

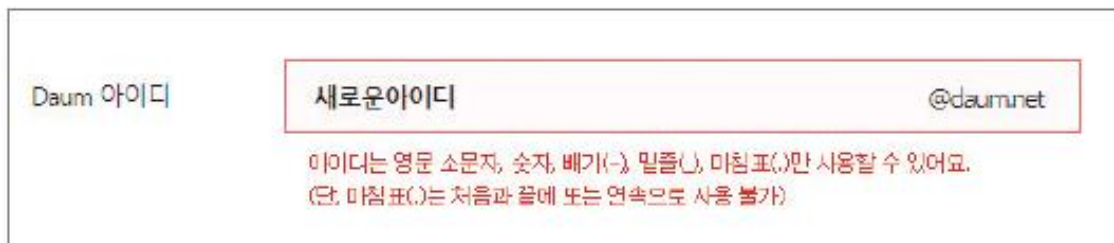
01

예외 처리

01. 예외 처리

■ 예외의 개념과 사례

- 예외(exception) 란 프로그램을 개발하면서 예상하지 못한 상황이 발생한 것이다. 프로그래밍의 예외는 크게 예측 가능한 예외와 예측 불가능한 예외로 나눌 수 있다.

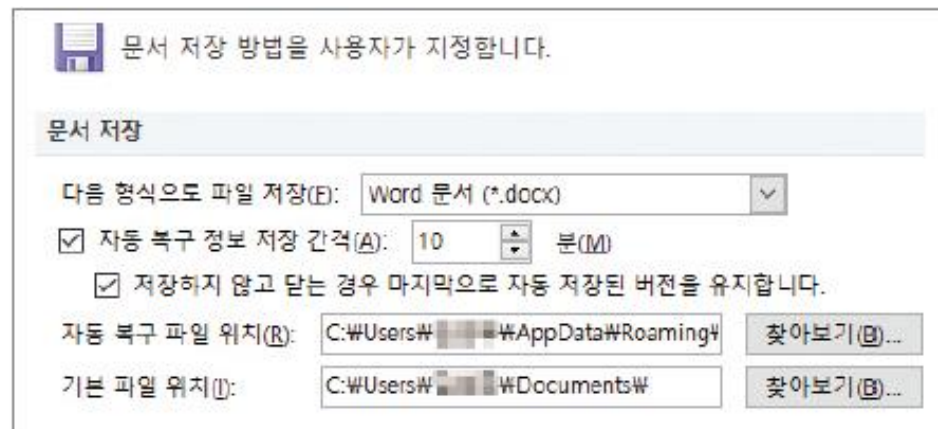


Daum 아이디

새로운아이디 @daumnet

아이디는 영문 소문자, 숫자, 배기(-), 밑줄(_), 마침표(.)만 사용할 수 있어요.
(단, 마침표(.)는 처음과 끝에 또는 연속으로 사용 불가)

(a) 아이디 생성 오류 입력



문서 저장 방법을 사용자가 지정합니다.

문서 저장

다음 형식으로 파일 저장(E): Word 문서 (*.docx)

☒ 자동 복구 정보 저장 간격(A): 10 분(M)

☒ 저장하지 않고 닫는 경우 마지막으로 자동 저장된 버전을 유지합니다.

자동 복구 파일 위치(R): C:\Users\...\AppData\Roaming\ 찾아보기(B)...

기본 파일 위치(I): C:\Users\... Documents\ 찾아보기(B)...

(b) 자동 저장 기능

01. 예외 처리

■ 예측 가능한 예외와 예측 불가능한 예외

- **예측 가능한 예외** : 발생 여부를 개발자가 사전에 인지할 수 있는 예외이다. 개발자는 예외를 예측하여 명시적으로 예외가 발생할 때는 어떻게 대응하라고 할 수 있다. 대표적으로 사용자 입력란에 값이 잘못 들어갔다면, if문을 사용하여 사용자에게 잘못 입력하였다고 응답하는 방법이 있다. 매우 쉽게 대응할 수 있다
- **예측 불가능한 예외** : 대표적으로 매우 많은 파일을 처리할 때 문제가 발생할 수 있다. 예측 불가능한 예외가 발생했을 경우, 인터프리터가 자동으로 이것이 예외라고 사용자에게 알려 준다. 대부분은 이러한 예외가 발생하면서 프로그램이 종료되므로 적절한 조치가 필요하다

01. 예외 처리

■ 예외 처리 구문 : `try -except`문

`try:`

예외 발생 가능 코드

`except` 예외 타입:

예외 발생 시 실행되는 코드

01. 예외 처리

■ 예외 처리 구문 : **try -except**문

- 0부터 9까지의 숫자를 i에 하나씩 할당하면서 10으로 나눈 값을 출력하는 코드
- '10÷0(10/0)'을 하면 0으로는 10을 나눌 수 없으므로 예외가 발생
- 예상 가능한 예외

```
for i in range(10):  
    try:  
        print(10/i)  
    except ZeroDivisionError:  
        print("Not divided by 0")
```

01. 예외 처리

■ 예외 처리 구문 : **try -except**문

- 0부터 9까지의 숫자를 i에 하나씩 할당하면서 10으로 나눈 값을 출력하는 코드
- '10÷0(10/0)'을 하면 0으로는 10을 나눌 수 없으므로 예외가 발생
- 예상 가능한 예외

```
for i in range(10):  
    try:  
        print(10/i)  
    except ZeroDivisionError:  
        print("Not divided by 0")
```

01. 예외 처리

■ 예외 처리 구문 : `try -except`문

```
for i in range(10):  
    try:  
        print(10/i)  
    except ZeroDivisionError:  
        print("Not divided by 0")
```

```
Not divided by 0  
10.0  
5.0  
3.3333333333333335  
2.5  
2.0  
1.6666666666666667  
1.4285714285714286  
1.25  
1.1111111111111112
```

01. 예외 처리

■ 예외 처리 구문 : 예외의 종류와 에러 메시지

예외 처리	예외 상황의 의미와 발생 원인
ArithmeticError	수의 연산과 관련된 문제가 발생할 때
EOFError	파일 등에서 읽어 들일 데이터가 더 이상 없을 때
Exception	대부분의 예외의 가장 상위 예외 처리 발생할 때
FileExistsError	이미 존재하는 파일이나 폴더를 새로 생성하려 할 때
FileNotFoundError	존재하지 않는 파일이나 폴더를 오픈하려 할 때
ImportError	모듈(라이브러리)을 불러올 수 없을 때
IndentationError	문법에서 들여쓰기가 잘못되었을 때
IndexError	잘못된 인덱스를 인덱싱할 때
NameError	존재하지 않는 변수를 호출할 때
ZeroDivisionError	0으로 숫자를 나눌 때
ValueError	변환할 수 없는 문자나 숫자를 변환할 때

01. 예외 처리

■ 예외 처리 구문 : 예외의 종류와 에러 메시지

```
for i in range(10):  
    try:  
        print(10/i)  
    except ZeroDivisionError as e :  
        print(e)  
        print("Not divided by 0")
```

```
division by zero  
Not divided by 0  
10.0  
5.0  
3.3333333333333335  
2.5  
2.0  
1.6666666666666667  
1.4285714285714286  
1.25  
1.1111111111111112
```

01. 예외 처리

■ 예외 처리 구문 : `try -except-else`문

- `try-except-else`문은 `if-else`문과 비슷
- 해당 예외가 발생하지 않을 경우 수행할 코드를 `else`문에 작성

`try:`

예외 발생 가능 코드

`except` 예외 타입:

예외 발생 시 실행되는 코드

`else:`

예외 발생하지 않을 때 실행되는 코드

01. 예외 처리

■ 예외 처리 구문 : try -except-else문

- 10을 i로 나누는 코드를 실행
 - 제대로 나누었을 경우 : else문에 의해 결과가 화면에 출력
 - 그렇지 않을 경우 : 사전에 정의된 except문에 의해 에러가 발생

```
for i in range(10):  
    try:  
        result = 10/i  
    except ZeroDivisionError:  
        print("Not divided by 0")  
    else:  
        print(result)
```

01. 예외 처리

■ 예외 처리 구문 : try -except-else문

```
for i in range(10):  
    try:  
        result = 10/i  
    except ZeroDivisionError:  
        print("Not divided by 0")  
    else:  
        print(result)
```

```
Not divided by 0  
10.0  
5.0  
3.3333333333333335  
2.5  
2.0  
1.6666666666666667  
1.4285714285714286  
1.25  
1.1111111111111112
```

01. 예외 처리

■ 예외 처리 구문 : **try-except-finally**문

■ finally문

- try-except문 안에 있는 수행 코드가 아무런 문제 없이 종료되었을 경우, 최종으로 호출하는 코드

try:

예외 발생 가능 코드

except 예외 타입:

예외 발생 시 실행되는 코드

finally:

예외 발생 여부와 상관없이 실행되는 코드

01. 예외 처리

■ 예외 처리 구문 : try-except-finally문

```
try:
    for i in range(1,10):
        result = 10//i
        print(result)
except ZeroDivisionError:
    print("Not divided by 0")
finally:
    print("종료되었다")
```

```
10
5
3
2
2
1
1
1
1
1
종료되었다
```

01. 예외 처리

■ 예외 처리 구문 : `raise()`문

- `raise`문은 `try-except`문과 달리 필요할 때 예외를 발생시키는 코드

`raise` 예외 타입(예외 정보)

01. 예외 처리

■ 예외 처리 구문 : **try-except-finally**문

```
while True:
    value = input("변환할 정수값을 입력 : ")
    for digit in value:
        if digit not in "0123456789":
            raise ValueError("숫자값을 입력하지 않았습니다")
    print("정수값으로 변환된 숫자-", int(value))
```

```
변환할 정수값을 입력 : 1
정수값으로 변환된 숫자- 1
변환할 정수값을 입력 : A
```

```
Traceback (most recent call last):
```

```
  File "test.py", line 5, in <module>
    raise ValueError("숫자값을 입력하지 않았습니다")
ValueError: 숫자값을 입력하지 않았습니다
```


PRACTICE

3번까지 있는 리스트 `a`를 만들고 (`a = [1, 2, 3]`)
`a[4]` 리스트를 호출할 경우 에러메시지를 출력하라

PRACTICE

3번까지 있는 리스트 a를 만들고 (a = [1, 2, 3])
a[4] 리스트를 호출할 경우 에러메시지를 출력하라

```
a = [1, 2, 3]
try:
    print(a[4])
except IndexError:
    print("없는 값을 호출하였습니다.")
```

PRACTICE

{이름 : 생년월일} 딕셔너리 생성 후 이름을 입력하면 생년월일을 출력하는 프로그램 작성 하라. 단, 해당 이름이 없는 경우 이름이 없다는 메시지를 출력하라

birth = {"홍길동":"2000년 3월 1일", "김춘추":"604년", "김유신":"595년"}

생일을 알고 싶은 사람을 입력하세요 : 홍길동

2000년 3월 1일

생일을 알고 싶은 사람을 입력하세요 : 김춘추

604년

생일을 알고 싶은 사람을 입력하세요 : 이황

데이터베이스에 존재하지 않는 이름입니다.

생일을 알고 싶은 사람을 입력하세요 : q

데이터베이스에 존재하지 않는 이름입니다.

PRACTICE

{이름 : 생년월일} 딕셔너리 생성 후 이름을 입력하면 생년월일을 출력하는 프로그램 작성 하라. 단, 해당 이름이 없는 경우 이름이 없다는 메시지를 출력하라

```
birth = {"홍길동":"2000년 3월 1일", "김춘추":"604년", "김유신":"595년"}
```

```
birth = {"홍길동":"2000년 3월 1일", "김춘추":"604년", "김유신":"595년"}  
a = ""
```

```
while a != 'q':  
    a = input("생일을 알고 싶은 사람을 입력하세요 : ")  
    try:  
        print(birth[a])  
    except KeyError:  
        print("데이터베이스에 존재하지 않는 이름입니다.")
```

02

파일 다루기

02. 파일 다루기

■ 파일의 종류

- 바이너리 파일(binary file)과 텍스트 파일(text file), 두 가지로 분류

바이너리 파일	텍스트 파일
<ul style="list-style-type: none">• 컴퓨터만 이해할 수 있는 형태인 이진(법) 형식으로 저장된 파일• 일반적으로 메모장으로 열면 내용이 깨져 보임(메모장에서 해석불가)• 엑셀파일, 워드파일	<ul style="list-style-type: none">• 사람도 이해할 수 있는 형태인 문자열 형식으로 저장된 파일• 메모장으로 열면 내용확인 가능• .txt 파일, .html파일, .py파일 등

02. 파일 다루기

■ 파일 읽기

- 텍스트 파일을 다루기 위해 `open()` 함수를 사용한다

```
저장할 변수명 = open("파일이름", "모드")  
변수명.close()
```

모드	설명	파일생성
r	파일을 읽기 전용 모드로 열기	
w	파일을 쓰기 모드로 열기	○
a	파일에 내용 추가하기	○
t	텍스트 모드로 파일 열기	
b	바이너리 모드로 파일 열기	
r+	읽기+쓰기 모드, 덮어쓰기로 파일을 쓴다.	
w+	읽기+쓰기 모드, 기존의 파일을 지우고 파일을 쓴다.	○
a+	읽기+쓰기 모드, 기존의 파일 끝에서부터 파일을 쓴다.	○

02. 파일 다루기

■ 파일 열기 / 닫기

- `open()` : 파일 열기

`open()`

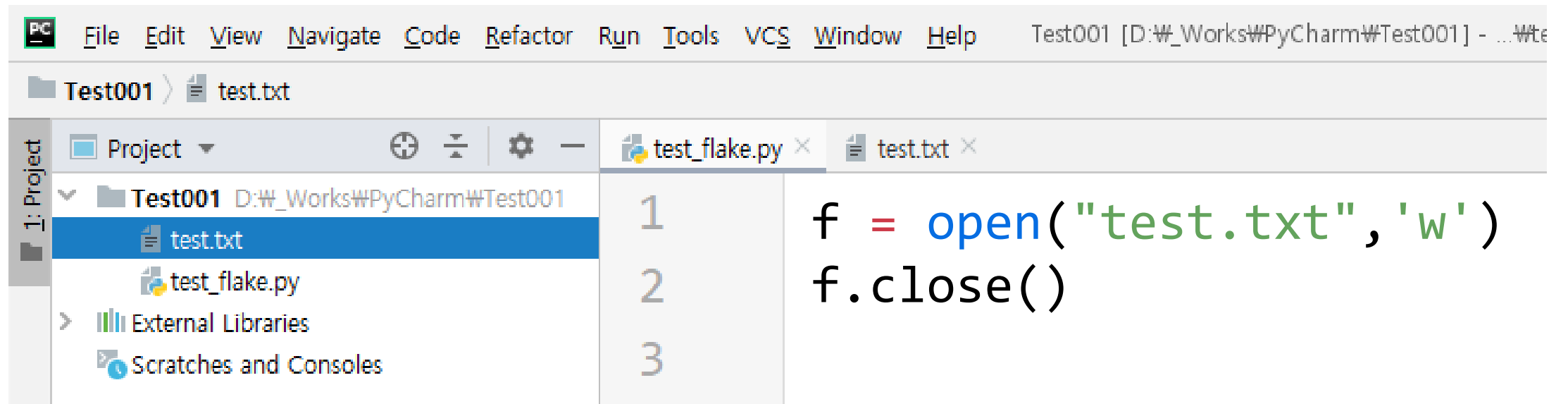
- `.close()` : 파일 닫기

`.close()`

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

test.txt 파일이 생성됨

02. 파일 다루기



02. 파일 다루기

■ 텍스트 파일에 글자 쓰기

- `write()` : 파일에 글자 쓰기

변수명.**`write("저장할 내용")`**

```
f = open("test.txt", 'w')  
f.write('파이썬 월드에 오신 것을 환영합니다.')  
f.close()
```

test.txt 파일이 생성됨

test.txt ×	
1	파이썬 월드에 오신 것을 환영합니다.

02. 파일 다루기

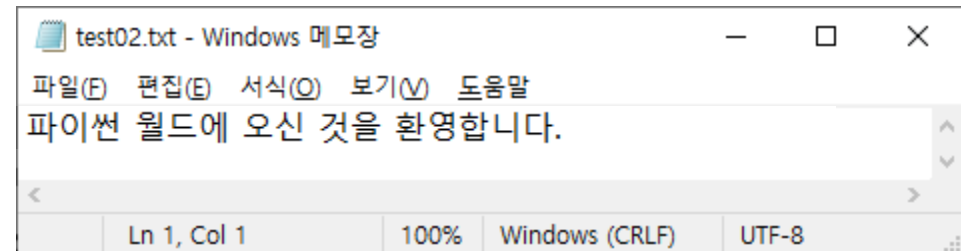
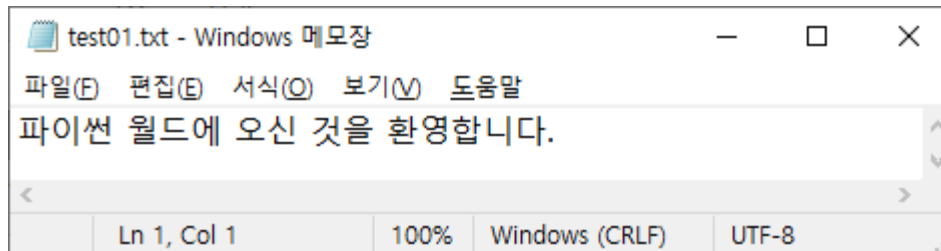
■ with 문 사용

- 파일을 open하면 무조건 close()해야함
- with문을 사용하면 close() 하지 않아도 with블록이 끝나면 자동으로 close해주어 편리하게 사용가능

with open("파일이름.txt", 'w') as 변수명:

```
f = open("test01.txt", 'w')  
f.write('파이썬 월드에 오신 것을 환영합니다.')  
f.close()
```

```
with open("test02.txt", 'w') as f:  
    f.write('파이썬 월드에 오신 것을 환영합니다.')
```

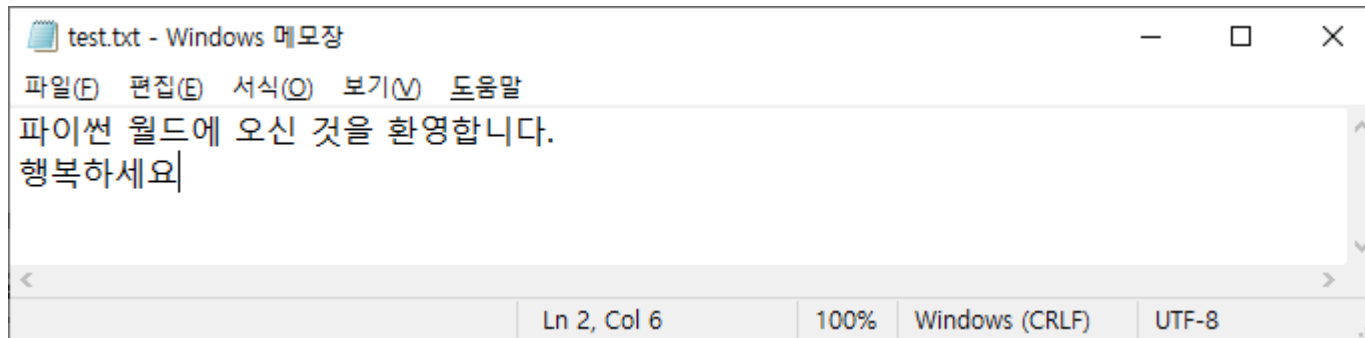


02. 파일 다루기

- 텍스트 파일에 글자 쓰기(여러 줄) : Wn

변수명.**write**("... \n ...")

```
f = open("test.txt", 'w')  
f.write('파이썬 월드에 오신 것을 환영합니다.\n행복하세요')  
f.close()
```

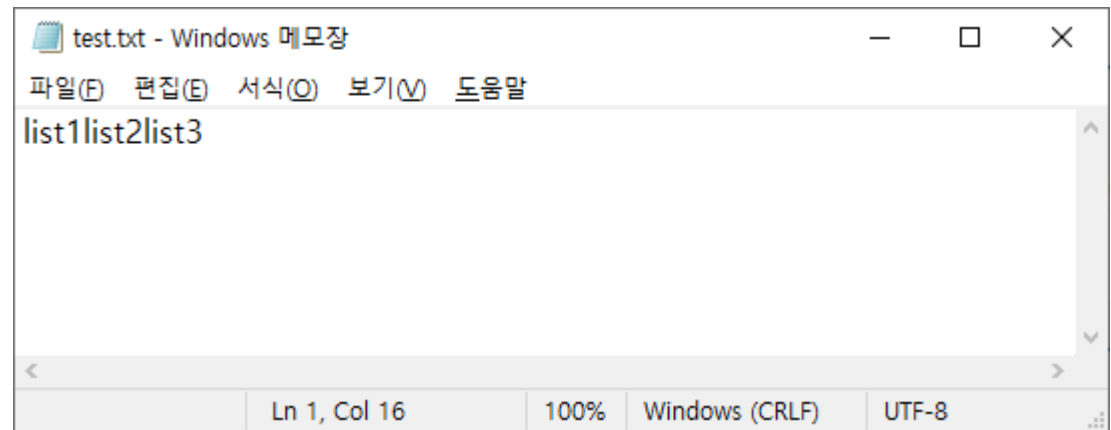


02. 파일 다루기

- 여러개의 리스트를 한줄에 쓰기

변수명.**writelines**("... \n ...")

```
f = open("test.txt", 'w')  
f.writelines(["list1", "list2", "list3"])  
f.close()
```



PRACTICE

새로운 텍스트 파일 `text01.txt`를 추가하고 1부터 10까지의 수가 입력되도록 저장하여라.

PRACTICE

새로운 텍스트 파일 text03.txt를 추가하고 1부터 10까지의 수가 입력되도록 저장하여라.

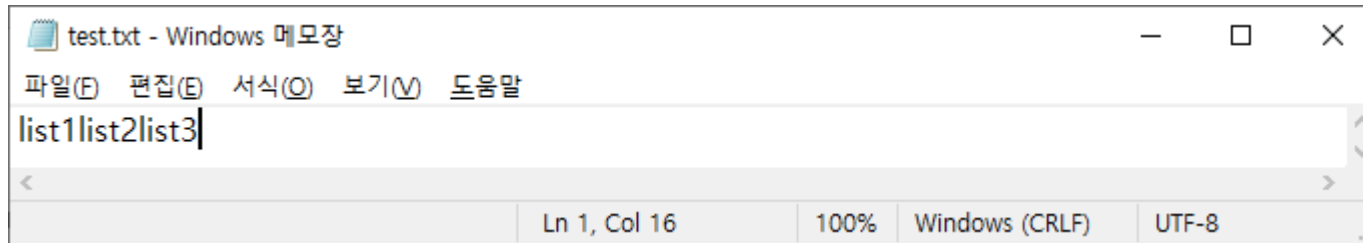
```
f = open("test03.txt", 'w')  
f.write('1 2 3 4 5 6 7 8 9 10')  
f.close()
```

```
with open("test03.txt", 'w') as f:  
    f.write('1 2 3 4 5 6 7 8 9 10')
```

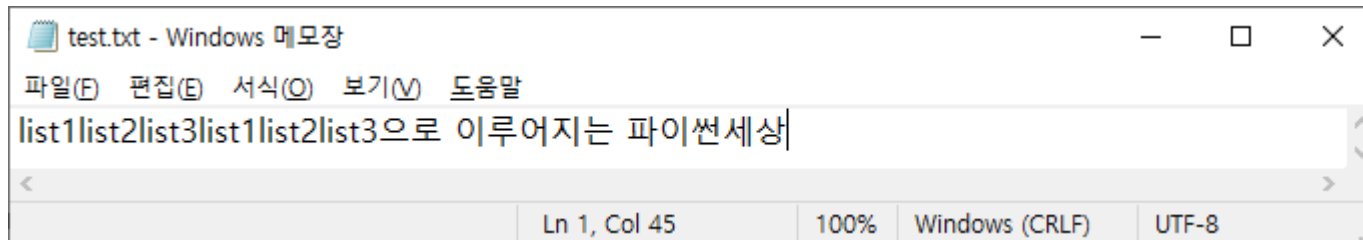
02. 파일 다루기

■ 파일에 내용 추가하기 - 파일 열기 모드 'a'로 새로운 글 추가하기

```
f = open('filename.txt', 'a')
```



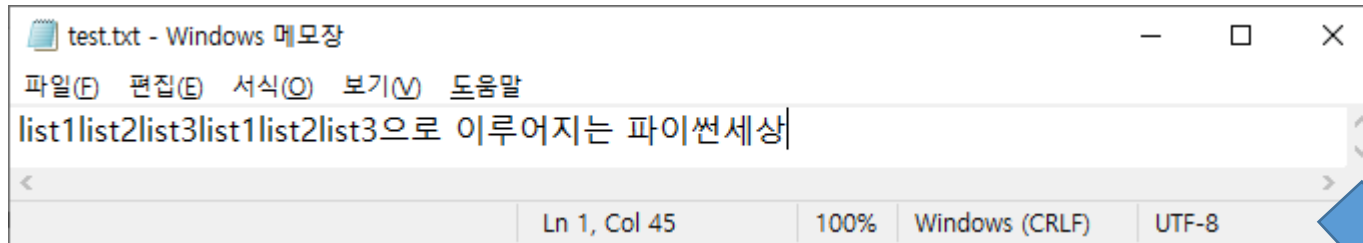
```
data = '으로 이루어지는 파이썬세상'  
f = open("test.txt", 'a')  
f.write(data)  
f.close()
```



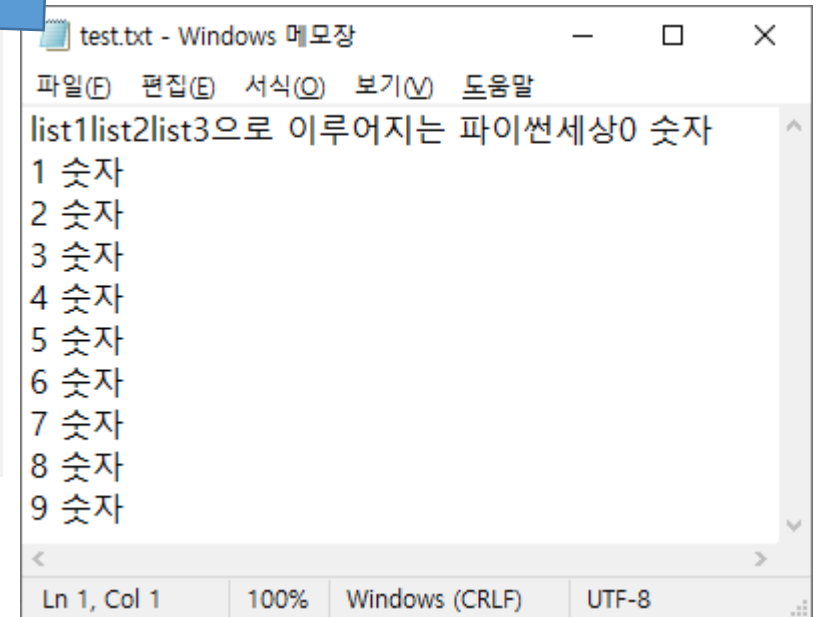
02. 파일 다루기

■ 파일에 내용 추가하기 - 반복문을 이용하여 내용 추가

```
f = open('filename.txt', 'a')
```



```
n = 0
f = open("test.txt", 'at')
while n < 10:
    data = "%d 숫자\n"%n
    f.write(data)
    a = a + 1
f.close()
```

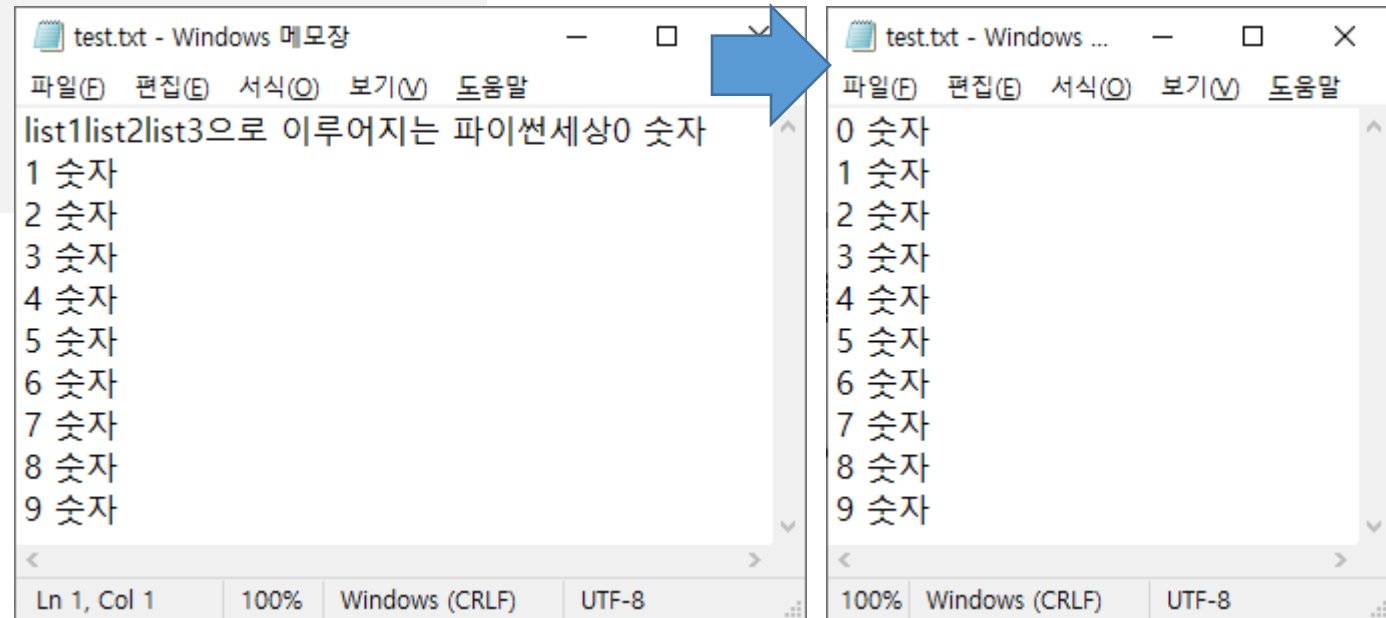


02. 파일 다루기

- 파일에 내용 추가하기 - 이전 내용을 지우고 반복문을 이용하여 내용 추가

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

```
n = 0
f = open("test.txt", 'w', encoding='utf-8')
while n < 10:
    data = "%d 숫자\n"%n
    f.write(data)
    n = n + 1
f.close()
```



PRACTICE

새로운 텍스트 파일 `loop.txt`를 생성하되, 이미 파일이 있는 경우 기존 파일의 내용을 추가한다.
1부터 100까지 한 칸씩만 띄우고 모두 한 줄에 저장한다.

PRACTICE

새로운 텍스트 파일 loop.txt를 생성하되, 이미 파일이 있는 경우 기존 파일의 내용을 추가한다.
1부터 100까지 한 칸씩만 띄우고 모두 한 줄에 저장한다.

```
f = open('open.txt', 'a')
i = 1
while i < 101:
    a = str(i) + " "
    f.write(a)
    i = i+1
f.close()
```

```
with open('open.txt', 'a') as f:
    i = 1
    while i < 101:
        a = str(i) + " "
        f.write(a)
        i = i+1
```

PRACTICE

`practice.txt`를 만들어 "제1의아해가무섭다" 부터 "제5의아해가무섭다" 까지 순서대로 한 줄에 하나씩 입력하여 저장하시오.

제1의아해가무섭다

제2의아해가무섭다

제3의아해가무섭다

제4의아해가무섭다

제5의아해가무섭다

PRACTICE

practice.txt를 만들어 "제1의아해가무섭다" 부터 "제5의아해가무섭다" 까지 순서대로 한 줄에 하나씩 입력하여 저장하시오.

```
f = open('practice.txt', 'a', encoding="utf-8")
for i in range(5):
    f.write("제%d의아해가무섭다\n"%(i+1))
f.close()
```

파이썬에서 utf-8로 txt를 저장하기 위해서는 `encoding="utf-8"`를 사용한다.

```
with open('practice.txt', 'a', encoding="utf-8") as f:
    for i in range(5):
        f.write("제%d의아해가무섭다\n"%(i+1))
```

02. 파일 다루기

■ 파일 읽기 : python code가 저장된 위치로 가기

The following table represents the data shown in the Windows Explorer window in the third screenshot:

이름	수정한 날짜	유형	크기
.idea	2019-11-19 오전 4:49	파일 폴더	
test_flake	2019-11-19 오전 6:03	JetBrains PyChar...	1 KB

02. 파일 다루기

- 파일 쓰기 : python code가 저장된 위치에 TEXT 파일 저장하기

practice.txt

```
with open('practice.txt', 'w', encoding="utf-8") as f:
    for i in range(5):
        f.write("제%d의아해가무섭다\n"%(i+1))
```


02. 파일 다루기

■ 텍스트 파일 읽기

- `read()` : 파일글자 읽기 / 모드 'r'

변수명.`read()`

```
f = open("practice.txt", 'r', encoding="utf-8")
a = f.read()
print(a)
f.close()
```

제1의아해가무섭다
제2의아해가무섭다
제3의아해가무섭다
제4의아해가무섭다
제5의아해가무섭다

02. 파일 다루기

■ readline() / readlines()

- readlines() : 한줄씩 읽어 리스트로 반환하기

변수명.readlines()

```
f = open('practice.txt', 'r', encoding='utf-8')
a = f.readlines()    #해당위치에서 전체내용 행렬로 리스트 부르기
print(type(a))
print(a)
f.close()
```

```
<class 'list'>
['제1의아해가무섭다\n', '제2의아해가무섭다\n', '제3의아해가무섭다\n', '제4의아해가무섭다\n', '제5의아해가무섭다\n']
```

02. 파일 다루기

■ readline() / readlines()

- readline() : 한줄씩 부르기

변수명.readline()

```
f = open('practice.txt', 'r', encoding='utf-8')
a = f.readline()      #해당위치에서 한줄씩 부르기
print(a, end="")
a = f.readline()
print(a, end="")
a = f.readline()
print(a, end="")
a = f.readline()
print(a, end="")
a = f.readline()
print(a, end="")
a = f.readline()
print(a, end="")
f.close()
```

기존txt파일에 \n이 포함되어 있어 print에 포함
된 줄바꿈이 적용되면 중복이 됨

제1의아해가무섭다
제2의아해가무섭다
제3의아해가무섭다
제4의아해가무섭다
제5의아해가무섭다

02. 파일 다루기

■ replace()

- replace : 바꾸기

변수명.**replace**("찾을값","바꿀값")

```
f = open('practice.txt', 'r', encoding='utf-8')
i=0
while True:
    line = f.readline()
    if not line:
        break
    print(line.replace("\n", ""))
    i=i+1

f.close()
```

제1의아해가무섭다
제2의아해가무섭다
제3의아해가무섭다
제4의아해가무섭다
제5의아해가무섭다

- .replace("찾을값","바꿀값",[바꿀횟수])

02. 파일 다루기

■ 파일 안 글자의 통계정보 출력하기

변수명.**split()** / **len()**

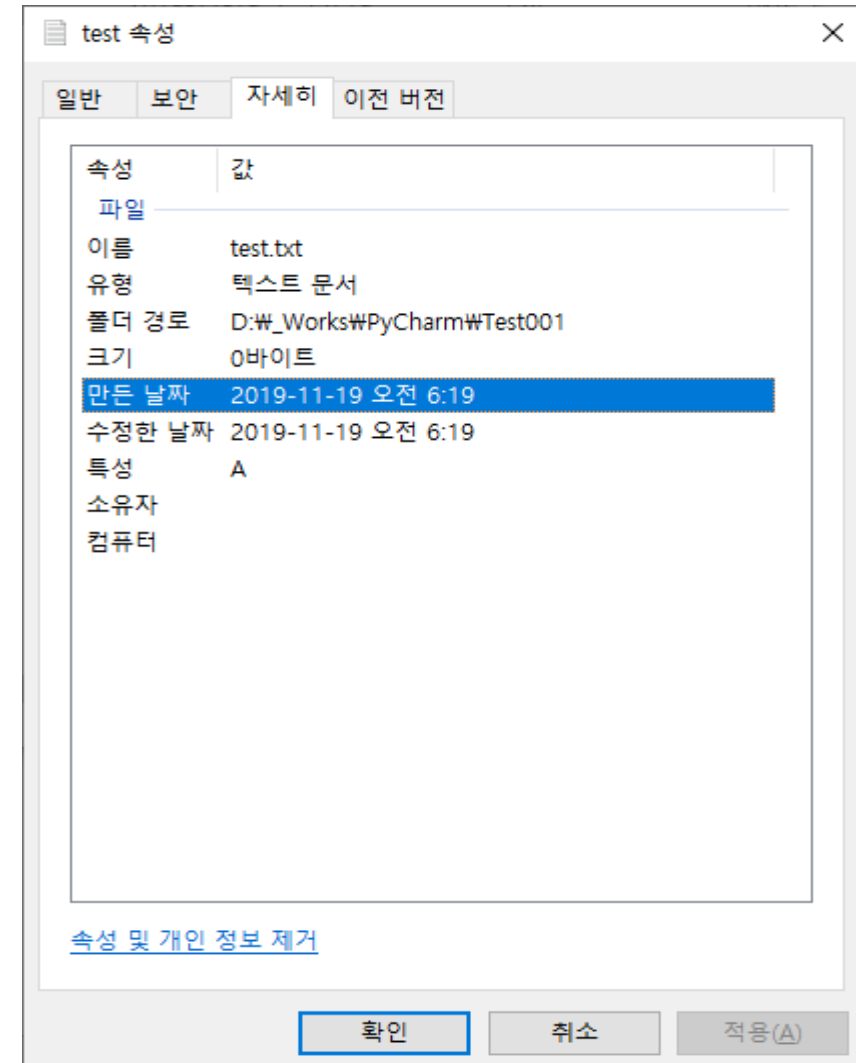
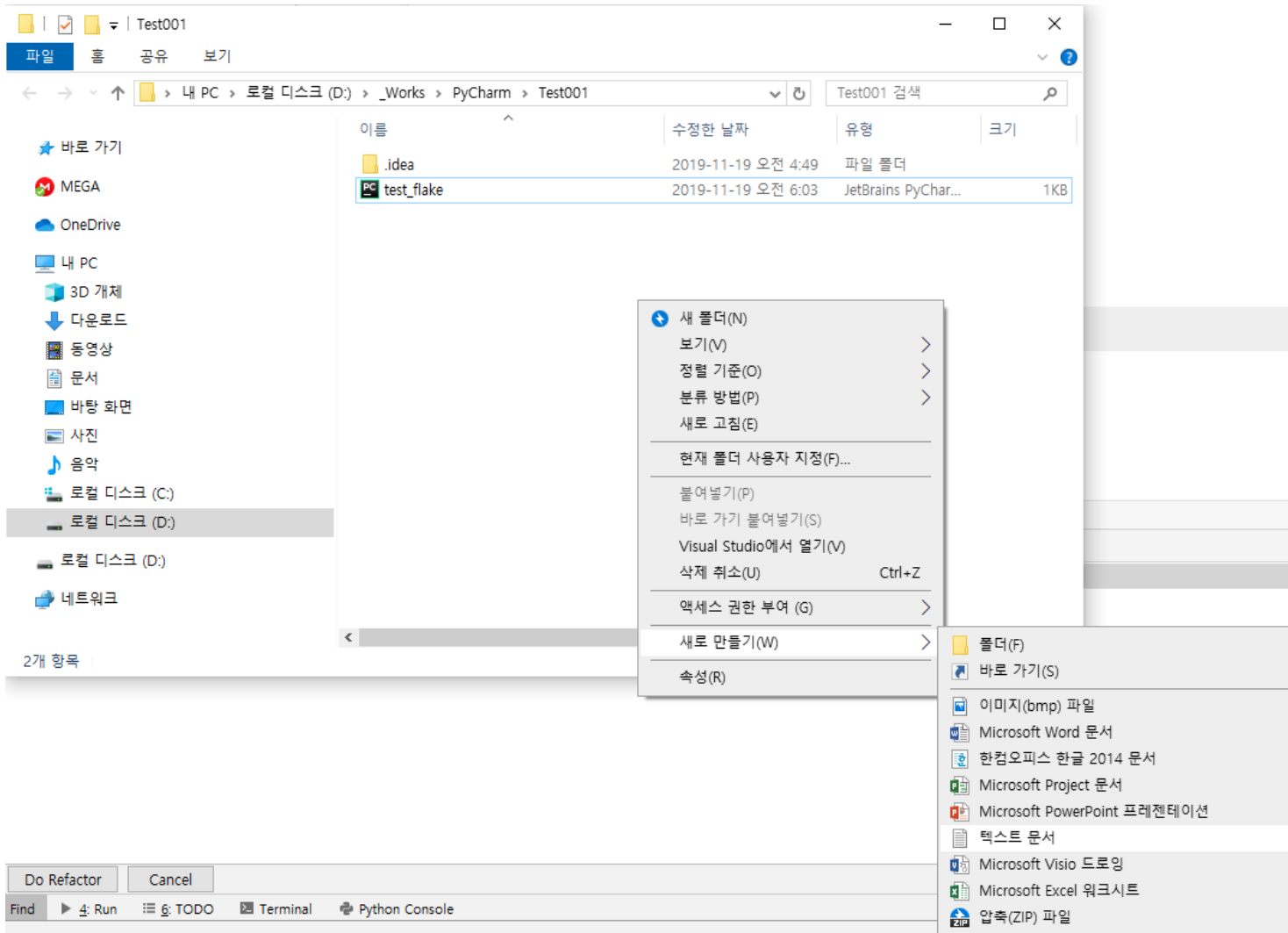
```
f = open('practice.txt', 'r', encoding='utf-8')
contents = f.read()
word_list = contents.split(" ")
line_list = contents.split("\n")

print("총글자수:", len(contents))
print("총단어의수:", len(word_list))
print("총줄의수:", len(line_list))
```

```
총글자수: 50
총단어의수: 1
총줄의수: 6
```

02. 파일 다루기

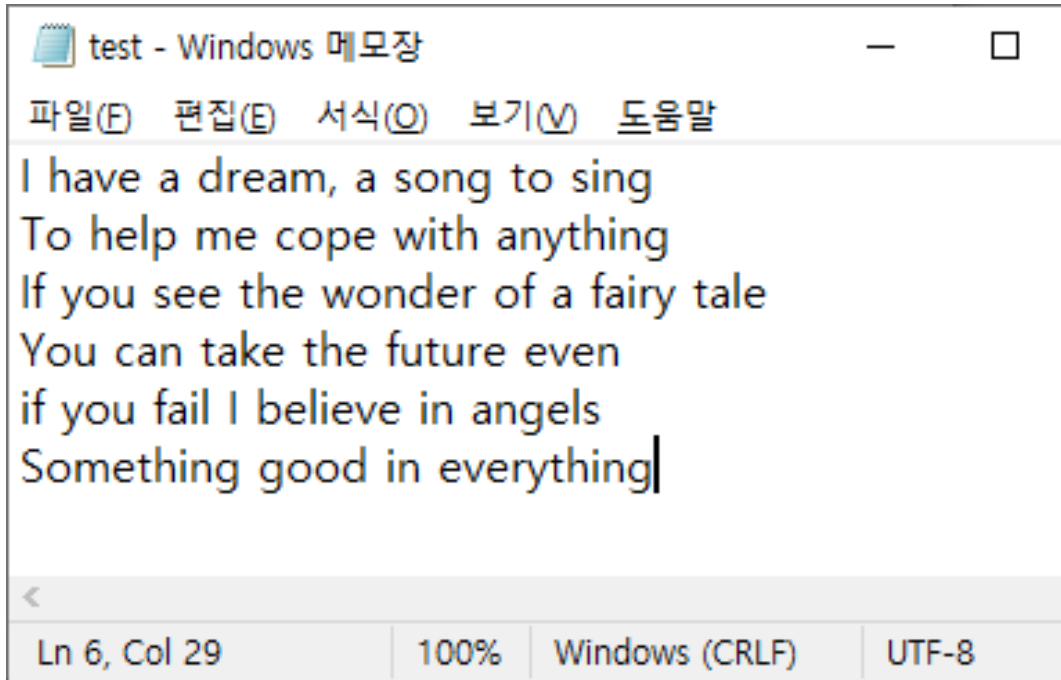
■ 파일 읽기 : python code가 저장된 위치에 TEXT 파일 저장하기



02. 파일 다루기

- 파일 읽기 : python code가 저장된 위치에 TEXT 파일 저장하기

test.txt



```
test - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말
I have a dream, a song to sing
To help me cope with anything
If you see the wonder of a fairy tale
You can take the future even
if you fail I believe in angels
Something good in everything|
<
Ln 6, Col 29 100% Windows (CRLF) UTF-8
```

```
f = open('test.txt', 'w', encoding="utf-8")
f.write("I have a dream, a song to sing\n\
To help me cope with anything\n\
If you see the wonder of a fairy tale\n\
You can take the future even\n\
if you fail I believe in angels\n\
Something good in everything\n")
f.close()
```

02. 파일 다루기

■ 텍스트 파일 읽기

```
f = open("test.txt", 'r')
contents = f.read()
print(contents)
f.close()
```

```
I have a dream, a song to sing
To help me cope with anything
If you see the wonder of a fairy tale
You can take the future even
if you fail I believe in angels
Something good in everything
```


02. 파일 다루기

■ 파일 읽기 with문과 함께 사용하기

- with문은 들여쓰기를 사용해 들여쓰기가 있는 코드에서는 open() 함수가 유지되고, 들여쓰기가 종료되면 open() 함수도 끝나는 방식

```
with open("test.txt", 'r') as my_file:  
    contents = my_file.read()  
    print(type(contents))  
    print(contents)
```

```
<class 'str'>  
I have a dream, a song to sing  
To help me cope with anything  
If you see the wonder of a fairy tale  
You can take the future even  
if you fail I believe in angels  
Something good in everything
```

02. 파일 다루기

■ 파일 읽기 **with**문과 함께 사용하기

```
with open("test.txt", 'r') as my_file:  
    contents = my_file.read()  
    print(contents)
```



```
my_file = open("test.txt", 'r')  
contents = my_file.read()  
print(contents)  
my_file.close()
```

01. 예외 처리

■ 예외 처리 구문 : `try -except`문

```
try:  
    f = open('python.txt', 'r')  
except:  
    print("파일을 읽을때 오류가 발생했습니다.")
```

파일을 읽을때 오류가 발생했습니다.

```
try:  
    f = open('python.txt', 'r')  
except FileNotFoundError:  
    print("파일을 찾을 수 없습니다.")
```

파일을 읽을때 오류가 발생했습니다.

01. 예외 처리

■ 예외 처리 구문 : `try -except-else`문

```
try:
    f = open('python.txt', 'r')
except FileNotFoundError:
    print("파일을 찾을 수 없습니다.")
else:
    a = f.read()
    print(a)
    f.close()
```

python.txt

Hello python!

파일이 없을 때

파일을 찾을 수 없습니다.

파일을 읽어 들였을 때(예외가 발생하지 않았을때)

Hello python!

01. 예외 처리

■ 예외 처리 구문 : try -except-else-finally문

```
try:
    f = open('python.txt', 'r')
except FileNotFoundError:
    pass
else:
    a = f.read()
    print(a)
    f.close()
finally:
    print("작업을 모두 마쳤습니다.")
```

python.txt

Hello python!

파일이 없을 때

작업을 모두 마쳤습니다.

파일을 읽어 들였을 때(예외가 발생하지 않았을때)

Hello python!
작업을 모두 마쳤습니다.

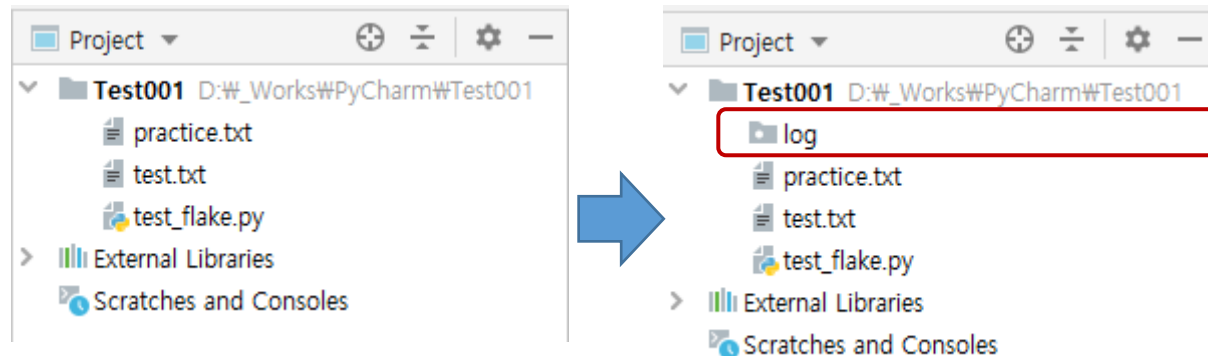
02. 파일 다루기

■ 파일쓰기 : 디렉터리 만들기

- 파이썬으로는 파일만 다루는 것이 아니라, 디렉터리도 함께 다룰 수 있다.
- os 모듈을 사용하면 디렉터리를 쉽게 만들 수 있다.

mkdir()

```
import os  
os.mkdir("log")
```



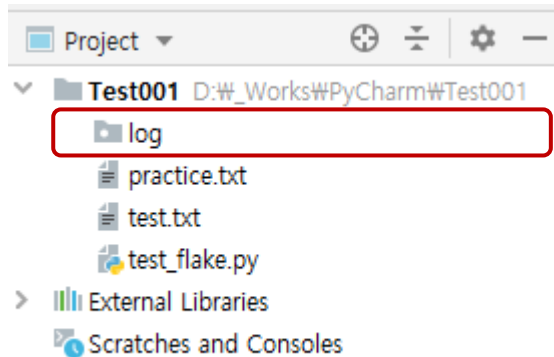
02. 파일 다루기

■ 파일쓰기 : 디렉터리 만들기

- 기존에 생성된 폴더를 생성하려고 하면 에러 발생

`mkdir()`

```
import os  
os.mkdir("log")
```



Traceback (most recent call last):
 File "D:_Works\PyCharm\Test001\test.py", line 2, in <module>
 os.mkdir("log")
FileExistsError: [WinError 183] 파일이 이미 있으므로 만들 수 없습니다:
'log'

02. 파일 다루기

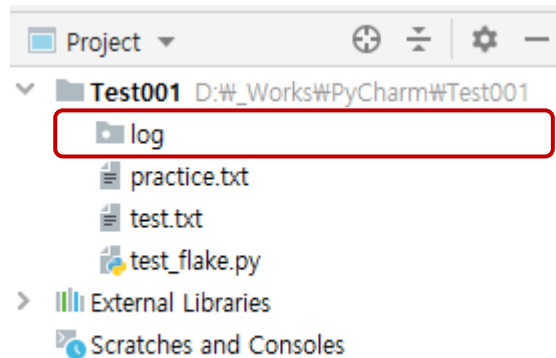
■ 파일쓰기 : 디렉터리 만들기

- 기존에 생성된 폴더가 있는지 확인하고 폴더 생성

`os.path.isdir()`

```
import os
os.mkdir("log")

if not os.path.isdir("log"):
    os.mkdir("log")
```



02. 파일 다루기

■ 파이썬 객체를 그대로 저장하는 pickle

- 자료형(리스트, 튜플, 딕셔너리 등)을 파일에 저장하면 발생하는 일

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

```
f.write(a)
```

```
Traceback (most recent call last):
```

```
File "<pyshell#27>", line 1, in <module>
```

```
f.write(a)
```

```
TypeError: write() argument must be str, not list
```

→ Pickle을 이용하면 자료형 그대로 저장할 수 있음

02. 파일 다루기

pickle로 저장하기

```
import pickle
a = {1:"a", 2:"b", 3:"c"}; b = [1, 2, 3, 4, 5]
with open('picklefile.bin', 'wb') as f:
    pickle.dump(a, f)
    pickle.dump(b, f)
```

pickle로 불러오기

```
import pickle
with open('picklefile.bin', 'rb') as f:
    data = pickle.load(f)
    print(data)
```

02. 파일 다루기

■ pickle 모듈

- 메모리에 로딩된 객체를 영속화 할 수 있도록 지원

dump()

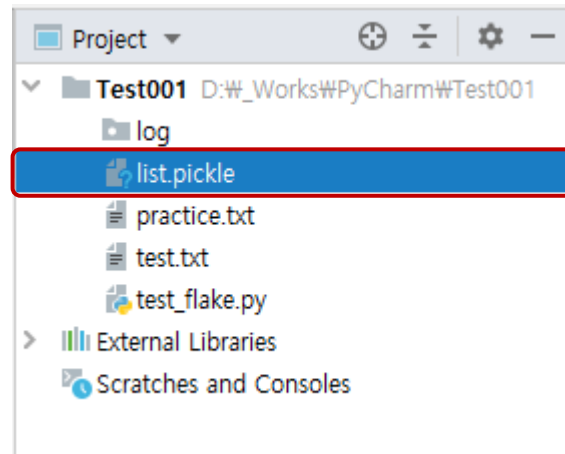
```
import pickle
```

```
f = open("list.pickle", 'wb') # pickle에 파일 저장시 'wb' 옵션 사용
```

```
test = [1, 2, 3, 4]
```

```
pickle.dump(test, f)
```

```
f.close()
```



02. 파일 다루기

■ pickle 모듈

- 메모리에 로딩된 객체를 영속화 할 수 있도록 지원

`pickle.load()`

```
import pickle
```

```
f = open("list.pickle", 'rb')    # pickle에 저장된 파일로드시 'rb'옵션 사용
test_pickle = pickle.load(f)
print(test_pickle)
f.close()
```

```
[1, 2, 3, 4]
```

PRACTICE

텍스트로 데미지를 입력받아 체력이 낮아지는 프로그램을 만들고 'save'라고 입력시 남은 체력을 save.txt에 저장하라. (초기 체력값은 300이다)

```
현재 체력은 300 입니다.  
데미지를 몇 입었습니까? : 10  
체력이 290 남았습니다.  
데미지를 몇 입었습니까? : 100  
체력이 190 남았습니다.  
데미지를 몇 입었습니까? : 50  
체력이 140 남았습니다.  
데미지를 몇 입었습니까? : save
```

PRACTICE

텍스트로 데미지를 입력받아 체력이 낮아지는 프로그램을 만들고 'save'라고 입력시 남은 체력을 save.txt에 저장하라. (초기 체력값은 300이다)

```
hp = 300
hit = ""
print("현재 체력은 %d 입니다."%hp)
while hit != "save" and hp > 0:
    hit = input("데미지를 몇 입었습니까 : ")
    if hit == "save":
        f = open('save.txt', 'w')
        f.write(str(hp))
        f.close()
    else:
        hit = int(hit)
        hp = hp - hit
        print("체력이 %d 남았습니다."%hp)
```

PRACTICE

위의 프로그램을 수정하여 프로그램을 실행시키면 이전에 저장한 체력에서 시작하도록 파일 읽어 오도록 프로그램을 수정하라(단, 파일이 없을 경우 "파일을 찾을 수 없습니다."를 출력)

세이브된 파일을 불러오는중...

현재 체력은 140 입니다.

데미지를 몇 입었습니까? : 30

체력이 110 남았습니다.

데미지를 몇 입었습니까? : 110

체력이 0 남았습니다.

PRACTICE

위의 프로그램을 수정하여 프로그램을 실행시키면 이전에 저장한 체력에서 시작하도록 파일 읽어 오도록 프로그램을 수정하라(단, 파일이 없을 경우 "파일을 찾을 수 없습니다."를 출력)

```
hp = 300
hit = ""
try:
    f = open('save.txt', 'r')
    hp = int(f.read())
    f.close()
    print("세이브 된 파일을 불러오는중...")
except:
    print("세이브 된 파일을 찾을 수 없습니다.")

print("현재 체력은 %d 입니다." % hp) # 이후부터 앞 부분과 동일
while hit != "save" and hp > 0:
    hit = input("데미지를 몇 입었습니까 : ")
    if hit == "save":
        f = open('save.txt', 'w')
        f.write(str(hp))
        f.close()
    else:
        hit = int(hit)
        hp = hp - hit
        print("체력이 %d 남았습니다." % hp)
```


PRACTICE

오류가 났을 때 오류의 내용을 지속적으로 ErrorLog.txt에 저장하는 프로그램 작성해 보자

리스트 `nation = ["한국", "미국", "일본", "중국", "러시아", "베트남"]` 가운데 나라이름이 있으면 그나라의 인덱스를 출력하고 나라이름이 없으면 오류를 `ErrorLog.txt`에 저장해보자 ('q'를 입력하면 종료한다.)

에러의 종류를 출력하고자 할 때에는 `[except 에러명 as 출력변수]`를 이용한다.

```
try... except ValueError as e
print(e)
```

나라이름을 입력 하세요 : 한국

0

나라이름을 입력 하세요 : 러시아

4

나라이름을 입력 하세요 : 베트남

5

나라이름을 입력 하세요 : 천조국

'천조국' in not in list 국가는 리스트에 존재하지 않습니다. 로그기록

나라이름을 입력 하세요 : 쌀국

'쌀국' in not in list 국가는 리스트에 존재하지 않습니다. 로그기록

나라이름을 입력 하세요 : q

'q' in not in list 국가는 리스트에 존재하지 않습니다. 로그기록

ErrorLog - 메모장

파일(E) 편집(E) 서식(O) 보기(V) 도움말(H)

천조국

쌀국

q

PRACTICE

오류가 났을 때 오류의 내용을 지속적으로 log.txt에 저장하는 프로그램 작성해 보자

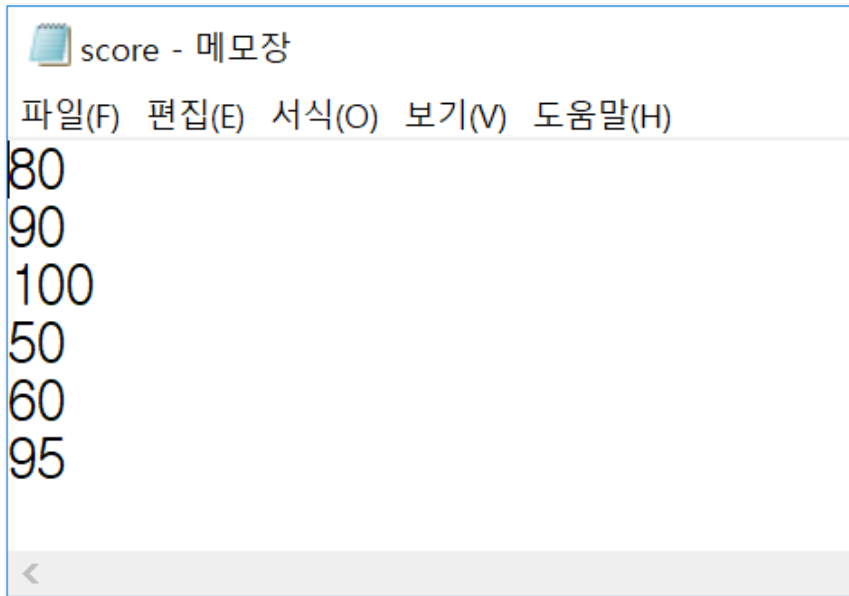
리스트 `nation = ["한국", "미국", "일본", "중국", "러시아", "베트남"]` 가운데 나라이름이 있으면 그나라의 인덱스를 출력하고 나라이름이 없으면 오류를 `log.txt`에 저장해보자 ('q'를 입력하면 종료한다.)

```
nation = ["한국", "미국", "일본", "중국", "러시아", "베트남"]
a = "a"
while a:
    if a == "q":
        break

    a = input("나라 이름을 입력하세요 : ")
    try:
        print(nation.index(a))
    except ValueError as e:
        f = open("ErrorLog.txt", "a")
        f.write("%s\n"%a)
        f.close()
        print("%s 국가는 리스트에 존재하지 않습니다. 로그기록"%e)
```

PRACTICE

텍스트파일(score.txt)을 생성하고 텍스트파일에 50~100사이의 임의의 숫자를 6개 저장해보자



PRACTICE

텍스트파일(score.txt)을 생성하고 텍스트파일에 50~100사이의 임의의 숫자를 6개 저장해보자

```
import random
with open('score.txt', 'w', encoding='utf-8') as score:
    for n in range(6) :
        score.write(str(random.randint(50,100))+"\n")
```

PRACTICE

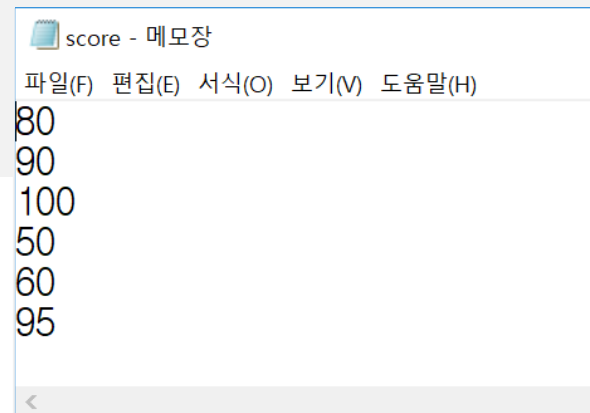
텍스트파일(score.txt)에 있는 숫자를 모두 불러 총합과 평균을 구하여 보자

map()

리스트 전체의 Type를 바꿀때는 반복문으로도 가능하지만 map함수 사용가능
map()함수는 반복가능한 객체들을 모두 펼쳐 한번에 바꿈

```
리스트이름 = list(map(int, 리스트이름))  
for i in range(len(리스트)):  
    리스트[i] = int(리스트[i])
```

전체의 합은 475입니다.
전체의 평균은 79 입니다.



PRACTICE

텍스트파일(score.txt)에 있는 숫자를 모두 불러 총합과 평균을 구하여 보자

```
f = open("score.txt", 'r')
score = f.readlines()           #읽어들여 리스트로 바꾸기
score = list(map(int, score))   #전부 정수형으로 바꾸기 (for 반복 가능)

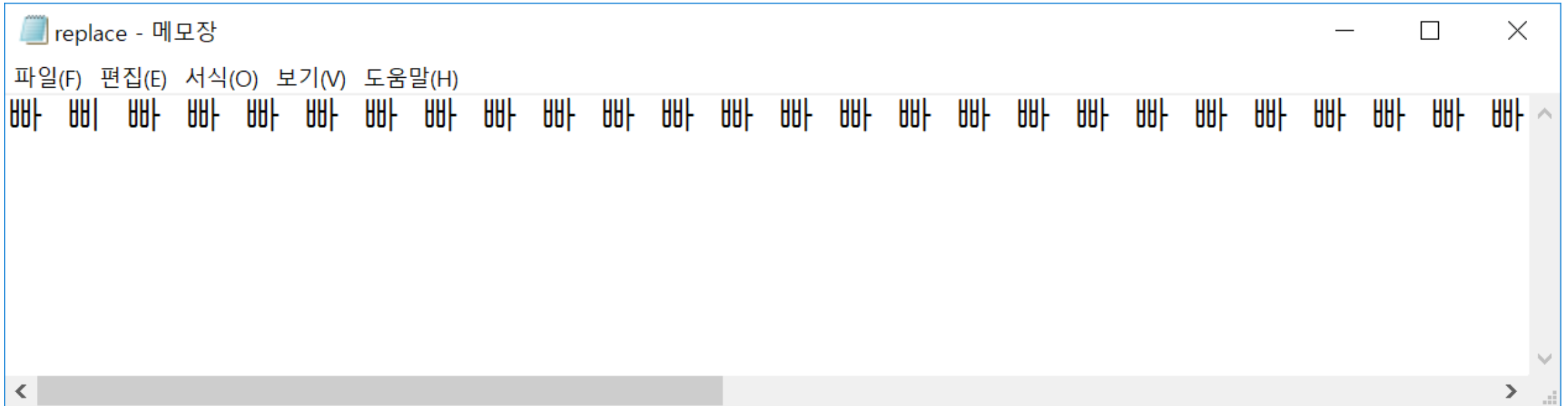
score_sum = 0
for i in score:                 #합계
    score_sum = score_sum + i

score_average = score_sum / len(score)

print("전체 합은 %d 입니다."%score_sum)
print("전체 평균은 %d 입니다."%score_average)
```

PRACTICE

replace.txt파일을 생성하고 "삐" 와 "빠"를 랜덤으로 100개 생성하여 저장하라



PRACTICE

replace.txt파일을 생성하고 "빠" 와 "파"를 랜덤으로 100개 생성하여 저장하라

```
import random
ch = ["빠", "파"]
with open('replace.txt', 'w', encoding='utf-8') as pipa:
    for n in range(100) :
        pipa.write(random.choice(ch)+" ")
```


PRACTICE

replace.txt파일을 읽어들이여 파일에 숨어있는 "뺑"를 "뺑"로 바꾸어라

```
f = open('replace.txt', 'r')
letter = f.read()
f.close()

letter = letter.replace("뺑", "뺑")

f = open('replace.txt', 'w')
f.write(letter)
f.close()
```

몫과 나머지 리턴 divmod(x, y)

9//2

4

9%2

1

a = divmod(9,2)

a

(4, 1) # 튜플

```
a = divmod(9,2)
print(a)
```

10진수 → 16진수

hex(15)

'0xf'

hex(169)

'0xa9'

```
num = 15
print(hex(num))
num = 169
print(hex(num))
```

자료형을 문자형으로 str(x)

```
str(12345)
```

```
'12345'
```

```
str(3.14)
```

```
'3.14'
```

```
str('abc')
```

```
'abc'
```

```
print(str(12345))
```

```
print(str(3.14))
```

```
print(str('abc'))
```

아스키코드→문자 chr(ASCII), 문자→아스키코드 ord(c)

```
chr(98), ord('P')
```

```
('b', 80)
```

반복가능한(iterable) 객체의 자료 전체 요소 참 확인 all(a)

```
a = [2,3,4]
```

```
b = [0,1,2,3,4]
```

```
c = ['a', '', 'b']
```

```
print(all(a))          # 리스트의 모든 요소 값 2,3,4에 0이 없음
```

```
True
```

```
print(all(b))          # 리스트의 요소 값에 0이 존재함, 논리값 0==False, 1==True
```

```
False
```

```
print(all(c))          # 문자열 리스트에 빈값 '' 이 존재함
```

```
False
```

반복가능한(iterable) 객체의 자료 최소 요소 참 확인 any(a)

```
a = [2, 0, 'abc', '', '3']
```

```
b = [0, 0, "", '']
```

```
print(any(a))
```

```
True
```

```
print(any(b))
```

```
False
```

객체의 전체 개수 확인 len(a)

```
a='apple'
```

```
print(len(a))
```

```
5
```

```
b = [1,2,3,"1","2","3"]
```

```
print(len(b))
```

```
6
```

리스트, 튜플 형태 반환 list(a), tuple(a)

```
print(list("Python"), tuple("Python"))  
(['P', 'y', 't', 'h', 'o', 'n'], ('P', 'y', 't', 'h', 'o', 'n'))
```

최대값과 최소값 max(a), min(a)

```
print(max("abcdxyz"), min(0,1,2,3,9,10))  
( 'z', 0)
```

정렬 sorted(a)

```
print(sorted((4,15,23,12,32)))      # 숫자형 자료는 튜플 자료만 정렬 가능  
[4, 12, 15, 23, 32]  
print(sorted("abc123xyz890"))      # 문자형 자료의 정렬  
['0', '1', '2', '3', '8', '9', 'a', 'b', 'c', 'x', 'y', 'z']
```

객체 요소 간의 짝 구성 zip(a, ...)

```
a=zip(['영희', '철수', '태령'], ['여', '남', '남'])
print(a)
<zip object at 0x0000028191ED5788>
print(list(a))
[('영희', '여'), ('철수', '남'), ('태령', '남')]
print(zip([1,2,3], ["A", "B", "C"], ["가", "나", "다"]))
<zip object at 0x00CACD50>
print(list(zip([1,2,3], ["A", "B", "C"], ["가", "나", "다"])))
[(1, 'A', '가'), (2, 'B', '나'), (3, 'C', '다')]
```

자료 순서와 값 반환 enumerate(a)

```
a = ["a", "b", "c", "d"]
print(list(enumerate(a)))
[(0, 'a'), (1, 'b'), (2, 'c'), (3, 'd')]
b = 'banana'
print(enumerate(b))
<enumerate object at 0x0000028191ED4F78>
print(list(b))
['b', 'a', 'n', 'a', 'n', 'a']
print(list(enumerate(b)))
[(0, 'b'), (1, 'a'), (2, 'n'), (3, 'a'), (4, 'n'), (5, 'a')]
```


조건에 알맞은 객체 반환 filter(def, a)

```
def num(a):  
    return type(a) == int  
a = [1, "a", "가", 5, 9.99, -10]  
print(filter(num, a))  
<filter object at 0x03942D50>  
print(list(filter(num, a)))  
[1, 5, -10]
```

객체의 함수 실행 결과 반환 map(def, a)

```
a = ['1', '2', '3']  
list(map(int, a))  
[1, 2, 3]
```

map() 에 함수 만들어 사용하기

```
def nn(a):  
    b=a*a  
    return b  
list(map(nn, [1,2,3,4,5]))  
[1, 4, 9, 16, 25]
```

숫자 범위의 객체 생성 range()

```
range(1, 10, 2)
```

```
range(1, 10, 2)
```

텍스트 계산기

문자 또는 숫자 수식 입력시 자동 계산 프로그램
단, 결과는 절대값, 반올림 후 아스키 코드로 변환하여 출력하라
만약 알파벳 범위가 아닐 경우 문자형으로 변환하여 출력하라
소문자 아스키코드 65~90, 대문자 아스키코드 97~122

cal()

수식 입력 : 45 + 74.22

w

텍스트 계산기

문자 또는 숫자 수식 입력시 자동 계산 프로그램
단, 결과는 절대값, 반올림 후 아스키 코드로 변환하여 출력하라
만약 알파벳 범위가 아닐 경우 문자형으로 변환하여 출력하라
소문자 아스키코드 65~90, 대문자 아스키코드 97~122

```
def cal():  
    a = eval(input("수식 입력 : ")) # 수식을 입력받아 eval()함수로 계산하기  
    a = abs(a) # 절대값 구하기  
    a = round(a, 0) # 0자리에서 반올림하여 정수 만들기  
    a = int(a) # 정수형으로 바꾸기  
  
    if 90 >= a >= 65 or 122 >= a >= 97:  
        print(chr(a))  
    else:  
        print(str(a))  
  
cal()
```

객체 정보 출력

함수에 리스트를 넣으면 여러 정보를 출력해주는 프로그램을 작성하라

```
simul([1, 3, 5, "A", "b"])
```

자료 요소 참 여부 : True

자료 길이 : 5

자료 중 최대값 : b

정렬 시 자료 순서 : ['1', '3', '5', 'A', 'b']

자료 번호 : [(0, '1'), (1, '3'), (2, '5'), (3, 'A'), (4, 'b')]

객체 정보 출력

함수에 리스트를 넣으면 여러 정보를 출력해주는 프로그램을 작성하라

```
simul([1, 3, 5, "A", "b"])
```

```
자료 요소 참 여부 : True
```

```
자료 길이 : 5
```

```
자료 중 최대값 : 5
```

```
정렬  
자료
```

```
def simul(a):  
    a = list(map(str,a))  
    print("자료 요소 참 여부 :", all(a))  
    print("자료 길이 :", len(a))  
    print("자료 중 최대 값 :", max(a))  
    print("정렬 시 자료 순서 :", sorted(a))  
    print("자료 번호 :", list(enumerate(a)))  
a = [1, 3, 5, "A", "b"]  
simul(a)
```

표준 라이브러리

외장 함수, 내장 라이브러리, 표준 라이브러리 = import 하여 사용

파일과 디렉터리 접근 : sys, os

데이터 파일 저장 : pickle

수학 및 랜덤 : math, random

인터넷 액세스 : webbrowser, urllib

날짜와 시간 : time, datetime

#우리가 배울 라이브러리

난수 발생하기 random 라이브러리



여러 범위에서 난수 발생시키기 `uniform(x, y)`, `randrange(x, y, z)`

```
import random
```

```
random.uniform(1,10)
```

uniform은 범위가 클 때 사용

```
4.489175521943265
```

```
random.randrange(1,10,2)
```

randrange 범위 내 나열된 무작위 숫자 고르기
(range와 용법 같음)

```
5
```

난수 발생하기 random 라이브러리

정해진 리스트 안에서 무작위로 고르기 choice(a), sample(a, x)

```
import random
```

```
lunch = ["Pasta", "Pizza", "Hamburger", "Chinese Cuisine"]
```

```
random.choice(lunch)
```

```
'Hamburger'
```

```
random.sample(lunch, 2)
```

a 객체 안에서 x만큼 구하기

```
['Chinese Cuisine', 'Pasta']
```

난수 발생하기 random 라이브러리

정해진 리스트 안에서 무작위로 고르기 choice(a), sample(a, x)

```
import random
```

```
lunch = ["Pasta", "Pizza", "Hamburger", "Chinese Cuisine"]
```

```
random.choice(lunch)
```

```
'Hamburger'
```

```
random.sample(lunch, 2)
```

a 객체 안에서 x만큼 구하기

```
['Chinese Cuisine', 'Pasta']
```

1~100사이 숫자 중 사용자가 추측하여 맞추는 Up&Down 게임을 만들어라. 문제의 숫자는 컴퓨터 임의대로 정한다.

다양한 수학 기능 지원 math 라이브러리



함수 종류 확인해보기 dir(math)

```
import math
```

```
dir(math)
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos',  
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos',  
'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial',  
'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose',  
'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p',  
'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan',  
'tanh', 'tau', 'trunc']
```

다양한 수학 기능 지원 math 라이브러리



math 라이브러리 중 대표적인 함수

<code>floor(x)</code>	x의 내림 연산	<code>log(x)</code>	로그 연산
<code>sin(x)</code>	$\sin x$ 값 반환	<code>degrees(x)</code>	라디안을 각도 값으로 변환
<code>pi</code>	원주율 반환	<code>e</code>	자연상수(e) 값 반환
<code>trunc(x)</code>	x의 정수부분만 반환	<code>copysign(x,y)</code>	y의 부호를 복사하여 x로
<code>factorial(x)</code>	x의 팩토리얼 값 반환	<code>fsum(a)</code>	값들의 합계 반환
<code>hypot(x,y)</code>	(x,y)까지의 거리 반환	<code>gcd(x,y)</code>	두 수의 최대 공약수

다양한 수학 기능 지원 math 라이브러리

10! 구해보기

```
import math  
  
print(math.factorial(10))  
  
3628800
```

1. math 모듈을 이용하여 지름이 10인 원의 넓이를 구하여라.
2. 원주율 값과 일치하는 라디안 값 정수 x 를 구하여라.

웹브라우저와 인터넷에 관련된 라이브러리 webbrowser,

웹 브라우저에서 특정 사이트 열어보기

```
import webbrowser
```

```
webbrowser.open("http://computing.or.kr")
```

open_url(주소) 는 새창으로 열기

```
True
```

웹에 있는 데이터 이용하기 urllib

urllib 라이브러리에 포함된 request 라이브러리 이용하기

```
import urllib.request
```

```
a = urllib.request.urlopen('http://computing.or.kr')
```

```
a.status
```

```
200
```

웹에 있는 데이터 이용하기 urllib

웹에 표시된 자료 긁어오기 .read()

```
import urllib.request
a = urllib.request.urlopen('http://computing.or.kr')
print(a.read())
... ..
```

서버에 관한 자료 불러오기 .getheaders

```
import urllib.request
a = urllib.request.urlopen("http://computing.or.kr")
a.getheaders()
[('Server', 'nginx'), ... ('Link', '<http://computing.or.kr/>; rel=shortlink')]
```


웹에 있는 데이터 이용하기 urllib

urllib에 있는 다양한 하위 모듈(라이브러리)

`urllib.request` : url을 읽고, 열고, 정보를 얻는다.

`urllib.error` : request에 의해 발생하는 예외(오류)를 포함한다.

`urllib.parse` : URL 구문 분석을 위한 도구를 제공한다.

`urllib.robotparser` : 관리자가 민감한 정보들을 미리 적어둔 `robots.txt`를 분석한다.

파이썬에서 날짜(datetime.date) 라이브러리 이용하기

오늘 날짜 확인하기

```
import datetime
datetime.date.today()
datetime.date(2019, 06, 08)
d = datetime.date.today()    # 객체(d)에 지정해두면 여러방법으로 사용가능
d.year, d.month, d.day, d.max
(2019, 06, 08, datetime.date(9999, 12, 31))
```

```
import datetime
d = datetime.date.today()
print(d)
print(d.year, d.month, d.day, d.max)
```

2019-07-14

2019 7 14 9999-12-31

파이썬에서 날짜(datetime.date) 라이브러리 이용하기

datetime.date.today() 를 다양하게 표현하기

```
d.isoformat()
```

#YYYY-MM-DD 형태로 반환

```
'2019-11-23'
```

```
d.ctime()
```

#날짜와 시간을 출력, 시간은 00:00:00로 초기화

```
'Sat Jun 08 00:00:00 2019'
```

```
d.strftime("%y년 %m월인데 %d일입니다.")
```

#표시 형태대로 출력

```
'19년 11월인데 23일입니다.'
```

```
import datetime
```

```
d = datetime.date.today()
```

```
print(d.isoformat())
```

#YYYY-MM-DD 형태로 반환

```
print(d.ctime())
```

#날짜와 시간을 출력, 시간은 00:00:00로 초기화

2019-11-23

Sat Nov 23 00:00:00 2019

파이썬에서 시간(datetime.time) 라이브러리 이용하기

time 형식의 객체 생성

```
import datetime
datetime.time(7)                # 시간만 입력
datetime.time(7, 0)
a = datetime.time(12, 30, 55)   # a에 시간형태의 시,분,초 입력
```

```
a = datetime.time(12, 30, 55)   # hh:mm:ss 형식 출력
print(a.hour, a.minute, a.second)
print(a.isoformat())
print(a.strftime("%H 시 %M 분 %S 초 입니다."))  # %H, %M, %S로 출력
```

파이썬에서 현재 시간(time)라이브러리 이용하기

time 라이브러리 내 time 함수

```
import time  
time.time()  
1546191371.5416882
```

#1970년부터 지난 초

날짜와 시간 형태로 변환하여 표시하기

```
time.localtime(time.time())  
time.struct_time(tm_year=2019, tm_mon=06, tm_mday=08, tm_hour=2, tm_min=37,  
tm_sec=7, tm_wday=0, tm_yday=365, tm_isdst=0)
```

파이썬에서 현재 시간(time)라이브러리 이용하기

strftime()을 이용하여 문자포맷으로 나타내기

```
a = time.localtime()  
time.strftime('%Y %m %d %c',a)  
'2019 06 08 Sat Jun 08 02:48:07 2019'
```

파이썬에서 시각차 구하기 timedelta

`datetime.datetime(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)`

```
d = datetime.datetime(2000, 8, 15)
d - datetime.timedelta(days=30)
datetime.datetime(2000, 7, 16, 0, 0)
```

파이썬에서 실행속도 제어하기

```
import time

for i in range(10):
    print(i)
    time.sleep(1)
```

파이썬에서 날짜 시간 라이브러리 연습

1. 오늘로부터 1000일 이후의 날짜를 구하라.
2. 우주선이 발사되도록 10부터 하나씩 감소되는 카운트다운 코드를 작성하라.

```
#1
import datetime
d = datetime.date.today() + datetime.timedelta(days=1000)
print(d)
```

```
#2
import time
print("우주선 발사 Countdown")
for i in range(10, 0, -1):
    print(i)
    time.sleep(1)
print("Launch")
```