단기 집중교육

3일차 실습

Computational Thinking

연세대학교
YONSEI MIRAE CAMPUS

파일 읽고 쓰기

YONSEI MIRAE CAMPUS

파일 읽고 쓰기

- f = open(file_name[, mode , encoding = 인코딩 방식])
  # 파일 열기(읽기 모드)
  (mode= r: 읽기, w: 쓰기)
- data = f.read() #파일의 내용 읽기
  f.write(str) #파일에 문자열 쓰기
- f.close() # 파일 닫기
- print(data) # 읽어온 파일 내용 출력

파일 읽기


```
f = open('C:/myPyExcel/data/ch04/read_test.txt', 'r')
# 파일 열기(읽기 모드)

data = f.read() # 파일의 내용 전체를 읽어서 변수에 할당
f.close()       # 파일 닫기

print(data)     # 읽어온 파일 내용 출력
```

```
All grown-up
were once children,
although few of them
remember it.
```

※ 한글 인코딩 방식 -> utf-8 / cp949

%%writefile C:₩myPyExcel₩data₩ch04₩read_test.txt (각자 파일 저장할 경로로 작성)

All grown-up

were once children,

although few of them

remember it.

파일 읽기

1. cp949로 인코딩된 한글 텍스트 파일 읽기

   파일 제목: 헌법_cp949

2. utf-8로 인코딩된 한글 텍스트 파일 읽기

   파일 제목: 헌법_utf8

파일 읽기- 한 줄씩 읽어 처리하기


```
file_name = 'C:/myPyExcel/data/ch04/read_test.txt'
# 파일 경로를 변수에 할당

f = open(file_name, 'r') # 파일 열기(읽기 모드)

line1 = f.readline()
# 파일의 내용을 한 줄씩 읽어서 변수에 할당

line2 = f.readline()
# 파일의 내용을 한 줄씩 읽어서 변수에 할당

f.close()  # 파일 닫기

print(line1, end='')
print(line2, end='')
```

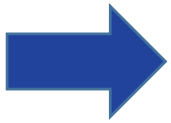
```
All grown-up
were once children,
```

readline()

-한 줄씩 읽어서 문자열로 반환
-실행 후 이어서 실행 시 다음 문자열 읽음
(마지막 줄까지 읽고 함수를 실행하면 빈 문자열 반환)
-실행 횟수만큼 문자열 한 줄씩 읽음

파일 읽기- 한 줄씩 읽어 처리하기

1. read_test.txt 파일을 읽어서 모든 문자열을 출력하기
   단, readline을 이용하여 한 줄씩 출력해야 함
   (hint! while, if문 사용)



출력화면

```
1: All grown-up
2: were once children,
3: although few of them
4: remember it.
```

파일 읽기- 한 줄씩 읽어 처리하기


```
file_name = 'C:/myPyExcel/data/ch04/read_test.txt'
# 파일 경로를 변수에 할당

f = open(file_name, 'r')    # 파일 열기(읽기 모드)
line_num = 0 # 줄 수 표시를 위한 변수 초기화

while True:
    line = f.readline()       # 파일의 내용을 한 줄씩 읽어서 변수에 할당
    if (line == ''):          # line이 빈 문자열인지 검사
        break                 # 빈 문자열이면 while문을 빠져나감
    line_num = line_num + 1   # line_num 을 1씩 증가
    print("{0}: {1}".format(line_num, line), end='')

f.close() # 파일 닫기
```

```
1: All grown-up
2: were once children,
3: although few of them
4: remember it.
```

파일 읽기- 한 줄씩 요소로 갖는 리스트

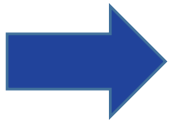

```
file_name = 'C:/myPyExcel/data/ch04/read_test.txt'

f = open(file_name, 'r') # 파일 열기(읽기 모드)
lines = f.readlines()    # 파일 전체의 내용을 읽어서 변수에 할당
f.close()    # 파일 닫기

print(lines)
```

```
['All grown-up₩n', 'were once children,₩n', 'although few of them₩n',
'remember it.₩n']
```

◆ realines() -> 모든 줄을 읽어서 한 줄씩 요소로 가짐
(개행문자 포함)

파일 읽기- 한 줄씩 요소로 갖는 리스트

1. read_test.txt 파일을 읽어서 모든 문자열을 출력하기
   단, readlines을 이용하여 한 줄씩 출력해야 함
   (hint! for문 사용)



출력화면

```
1: All grown-up
2: were once children,
3: although few of them
4: remember it.
```

파일 읽기- 한 줄씩 요소로 갖는 리스트


```
file_name = 'C:/myPyExcel/data/ch04/read_test.txt'

f = open(file_name, 'r') # 파일 열기(읽기 모드)
lines = f.readlines()    # 파일 전체의 내용을 읽어서 변수에 할당
f.close() # 파일 닫기

line_num = 0 # 줄 수 표시를 위한 변수 초기화
for line in lines:
    line_num = line_num + 1  # line_num 을 1씩 증가
    print("{0}: {1}".format(line_num, line), end='')
```

```
1: All grown-up
2: were once children,
3: although few of them
4: remember it.
```

파일 쓰기

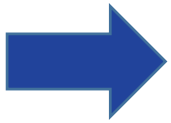

```
file_name = 'C:/myPyExcel/data/ch04/write_test.txt'
# 파일 경로를 변수에 할당

f = open(file_name, 'w')  # 파일 열기(쓰기 모드)
f.write("Python is powerful... and fast;₩n")
# 문자열을 파일에 쓰기

f.write("plays well with others;₩n")
f.write("runs everywhere;₩n")
f.write("is friendly & easy to learn;₩n")
f.write("is Open.₩n")
f.close() # 파일 닫기

print("생성한 파일:", file_name) # 생성한 파일 이름 출력
```

```
생성한 파일: C:/myPyExcel/data/ch04/write_test.txt
```

◆ Write 함수는 자동 줄 바꿈이 되지 않기 때문에 개행문자를 추가해야 함(₩n)

파일 쓰기

1. write()함수를 사용해서 아래 출력화면 내용이 작성된 텍스트 파일을 작성하시오.
   (for문을 이용하여 구구단 작성)



출력화면

```
[구구단 2단의 일부]
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
```

파일 쓰기

1. write()함수를 사용해서 아래 출력화면 내용이 작성된 텍스트 파일을 작성하시오.

   (for문을 이용하여 구구단 작성)

```
file_name = 'C:/myPyExcel/data/ch04/two_times.txt'

f = open(file_name, 'w') # 파일 열기(쓰기 모드)
f.write("[구구단 2단의 일부]₩n")
for num in range(1,6):  # for문: num이 1~5까지 반복
    format_string = "2 x {0} = {1}₩n".format(num, 2 * num)
    # 저장할 문자열 생성

    f.write(format_string)  # 파일에 문자열 쓰기
f.close() # 파일 닫기

print("생성한 파일:", file_name) # 생성한 파일 이름 출력
```

with 문을 사용한 파일 읽고 쓰기

◆ with open(file_name [, mode, encoding= 인코딩 방식]) as f:

-> 파일 객체(f)를 이용해 파일을 읽거나 쓰는 코드

**close()를 사용할 필요가 없음

```
with open(file_name, 'r') as f:  # 파일 열기(읽기 모드)
    data = f.read()              # 파일에서 문자열 읽기
    print(data)
```

```
[구구단 2단의 일부]
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
```

문자열 처리

YONSEI MIRAE CAMPUS

문자열 처리

1. split함수를 사용하여 “에스프레소,아메리카노,카페라테,카푸치노” 문자열을 단어로 분리해서 리스트로 반환

2. strip함수를 사용하여 "₩n Python ₩n₩n” 문자열을 공백과 개행문자 없이 반환

3. 리스트 ["서울시","서초구","반포대로","201(반포동)"]를 개행문자(₩n)을 요소 사이에 추가하여 문자열로 반환 및 출력

문자열 처리


```
"에스프레소,아메리카노,카페라테,카푸치노".split(',')
```

```
['에스프레소', '아메리카노', '카페라테', '카푸치노']
```

```
"₩n  Python  ₩n₩n".strip()
```

```
'Python'
```

```
joined_str = "₩n".join(["서울시","서초구","반포대로","201(반포동)"])
print(joined_str)
```

```
서울시
서초구
반포대로
201(반포동)
```

문자열 처리

◆ 문자열 찾기

- Str.find(search_str[, start, end])
  > 문자열의 위치 찾기
- Str.count(search_str[, start, end])
  > 문자열에 있는 검색 문자열의 개수 찾기
- Str.startwith(prefix_str[, start, end])
  > 문자열이 지정한 문자열로 시작하면 true 반환
- Str.endwith(suffix_str[, start, end])
  > 문자열이 지정한 문자열로 끝나면 true 반환

◆ 문자열 바꾸기

- str.replace(old_str, new_str[, count])
  > 주어진 문자열에서 지정한 문자열을 찾아 다른 문자열로 변경
  count는 변경 횟수

문자열 처리

◆ 문자열 찾기

str_p = "Python is powerful. Python is easy."

1. 문자열에서 easy라는 단어의 위치 출력
2. 시작위치 10, 끝범위 30에서 python 단어가 몇 개 있는지 출력
3. 문자열이 powerful로 시작하는지 출력
4. str_e = "[Python] [is] [easy] [to] [learn.]“ 문자열에서 [] 제거해서 출력하기

문자열 처리


```
str_p = "Python is powerful. Python is easy."
print(str_p.find("easy"))
print(str_p.count("Python", 10, 30))
print("- 문자열이 'powerful'로 시작?",str_p.startswith("powerful"))

str_e = "[Python] [is] [easy] [to] [learn.]"
str_e1 = str_e.replace("[", "")    # 문자열(str_e)에서 '[' 제거하고 결과를 변수에 할당
str_e2 = str_e1.replace("]", "")   # '['를 제거한 결과에 다시 ']'를 제거
print(str_e2)
```

```
30
1
- 문자열이 'powerful'로 시작? False
Python is easy to learn.
```

배열 데이터

YONSEI MIRAE CAMPUS

배열 데이터

◆ numpy

: 배열의 수치 연산을 쉽고 빠르게 수행하는 파이썬 패키지

패키지 설치가 필요한 경우 – cmd에서 pip install numpy

```
C:₩Users₩user>pip install numpy
Collecting numpy
  Downloading numpy-1.23.1-cp310-cp310-win_amd64.whl (14.6 MB)
     ---------------------------------------- 14.6/14.6 MB 4.5 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-1.23.1
```

jupyter에서 사용 전에 반드시 import numpy 실행

```
import numpy as np
```

배열 데이터 생성

◆ **arr= np.array(list_data)**

> 리스트 데이터(또는 튜플)를 인수로 받아 numpy 배열 객체 생성

(실수와 정수가 혼합되어 있으면 전체다 실수로 변환

문자열과 숫자가 혼합되어 있으면 문자열로 변환됨)

```
a4 = np.array([10, 'hello', 'python', 3.14])
a4
```

```
array(['10', 'hello', 'python', '3.14'], dtype='<U32')
```

배열 데이터 생성

◆ **arr= np.arange([start,] stop [,step])**

> 배열의 시작과(또는 시작과 끝) 간격을 지정해 배열을 생성

```
np.arange(0, 10, 1)  # start, stop, step 모두 지정
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

◆ **arr=np.linspace(start, stop[, num])**

> 배열의 범위와 요소의 개수를 지정해 넘파일 배열을 생성

```
np.linspace(1, 10, 10) # start, stop, num 지정
```

```
array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

배열 데이터 생성

1. 0부터 5전까지 0.5씩 더하여 넘파이 배열을 생성하시오.

2. 0부터 원주율(π)까지 동일한 간격으로 나눈 20개의 요소를 갖는 배열을 생성하시오.
   (np.pi -> numpy에서 π 입력하는 방법)

```
np.pi
```

```
3.141592653589793
```

배열 데이터 생성

1. 0부터 5전까지 0.5씩 더하여 넘파이 배열을 생성하시오.
2. 0부터 원주율(π)까지 동일한 간격으로 나눈 20개의 요소를 갖는 배열을 생성하시오.

   (np.pi -> numpy에서 π 입력하는 방법)

```
np.arange(0, 5, 0.5) # start, stop, step 모두 지정
```

```
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])
```

```
np.linspace(0, np.pi, 20)
```

```
array([0.        , 0.16534698, 0.33069396, 0.49604095, 0.66138793,
       0.82673491, 0.99208189, 1.15742887, 1.32277585, 1.48812284,
       1.65346982, 1.8188168 , 1.98416378, 2.14951076, 2.31485774,
       2.48020473, 2.64555171, 2.81089869, 2.97624567, 3.14159265])
```

배열 데이터 형태 재구성

- arr.shape -> 배열의 형태 출력

```
arr1 = np.array([0, 1, 2, 3, 4, 5])      # 1차원 배열
arr1.shape
```

```
(6,)
```

```
arr2 = np.array([[0, 1, 2], [3, 4, 5]]) # 2차원 배열
arr2.shape
```

```
(2, 3)
```

- arr.reshape() -> 배열의 형태를 재구성 함

```
arr1.reshape(2, 3) # 1차원 배열을 2차원 배열로 재구성
```

```
array([[0, 1, 2],
       [3, 4, 5]])
```

```
arr2.reshape(6,) # 2차원 배열을 1차원 배열로 재구성
```

```
array([0, 1, 2, 3, 4, 5])
```

배열 데이터 형태 재구성

1. 0부터 24까지의 숫자를 가진 1차원 배열을 생성하고, 그 배열을 3차원 배열로 재구성 하시오.

출력 결과

```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

배열 데이터 형태 재구성

1. 0부터 24까지의 숫자를 가진 1차원 배열을 생성하고, 그 배열을 3차원 배열로 재구성 하시오.

```
np.arange(0, 24, 1).reshape(2, 3, 4) # 1차원 배열을 3차원 배열로 재구성
```

```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

배열 데이터 형태 재구성

◆ arr.reshape() 예시

> reshape(m,n)을 적용할 때 m이나 n 중 하나의 값만 입력하는 경우에 나머지에 -1을 대입하면 배열(array)의 개수에 따라 자동으로 값이 계산되어 대입됨

```
arr1.reshape(2, -1) # arr1.reshape(2, 3)과 동일
```

```
array([[0, 1, 2],
       [3, 4, 5]])
```

```
arr1.reshape(-1, 1) # arr1.reshape(6, 1)과 동일
```

```
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5]])
```

배열 데이터 연산

◆ 기본 연산

+, - , *, / -> 두 배열의 형태가 같다면 두 배열의 요소끼리 연산

```
import numpy as np

arr1 = np.array([10, 20, 30, 40])
arr2 = np.array([1, 2, 3, 4])

arr1 + arr2  # 덧셈
```

```
array([11, 22, 33, 44])
```

1. 위의 나머지 뺄셈, 곱셈, 나눗셈을 계산하세요.

배열 데이터 연산

◆ 상수의 연산

+, - , *, / -> 배열의 각 요소와 상수의 연산을 수행

```
arr2 = np.array([1, 2, 3, 4])

arr2 + 10 # 배열의 각 요소에 상수 덧셈
```

```
array([11, 12, 13, 14])
```

1. 위의 나머지 뺄셈, 곱셈, 나눗셈, 거듭제곱, np.sin(x)을 계산하세요.

배열 데이터 연산

◆ 집계 및 통계 연산

```
import numpy as np

arr3 = np.array([1, 2, 3, 4, 5]) # 배열의 생성
[arr3.sum(), arr3.mean()]        # 합, 평균
```

```
[15, 3.0]
```

- 합, 평균 = .sum(), .mean()
- 표준편차, 분산 = .std(), .var()
- 최소, 최대 = .min(), .max()
- 누적 합 = .cumsum()
- 누적 곱 = .cumprod()

  -> 해당 값들을 구하시오.

배열 데이터 선택

◆ 배열 인덱싱

- 배열명[i] > 위치를 지정하여 배열의 요소를 가져옴
- 배열명[[i,j,k...]] > 원하는 여러 개의 요소를 가져옴(위치를 리스트로 지정)

```
import numpy as np

a1 = np.array([0, 10, 20, 30, 40, 50]) # 1차원 배열 생성
print([a1[0], a1[3], a1[5], a1[-1], a1[-2]]) # 배열 인덱싱의 다양한 예
print(a1[[4, 0, 5, -1, -2]]) # 배열의 위치로 여러 개의 요소를 선택
```

```
[0, 30, 50, 50, 40]
[40  0 50 50 40]
```

배열 데이터 선택

◆ 불(bool) 인덱싱

- 배열명[conditions] > 조건을 만족하는 배열의 요소를 선택

```
a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
a[a >= 5]
```

```
array([5, 6, 7, 8, 9])
```

1. A 배열에서 짝수인 요소만 선택
2. A 배열에서 짝수이면서 5 초과인 요소를 선택
3. A 배열에서 짝수이거나 5 초과인 요소를 선택

배열 데이터 선택


```
a[(a % 2) == 0]
```

```
array([0, 2, 4, 6, 8])
```

```
a[ ((a % 2)==0) & (a > 5) ] # 두 조건을 동시에 만족하는 요소만 선택
```

```
array([6, 8])
```

```
a[ ((a % 2)==0) | (a > 5) ] # 두 조건 중 하나만 만족해도 요소 선택
```

```
array([0, 2, 4, 6, 7, 8, 9])
```

Question and Answer

YONSEI MIRAE CAMPUS