파이썬 크롤링

파이썬기초-파일입출력

파일객체 생성

- **파일객체 생성**

  파일 객체 = open(파일 이름, 파일 열기 모드)

파일열기모드	설명
r	읽기모드 - 파일을 읽기만 할 때 사용
w	쓰기모드 - 파일에 내용을 쓸 때 사용
a	추가모드 - 파일의 마지막에 새로운 내용을 추가 시킬 때 사용

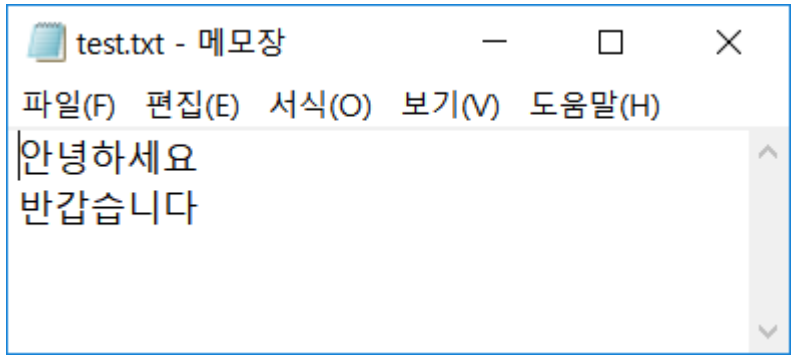
- **파일객체 닫기**

  파일 객체.close()

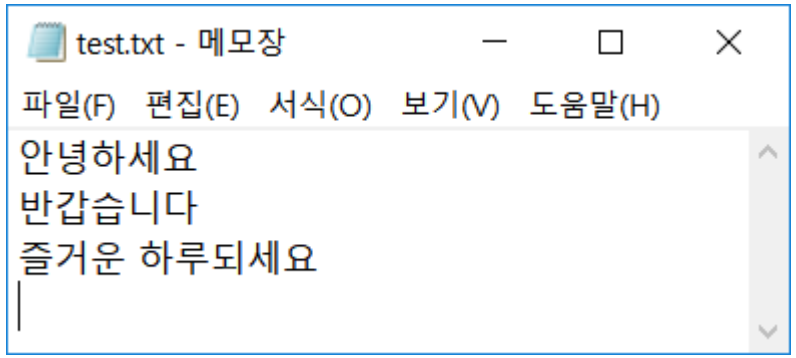
파일 출력

- **파일출력함수 : write()**
  - 파일열기모드 : w

```
f = open("test.txt","w")
inData = input("기 록 내 용 을  입 력 하 세 요 ")
f.write(inData+"\n")
inData = input("기 록 내 용 을  입 력 하 세 요 ")
f.write(inData)
f.close()
```




```
f = open("test.txt","a")
inData = input("기 록 내 용 을  입 력 하 세 요 ")
f.write(inData+"\n")
f.close()
```



파일 입력

- 파일입력함수
  - readline(): 파일의 첫 번째 줄을 읽어 출력하는 경우
  - readlines(): 파일의 모든 라인을 읽어서 각각의 줄을 요소로 갖는 리스트로 리턴
  - f.read(): 파일의 내용 전체를 문자열로 리턴

```
inData = f.readline()
print(inData)

while True:
    inData = f.readline()
    if not inData : break
    print(inData)
```

```
inData = f.readlines()
for line in inData :
    print(line)
```

```
inData = f.read()
print(inData)
```

파이썬 기초

변수와 자료형

- **변수**
  - 값을 저장하는 공간
- **변수명**
  - 숫자, 알파벳, 밑줄 기호(_), 한글 등 다양한 문자를 사용
  - 연산자로 사용되는 기호(+, -, = 등)와 공백 문자()는 사용할 수 없음
  - 숫자로 시작해서는 안됨
  - 시스템 예약어(for, if, while 등)를 이름으로 사용할 수 없음
- **변수 선언 및 초기화**
  - 자료형을 명시하지 않아도 자동으로 결정
  - 자료형 확인
    - Type(변수명)

```
x = 3              #정수형 변수 선언
x = 1.3            #실수형 변수 선언
x = "안녕"         # 문자형 변수 선언
x = True           # bool형 변수 선언
x, y = 10, 20.5
```

Byte 데이터

- 1바이트(0~255사이 코드)로 표현되는 문자 표현
- 문자열에서 사용하는 연산을 거의 제공
  - 인덱싱/슬라이싱, In, Upper(), Split()
- 지원 메소드
  - 바이트->문자열로 변환 : decode(인코딩값)
  - 문자열->바이트로 변환 : encode()

```
1   cstr = "안녕하세요"
2   bstr = cstr.encode()
3
4   print("-"*20)
5   print(type(cstr))
6   print(cstr)
7
8   print("-"*20)
9   print(type(bstr))
10  print(bstr[0])
11  print(str(bstr)[0])
12  print(bstr.decode("ascii", errors="replace"))
13  print(bstr.decode("utf-8"))
```

```
--------------------
<class 'str'>
안녕하세요
--------------------
<class 'bytes'>
236
b
������������
안녕하세요
```

기본 산술 연산자 및 우선순위

우선순위	연산기호	설명
1	()	괄호
2	**	지수 연산
3	+, -	양수, 음수 부호
4	*, /, //, %	곱셈, 나눗셈, 나눗셈(정수만), 나머지
5	+, -	덧셈, 뺄셈

range() 함수

- **range() 함수**
  - 정수 범위를 표현
  - 시작 숫자와 끝 숫자를 지정했을 때 끝 숫자는 범위에 포함되지 않음

• range(숫자)
: 0부터 숫자미만 까지 알려줍니다.
• range(시작숫자,끝숫자)
: 시작 숫자부터 끝숫자 전 까지 알려줍니다.

예제	파라미터 의미	리턴값
range(3)	Stop	0, 1, 2
range(3,6)	Start, Stop	3, 4, 5
range(2,11,2)	Start, Stop, Step	2, 4, 6, 8, 10

문자열

- 문자, 단어 등으로 구성된 문자들의 집합
- 단일라인
  - 큰따옴표("), 작은따옴표(')
- 멀티라인
  - 큰따옴표 3개를 연속("""), 작은따옴표 3개를 연속(''')
- 이스케이프 시퀀스(escape sequence)
  - 프로그래밍할 때 사용할 수 있도록 미리 정의해 둔 문자 조합
  - ₩와 특정 문자를 결합하여 문자를 표시
    - ₩n : 문자열 안에서 줄을 바꿀 때 사용
    - ₩t :문자열 사이에 탭 간격을 줄 때 사용
    - ₩₩ :문자 ₩를 그대로 표현할 때 사용
    - ₩' :작은따옴표(')를 그대로 표현할 때 사용
    - ₩" :큰따옴표(")를 그대로 표현할 때 사용
    - ₩r :캐리지 리턴(줄 바꿈 문자, 현재 커서를 가장 앞으로 이동)
    - ₩f :폼 피드(줄 바꿈 문자, 현재 커서를 다음 줄로 이동)

문자열 연산

```
str1 = "Hello"
str2 = "World!!"

#문자열 더하기
print(str1 + str2)

#문자열 곱하기
print("=" * 20)

#문자열 길이 구하기
print(len(str1))

#문자열 인덱싱
print(str1[0])
print(str1[-1])

#문자열 슬라이싱
print(str1[1:4])    #마지각 항목 제외
print(str1[:3])     #처음부터 추출 마지각 항목 제외
print(str1[3:])     #마지막까지 추출

#문자열의 요소 값은 변경 불가
#str1[0] = "h"      #오류
```

```
HelloWorld!!
====================
5
H
o
ell
Hel
lo
```

문자열 관련 함수

```
name = input("이 름 을  입 력 하 세 요 .")
print(name)

#문 자 열  길 이
print(len(name))

#문 자  갯 수
print(name.count('김 '))

#위 치  찾 기 c
print(name.find('김 '))   #찾 는  문 자 없 으 면  -1
#print(name.index('김 '))  #찾 는  문 자 없 으 면  오 류

#대 소 문 자  변 경
print(name.upper())
print(name.lower())

#공 백  지 우 기
print(name.strip())      #양 쪽  공 백  제 거
print(name.lstrip())     #왼 쪽  공 백  제 거
print(name.rstrip())     #오 른 쪽  공 백  제 거

#문 자 열  바 꾸 기
print(name.replace(" ","."))

#문 자 열  삽 입
name = ",".join(name.strip().replace(" ",""))

#문 자 열  나 누 기
print(name.split(","))
```

```
 a b c
7
0
-1
 A B C
 a b c
a b c
a b c
 a b c
,a,b,c,
['a', 'b', 'c']
```

리스트(list) 생성

- 변수에 값을 저장할 때 [ ](대괄호)로 묶고 각 값을 ,(콤마)로 구분
  - 요소(element) : 리스트에 저장된 각 값

```
#리스트 생성
lst1 = []
lst2 = list()
lst3 = list(range(1, 11, 2))
lst4 = list(range(11, 1, -2))

print(lst1)
print(lst2)
print(lst3)
print(lst4)
```

```
[]
[]
[1, 3, 5, 7, 9]
[11, 9, 7, 5, 3]
```

리스트(list) 연산 / 인덱싱

```
#리스트 연산
lst = lst3 + lst4
print(lst)

lst = lst3 * 3
print(lst)

#리스트 갯수
print("리스트 전체 개수:", len(lst))

#특정항목 개수
print("3의 개수:", lst.count(3))
```

```
[1, 3, 5, 7, 9, 11, 9, 7, 5, 3]
[1, 3, 5, 7, 9, 1, 3, 5, 7, 9, 1, 3, 5, 7, 9]
리스트 전체 개수: 15
3의 개수: 3
```

```
#리스트 인덱싱/ 슬라이싱
lst = list(range(5))
print("리스트 :" , lst)
print("리스트 첫번째 항목 :", lst[0])
print("리스트 마지막 항목 :", lst[-1])

lst[1] = 2
print("리스트 항목 수정 :", lst)

print("2번째에서 3번째 :", lst[2:4])
print("처음부터 3개 항목 :", lst[:3])
print("두번째에서 마지막까지 :", lst[1:])
```

```
리스트 : [0, 1, 2, 3, 4]
리스트 첫번째 항목 : 0
리스트 마지막 항목 : 4
리스트 항목 수정 : [0, 2, 2, 3, 4]
2번째에서 3번째 : [2, 3]
처음부터 3개 항목 : [0, 2, 2]
두번째에서 마지막까지 : [2, 2, 3, 4]
```

리스트 항목 추가/삭제

```python
#리스트 원소 추가/ 삭제
lst = list(range(1,6))
print("리스트 :", lst)

#마지막에 리스트 추가
lst.append(100)
print("리스트 :", lst)

#특정 위치에 추가
lst.insert(1,100)
print("리스트 :", lst)

#리스트 삭제 => 항목이 없으면 오류
lst.remove(100)
print("리스트 :", lst)

#특정 위치 항목 삭제
idx = lst.index(100)  #항목 값 찾기
del lst[idx]
print("리스트 :", lst)

#리스트 마지막 항목 삭제 및 특정 항목 삭제
lst.pop()
print("리스트 :", lst)

lst.pop(0)
print("리스트 :", lst)
```

```
리스트 : [1, 2, 3, 4, 5]
리스트 : [1, 2, 3, 4, 5, 100]
리스트 : [1, 100, 2, 3, 4, 5, 100]
리스트 : [1, 2, 3, 4, 5, 100]
리스트 : [1, 2, 3, 4, 5]
리스트 : [1, 2, 3, 4]
리스트 : [2, 3, 4]
```

파이썬 내장함수

- **sum(), max(), min()**
  - 입력으로 받은 리스트나 튜플의 모든 요소의 합,최대, 최소값를 리턴하는 함수
- **sorted()**
  - 리스트 정렬
  - sorted(리스트,reverse=True)
    - 내림차순정렬

```
list1 = ['a', 'b','c']
list2 = [1, 3, 2]

#합 계
print(sum(list2))
#최 대
print(max(list2))
#최 소 값
print(min(list2))
#정 렬 (
print(sorted(list2))
```

```
6
3
1
[1, 2, 3]
```

튜플(tuple)

- **변경할 수 없는 순서 있는 객체의 집합**
  - 리스트와 비슷하지만 한번 생성되면 변경할 수 없음
  - ()를 사용하여 정의

```
tpl = ()
print("튜플 생성 :", tpl)

tpl = tuple()
print("튜플 생성 :", tpl)

tpl = tuple(range(1,6))
print("튜플 생성 :", tpl)

print("튜플 :" , tpl)
print("튜플 첫번째 항목 :", tpl[0])
print("튜플 마지막 항목 :", tpl[-1])

print("2번째에서 3번째 :", tpl[2:4])
print("처음부터 3개 항목 :", tpl[:3])
print("두번째에서 마지막까지 :", tpl[1:])

#항목 추가
tpl = tpl + (100,)
print("튜플 :" , tpl)

#항목 수정 불가
tpl[1] = 2
print("튜플 항목 수정 :", tpl)
```

```
튜플 생성 : ()
튜플 생성 : ()
튜플 생성 : (1, 2, 3, 4, 5)
튜플 : (1, 2, 3, 4, 5)
튜플 첫번째 항목 : 1
튜플 마지막 항목 : 5
2번째에서 3번째 : (3, 4)
처음부터 3개 항목 : (1, 2, 3)
두번째에서 마지막까지 : (2, 3, 4, 5)
튜플 : (1, 2, 3, 4, 5, 100)
```

```
TypeError                                 Traceback (most recent call last)
<ipython-input-62-ace6ac3ff93c> in <module>()
     21
     22 #항목 수정 불가
---> 23 tpl[1] = 2
     24 print("튜플 항목 수정 :", tpl)

TypeError: 'tuple' object does not support item assignment
```

딕션너리(dictionary)

- Key와 Value라는 것을 한 쌍으로 갖는 자료형
  - {Key1:Value1, Key2:Value2, Key3:Value3, ...}
  - 리스트나 튜플처럼 순차적으로(sequential) 해당 요소값을 구하지 않고 Key를 통해 Value를 얻음
  - Key는 고유한 값이므로 중복되는 Key 값을 설정해 놓으면 하나를 제외한 나머지 것들이 모두 무시

조건문 : if

- if 조건문 뒤에는 반드시 콜론(:)
- if 조건문: 바로 아래 문장부터 if문에 속하는 모든 문장에 들여쓰기(indentation)를 해야 함
- 파이썬에는 다른 언어에 있는 switch 문이 존재하지 않음
- switch 문 기능은 if...elif...elif... 문으로 수행
- pass
  - def 문이나 if 문처럼 코드 블록을 본문으로 갖는 표현에서 본문을 비워 둘 때 사용

```
1  if x < 10:
2      print(x)
3      print("한자리수")
4
5  # 한 라인에서 표현된 if 문
6  if x < 100:  print(x)
```

```
1  x = 10
2  if x < 10:
3      print("한자리수")
4  elif x < 100:
5      print("두자리수")
6  else:
7      print("세자리 이상")
```

```
1  if n < 10:
2      pass
3  else:
4      print(n)
```

비교연산자/논리연산자/in연산자

비교 연산자	의미
x < y	x가 y보다 작다
x > y	x가 y보다 크다
x == y	x와 y가 같다
x != y	x와 y가 같지 않다
x >= y	x가 y보다 크거나 같다
x <= y	x가 y보다 작거나 같다

논리 연산자	의미
and	A and B : A와 B 모두 True일때 True
or	A or B : A 혹은 B가 True일때 True
not	not A : A가 True면 False, False면 True

in	not in
x in 리스트	x not in 리스트
x in 튜플	x not in 튜플
x in 문자열	x not in 문자열

반복문 - for

- **for**
  - 컬렉션으로부터 하나씩 요소(element)를 가져와, 루프 내의 문장들을 실행
  - 리스트, Tuple, 문자열 등의 컬렉션

```
sum = 0
for i in range(11):
    sum += i
print(sum)
```

반복문 while

- **while**
  - while 키워드 다음의 조건식이 참일 경우 계속 while 안의 블럭을 실행

```
sum = 0
i = 1
while i <= 10:
    sum = sum + i
    print(i)
    i = i + 1

print("합계 : " + str(sum))
```

break/continue

- **break 문**
  - 반복문 안에서 루프를 빠져나오기 위해 을 사용
- **continue문**
  - 루프 블럭의 나머지 문장들을 실행하지 않고 다음 루프로 직접 돌아가게 함

```
#반 복 문

for i in range(1, 101):
    if ( i % 2 == 1) : continue
    if ( i > 10 ) : break
    print(i)
```

```
2
4
6
8
10
```

내장함수

함수	설명	예시
abs(x)	어떤 숫자를 입력으로 받았을 때, 그 숫자의 절대값을 돌려주는 함수	abs(-1)
divmod(a, b)	a를 b로 나눈 몫과 나머지를 튜플 형태로 리턴하는 함수	divmod(10, 3) => (3, 1)
float(x)	데이터를 실수로 변환	
int(x)	문자열 형태의 숫자나 소수점이 있는 숫자 등을 정수 형태로 변환	
len(s)	입력값 s의 길이(요소의 전체 개수)를 리턴하는 함수	len("python"), len([1,2,3])
max(iterable)	인수로 반복 가능한 자료형을 입력받아 그 최대값을 리턴하는 함수	max([1, 2, 3]) , max("python")
min(iterable)	인수로 반복 가능한 자료형을 입력받아 그 최소값을 리턴하는 함수	min([1, 2, 3]), min("python")
pow(x, y)	x의 y 제곱한 결과값을 리턴하는 함수	
round(n[,nd])	숫자를 입력받아 반올림 해 주는 함수	
sorted(iterable)	입력값을 정렬한 후 그 결과를 리스트로 리턴하는 함수	
str(object)	문자열 형태로 객체를 변환하여 리턴하는 함수	
sum(iterable)	입력으로 받은 리스트나 튜플의 모든 요소의 합을 리턴하는 함수	

문자열 메소드

- 문자열을 변경이 불가능(immutable)하기 때문에 직접 문자열을 수정하는 방식이 아닌 변경된 다른 문자열을 리턴하는 방식

함수	설명
upper()	대문자로 변경
lower()	소문자로 변경
replace()	문자열 특정 부분을 변경
split()	전달한 문자로 문자열을 나눔, 결과는 리스트 (구분자 포함 안됨)
count()	특정 단어(문자열)의 수를 구함 (없으면 0을 반환)
find()	특정 단어를 찾아 인덱스를 리턴 (없으면 -1을 리턴)
index()	find와 동일하지만 없을 때 예외를 발생시킴
in, not in	사용하면 특정 단어가 있는지 없는지 확인 가능

함수

- 특정 기능을 하나로 묶어서 따로 관리하기 위해 사용
  - 반복되는 내용을 효율적으로 처리
  - 코드의 가독성을 높임
- 함수 구조

```
def 함수명(인자1, 인자2 ...):
        #함수 코드
        return 리턴값
```

```
1  #함수 정의하기
2  def gugu(dan) :
3      for i in range(1,10):
4          print(dan , "x", i,"=", (dan*i))
5
6  dan = int(input("단을 입력하세요=>"))
7  gugu(dan)
```

함수 매개변수와 반환값

반환 값 없음

```python
def cal(num1, num2, op) :
    ans = 0
    if op == "+" : ans = num1 + num2
    elif op == "-" : ans = num1 - num2
    elif op == "*" : ans = num1 * num2
    elif op == "/" : ans = num1 / num2

    print(num1, op, num2, "=", ans)


if __name__ == "__main__" :
    while(True) :

        num1 = int(input("숫자를 입력하세요=>"))
        num2 = int(input("숫자를 입력하세요=>"))
        op = input("연산자를 입력하세요(+, -, *, /)=>")

        if ( op not in ["+", "-", "*", "/"]): break

        cal(num1, num2, op)
```

반환 값 1개

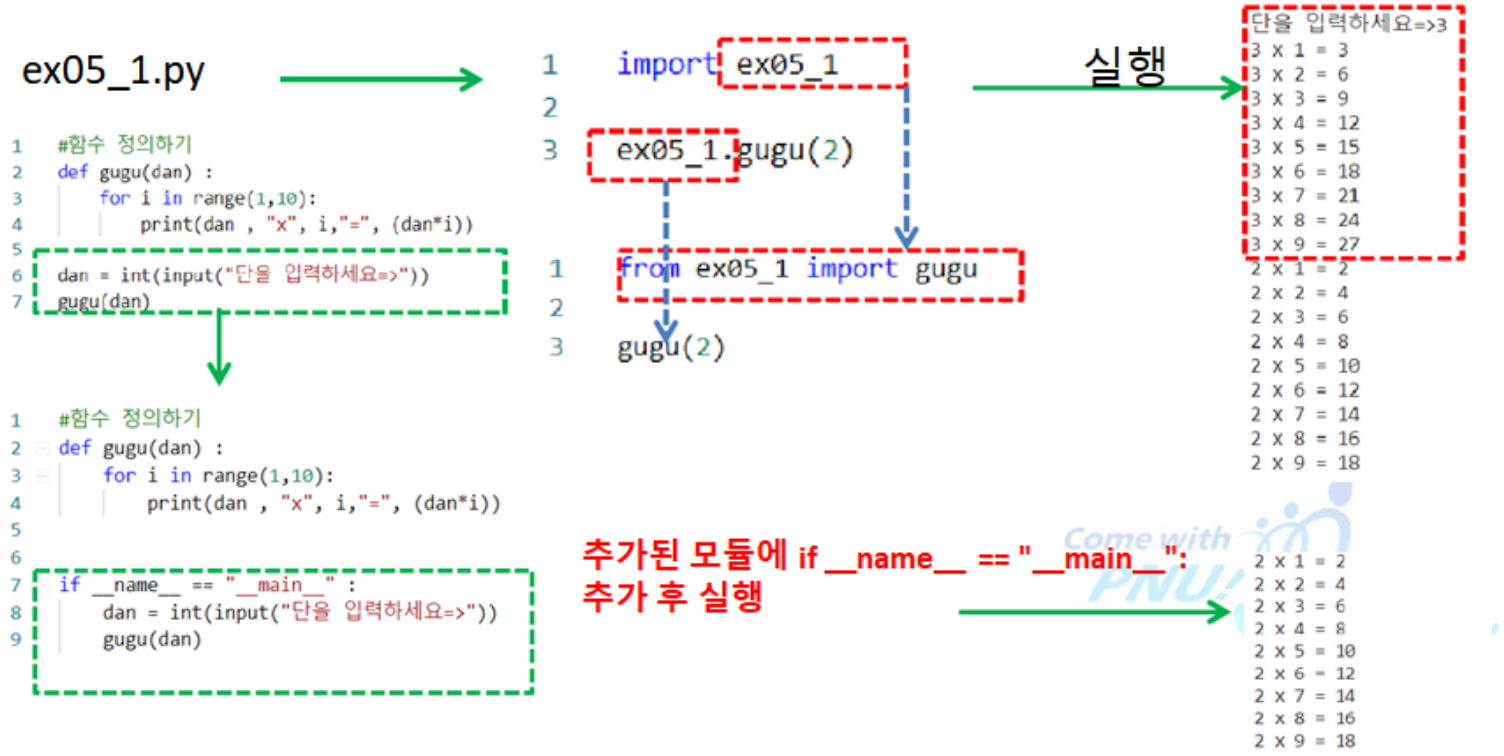
```python
def cal(num1, num2, op) :
    ans = 0
    if op == "+" : ans = num1 + num2
    elif op == "-" : ans = num1 - num2
    elif op == "*" : ans = num1 * num2
    elif op == "/" : ans = num1 / num2

    return ans


if __name__ == "__main__" :
    while(True) :

        num1 = int(input("숫자를 입력하세요=>"))
        num2 = int(input("숫자를 입력하세요=>"))
        op = input("연산자를 입력하세요(+, -, *, /)=>")

        if ( op not in ["+", "-", "*", "/"]): break

        print(num1, op, num2, "=", cal(num1, num2, op))
```

반환 값 2개

```python
def cal(num1, num2, op) :
    ans = 0
    if op == "+" : ans = num1 + num2
    elif op == "-" : ans = num1 - num2
    elif op == "*" : ans = num1 * num2
    elif op == "/" : ans = num1 / num2

    strans = str(num1) + op + str(num2) + "="
    return strans, ans


if __name__ == "__main__" :
    while(True) :

        num1 = int(input("숫자를 입력하세요=>"))
        num2 = int(input("숫자를 입력하세요=>"))
        op = input("연산자를 입력하세요(+, -, *, /)=>")

        if ( op not in ["+", "-", "*", "/"]): break

        strans, ans = cal(num1, num2, op)
        print(strans, ans)
```

모듈

- 함수나 변수 또는 클래스 들을 모아 놓은 파일
- 다른 파이썬 프로그램에서 불러와 사용할 수 있게끔 만들어진 파이썬 파일



실습

- 파일명을 입력 받아서 파일 내용을 문자열로 반환하는 함수를 작성하시오
  - fileToStr()

파이썬 RE

정규 표현식 (Regular Expression)

- 특정한 규칙을 가진 문자열의 패턴을 표현하는 데 사용하는 표현식(Expression)
  - 텍스트에서 특정 문자열을 검색하거나 치환할 때 사용
  - 메타 문자
    - 원래 그 문자가 가진 뜻이 아닌 특별한 용도로 사용하는 문자
      - . ^ $ * + ? { } [ ] ₩ | ( )
    - 정규 표현식에 위 메타 문자를 사용하면 특별한 의미를 갖게 됨

정규 표현식 (Regular Expression)

- **메타문자 []**
  - 문자 클래스(character class)
  - [ ] 사이의 문자들과 매치
    - 예) [abc] : a, b, c 중 한 개의 문자와 매치
  - [ ] 안의 두 문자 사이에 하이픈(-)을 사용
    - 두 문자 사이의 범위(From - To)를 의미
    - 예) [a-zA-Z] : 알파벳 모두, [0-9] : 숫자

- **자주 사용하는 문자 클래스**
  - ₩d : 숫자와 매치, [0-9]와 동일한 표현식
  - ₩D : 숫자가 아닌 것과 매치, [^0-9]와 동일한 표현식
  - ₩s : whitespace 문자와 매치, [₩t₩n₩r₩f₩v]와 동일한 표현식
    맨 앞의 빈 칸은 공백문자(space)를 의미
  - ₩S : whitespace 문자가 아닌 것과 매치
    [^ ₩t₩n₩r₩f₩v]와 동일한 표현식
  - ₩w : 문자+숫자(alphanumeric)와 매치, [a-zA-Z0-9_]와 동일한 표현식
  - ₩W : 문자+숫자(alphanumeric)가 아닌 문자와 매치, [^a-zA-Z0-9_]와 동일한 표현식

정규 표현식 (Regular Expression)

- **메타문자 : ^**
  - 반대(not)
  - 예) [^0-9]라는 정규 표현식은 숫자가 아닌 문자만 매치
- **메타문자 : Dot(.)**
  - 줄바꿈 문자인 ₩n을 제외한 모든 문자와 매치됨을 의미
    - 예) a.b : a와 b라는 문자 사이에 어떤 문자가 들어가도 모두 매치
  - 문자 클래스([]) 내에 Dot(.) 메타 문자가 사용
    - "모든 문자"라는 의미가 아닌 문자 . 그대로를 의미
    - 예) a[.]b :

정규 표현식 (Regular Expression)

- **메타문자 : 반복 (*)**
  - 예) ab*c :
    문자 b가 0부터 무한대로 반복될 수 있다는 의미로 b가 한번도 나오지 않는 ac도 매치
- **메타문자 : 반복 (+)**
  - 예) ab+c : 최소 1번 이상 반복될 때 사용하므로 ac는 매치가 되지 않음
- **메타문자:반복 {m,n}**
  - 반복 횟수를 고정
  - {m, n} 정규식을 사용하면 반복 횟수가 m부터 n까지 매치
  - m 또는 n을 생략하면 m은 0과 동일하며, 생략된 n은 무한대의 의미
  - 예) ab{2}c : b가 2회만 반복되면 매치
  - 예) ab{2,5}c : b가 2회에서 5회까지 반복되면 매치
- **메타문자: ?**
  - 없거나 하나만 있으면 매치
  - 예) ab?c : b가 없거나 하나만 있는 경우 매치
- **메타문자:()**
  - 그룹을 의미
  - group() 혹은 group(0) : 전체 가져오기, group(n) : 첫번째 그룹 가져오기

파이썬 re모듈

- 정규 표현식을 지원하기 위해 re(regular expression의 약어) 모듈
  - re 모듈은 파이썬을 설치할 때 자동으로 설치되는 기본 라이브러리
  - 패턴과 매치하는 텍스트를 찾고 조작하는 기능을 제공

- 정규식 처리 메서드
  - .compile() : 정규 표현식 컴파일
  - .match():문자열의 처음부터 정규식과 매치되는지 조사
  - .search():문자열 전체를 검색하여 정규식과 매치되는지 조사
  - .findall():정규식과 매치되는 모든 문자열(substring)을 리스트로 반환
  - .group():매치된 문자열을 반환
  - .start():매치된 문자열의 시작 위치를 반환
  - .end():매치된 문자열의 끝 위치를 반환
  - .span():매치된 문자열의 (시작, 끝)에 해당하는 튜플을 반환

파이썬 re 모듈 예제

```
1   import re
2
3   #파일 열기
4   #f = open("binarytest.html", "rb")
5   f = open("01_html_css_b.html", "rb")
6   data = f.read()
7   #파일 종료
8   f.close()
9
10  dataEncoding = data[:1024].decode()
11  print(type(dataEncoding))
12
13  #정규표현식
14  #charset=뒤에 "," 문자가가 없거나 하나가 있고
15  #알파벳문자나 -문자가 최소 한번이상 나오는 것
16  #이때 알파벳문자나 -문자는 그룹
17  p = re.compile('charset=["\']?([\w-]+)')
18  encoding = p.search(dataEncoding).group(1)
19
20  print(encoding)
21
22  #r' 접두어 : 문자열에 사용된 이스케이프 시퀀스들을
23  #            이스케이프 시퀀스들이 아닌 원시 문자열로 취급
24  encoding=re.search(r'charset=["\']?([\w-]+)', dataEncoding).group(1)
25  print(encoding)
```

파이썬 URLLIB

웹 동작 방식

파이썬 urllib

- URL과 웹 요청에 관련된 모듈들 패키지로 묶어 제공
  - urllib.parse:
    - URL 해석·조작 기능을 담은 모듈
  - urllib.request:
    - HTTP 요청 기능을 담은 모듈

urllib.request 모듈

- **urllib.request.urlopen() 함수**
  - 웹 서버에 정보를 요청한 후, 돌려받은 응답을 저장하여 '응답 객체(HTTPResponse)'를 반환
- **반환된 응답 객체의 read() 메서드를 실행**
  - 웹 서버가 응답한 데이터를 바이트 배열로 읽어들임
  - 읽어들인 바이트 배열
    - 이진수로 이루어진 수열이어서 텍스트 형식의 데이터를 decode() 메서드를 실행하여 문자열로 변환

urllib.request로 문서 가져오기

```
1   from urllib.request import urlopen
2
3   response = urlopen("https://movie.naver.com/movie/sdb/rank/rmovie.nhn")
4   data = response.read()
5   print(data)
```

```
b'\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\t\r\n\t\r\n\t\r\n\t\r\n\t\r\n\t\r\r
\n\t\r\n\t\r\n\r\n\t\r\n\t\r\n\t\r\n\t\r\n\t\r\n\r\n\t\r\n\t\r\n\t\r\n\r
\t\t\t\t\t\r\n\t\t\t\t\r\n\t\t\t\r\n\t\t\t\r\n\t\t\t\r\n\t\t\r\n\t\r\n\t
n<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">\r\
<meta property="me2:image" content="http://imgmovie.naver.com/today/nave
xcc\xb9\xf6\xbf\xb5\xc8\xad "/>\r\n<meta property="me2:category1" conter
ontent=""/>\r\n\r\n\r\n\r\n\r\n\r\n\t\r\n\t\r\n\t\t<meta property="og:t
t\t\r\n\t\t\t\r\n\t\t\t<meta property="og:description" content="\xbf\x
c7\xc7\xbd\xba \xb7\xa9\xc5\xb7 \xc1\xa4\xba\xb8 \xc1\xa6\xb0\xf8" />\r\
"og:type" content="article"/>\r\n<meta property="og:url" content="https
ty="og:image" content="http://static.naver.net/m/movie/icons/OG_270_270.
```

실습

- URL 주소를 입력 받아서 문자열을 반환하는 함수를 작성하시오.
  - urlToStr()
  - urlCharset()
    - 해당 문서의 charset을 찾아서 디코딩