```
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
6번째 줄입니다.
7번째 줄입니다.
8번째 줄입니다.
9번째 줄입니다.
10번째 줄입니다.
```

파일 읽고 쓰기

- readlines() 함수: list 객체로 반환

```
f = open('new.txt', 'r')

lines = f.readlines()
print(lines)

for line in lines:
    print(line)
f.close()
```

```
['1번째 줄입니다.\n', '2번째 줄입니다.\n', '3번째 줄입니다.\n', '4번째 줄입니다.\n', '5번째 줄입니다.\n', '6번째 줄입니다.\n', '7번째 줄입니
다.\n', '8번째 줄입니다.\n', '9번째 줄입니다.\n', '10번째 줄입니다.\n']
1번째 줄입니다.

2번째 줄입니다.

3번째 줄입니다.

4번째 줄입니다.

5번째 줄입니다.

6번째 줄입니다.

7번째 줄입니다.

8번째 줄입니다.

9번째 줄입니다.

10번째 줄입니다.
```

파일 읽고 쓰기

- 파일에 새로운 내용 추가하기

  - 쓰기 모드로 파일을 열면 그 파일의 내용이 모두 사라지기 때문에 원래 있던 값을 유지하면서 새로운 값을 추가할 때는 추가 모드('a')로 파일을 오픈

```
f = open('new.txt', 'a')
for i in range(11, 20):
    data = '%d번째 줄입니다.\n' %i
    f.write(data)
f.close()
```

파일 읽고 쓰기

- write() 함수
  - 쓰기 모드로 파일을 열어서 파일에 내용을 작성할 때 사용

```
file = open('new2.txt', mode='w')
file.write('Start\n')
file.write('To be, or not to be - that is the question.\n')
file.write('End')
file.close()
```

```
file = open('new2.txt', mode='r')
print(file.read())
file.close()
```

```
Start
To be, or not to be - that is the question.
End
```

파일 읽고 쓰기

- split(sep=None, maxplit=-1)
  - 문자열을 특정 구분자(sep)를 기준으로 나눈 후 리스트 형태로 반환
    - 특정 구분자가 주어지지 않으면 화이트스페이스 기준으로 분할
  - 최대분할값(maxplit)이 주어지면 그 값만큼 분할되며, 값이 주어지지 않으면(또는 -1) 모두 분할

```
eng = 'Introduction to Python'
```

```
eng.split()
```

```
['Introduction', 'to', 'Python']
```

```
eng.split('t', 2)
```

```
['In', 'roduc', 'ion to Python']
```

```
eng.rsplit('o', 2)
```

```
['Introduction t', ' Pyth', 'n']
```

.splitlines()
각 줄의 마지막에
새줄바꿈 등
화이트 스페이스 없이
줄 단위로 나눈 후
리스트 형식으로 반환

파일 읽고 쓰기

- 데이터 파일 파싱(parsing data files)
  - 대부분의 경우 데이터셋은 여러 column으로 이루어져 있음
  - 각 row를 리스트의 객체로 반환

```
file = open('data/grades.txt', mode='r')
data = file.read().splitlines()
file.close()

print(len(data))
print(data)
```

```
5
['성명, 점수, 성적', '제이지, 97, A+', '어피치, 80, B+', '무지, 60, C-', '튜브, 95, A0']
```

  - 텍스트 파일의 모든 내용은 문자열로 인식되므로 숫자 연산이 필요할 경우 형 변환이 필요

파일 읽고 쓰기

- with open(파일이름, 모드) as 변수:

```
with open('data/new.txt', mode='r')as f:
    data = f.read()
    print(data)
```

```
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
6번째 줄입니다.
7번째 줄입니다.
8번째 줄입니다.
9번째 줄입니다.
10번째 줄입니다.
11번째 줄입니다.
12번째 줄입니다.
13번째 줄입니다.
14번째 줄입니다.
15번째 줄입니다.
16번째 줄입니다.
17번째 줄입니다.
18번째 줄입니다.
19번째 줄입니다.
```

파일 읽고 쓰기

- join() 함수
  - 문자열로 순환형 문자열 객체들을 결합
  - 많은 수의 문자열을 한번에 연결하는 것이 가능

```
languages = ['Python', 'Java', 'Ruby']
```

```
' '.join(languages)
```

```
'Python Java Ruby'
```

```
','.join(languages)
```

```
'Python,Java,Ruby'
```

```
'-'.join(reversed(languages))
```

```
'Ruby-Java-Python'
```

Practice 4-1 🤓

- 파일의 각 줄 번호 출력

  - 사용자로부터 텍스트 파일의 이름('고향의 봄.txt')을 입력 받아(input 활용) 해당 파일을 읽은 후 각 줄의 줄 번호와 내용을 출력하는 프로그램 작성

```
1: ======= 고향의 봄 =======

2:

3: 나의 살-던 고향은 꽃피는-산골-

4: 복숭아-꽃 살-구꽃 아-기 진-달래-

5: 울긋불-긋 꽃 대궐 차리인-동네-

6: 그 속에서 놀던 때가 그립습니다-

7:

8: 꽃-동-네 새 동네 나의 옛-고향-

9: 파-란-들 남쪽에서 바-람이-불면-

10: 냇-가에 수양버들 춤추는-동네-

11: 그 속에서 놀던 때가 그립습니다-
```

Practice 4-2 🤓

- 파일의 첫 몇 줄 읽고 출력
  - 대상 파일('고향의 봄.txt')의 처음 몇 줄만 출력하는 프로그램을 작성
  - input() 함수를 활용해 숫자를 입력 후, 그 줄까지만 출력
    - 숫자 4를 입력했을 경우

```
======= 고향의 봄 =======


나의 살-던 고향은 꽃피는-산골-

복숭아-꽃 살-구꽃 아-기 진-달래-
```

Practice 4-3 🤓

- 파일에서 지정된 몇 줄을 읽고 출력
  - 대상 파일(‘고향의 봄.txt’)에서 지정된 몇 줄만 출력하는 프로그램을 작성
    - 3과 6을 입력했을 경우

나의 살-던 고향은 꽃피는-산골-

복숭아-꽃 살-구꽃 아-기 진-달래-

울긋불-긋 꽃 대궐 차리인-동네-

그 속에서 놀던 때가 그립습니다-

Practice 4-4 🤓

- 투수 통계 데이터 파싱
  - ‘pitcher_stats.txt’ 파일로부터 투수 통계 데이터를 읽음
  - 아래와 같이 데이터를 파싱
    - Name은 문자열로, W와 L은 정수로, ERA는 실수로 처리
    - 텍스트 파일의 단락이 ‘/’로 구분되어져 있음

Name	W	L	ERA
Kershaw	18	4	2.31
Jansen	5	0	1.32
Wood	16	3	2.27
Hill	12	8	3.32

```
[['Kershaw', 18, 4, 2.31], ['Jansen', 5, 0, 1.0, '32'], ['Wood', 16, 3, 2.27], ['Hill', 12, 8, 3.32]]
```

HYUNDAI MOBIS

Practice 4-5 🤓

- ‘zen.txt’ 파일 다루기

  - 각 줄을 마침표(.)로 구분해서 분할한 후 한 줄씩 새로운 텍스트 파일에 저장 (zen_of_python.txt)

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
```