

파이썬프로그래밍

폴더의 파일 종류 및 내용 분석 프로그램

2019305059

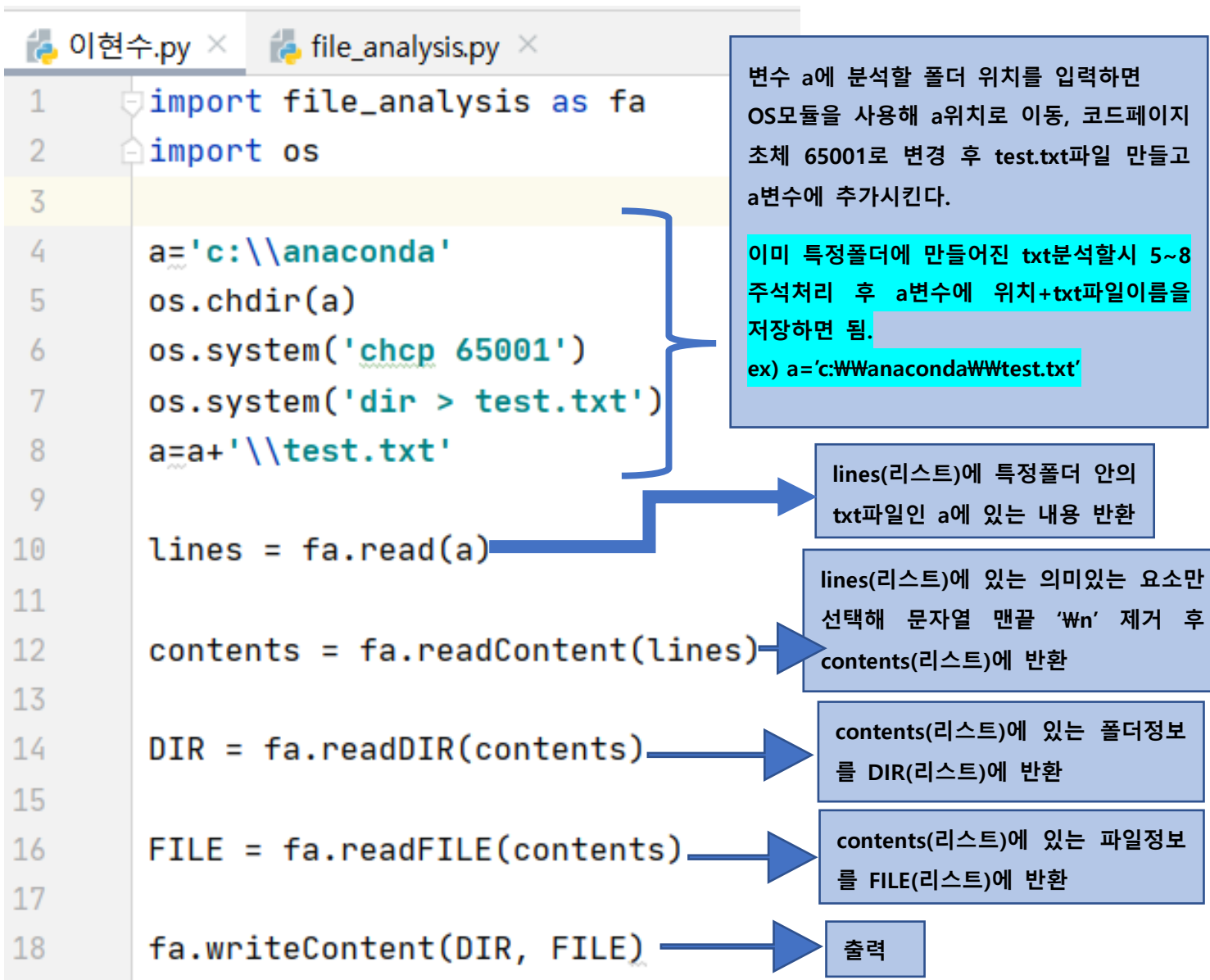
이현수

■제목 : 폴더의 파일 종류 및 내용 분석 프로그램

■내용

변수 a가 지정하는 특정 폴더 안에 있는 특정 파일(예: pptx, py 파일 등)의 개수(수량) 및 그 확장자를 갖는 모든 파일의 용량(바이트)를 출력하는 프로그램을 작성.

1. 프로그램 전체 흐름



메인 파일은 이현수.py 이다.

핵심 코드는 함수화시켜서 모듈 file_analysis.py에 위치시켰다.

모듈 file_analysis.py를 사용하기 위해서 코드1줄에 **import file_analysis as fa**를 삽입했다.

변수 a에는 폴더이름이 담겨있는 문자열 변수이다.

```

1 def read(filename):
2     f = open(filename, 'rt', encoding='utf-8') # 파일 열기
3     lines = f.readlines() # 파일 내용 읽기
4     f.close() # 파일 닫기
5     return lines # 리스트 반환

```

파일 열기, 파일내용 저장, 파일 닫기 함수

```

7 def readContent(lines):
8     contents=[]
9     for line in lines:
10         if '오후' in line or '오전' in line: # 각 줄에 '오후' 혹은 '오전'이 있으면
11             contents.append(line) # contents리스트에 추가
12     for i in range(len(contents)):
13         contents[i] = contents[i].replace('\n', '') # 각 줄 맨뒤에있는 '\n' 제거
14     return contents # 리스트 반환

```

처리할 문자열 저장 및 각 문자열 맨뒤에 있는 'Wn' 제거 함수

```

16 def readDIR(contents):
17     DIR=[]
18     for content in contents:
19         if '<DIR>' in content: # '<DIR>'이 있는 문자열일 경우==폴더일 경우
20             i = content.rfind(' ') # 마지막부터 ' '을 찾아 변수i에 저장
21             if content[i + 1:len(content)] != '.' and content[i + 1:len(content)] != '..':
22                 DIR.append(content[i + 1:len(content)]) # 폴더이름을 DIR리스트에 저장
23     return DIR # 리스트 반환

```

폴더이름 저장 함수

```

25 def readFILE(contents):
26     file=[]
27     for content in contents:
28         if '.' in content and '<DIR>' not in content: # .(확장자)이 있고, <DIR>(폴더)가 없는경우에
29             i = 20 # 시간 이후 문자열 인덱스
30             content_temp = content[20:] # 그 이후 문자열 슬라이싱해서 content_temp에 저장
31             content_temp = content_temp.lstrip() # 왼쪽의 공백 제거
32             j = content_temp.find(' ') # ' '찾기
33             size_str = content_temp[:j] # 처음부터 j-1까지 슬라이싱해서 size_str에 저장
34             while ',' in size_str:
35                 size_str = size_str.replace(',', '') # 용량에서 ','를 반복문으로 모두 제거
36             size = int(size_str) # size변수에 size_str문자열 정수형으로 변환
37             i = content_temp.rfind('.') # 뒤에서부터 . 찾고 인덱스 반환
38             name = content_temp[i + 1:] # i+1부터 끝까지 슬라이싱=확장자명
39             b = False # 기존에 있는 중복파일명인지 선별해줄 bool형변수
40             k = 0 # 기존에 있는 중복파일명이라면 그 파일명의 위치 인덱스 저장할 변수
41             for j in range(len(file)):
42                 if file[j][0] == name: # 만약 파일이름이 이미 저장되었는 문자열이라면
43                     b = True # b를 True로
44                     k = j # k에 찾은 인덱스 j를 저장
45                     break # 반복문 빠져나감
46             if b == True: # 만약 파일이름이 이미 file 이중리스트에 저장되었다면
47                 file[k][1] += 1 # 개수 1 증가
48                 file[k][2] += size # 사이즈 추가
49             else: # 파일이름이 새로운 것이라면
50                 file.append([name, 1, size]) # [파일이름, 개수 초기값1, 용량] 저장
51     file = sorted(file, key=lambda filename: filename[0]) # 확장자 알파벳순서로 정렬
52     return file # 이중리스트 반환

```

파일 이름, 개수, 용량 저장 함수

```

54 def writeContent(DIR, file):
55     print("{0:>10s}{1:>10s}{2:>10s}{3:>13s}".format("파일:", "개수:", "파일크기", "퍼센트용량"))
56     sum=0 #파일 전체용량 저장변수
57     totalfile = 0 # 전체파일개수 저장변수
58     for i in file:
59         totalfile+=i[1] #파일개수 누적저장
60     for i in file:
61         sum+=i[2] #파일 전체용량 저장
62     if sum==0:
63         for i in file:
64             print("{0:>10s}{1:>10}{2:>15}{3:>15.5f} %".format(i[0], printNum(i[1]), printNum(i[2]),
65                                                         (i[1]/totalfile)*100))
66     else:
67         for i in file:
68             print("{0:>10s}{1:>10}{2:>15}{3:>15.5f} %".format(i[0], printNum(i[1]), printNum(i[2]),
69                                                         ((i[2] / sum) * 100)))
70     print()
71     if len(DIR)>0: #폴더가 있다면
72         print("{0:>10} <{1}>".format("폴더:", DIR[0]))
73         for i in range(1, len(DIR)):
74             temp_str = "<"
75             temp_str += DIR[i]
76             temp_str += ">"
77             print("{0:12}{1:<12}".format(" ", temp_str))
78     else: #폴더가 없다면
79         print("{0:>10} {1}".format("폴더:", "없음"))
80
81     print()
82     print("\t 확장자의 수={0}, 폴더의 수={1}, 총 파일의 수={2}, 총 용량={3}".format(printNum(len(file)), printNum(len(DIR)),
83                                         printNum(totalfile), printNum(sum)))
84
85 def printNum(num): #숫자를 천단위로 , 를 넣어 문자열로 반환하는 함수
86     str_num = str(num) #정수를 문자열로 변환
87     num_len = len(str_num) #문자열 길이
88     if num_len < 4: #문자열길이가 4미만이라면
89         return str_num #그냥 반환
90     else: #문자열길이가 4이상이라면
91         numlist=list(str_num) #문자열을 리스트로 변환
92         index = num_len-1 - 3 + 1
93         numlist.insert(index,"") #삽입
94         while True:
95             index = index - 3 #인덱스 3빼기
96             if index<=0: #인덱스가 맨앞자리거나 음수면 반복문 빠져나감
97                 break
98             numlist.insert(index,"") #아니면 인덱스에 , 삽입
99     str_num=''.join(numlist) #리스트를 문자열로 변환
100    return str_num #문자열로 반환

```

출력 함수

숫자를 천단위로 , 를 넣어 문자열로 반환하는 함수

file_analysis.py 모듈에는 총 6개의 함수가 있다.

- ① **def read(filename) :**
변수에 있는 txt 파일을 열어서 모든 내용을 저장후 리스트 형태로 변환한다.
- ② **def readContent(lines) :**
lines 리스트에 있는 문자열들(txt파일의 각줄) 중 의미있는 문자열만 선택하고 그 문자열 맨뒤에 붙어있는 'Wn' 문자 삭제 후 리스트 형태로 반환
- ③ **def readDIR(contents) :**
contents리스트에 있는 문자열 중 폴더에 대해서 폴더 이름을 저장. 이때 './폴더(현재폴더), '../폴더(상위폴더)는 제외.
- ④ **def readFILE(contents) :**
contents리스트에 있는 문자열 중 파일에 관한 문자열에서 확장자, 크기, 개수 등을 분석해 이중리스트 형태로 반환한다. 이때 이중리스트는 [[확장자, 확장자파일개수, 확장자파일총용량], [...], [...],] 형태이다.
- ⑤ **def writeContent(DIR, file) :**
정해진 형태로 출력하는 함수
- ⑥ **def printNum(num):**
숫자를 천단위마다 , 를 삽입하기 위해서 정수를 매개변수로 받으면 , 삽입 후 문자열로 반환하는 함수

2. 코드 설명

① `def read(filename):`

```
이현수.py x file_analysis.py x
1 def read(filename):
2     f = open(filename, 'rt', encoding='utf-8') # 파일 열기
3     lines = f.readlines() # 파일 내용 읽기
4     f.close() # 파일 닫기
5     return lines # 리스트 반환
```

filename을 인수로 받는 함수이다. open()함수를 통해 파일을 읽는다. 이때 encoding='utf-8'을 해준다.

그리고 lines=f.readlines()을 통해 txt파일의 각줄을 한 개의 문자열로 리스트에 저장한다.

그 후 파일을 닫고 txt파일의 각 줄 문자열이 들어있는 리스트 lines를 반환한다.

② `def readContent(lines):`

```
7 def readContent(lines):
8     contents=[]
9     for line in lines:
10         if '오후' in line or '오전' in line: # 각 줄에 '오후' 혹은 '오전'이 있으면
11             contents.append(line) # contents리스트에 추가
12     for i in range(len(contents)):
13         contents[i] = contents[i].replace('\n', '') # 각줄 맨뒤에있는 '\n'제거
14     return contents # 리스트 반환
```

`def read(filename):` 함수에서 반환된 lines리스트를 인수로 받는다. 그 후 contents=[]이라는 리스트를 선언해준다. 이 리스트에는 분석에 필요한 문자열만 저장해 반환할 것이다.

```
2020-10-14 오전 02:10 <DIR> .
2020-10-14 오전 02:10 <DIR> ..
2020-10-14 오전 12:37      1,861 @escape_sequence.py
2020-10-14 오전 02:10      1,599 @getcwd.py
2020-10-14 오전 01:52      6,326 @str_methods.py
2020-10-14 오전 02:00      5,268 @str_methods_etc.py
2019-10-08 오전 05:51      1,272 dir.txt
```

(코드 9-11줄) txt파일을 보면 폴더, 파일에 관한 정보가 있는 라인에는 항상 '오전' 혹은 '오후'라는 문자열이 있다. 그래서 '오후' 혹은 '오전' 중 한문자열이라도 존재하면 그 문자열은 분석할만한 의미있는 문자열이다. 그래서 append()함수를 통해서 contents리스트에 저장한다.

(코드 12-13줄) f.readlines()로 받은 문자열에는 마지막에 \n이 붙어있다. 그래서 replace()함수를 통해 \n를 삭제시킨다.

그러면 contents리스트에는 분석할 문자열(맨뒤에 \n을 제거한)들만 들어있다. 그 리스트를 반환한다.

③def readDIR(contents):

```
16 def readDIR(contents):
17     DIR=[]
18     for content in contents:
19         if '<DIR>' in content: # '<DIR>'이 있는 문자열일 경우==폴더일 경우
20             i = content.rfind(' ') # 마지막부터 ' '을 찾아 변수i에 저장
21             if content[i + 1:len(content)] != '.' and content[i + 1:len(content)] != '..' :
22                 DIR.append(content[i + 1:len(content)]) # 폴더이름을 DIR리스트에 저장
23     return DIR #리스트 반환
24
```

(코드 18-22줄) 폴더는 이름만 저장시키면 된다. DIR리스트에 폴더이름을 저장할 것이다. 폴더를 식별하기 위해서는 <DIR>의 유무로 판단할 수 있다. <DIR>이 있다면 1차적으로 분석할 폴더의 대상이다.

2020-10-14 오전 02:10 <DIR> .. ->content 중 하나

Rfind(' ')을 통해 문자열 뒤부터 공백을 찾고 문자열 i+1부터 끝까지 슬라이싱을 진행한다. 그렇게 되면 폴더의 이름을 얻을 수 있다. 하지만 <.> <..> 폴더는 제외해야 하기 때문에 21줄 조건문 코드를 통해 제외시킨 후 DIR리스트에 폴더이름을 추가시킨다.

④def readFILE(contents):

```
25 def readFILE(contents):
26     file=[]
27     for content in contents:
28         if '.' in content and '<DIR>' not in content: # .(확장자)이 있고, <DIR>(폴더)가 없는경우에
29             i = 20 # 시간 이후 문자열 인덱스
30             content_temp = content[20:] # 그 이후 문자열 슬라이싱해서 content_temp에 저장
31             content_temp = content_temp.lstrip() # 왼쪽의 공백 제거
32             j = content_temp.find(' ') # ' '찾기
33             size_str = content_temp[:j] # 처음부터 j-1까지 슬라이싱해서 size_str에 저장
34             while ',' in size_str:
35                 size_str = size_str.replace(',', '') # 용량에서 ','를 반복문으로 모두 제거
36             size = int(size_str) # size변수에 size_str문자열 정수형으로 변환
```

파일데이터를 저장할 file리스트를 저장한다. file리스트는 결과적으로 이중리스트가 된다.

file리스트 구조를 보면 file = [[확장자이름, 개수, 용량], [...], [...],]

27줄부터는 for 반복문에서 조건문을 사용해서 만약 문자열에 '.'이 있고 '<DIR>'이 없는 줄만 데이터를 처리하게 한다. 여기서 '<DIR>'이 없다는 조건을 추가시킨 이유는 폴더이면서 폴더명에 '.'이 들어갈 수 있기 때문이다.

2020-10-14 오전 12:37 1,861 @escape_sequence.py

그 후 i에 20을 저장한다. 여기서 20은 시간문자열 이후의 인덱스이다. 즉 위 사진에서보면 '오전 12:37'에서 숫자 7이후의 인덱스 값이다.

(코드 30줄)그 후 content_temp에 인덱스 20부터 끝까지 저장시킨다. 그렇게 되면...

content_temp = 1,861 @escape_sequence.py 이렇게 저장된다.

(코드 31줄)그 후 content_temp.lstrip()을 통해 왼쪽의 공백을 모두 제거시킨다. 그 결과로...

content_temp = 1,861 @escape_sequence.py 이렇게 저장된다.

(코드 33줄)에서 변수 j에 find메소드를 사용해 공백이 있는 인덱스를 반환받고, size_str문자열에 문자를 저장한다. 그 결과 size_str = 1,861 이렇게 저장된다.

(코드 34줄)에서 while 반복문을 통해 size_str에 ','가 없을 때까지 replace메소드를 통해서 ','를 제거시킨다. 그 후 size변수에 size_str을 정수형으로 형변환을 한다.

```

37 i = content_temp.rfind('.') # 뒤에서부터 . 찾고 인덱스 반환
38 name = content_temp[i + 1:] # i+1부터 끝까지 슬라이싱=확장자명
39 b = False # 기존에 있는 중복파일명인지 선별해줄 bool 형변수
40 k = 0 # 기존에 있는 중복파일명이라면 그 파일명의 위치 인덱스 저장할 변수
41 for j in range(len(file)):
42     if file[j][0] == name: # 만약 파일이름이 이미 저장되어있는 문자열이라면
43         b = True # b를 True로
44         k = j # k에 찾은 인덱스 j를 저장
45         break # 반복문 빠져나감
46 if b == True: # 만약 파일이름이 이미 file 이중리스트에 저장되었다면
47     file[k][1] += 1 # 개수 1 증가
48     file[k][2] += size # 사이즈 추가
49 else: # 파일이름이 새로운 것이라면
50     file.append([name, 1, size]) # [파일이름, 개수초기값1, 용량] 저장
51 file = sorted(file, key=lambda filename: filename[0]) # 확장자 알파벳순서로 정렬
52 return file # 이중리스트 반환
53

```

(코드 37줄)에서 파일 확장자 이름을 저장하기 위해서 `content_temp.rfind('.')`을 통해 문자열 뒤에서부터 '.'이 있는 위치의 인덱스를 변수 `i`에 반환한다.

`content_temp = 1,861 @escape_sequence.py`

그 후 `content_temp[i+1:]`를 통해 `i+1`부터 끝까지를 문자열 슬라이싱을 통해 `name` 변수에 저장한다.

(코드 41줄)에서 반복문을 통해 `file` 이중리스트에 변수 `name`에 있는 문자열과 같은 확장자 이름이 있는지 살펴본다. 만약에 같은 확장자 이름이 발견된 경우에만 bool형 변수 `b`를 True로 변경시키고 현재 인덱스를 저장하기 위해 변수 `k`에 인덱스 정보를 저장한다. 그리고 반복문에서 빠져나온다.

그 후 `b`가 True일 경우(file리스트에 이미 같은 이름의 확장자 존재) `file[k][1]`을 1증가시켜 확장자 개수를 증가시키고 `file[k][2]`를 통해 용량을 누적합을 한다.

그것이 아닐 경우에는 `file`리스트에 `[name, 1, size]`리스트를 추가한다.

이 과정을 `contents`에 있는 모든 문자열에 대해서 작업한다.

그 결과

DIR = [폴더이름1, 폴더이름2,] (예시)

file = [['py', 11, 17824], ['txt', 4, 3526]] (예시)

이렇게 저장된다.

그 후 `sorted()`함수를 통해 `file`리스트를 확장자 알파벳순으로 정렬한다.
그리고 `file`리스트를 반환한다.

⑤ def writeContent(DIR, file):

```

54 def writeContent(DIR, file):
55     print("{0:>10s}{1:>10s}{2:>10s}{3:>13s}".format("파일:", "개수:", "파일크기의합", "퍼센트용량"))
56     sum=0 #파일 전체용량 저장변수
57     totalfile = 0 # 전체파일개수 저장변수
58     for i in file:
59         totalfile+=i[1] #파일개수 누적저장
60     for i in file:
61         sum+=i[2] #파일 전체용량 저장
62     if sum==0:
63         for i in file:
64             print("{0:>10s}{1:>10s}{2:>15s}{3:>15.5f} %".format(i[0], printNum(i[1]), printNum(i[2]),
65                                                         (i[1]/totalfile)*100))
66     else:
67         for i in file:
68             print("{0:>10s}{1:>10s}{2:>15s}{3:>15.5f} %".format(i[0], printNum(i[1]), printNum(i[2]),
69                                                         ((i[2] / sum) * 100)))
70     print()
71     if len(DIR)>0: #폴더가 있다면
72         print("{0:>10} <{1}>".format("폴더:", DIR[0]))
73         for i in range(1, len(DIR)):
74             temp_str = "<"
75             temp_str += DIR[i]
76             temp_str += ">"
77             print("{0:12}{1:<12}".format(" ", temp_str))
78     else: #폴더가 없다면
79         print("{0:>10} {1}".format("폴더:", "없음"))
80
81     print()
82     print("\t 확장자의 수={0}, 폴더의 수={1}, 총 파일의 수={2}, 총 용량={3}".format(printNum(len(file)), printNum(len(DIR)),
83                                                         printNum(totalfile), printNum(sum)))

```

출력 내용을 정렬할 때 format()함수의 패딩을 통해서 정렬했다. 또한 조건문을 사용해 폴더가 없는 경우 '없음'을 출력하도록 코딩했다.

⑥ def printNum(num):

```

84 def printNum(num): #숫자를 천단위로 , 를 넣어 문자열로 반환하는 함수
85     str_num = str(num) #정수를 문자열로 변환
86     num_len = len(str_num) #문자열 길이
87     if num_len < 4: #문자열길이가 4미만이라면
88         return str_num #그냥 반환
89     else: #문자열길이가 4이상이라면
90         numlist=list(str_num) #문자열을 리스트로 변환
91         index = num_len-1 - 3 + 1
92         numlist.insert(index, ",") # , 삽입
93         while True:
94             index = index - 3 #인덱스 3빼기
95             if index<=0: #인덱스가 맨앞자리거나 음수면 반복문 빠져나감
96                 break
97             else:
98                 numlist.insert(index, ",")#아니면 인덱스에 , 삽입
99         str_num=''.join(numlist) #리스트를 문자열로 변환
100        return str_num #문자열로 반환

```

숫자를 천단위로 , 를 넣어 문자열로 반환하는 함수이다. 우선 정수를 문자열로 변환한다.

(코드 81줄) 우선 100 과같이 숫자길이가 3이하라면 ,를 붙일 필요가 없다. 그래서 바로 return 해준다.

(코드 83줄) 만약에 1000 처럼 ,을 붙여야 할경우에 문자열을 리스트로 변환시킨다.

예로 들어 1000일경우 ['1', '0', '0', '0'] 로 바꾼다. 그 후 $index = 4-1-3+1 = 1$ 을 통해 리스트 인덱스 1에 ,를 삽입한다. 그렇게 되면 ['1', ',', '0', '0', '0'] 로 바뀐다. 무한반복문을 통해 index값을 3씩 빼준다. 만약에 그 값이 0(문자열의 맨앞) 혹은 음수가 될 경우 while반복문을 빠져나온다.

그리고 "join(numlist)를 통해 간격없이 리스트를 합쳐 문자열로 바꾼다. 그리고 문자열을 반환한다.

3. 실행

1)

이현수.py x file_analysis.py x

1import file_analysis as fa

2import os

3

4a='c:\\anaconda'

5os.chdir(a)

6os.system('chcp 65001')

7os.system('dir > test.txt')

8a=a+'\\test.txt'

9

10lines = fa.read(a)

11

12contents = fa.readContent(lines)

13

14DIR = fa.readDIR(contents)

15

16FILE = fa.readFILE(contents)

17

18fa.writeContent(DIR, FILE)

내 PC > 로컬 디스크 (C:) > anaconda

anaconda 검색

이름	수정한 날짜	유형	크기
msvcp140_2.dll	2018-11-20 오전 2:57	응용 프로그램 확장	201KB
python	2020-07-03 오전 1:31	응용 프로그램	93KB
python.pdb	2020-07-03 오전 1:31	Program Debug ...	436KB
python3.dll	2020-07-03 오전 1:31	응용 프로그램 확장	51KB
python38.dll	2020-07-03 오전 1:31	응용 프로그램 확장	4,111KB
python38.pdb	2020-07-03 오전 1:31	Program Debug ...	11,796KB
pythonw	2020-07-03 오전 1:31	응용 프로그램	92KB
pythonw.pdb	2020-07-03 오전 1:31	Program Debug ...	436KB
qt.conf	2020-09-08 오후 7:42	CONF 파일	1KB
test	2020-10-25 오전 11:00	텍스트 문서	6KB
ucrtbase.dll	2018-04-21 오전 12:37	응용 프로그램 확장	993KB

분석하려는 폴더 위치를 a변수에 입력 후(a='c:\\anaconda') 프로그램 실행 시 anaconda폴더에 test.txt가 생겼다.

●실행결과 ↓

Active code page: 65001

파일:	개수:	파일크기의합	퍼센트용량
conf:	1	203	0.00048 %
dll:	52	8,635,416	20.57869 %
exe:	6	20,341,409	48.47475 %
nonadmin:	1	0	0.00000 %
pdb:	3	12,972,032	30.91310 %
py:	1	1,063	0.00253 %
txt:	2	12,775	0.03044 %

폴더: <bin>

<conda-meta>

<condabin>

<DLLs>

<envs>

<etc>

<include>

<Lib>

<Library>

<libs>

<man>

<Menu>

<pkgs>

<Scripts>

<share>

<shell>

<sip>

<tcl>

<Tools>

확장자의 수=7, 폴더의 수=19, 총 파일의 수=66, 총 용량=41,962,898

●실행평가

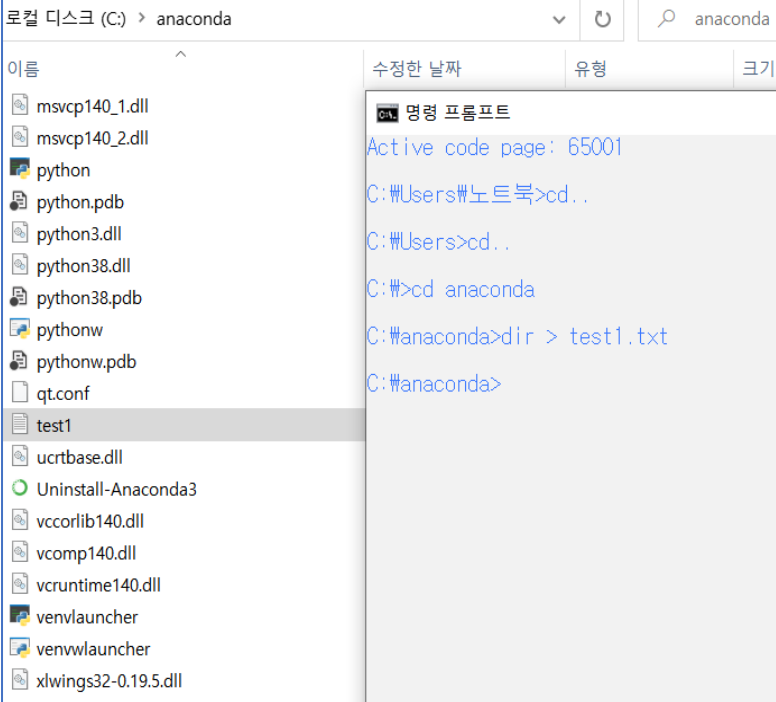
2020-07-05 오후 07:05	381,440	xlwings64-0.19.5.dll
2020-07-18 오전 07:51	18,798,141	_conda.exe
66 File(s)	41,962,898 bytes	
21 Dir(s)	48,208,076,800 bytes free	

test.txt 파일의 맨 하단의 정보를 비교해보면 모든정보가 똑같다.

***<.>, <..> 폴더를 제외했기 때문에 위사진에 보이는 21Dir(s)와 비교해 프로그램 실행결과인 폴더의 수=19 는 정상적인 폴더수이다.**

정렬정도, 숫자 천단위로 ‘ ’ 찍기, 파일확장자 알파벳 순 정렬, 퍼센트, 폴더 <.>, <..> 제외 등 모두 정상적으로 출력된다.

1) 이미 만들어진 특정폴더 내 txt파일 분석 할 경우



```

1 import file_analysis as fa
2
3 a="c:/anaconda/test1.txt"
4 lines = fa.read(a)
5
6 contents = fa.readContent(lines)
7
8 DIR = fa.readDIR(contents)
9
10 FILE = fa.readFILE(contents)
11
12 fa.writeContent(DIR, FILE)

```

cmd 창에서 코드페이지를 UTF-8로 변경후 test1.txt파일을 만든다. test1.txt파일이 정상적으로 만들어졌다. 그 후 이미 txt파일이 만들어졌기 때문에 코드5-8 제거 후 변수 a에 위 사진과 같이 저장한다.

●실행결과 ↓

"C:\Program Files\Python38\python.exe" C:/Users/노트북/Desktop/이현수/이현수.py

파일:	개수:	파일크기의합	퍼센트용량
conf:	1	203	0.00048 %
dll:	52	8,635,416	20.57869 %
exe:	6	20,341,409	48.47475 %
nonadmin:	1	0	0.00000 %
pdb:	3	12,972,032	30.91310 %
py:	1	1,063	0.00253 %
txt:	2	12,775	0.03044 %

폴더: <bin>
 <conda-meta>
 <condabin>
 <DLLs>
 <envs>
 <etc>
 <include>
 <Lib>
 <Library>
 <libs>
 <man>
 <Menu>
 <pkgs>
 <Scripts>
 <share>
 <shell>
 <sip>
 <tcl>
 <Tools>

확장자의 수=7, 폴더의 수=19, 총 파일의 수=66, 총 용량=41,962,898

●실행평가

2020-07-18 오전 07:51 18,798,141 _conda.exe
 66 File(s) 41,962,898 bytes
 21 Dir(s) 48,141,189,120 bytes free

test1.txt 파일의 맨 하단의 정보를 비교해보면 모든정보가 똑같다.

***<.>, <..> 폴더를 제외했기 때문에 위 사진에 보이는 21Dir(s)와 비교해 프로그램 실행결과인 폴더의수=19 는 정상적인 폴더수이다.**

정렬정도, 숫자 천단위로 ' 찍기, 파일확장자 알파벳순 정렬, 퍼센트, 폴더 <.>, <..> 제외 등 모두 정상적으로 출력된다.

2)














```

1 import file_analysis as fa
2 import os
3
4 a='c:\\Ch06'
5 os.chdir(a)
6 os.system('chcp 65001')
7 os.system('dir > test.txt')
8 a=a+'\\test.txt'
9
10 lines = fa.read(a)
11
12 contents = fa.readContent(lines)
13
14 DIR = fa.readDIR(contents)
15
16 FILE = fa.readFILE(contents)
17
18 fa.writeContent(DIR, FILE)

```

내 PC > 로컬 디스크 (C:) > Ch06

이름

-  @escape_sequence
-  @getcwd
-  @str_methods
-  formatting1
-  formatting2
-  formatting3
-  formatting4
-  formatting5
-  padding
-  test
-  yesterday
-  yesterday
-  한글

분석하려는 폴더 위치를 a변수에 입력 후(a='c:\\Ch06') 프로그램 실행 시 anaconda폴더에 test.txt가 생겼다.

●실행결과 ↓

Active code page: 65001

파일:	개수:	파일크기의합	퍼센트용량
py:	10	12,900	85.18788 %
txt:	3	2,243	14.81212 %

폴더: 없음

확장자의 수=2, 폴더의 수=0, 총 파일의 수=13, 총 용량=15,143

●실행평가

정렬정도, 확장자별 알파벳순 정렬, 개수, 퍼센트 표시, 숫자 천단위마다 쉼표 찍기가 정상적으로 출력된다.

폴더는 없기 때문에 폴더가 없음이라고 정상적으로 출력된다.

2018-11-19 오후 12:06 703 yesterday.txt

2020-10-14 오전 02:46 1,540 한글.txt

13 File(s) 15,143 bytes

2 Dir(s) 48,203,567,104 bytes free

->test.txt 하단의 내용

테스트2.txt 하단내용과 실행결과를 비교해보면 파일수 13, 총용량 15,143으로 동일하게 출력된다.

폴더개수의 경우 <.>, <..>를 제외시켰기 때문에 실행결과 폴더의수=0으로 출력되는 것은 정상이다.

2) 이미 만들어진 특정폴더 내 txt파일 분석 할 경우

로컬 디스크 (C:) > Ch06

이름
수정일
@escape_sequence
2020-
@getcwd
2020-
@str_methods
2020-
formatting1
2018-
formatting2
2018-
formatting3
2018-
formatting4
2018-
formatting5
2018-
padding
2020-
yesterday
2019-
yesterday
2018-
테스트2
2020-
한글
2020-

명령 프롬프트
Active code page: 65001
C:\Users\#노트북>cd..
C:\Users>cd..
C:\>cd Ch06
C:\Ch06>dir > 테스트2.txt
C:\Ch06>

이현수.py x file_analysis.py x

```

1 import file_analysis as fa
2
3 a="c:/Ch06/테스트2.txt"
4 lines = fa.read(a)
5
6 contents = fa.readContent(lines)
7
8 DIR = fa.readDIR(contents)
9
10 FILE = fa.readFILE(contents)
11
12 fa.writeContent(DIR, FILE)

```

cmd 창에서 코드페이지를 UTF-8로 변경후 테스트2.txt파일을 만든다. 테스트2.txt파일이 정상적으로 만들어졌다.

●실행결과

"C:\Program Files\Python38\python.exe" C:/Users/노트북/Desktop/이현수/이현수.py

파일:	개수:	파일크기의합	퍼센트용량
py:	10	12,900	85.18788 %
txt:	3	2,243	14.81212 %

폴더: 없음

확장자의 수=2, 폴더의 수=0, 총 파일의 수=13, 총 용량=15,143

Process finished with exit code 0

●실행평가

정렬정도, 확장자별 알파벳순 정렬, 개수, 퍼센트 표시, 숫자 천단위마다 콤마 찍기가 정상적으로 출력된다.

폴더는 없기 때문에 폴더가 없음이라고 정상적으로 출력된다.

2020-10-14 오전 02:46
1,540 한글.txt
13 File(s)
15,143 bytes
2 Dir(s)
48,158,089,216 bytes free

->테스트2.txt 하단의 내용

테스트2.txt 하단내용과 실행결과를 비교해보면 파일수 13, 총용량 15,143으로 동일하게 출력된다.

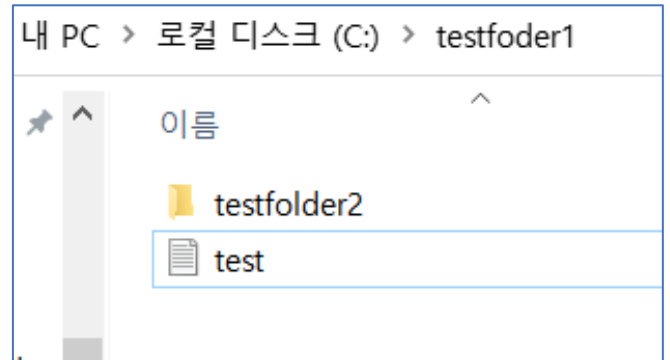
폴더개수의 경우 <.>, <..>를 제외시켰기 때문에 실행결과 폴더의수=0으로 출력되는 것은 정상이다.

3)

```

이현수.py x file_analysis.py x
1 import file_analysis as fa
2 import os
3
4 a='c:\\testfoder1'
5 os.chdir(a)
6 os.system('chcp 65001')
7 os.system('dir > test.txt')
8 a=a+'\\test.txt'
9
10 lines = fa.read(a)
11
12 contents = fa.readContent(lines)
13
14 DIR = fa.readDIR(contents)
15
16 FILE = fa.readFILE(contents)
17
18 fa.writeContent(DIR, FILE)

```



분석하려는 폴더 위치를 a변수에 입력 후(a='c:\\WWCh06') 프로그램 실행 시 anaconda폴더에 test.txt가 생겼다.

●실행결과

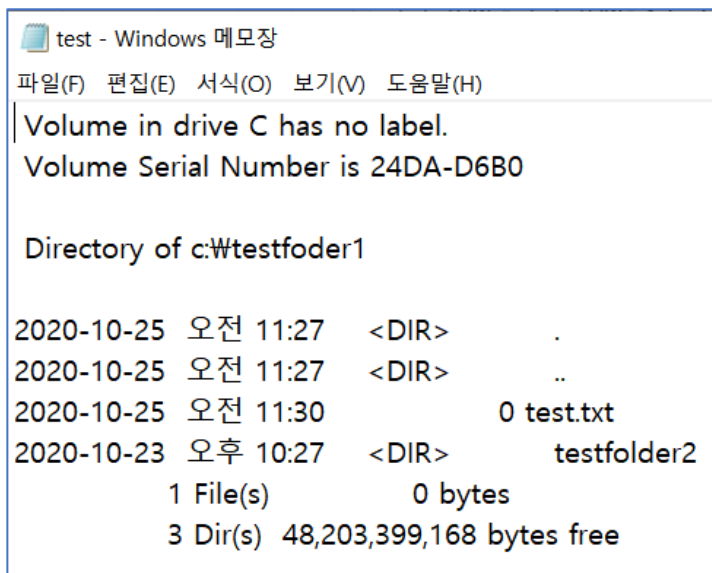
Active code page: 65001

파일:	개수:	파일크기의합	퍼센트용량
txt:	1	0	100.00000 %

폴더: <testfolder2>

확장자의 수=1, 폴더의 수=1, 총 파일의 수=1, 총 용량=0

●실행평가



정렬정도, 확장자별 알파벳순 정렬, 개수, 퍼센트 표시가 정상적으로 출력된다.

폴더는 <.>, <..>를 제외하고 폴더가 1개 있기 때문에 폴더1개만 정상적으로 출력 된다.

4. 추가(메일질문 사항)

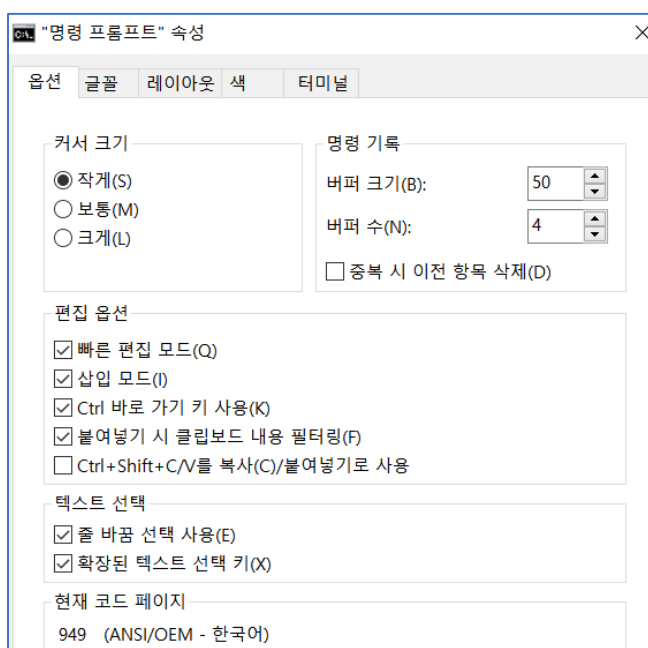
F = open(filename, 'rt', encoding='utf-8') 으로 파일을 열 때

```
1 import file_analysis as fa
2 import os
3 a='c:\\testfoder1'
4 #'''
5 os.chdir(a)
6 #os.system('chcp 65001')
7 os.system('dir > test.txt')
8 a=a+'\\test.txt'
9 #'''
10 lines = fa.read(a) #txt파일 열어서 내용 저장
11
12 contents = fa.readContent(lines) #분석할 의미있는 내용만 저장
13
14 DIR = fa.readDIR(contents) #폴더 정보 저장
15
16 FILE = fa.readFILE(contents) #파일 정보 저장
```

os.system('chcp 65001')을 생략하게 되면

```
File "C:\Program Files\Python38\lib\codecs.py", line 322, in decode
    (result, consumed) = self._buffer_decode(data, self.errors, final)
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xb5 in position 3: invalid start byte
```

생성된 test.txt파일을 열 때 이러한 오류가 생긴다.



명령 프롬프트에서 오른쪽마우스->속성으로 코드페이지를 확인할 수 있다. 'chcp 65001'을 입력해 코드페이지를 65001 (UTF-8)로 변경해야 F = open(filename, 'rt', encoding='utf-8') 코드로 정상적으로 파일을 열 수 있다.