REPORT

2019 전공기초프로젝트2 1차 검사 보고서

과목명 | 전공기초프로젝트2

담당교수 | 차리서 교수님

학과 | 소프트웨어학과

소속 | 9팀

팀원 | 201714150 김동진

201714151 박민기

201714154 오병현

201714156 이예지

201715007 안지호

주제 | 사진 파일 분류기

Contents

1. 개요

프로그램 실행 시 폴더가 생성되고 그 폴더에 사진 파일들을 넣은 후, 사용자가 분류기준(크기, 이름, 생성날짜, 수정날짜, 확장자, GPS 등)을 정하면 분류를 수행합니다. 분류 과정은 실행 시 생성된 폴더 안에서 분류기준에 따라 여러 폴더를 만들어 사진들을 분류하고 만약 한 개의 기준이 아닌 여러 분류기준을 선택했을 시 각각의 분류된 폴더 안에서 또 다른 분류기준을 적용하여 하위 폴더를 만들어가는 방식입니다.

2. 단위검사

2.1mainModule.py

2.1.1 Createfolder(directory) Function #1 - 수정 無

C 드라이브 바로 밑에 작업 폴더를 생성한다.

부분 설계

폴더명이 중복되지 않을 경우 정상적으로 폴더를 생성한다.

검사방법

Start() 함수의 폴더 생성부분에 해당 함수를 호출하여 리턴값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 폴더명 중복 시 예외처리

입력 (호출)	예상결과	실제 결과	비고
Program Files	OSError	OSError	
PerfLogs	OSError	OSError	
Program Files(x86)	OSError	OSError	
Temp	OSError	OSError	
tmp	OSError	OSError	
web	OSError	OSError	
Windows	OSError	OSError	

사용자	OSError	OSError

2.1.2 start() Function #2 - 수정 전

사용자에게 프로그램 시작을 알리고 작업 폴더명 입력을 받는다.

부분 설계 1

폴더명 중복의 경우 예외 처리한다.

"파일명이 중복되었습니다. 다시 지정해주세요." 출력 (= 파일명중복오류메세지)

이후 사용자에게 재입력을 받는다.

부분 설계 2

폴더명 길이가 0이거나 50이상인 경우 예외 처리한다.

"파일명 지정은 50자 이내 입니다. 다시 설정해주세요." 출력 (= 파일명길이오류메세지)

이후 사용자에게 재입력을 받는다.

부분 설계 3

폴더명에 사용할 수 없는 특수문자를 붙인 경우 예외 처리한다.

"{/ , ₩ , : , . , | , < , > , ?} 는 파일명에 포함될 수 없습니다. 다시 지정해주세요." 출력

(= 파일명오류메세지) 이후 사용자에게 재입력을 받는다.

검사방법

mainModule.py를 Python shell에 로드한 후 프롬프트에 start() 함수를 직접 호출하고 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 정상적으로 작업 폴더를 생성하는지 확인

입력 (호출)	예상결과	실제 결과	비고
Program File	파일명중복오류메세지	파일명중복오류메세지	
PerfLogs	파일명중복오류메세지	파일명중복오류메세지	
Program Files(x86)	파일명중복오류메세지	파일명중복오류메세지	

Temp	파일명중복오류메세지	파일명중복오류메세지	
tmp	파일명중복오류메세지	파일명중복오류메세지	
web	파일명중복오류메세지	파일명중복오류메세지	
Windows	파일명중복오류메세지	파일명중복오류메세지	
사용자	파일명중복오류메세지	파일명중복오류메세지	
	파일길이오류메세지	OSError	정상처리 x
(빈칸)	파일길이오류메세지	FileNotFoundError	정상처리 x
1*51(51개숫자)	파일길이오류메세지	파일길이오류메세지	
ㅁ*51(51개한글)	파일길이오류메세지	파일길이오류메세지	
R*51(51개영문)	파일길이오류메세지	파일길이오류메세지	
安*51(51개한문)	파일길이오류메세지	파일길이오류메세지	
ほ*51(51개일문)	파일길이오류메세지	파일길이오류메세지	
Ò*51(51개독문)	파일길이오류메세지	파일길이오류메세지	
Aaa,	파일명오류메세지	파일명오류메세지	
Aaa/	파일명오류메세지	파일명오류메세지	
Aaa₩	파일명오류메세지	OSError & ₩문자제외폴더생성	정상처리x
Aaa:	파일명오류메세지	파일명오류메세지	
Aaa.	파일명오류메세지	파일명오류메세지	
Aaa|	파일명오류메세지	파일명오류메세지	
Aaa<	파일명오류메세지	파일명오류메세지	
Aaa>	파일명오류메세지	파일명오류메세지	
Aaa?	파일명오류메세지	파일명오류메세지	

부연설명

역슬래쉬(₩) , 빈칸입력(SpaceBar), 입력이 없는 경우에 대한 예외처리가 필요하다.

다음의 경우 Error 가 발생하며 정상 작동하지 않고 프로그램이 종료한다.

2.1.3 Start() Function#2 - 수정 후

사용자에게 프로그램 시작을 알리고 작업 폴더명 입력을 받는다.

부분 설계 1

폴더명 중복의 경우 예외 처리한다.

"파일명이 중복되었습니다. 다시 지정해주세요." 출력 (= 파일명중복오류메세지)

이후 사용자에게 재입력을 받는다.

부분 설계 2

폴더명 길이가 0이거나 50이상인 경우 예외 처리한다.

폴더명 길이가 0이거나 50이상 경우 :

"파일명 지정은 50자 이내 입니다.(파일명 미지정도 불가합니다 다시 설정해주세요." 출력

(= 파일명길이오류메세지_1) 이후 사용자에게 재입력을 받는다.

폴더명에 공백을 포함하여 지정한 경우:

"파일명에 공백이 존재하면 안됩니다." 출력 (= 파일길이오류메세지_2) 이후 사용자에게 재입력을 받는다.

부분 설계 3

폴더명에 사용할 수 없는 특수문자를 붙인 경우 예외 처리한다.

"{/ , ₩ , : , . , | , < , > , ?} 는 파일명에 포함될 수 없습니다. 다시 지정해주세요." 출력

(= 파일명오류메세지) 이후 사용자에게 재입력을 받는다.

검사방법

mainModule.py를 Python shell에 로드한 후 프롬프트에 start() 함수를 직접 호출하고 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 정상적으로 작업 폴더를 생성하는지 확인

입력 (호출)	예상결과	실제 결과	비고
Program File	파일명중복오류메세지	파일명중복오류메세지	
PerfLogs	파일명중복오류메세지	파일명중복오류메세지	
Program Files(x86)	파일명중복오류메세지	파일명중복오류메세지	
Temp	파일명중복오류메세지	파일명중복오류메세지	
tmp	파일명중복오류메세지	파일명중복오류메세지	
web	파일명중복오류메세지	파일명중복오류메세지	
Windows	파일명중복오류메세지	파일명중복오류메세지	
사용자	파일명중복오류메세지	파일명중복오류메세지	
	파일길이오류메세지_1	파일길이오류메세지_1	
(빈칸)	파일길이오류메세지_2	파일길이오류메세지_2	
1*51(51개숫자)	파일길이오류메세지_1	파일길이오류메세지_1	
ㅁ*51(51개한글)	파일길이오류메세지_1	파일길이오류메세지_1	
R*51(51개영문)	파일길이오류메세지_1	파일길이오류메세지_1	
安*51(51개한문)	파일길이오류메세지_1	파일길이오류메세지_1	
ほ*51(51개일문)	파일길이오류메세지_1	파일길이오류메세지_1	
Ò*51(51개독문)	파일길이오류메세지_1	파일길이오류메세지_1	
Aaa,	파일명오류메세지	파일명오류메세지	
Aaa/	파일명오류메세지	파일명오류메세지	
Aaa₩	파일명오류메세지	파일명오류메세지	
Aaa:	파일명오류메세지	파일명오류메세지	
Aaa.	파일명오류메세지	파일명오류메세지	
Aaa|	파일명오류메세지	파일명오류메세지	
Aaa<	파일명오류메세지	파일명오류메세지	

Aaa>	파일명오류메세지	파일명오류메세지
Aaa?	파일명오류메세지	파일명오류메세지

2.1.4 photochecker(fileName) Function#3 – 수정 전

C 드라이브에 생성한 작업폴더에 분류대상인 사진 파일이 존재하는 지 확인한다.

부분 설계 1

Jpg, bmp, jpeg, png, gif 파일이 있는지를 확인한다.

없다면 "사진파일 또는 사진 폴더가 작업 폴더에 존재하지 않습니다. 사진 파일 또는 사진 폴더를 넣어주세요."(= 출력1) 라는 메시지를 출력한다.

부분 설계 2

"사진 파일을 넣었으면 ok를 입력하여 주십시오."(= 출력 2) 를 출력하고 유효한 값을 입력 시 다음 작업으로 넘어가고, 유효하지 않은 값을 입력 시 "ok,Ok,OK,oK 중 하나를 입력하세요."(= 출력 3)을 출력하고 재입력을 받는다.

검사방법

mainModule.py를 Python shell에 로드한 후 프롬프트에 photochecker(fileame) 함수를 직접 호출하고 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 작업 폴더 내 사진 파일 존재 유무 확인

입력 (호출)	예상결과	실제 결과	비고
1.jpg	사진 파일로 인식	사진 파일로 인식	
1.bmp	사진 파일로 인식	사진 파일로 인식	
1.jpeg	사진 파일로 인식	사진 파일로 인식	
1.png	사진 파일로 인식	사진 파일로 인식	
1.gif	사진 파일로 인식	사진 파일로 인식	
1.java	사진 파일로 인식x	사진 파일로 인식x	

1.mp4	사진 파일로 인식x	사진 파일로 인식x
1.o	사진 파일로 인식x	사진 파일로 인식x
1.so	사진 파일로 인식x	사진 파일로 인식x
1.mp3	사진 파일로 인식x	사진 파일로 인식x
1.c	사진 파일로 인식x	사진 파일로 인식x
1.cpp	사진 파일로 인식x	사진 파일로 인식x
1.dll	사진 파일로 인식x	사진 파일로 인식x
1.css	사진 파일로 인식x	사진 파일로 인식x
1.html	사진 파일로 인식x	사진 파일로 인식x
1.doc	사진 파일로 인식x	사진 파일로 인식x
1.avi	사진 파일로 인식x	사진 파일로 인식x
1.css	사진 파일로 인식x	사진 파일로 인식x
1.ppt	사진 파일로 인식x	사진 파일로 인식x
1.eps	사진 파일로 인식x	사진 파일로 인식x
1.zip	사진 파일로 인식x	사진 파일로 인식x
1.egg	사진 파일로 인식x	사진 파일로 인식x

부연 설명

이 함수에서의 문제점은 파일을 인식하는 데에서는 없었으나 함수가 사진 파일을 인식하기 전까지 사용자에게 계속해서 "사진파일 또는 사진 폴더가 작업 폴더에 존재하지 않습니다. 사진 파일 또는 사진 폴더를 넣어주세요." 가 출력되는 (기획 : 1 번만 출력 -> ok 사인을 받으면 재확인) 문제점이 발생했다.

2.1.5 photochecker(fileName) Function#3 – 수정 후

C 드라이브에 생성한 작업폴더에 분류대상인 사진 파일이 존재하는 지 확인한다.

부분 설계 1

Jpg, bmp, jpeg, png, gif 파일이 있는지를 확인한다.

없다면 "사진파일 또는 사진 폴더가 작업 폴더에 존재하지 않습니다. 사진 파일 또는 사진 폴더를 넣어주세요."(= 출력1) 라는 메시지를 출력한다.

부분 설계 2

"사진 파일을 넣었으면 ok를 입력하여 주십시오."(= 출력 2) 를 출력하고 유효한 값을 입력 시 다음 작업으로 넘어가고, 유효하지 않은 값을 입력 시 "ok,Ok,OK,oK 중 하나를 입력하세요."(= 출력 3)을 출력하고 재입력을 받는다. 입력이 완료되면 "사진이 확인되었습니다." 출력(= 출력 4)

검사방법

mainModule.py를 Python shell에 로드한 후 프롬프트에 photochecker(fileame) 함수를 직접 호출하고 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적 1 : 작업 폴더 내 사진 파일 존재 유무 확인

입력 (호출)	예상결과	실제 결과	비고
1.jpg	사진 파일로 인식	사진 파일로 인식	
1.bmp	사진 파일로 인식	사진 파일로 인식	
1.jpeg	사진 파일로 인식	사진 파일로 인식	
1.png	사진 파일로 인식	사진 파일로 인식	
1.gif	사진 파일로 인식	사진 파일로 인식	
1.java	사진 파일로 인식x	사진 파일로 인식x	
1.mp4	사진 파일로 인식x	사진 파일로 인식x	
1.o	사진 파일로 인식x	사진 파일로 인식x	

1.so	사진 파일로 인식x	사진 파일로 인식x
1.mp3	사진 파일로 인식x	사진 파일로 인식x
1.c	사진 파일로 인식x	사진 파일로 인식x
1.cpp	사진 파일로 인식x	사진 파일로 인식x
1.dll	사진 파일로 인식x	사진 파일로 인식x
1.css	사진 파일로 인식x	사진 파일로 인식x
1.html	사진 파일로 인식x	사진 파일로 인식x
1.doc	사진 파일로 인식x	사진 파일로 인식x
1.avi	사진 파일로 인식x	사진 파일로 인식x
1.css	사진 파일로 인식x	사진 파일로 인식x
1.ppt	사진 파일로 인식x	사진 파일로 인식x
1.eps	사진 파일로 인식x	사진 파일로 인식x
1.zip	사진 파일로 인식x	사진 파일로 인식x
1.egg	사진 파일로 인식x	사진 파일로 인식x

목적 2 : 사진 파일 폴더에 삽입 후 다음 과정 실행 입력 (출력 2는 공통)

입력 (호출)	예상결과	실제 결과	비고
Ok	출력 4	출력 4	
ok	출력 4	출력 4	
OK	출력 4	출력 4	
oK	출력 4	출력 4	
O k	출력 3	출력 3	
O K	출력 3	출력 3	
okk	출력 3	출력 3	

Oking	출력 3	출력 3
.	출력 3	출력 3
₩	출력 3	출력 3
O k	출력 3	출력 3
BokB	출력 3	출력 3

부연 설명

출력 2 가 1 번만 출력되고, 사용자가 ok 입력을 할 시에 다시 재확인 하여 함수를 진행한다.

→내부코드 문제 수정 해결

2.1.6 numchecker() Function #4 – 수정 전

사용자에게 분류기준을 입력받는다.

부분 설계

1,2,3,4,5 중 1개이상 5개 이하의 숫자를(중복선택 불가) 입력받고 각 분류기준 번호에 대한 함수를 호출해준다. 올바르지 않은 입력 시 "올바른 형식의 입력이 아닙니다. 다시 입력하세요."(= 출력1) 출력하고 재입력을 받는다.

검사방법

mainModule.py를 Python shell에 로드한 후 프롬프트에 함수를 직접 호출하고 numchecker() 함수 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 분류 기준 입력 받기

입력 (호출)	예상결과	실제 결과	비고
1	1에 대한 함수 호출	1에 대한 함수 호출	
2	2에 대한 함수 호출	2에 대한 함수 호출	

3	3에 대한 함수 호출	3에 대한 함수 호출
4	4에 대한 함수 호출	4에 대한 함수 호출
5	5에 대한 함수 호출	5에 대한 함수 호출
1	1에 대한 함수 호출	1에 대한 함수 호출
1,2	1,2 순 함수 호출	1,2 순 함수 호출
2,4	2,4 순 함수 호출	2,4 순 함수 호출
5,1	5,1 순 함수 호출	5,1 순 함수 호출
1, 4	1,4 순 함수 호출	1,4 순 함수 호출
5 , 2	5,2 순 함수 호출	5,2 순 함수 호출
1,2,3	1,2,3 순 함수 호출	1,2,3 순 함수 호출
1 , 4, 5	1,4,5 순 함수 호출	1,4,5 순 함수 호출
5, 4 , 1	5,4,1 순 함수 호출	5,4,1 순 함수 호출
1,2,3,4	1,2,3,4 순 함수 호출	1,2,3,4 순 함수 호출
5,3,2,1	5,3,2,1 순 함수 호출	5,3,2,1 순 함수 호출
1, 2 , 3 ,5	1,2,3,5 순 함수 호출	1,2,3,5 순 함수 호출
1,4 ,2 , 3	1,4,2,3 순 함수 호출	1,4,2,3 순 함수 호출
1,2,3,4,5	1,2,3,4,5 순 함수 호출	1,2,3,4,5 순 함수 호출
5,3,2,1,4	5,3,1,2,4 순 함수 호출	5,3,1,2,4 순 함수 호출
1 ,3 , 2 , 4 , 5	1,3,2,4,5 순 함수 호출	1,3,2,4,5 순 함수 호출
	출력1	출력1
1,,2	출력1	출력1
1/2/3/4/5	출력1	출력1
One, two	출력1	출력1

,3,4	출력1	출력1	
!	출력1	출력1	
1,2,3,4,5,6,7,8,9,10	출력1	정상 작동 이후 6번부터 오류	정상작동x
Abc	출력1	출력1	
1,5, 10	출력1	정상 작동 이후 10번 오류	정상작동x
19	출력1	출력1	
가나다	출력1	출력1	
1,가,2,3,나	출력1	가,나에 대하여 오류 발생	정상작동x
가,1,2,3	출력1	출력1	
1,m,1,2,3,k	출력1	m,k에 대하여 오류발생	정상작동x
001,00002,00003	출력1	출력1	
1. , 2. , 3.	출력1	ValueError	정상작동x
1.0	출력1	ValueError	정상작동x

비고 내용

중복 입력에 대한 부분은 check_dupicate(number_array) 함수에서 다룬다.

첫번째 입력 이외의 입력에 올바른 입력과 올바르지 않은 입력이 뒤섞여도 정상작동으로 넘어간다. 또한 int 형 인자가 아닌 float 형 인자가 들어갔을 때 오류가 발생한다.

2.1.7 numchecker() Function #4 – 수정 후

사용자에게 분류기준을 입력받는다.

부분 설계

1,2,3,4,5 중 1개이상 5개 이하의 숫자를(중복선택 불가) 입력받고 각 분류기준 번호에 대한 함수를 호출해준다. 올바르지 않은 입력 시 "올바른 형식의 입력이 아닙니다. 다시 입력하세요."(= 출력1) 출력하고 재입력을 받는다.

검사방법

mainModule.py를 Python shell에 로드한 후 프롬프트에 함수를 직접 호출하고 numchecker() 함수 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 폴더명 중복 시 예외처리

입력 (호출)	예상결과	실제 결과	비고
1	1에 대한 함수 호출	1에 대한 함수 호출	
2	2에 대한 함수 호출	2에 대한 함수 호출	
3	3에 대한 함수 호출	3에 대한 함수 호출	
4	4에 대한 함수 호출	4에 대한 함수 호출	
5	5에 대한 함수 호출	5에 대한 함수 호출	
1	1에 대한 함수 호출	1에 대한 함수 호출	
1,2	1,2 순 함수 호출	1,2 순 함수 호출	
2,4	2,4 순 함수 호출	2,4 순 함수 호출	
5,1	5,1 순 함수 호출	5,1 순 함수 호출	
1, 4	1,4 순 함수 호출	1,4 순 함수 호출	
5 , 2	5,2 순 함수 호출	5,2 순 함수 호출	
1,2,3	1,2,3 순 함수 호출	1,2,3 순 함수 호출	
1 , 4, 5	1,4,5 순 함수 호출	1,4,5 순 함수 호출	
5, 4 , 1	5,4,1 순 함수 호출	5,4,1 순 함수 호출	
1,2,3,4	1,2,3,4 순 함수 호출	1,2,3,4 순 함수 호출	
5,3,2,1	5,3,2,1 순 함수 호출	5,3,2,1 순 함수 호출	
1, 2 , 3 ,5	1,2,3,5 순 함수 호출	1,2,3,5 순 함수 호출	
1,4 ,2 , 3	1,4,2,3 순 함수 호출	1,4,2,3 순 함수 호출	

1,2,3,4,5	1,2,3,4,5 순 함수 호출	1,2,3,4,5 순 함수 호출
5,3,2,1,4	5,3,1,2,4 순 함수 호출	5,3,1,2,4 순 함수 호출
1 ,3 , 2 , 4 , 5	1,3,2,4,5 순 함수 호출	1,3,2,4,5 순 함수 호출
	출력1	출력1
1,,2	출력1	출력1
1/2/3/4/5	출력1	출력1
One, two	출력1	출력1
,3,4	출력1	출력1
!	출력1	출력1
1,2,3,4,5,6,7,8,9,10	출력1	출력1
Abc	출력1	출력1
1,5, 10	출력1	출력1
19	출력1	출력1
가나다	출력1	출력1
1,가,2,3,나	출력1	출력1
가,1,2,3	출력1	출력1
1,m,1,2,3,k	출력1	출력1
001,00002,00003	출력1	출력1
1. , 2. , 3.	출력1	출력1
1.0	출력1	출력1

2.1.8 check_duplicate(number_array) Function #4 – 수정 無

사용자에게 분류기준을 입력받는 것에 대하여 중복검사를 수행한다.

부분 설계

중복되는 숫자를 입력 시 "중복된 입력입니다. 다시 입력하세요."(=출력 1)를 출력한다. 그리고 False를 반환한다.

이후 사용자에게 재입력을 받는다.

검사방법

mainModule.py를 Python shell에 로드한 후 프롬프트에 함수를 직접 호출하고

check_duplicate(nubmer_array) 함수리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 분류 기준 중복 입력 시 예외처리

입력 (호출)	예상결과	실제 결과	비고
1,1	출력 1	출력 1	
1,2,2	출력 1	출력 1	
1,3,4,3	출력 1	출력 1	
1, 3, 2, 1	출력 1	출력 1	
5,5,5,5,5	출력 1	출력 1	
4, 4.0, 3	출력 1	출력 1	
3,3.0,3.00,3.000	출력 1	출력 1	

2.1.9 standard_filename() Function #4 – 수정 無

파일 이름으로 분류할 때 분류 기준을 사용자에게 입력받는다.

부분 설계

유효한 기준 이외에 입력에 대하여 "유효한 값이 아닙니다. 혹은 2개 이상의 기준을 입력하셨습니다. 다시 입력해주세요."(= 출력1) 을 출력한다. 이후 재입력을 받는다. 유효한 기준을 입력 시 해당 기준에 대하여 분류 작업을 수행하는 함수를 불러온다.

검사방법

mainModule.py를 Python shell에 로드한 후 프롬프트에 함수를 직접 호출하고

standard_filename() 함수리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 파일 이름으로 분류 시 유효 값 이외의 값들에 대하여 예외처리

입력 (호출)	예상결과	실제 결과	비고
한글	변수값 넘겨줌	변수값 넘겨줌	정상작동
한	변수값 넘겨줌	변수값 넘겨줌	정상작동
kor	변수값 넘겨줌	변수값 넘겨줌	정상작동
KOREAN	변수값 넘겨줌	변수값 넘겨줌	정상작동
KOR	변수값 넘겨줌	변수값 넘겨줌	정상작동
영어	변수값 넘겨줌	변수값 넘겨줌	정상작동
영	변수값 넘겨줌	변수값 넘겨줌	정상작동
eng	변수값 넘겨줌	변수값 넘겨줌	정상작동
english	변수값 넘겨줌	변수값 넘겨줌	정상작동
Eng	변수값 넘겨줌	변수값 넘겨줌	정상작동
English	변수값 넘겨줌	변수값 넘겨줌	정상작동
ENGLISH	변수값 넘겨줌	변수값 넘겨줌	정상작동

숫자	변수값 넘겨줌	변수값 넘겨줌	정상작동
num	변수값 넘겨줌	변수값 넘겨줌	정상작동
number	변수값 넘겨줌	변수값 넘겨줌	정상작동
Number	변수값 넘겨줌	변수값 넘겨줌	정상작동
Num	변수값 넘겨줌	변수값 넘겨줌	정상작동
NUM	변수값 넘겨줌	변수값 넘겨줌	정상작동
NUMBER	변수값 넘겨줌	변수값 넘겨줌	정상작동
한글1	출력 1	출력 1	
한 영	출력 1	출력 1	
영0	출력 1	출력 1	
K1or	출력 1	출력 1	
KOR영어	출력 1	출력 1	
numengkor	출력 1	출력 1	
1	출력 1	출력 1	
?!?!?	출력 1	출력 1	
	출력 1	출력 1	
#	출력 1	출력 1	
%	출력 1	출력 1	
^	출력 1	출력 1	
Enumng	출력 1	출력 1	
한영글숫자	출력 1	출력 1	
숫자,num	출력 1	출력 1	
ㅎ	출력 1	출력 1	

2.1.10 date() Function #4 – 수정 無

분류 기준 3 ,4 번 선택 시 날짜 분류 기준에 대하여 연도별 분류를 할 것인지 월별 분류를 할 것인지 사용자에게 입력받는다..

부분 설계

유효한 기준 이외에 입력에 대하여 "유효한 값이 아닙니다. 혹은 2개 이상의 기준을 입력하셨습니다. 다시 입력해주세요."(= 출력 1) 을 출력한다. 이후 재입력을 받는다. 유효한 기준을 입력 시 해당 기준에 대하여 분류 작업을 수행하는 함수를 불러온다.

검사방법

mainModule.py를 Python shell에 로드한 후 프롬프트에 함수를 직접 호출하고

date() 함수리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 분류 기준 3,4 번에 대하여 유효값을 제외한 입력에 대한 예외처리

입력 (호출)	예상결과	실제 결과	비고
년	변수값 넘겨줌	변수값 넘겨줌	정상작동
년도	변수값 넘겨줌	변수값 넘겨줌	정상작동
Y	변수값 넘겨줌	변수값 넘겨줌	정상작동
Year	변수값 넘겨줌	변수값 넘겨줌	정상작동
y	변수값 넘겨줌	변수값 넘겨줌	정상작동
year	변수값 넘겨줌	변수값 넘겨줌	정상작동
YEAR	변수값 넘겨줌	변수값 넘겨줌	정상작동
월	변수값 넘겨줌	변수값 넘겨줌	정상작동
달	변수값 넘겨줌	변수값 넘겨줌	정상작동
M	변수값 넘겨줌	변수값 넘겨줌	정상작동
Month	변수값 넘겨줌	변수값 넘겨줌	정상작동

m	변수값 넘겨줌	변수값 넘겨줌	정상작동
MONTH	변수값 넘겨줌	변수값 넘겨줌	정상작동
	출력 1	출력 1	
년 월	출력 1	출력 1	
월달	출력 1	출력 1	
Ymym	출력 1	출력 1	
월 월 월	출력 1	출력 1	
ㅁ	출력 1	출력 1	
^	출력 1	출력 1	
!	출력 1	출력 1	
h	출력 1	출력 1	
間	출력 1	출력 1	
년,	출력 1	출력 1	
,년,	출력 1	출력 1	
(Space)	출력 1	출력 1	
month	출력 1	출력 1	
m m	출력 1	출력 1	

2.1.11 furtheration_check() Function #4 – 수정 無

사용자에게 추가 작업 여부를 묻고 추가 작업을 진행하는 유효값 입력 시 처음부터 프로그램을 다시 실행하며 추가 작업을 더 이상 진행하지 않는 유효값을 입력 시 작업을 종료한다.

부분 설계

유효한 기준 이외에 입력에 대하여 " 'Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO' 중 1개의 값을 입력해 주세요. "(=출력 1)을 출력한다. 이후 재입력을 받는다. 유효한 기준을 입력 시 추가 작업 진행 여부를 결정한다.

검사방법

mainModule.py를 Python shell에 로드한 후 프롬프트에 함수를 직접 호출하고 furtheratrion_check() 함수리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 폴더명 중복 시 예외처리

입력 (호출)	예상결과	실제 결과	비고
Y	추가작업 진행	추가작업 진행	
y	추가작업 진행	추가작업 진행	
Yes	추가작업 진행	추가작업 진행	
yes	추가작업 진행	추가작업 진행	
네	추가작업 진행	추가작업 진행	
예	추가작업 진행	추가작업 진행	
YES	추가작업 진행	추가작업 진행	
N	프로그램 종료	프로그램 종료	
n	프로그램 종료	프로그램 종료	
No	프로그램 종료	프로그램 종료	
no	프로그램 종료	프로그램 종료	
아니오	프로그램 종료	프로그램 종료	

아니요	프로그램 종료	프로그램 종료
NO	프로그램 종료	프로그램 종료
Nono	출력 1	출력 1
네아니오네	출력 1	출력 1
	출력 1	출력 1
(Space)	출력 1	출력 1
#	출력 1	출력 1
!Y	출력 1	출력 1
"no"	출력 1	출력 1
₩	출력 1	출력 1
龍	출력 1	출력 1
아니오 아니오	출력 1	출력 1
123	출력 1	출력 1
네니요	출력 1	출력 1

2.2FileMoverToRoot.py

2.2.1 __init__ 메소드 #1 (FileMoverToRoot class) – 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

검사방법

FileMoverToRoot 모듈(FileMoverToRoot.py)을 Pythonshell에 로드한 후 프롬프트에 __init__ 메소드를 직접 호출하고 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
FileMoverToRoot ('(1)')	TypeError 없음	TypeError 없음	
FileMoverToRoot ('(1)', '(2)')	TypeError	TypeError	
FileMoverToRoot ('(1)', '(2)', '(3)')	TypeError	TypeError	
FileMoverToRoot ('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
FileMoverToRoot (1)	AssertionError	AssertionError	assert문이우선시됨
FileMoverToRoot ()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
FileMoverToRoot (123)	AssertionError	AssertionError	
FileMoverToRoot (123.4)	AssertionError	AssertionError	
FileMoverToRoot(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
FileMoverToRoot('abc')	AssertionError 없음	AssertionError 없음	

2.2.2 hasLabel 메소드 #1 (FileMoverToRoot class) - 수정 전

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, 예외 없이 언제나 "파일명에 라벨이 포함되어 있는지" 여부를 진리값으로 리턴 해야함

검사방법

FileMoverToRoot 모듈(FileMoverToRoot.py)을 Pythonshell에 로드한 후 프롬프트에 hasLabel 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
hasLabel('(1)')	TypeError 없음	TypeError 없음	
hasLabel('(1)', '(2)')	TypeError	TypeError	
hasLabel('(1)', '(2)', '(3)')	TypeError	TypeError	
hasLabel('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
hasLabel(1)	AssertionError	AssertionError	assert문이 우선시됨
hasLabel()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
hasLabel(123)	AssertionError	AssertionError	
hasLabel(123.4)	AssertionError	AssertionError	
hasLabel(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
hasLabel('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (파일명에 라벨이 포함되어 있는지) 확인

입력 (호출)	예상결과	실제 결과	비고
hasLabel('a(1).txt')	True	True	라벨 '(1)' 이 존재함
hasLabel('a1).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a(1.txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a(1).')	True	False	라벨 '(1)' 이 존재하나 False 반환
hasLabel('a(1)')	True	True	라벨 '(1)' 이 존재함
hasLabel('(1).txt')	True	True	라벨 '(1)' 이 존재함
hasLabel('1).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('(1.txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('(1).')	True	False	라벨 '(1)' 이 존재하나 False 반환
hasLabel('(1)')	True	True	라벨 '(1)' 이 존재함
hasLabel('a(가).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a(!!!!).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a(a).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a( ).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a(1).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a(1).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a((1)).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음

2.2.3 hasLabel 메소드 #2 (FileMoverToRoot class) - 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

라벨 뒤에 '.' 만 있더라도 True 로 반환하도록 변경

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야 함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, 예외 없이 언제나 "파일명에 라벨이 포함되어 있는지" 여부를 진리값으로 리턴 해야함

검사방법

FileMoverToRoot 모듈(FileMoverToRoot.py)을 Pythonshell에 로드한 후 프롬프트에 hasLabel 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
hasLabel('(1)')	TypeError 없음	TypeError 없음	
hasLabel('(1)', '(2)')	TypeError	TypeError	
hasLabel('(1)', '(2)', '(3)')	TypeError	TypeError	
hasLabel('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
hasLabel(1)	AssertionError	AssertionError	assert문이 우선시됨
hasLabel()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
hasLabel(123)	AssertionError	AssertionError	
hasLabel(123.4)	AssertionError	AssertionError	
hasLabel(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선

hasLabel('abc')	AssertionError 없음	AssertionError 없음

목적: 결과조건 (파일명에 라벨이 포함되어 있는지) 확인

입력 (호출)	예상결과	실제 결과	비고
hasLabel('a(1).txt')	True	True	라벨 '(1)' 이 존재함
hasLabel('a1).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a(1.txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a(1).')	True	True	라벨 '(1)' 이 존재함
hasLabel('a(1)')	True	True	라벨 '(1)' 이 존재함
hasLabel('(1).txt')	True	True	라벨 '(1)' 이 존재함
hasLabel('1).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('(1.txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('(1).')	True	True	라벨 '(1)' 이 존재함
hasLabel('(1)')	True	True	라벨 '(1)' 이 존재함
hasLabel('a(가).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a(!!!!).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a(a).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a( ).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a(1).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a(1).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음
hasLabel('a((1)).txt')	False	False	라벨 형식 '(숫자)' 이 존재하지 않음

2.2.4 findLabelNumPos 메소드 #1 (FileMoverToRoot class) – 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야 함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, 예외 없이 언제나 "라벨의 '(' 이 있는 문자의 인덱스 번호와 라벨의 ')' 이 있는 문자 바로 이전 인덱스 번호" 를 정수형 인자 두개 튜플 타입으로 리턴 해야함.

만약 라벨이 존재하지 않을 경우 두개의 인자 (None, None) 을 반환함.

검사방법

FileMoverToRoot 모듈(FileMoverToRoot.py)을 Python shell에 로드 한 후 프롬프트에 findLabelNumPos 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
findLabelNumPos('(1)')	TypeError 없음	TypeError 없음	
findLabelNumPos ('(1)', '(2)')	TypeError	TypeError	
findLabelNumPos ('(1)', '(2)', '(3)')	TypeError	TypeError	
findLabelNumPos ('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
findLabelNumPos (1)	AssertionError	AssertionError	assert문이우선시됨
findLabelNumPos ()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
findLabelNumPos(123)	AssertionError	AssertionError	
findLabelNumPos(123.4)	AssertionError	AssertionError	

findLabelNumPos(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
findLabelNumPos('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (라벨이 존재하는 위치의 인덱스 반환) 확인

입력 (호출)	예상결과	실제 결과	비고
findLabelNumPos('a(1).txt')	(2,3)	(2,3)	라벨 '(1)' 이 존재함
findLabelNumPos('a1).txt')	(None, None)	(None, None)	라벨이 존재하지 않음
findLabelNumPos('a(1.txt')	(None, None)	(None, None)	라벨이 존재하지 않음
findLabelNumPos('a(1).')	(2,3)	(2,3)	라벨 '(1)' 이 존재함
findLabelNumPos('a(1)')	(2,3)	(2,3)	라벨 '(1)' 이 존재함
findLabelNumPos('(1).txt')	(1,2)	(1,2)	라벨 '(1)' 이 존재함
findLabelNumPos('1).txt')	(None, None)	(None, None)	라벨이 존재하지 않음
findLabelNumPos('(1.txt')	(None, None)	(None, None)	라벨이 존재하지 않음
findLabelNumPos('(1).')	(1,2)	(1,2)	라벨이 존재하지 않음
findLabelNumPos('(1)')	(1,2)	(1,2)	라벨 '(1)' 이 존재함
findLabelNumPos('a(가).txt')	(None, None)	(None, None)	라벨이 존재하지 않음
findLabelNumPos('a(!!!!).txt')	(None, None)	(None, None)	라벨이 존재하지 않음
findLabelNumPos('a(a).txt')	(None, None)	(None, None)	라벨이 존재하지 않음
findLabelNumPos('a( ).txt')	(None, None)	(None, None)	라벨이 존재하지 않음
findLabelNumPos('a(1).txt')	(None, None)	(None, None)	라벨이 존재하지 않음
findLabelNumPos('a(1).txt')	(None, None)	(None, None)	라벨이 존재하지 않음
findLabelNumPos ('a((1)).txt')	(None, None)	(None, None)	라벨이 존재하지 않음

2.2.5 getLabelNum 메소드 #1 (FileMoverToRoot class) – 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, 예외 없이 언제나 "라벨 안에 존재하는 정수" 를 정수형 값으로 리턴 해야 함

만약 라벨이 존재하지 않을 경우 None을 반환

검사방법

FileMoverToRoot 모듈(FileMoverToRoot.py)을 Pythonshell에 로드한 후 프롬프트에 getLabelNum 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
getLabelNum('(1)')	TypeError 없음	TypeError 없음	
getLabelNum('(1)', '(2)')	TypeError	TypeError	
getLabelNum('(1)', '(2)', '(3)')	TypeError	TypeError	
getLabelNum('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
getLabelNum(1)	AssertionError	AssertionError	assert문이 우선시됨
getLabelNum()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
getLabelNum(123)	AssertionError	AssertionError	
getLabelNum(123.4)	AssertionError	AssertionError	

getLabelNum(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
getLabelNum('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (파일명에 라벨의 번호가 반환) 확인

입력 (호출)	예상결과	실제 결과	비고
getLabelNum('a(1).txt')	1	1	라벨 '(1)' 이 존재함
getLabelNum('a1).txt')	None	None	라벨 형식 '(숫자)' 이 존재하지 않음
getLabelNum('a(1.txt')	None	None	라벨 형식 '(숫자)' 이 존재하지 않음
getLabelNum('a(1).')	1	1	라벨 '(1)' 이 존재하나 False 반환
getLabelNum('a(1)')	1	1	라벨 '(1)' 이 존재함
getLabelNum('(1).txt')	1	1	라벨 '(1)' 이 존재함
getLabelNum('1).txt')	None	None	라벨 형식 '(숫자)' 이 존재하지 않음
getLabelNum('(1.txt')	None	None	라벨 형식 '(숫자)' 이 존재하지 않음
getLabelNum('(1).')	1	1	라벨 '(1)' 이 존재하나 False 반환
getLabelNum('(1)')	1	1	라벨 '(1)' 이 존재함
getLabelNum('a(가).txt')	None	None	라벨 형식 '(숫자)' 이 존재하지 않음
getLabelNum('a(!!!!).txt')	None	None	라벨 형식 '(숫자)' 이 존재하지 않음
getLabelNum('a(a).txt')	None	None	라벨 형식 '(숫자)' 이 존재하지 않음
getLabelNum('a( ).txt')	None	None	라벨 형식 '(숫자)' 이 존재하지 않음
getLabelNum('a(1).txt')	None	None	라벨 형식 '(숫자)' 이 존재하지 않음
getLabelNum('a(1).txt')	None	None	라벨 형식 '(숫자)' 이 존재하지 않음
getLabelNum('a((1)).txt')	None	None	라벨 형식 '(숫자)' 이 존재하지 않음

2.2.6 findExtPos 메소드 #1 (FileMoverToRoot class) - 수정 전

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, 예외 없이 언제나 "확장자 앞에 붙는 '.' 의 인덱스 번호" 를 정수형 값으로 리턴 해야함

만약, 확장자가 존재하지 않으면 None을 반환함

검사방법

FileMoverToRoot 모듈(FileMoverToRoot.py)을 Pythonshell에 로드한 후 프롬프트에 findExtPos

메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
findExtPos('(1)')	TypeError 없음	TypeError 없음	
findExtPos('(1)', '(2)')	TypeError	TypeError	
findExtPos('(1)', '(2)', '(3)')	TypeError	TypeError	
findExtPos('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
findExtPos(1)	AssertionError	AssertionError	assert문이 우선시됨
findExtPos()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
findExtPos(123)	AssertionError	AssertionError	
findExtPos(123.4)	AssertionError	AssertionError	

findExtPos(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
findExtPos('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (파일명에 확장자의 인덱스 번호가 반환) 확인

입력 (호출)	예상결과	실제 결과	비고
findExtPos('a.txt')	1	1	마지막 '.' 뒤에 확장자 존재
findExtPos('a.')	None	None	마지막 '.' 뒤에 확장자 미존재
findExtPos('a.123')	1	1	마지막 '.' 뒤에 확장자 존재
findExtPos('a.abc')	1	1	마지막 '.' 뒤에 확장자 존재
findExtPos('a.()')	1	1	마지막 '.' 뒤에 확장자 존재
findExtPos('a.....')	None	1	'.' 뒤에 '.'은 확장자가 아님
findExtPos('a...')	None	1	'.' 뒤에 '.'은 확장자가 아님
findExtPos('a')	None	None	마지막 '.' 뒤에 확장자 미존재
findExtPos('a.abc.a')	5	1	마지막 '.' 뒤에 확장자 존재
findExtPos('a.ab.ab')	4	1	마지막 '.' 뒤에 확장자 인식 오류
findExtPos('a.ab.ab.ab')	7	1	마지막 '.' 뒤에 확장자 인식 오류
findExtPos('a_._._')	4	1	마지막 '.' 뒤에 확장자 인식 오류
findExtPos('..a')	1	0	마지막 '.' 뒤에 확장자 인식 오류
findExtPos('.a')	0	0	마지막 '.' 뒤에 확장자 존재
findExtPos('...a')	2	0	마지막 '.' 뒤에 확장자 인식 오류
findExtPos('a...a')	3	1	마지막 '.' 뒤에 확장자 인식 오류
findExtPos('a(1)..)	None	1	'.' 뒤에 '.'은 확장자가 아님
findExtPos('a(1).t')	4	4	마지막 '.' 뒤에 확장자 존재
findExtPos('a(1).(1).t')	8	4	마지막 '.' 뒤에 확장자 인식 오류

2.2.7 findExtPos 메소드 #1 (FileMoverToRoot class) – 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

가장 마지막 '.' 뒤가 확장자 명이라는 조건 아래 수정

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, 예외 없이 언제나 "확장자 앞에 붙는 '.' 의 인덱스 번호" 를 정수형 값으로 리턴 해야함

만약, 확장자가 존재하지 않으면 None을 반환함

검사방법

FileMoverToRoot 모듈(FileMoverToRoot.py)을 Pythonshell에 로드한 후 프롬프트에 findExtPos

메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
findExtPos('(1)')	TypeError 없음	TypeError 없음	
findExtPos('(1)', '(2)')	TypeError	TypeError	
findExtPos('(1)', '(2)', '(3)')	TypeError	TypeError	
findExtPos('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
findExtPos(1)	AssertionError	AssertionError	assert문이 우선시됨
findExtPos()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
findExtPos(123)	AssertionError	AssertionError	

findExtPos(123.4)	AssertionError	AssertionError	
findExtPos(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
findExtPos('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (파일명에 확장자의 인덱스 번호가 반환) 확인

입력 (호출)	예상결과	실제 결과	비고
findExtPos('a.txt')	1	1	마지막 '.' 뒤에 확장자 존재
findExtPos('a.')	None	None	마지막 '.' 뒤에 확장자 미존재
findExtPos('a.123')	1	1	마지막 '.' 뒤에 확장자 존재
findExtPos('a.abc')	1	1	마지막 '.' 뒤에 확장자 존재
findExtPos('a.()')	1	1	마지막 '.' 뒤에 확장자 존재
findExtPos('a.....')	None	None	'.' 뒤에 '.'은 확장자가 아님
findExtPos('a...')	None	None	'.' 뒤에 '.'은 확장자가 아님
findExtPos('a')	None	None	마지막 '.' 뒤에 확장자 미존재
findExtPos('a.abc.a')	5	5	마지막 '.' 뒤에 확장자 존재
findExtPos('a.ab.ab')	4	4	마지막 '.' 뒤에 확장자 존재
findExtPos('a.ab.ab.ab')	7	7	마지막 '.' 뒤에 확장자 존재
findExtPos('a_._._')	4	4	마지막 '.' 뒤에 확장자 존재
findExtPos('..a')	1	1	마지막 '.' 뒤에 확장자 존재
findExtPos('.a')	0	0	마지막 '.' 뒤에 확장자 존재
findExtPos('...a')	2	2	마지막 '.' 뒤에 확장자 존재
findExtPos('a...a')	3	3	마지막 '.' 뒤에 확장자 존재
findExtPos('a(1)..)	None	None	'.' 뒤에 '.'은 확장자가 아님
findExtPos('a(1).t')	4	4	마지막 '.' 뒤에 확장자 존재

findExtPos('a(1).(1).t')	8	8	마지막 '.' 뒤에 확장자 존재

2.2.8 hasExt 메소드 #1 (FileMoverToRoot class) – 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, 예외 없이 언제나 "확장의 유무" 를 진리 값으로 리턴 해야함

검사방법

FileMoverToRoot 모듈(FileMoverToRoot.py)을 Pythonshell에 로드한 후 프롬프트에 hasExt 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
hasExt('(1)')	TypeError 없음	TypeError 없음	
hasExt('(1)', '(2)')	TypeError	TypeError	
hasExt('(1)', '(2)', '(3)')	TypeError	TypeError	
hasExt('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
hasExt(1)	AssertionError	AssertionError	assert문이 우선시됨
hasExt()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
hasExt(123)	AssertionError	AssertionError	

hasExt(123.4)	AssertionError	AssertionError	
hasExt(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
hasExt('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (파일명에 확장자가 포함되어 있는지) 확인

입력 (호출)	예상결과	실제 결과	비고
hasExt('a.txt')	True	True	마지막 '.' 뒤에 확장자 존재
hasExt('a.')	False	False	마지막 '.' 뒤에 확장자 미존재
hasExt('a.123')	True	True	마지막 '.' 뒤에 확장자 존재
hasExt('a.abc')	True	True	마지막 '.' 뒤에 확장자 존재
hasExt('a.()')	True	True	마지막 '.' 뒤에 확장자 존재
hasExt('a.....')	False	False	'.' 뒤에 '.'은 확장자가 아님
hasExt('a...')	False	False	'.' 뒤에 '.'은 확장자가 아님
hasExt('a')	False	False	마지막 '.' 뒤에 확장자 미존재
hasExt('a.abc.a')	True	True	마지막 '.' 뒤에 확장자 존재
hasExt('a.ab.ab')	True	True	마지막 '.' 뒤에 확장자 존재
hasExt('a.ab.ab.ab')	True	True	마지막 '.' 뒤에 확장자 존재
hasExt('a_._._')	True	True	마지막 '.' 뒤에 확장자 존재
hasExt('..a')	True	True	마지막 '.' 뒤에 확장자 존재
hasExt('.a')	True	True	마지막 '.' 뒤에 확장자 존재
hasExt('...a')	True	True	마지막 '.' 뒤에 확장자 존재
hasExt('a...a')	True	True	마지막 '.' 뒤에 확장자 존재
hasExt('a(1)..)	False	False	'.' 뒤에 '.'은 확장자가 아님
hasExt('a(1).t')	True	True	마지막 '.' 뒤에 확장자 존재

hasExt('a(1).(1).t')	True	True	마지막 '.' 뒤에 확장자 존재

2.2.9 extractExt 메소드 #1 (FileMoverToRoot class) – 수정 전

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, 예외 없이 언제나 "확장자('.' 포함)" 를 문자열 값으로 리턴 해야함

존재하지 않을 경우 빈 문자열('')을 반환함

검사방법

FileMoverToRoot 모듈(FileMoverToRoot.py)을 Pythonshell에 로드한 후 프롬프트에 extractExt 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
extractExt('(1)')	TypeError 없음	TypeError 없음	
extractExt('(1)', '(2)')	TypeError	TypeError	
extractExt('(1)', '(2)', '(3)')	TypeError	TypeError	
extractExt('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
extractExt(1)	AssertionError	AssertionError	assert문이 우선시됨
extractExt()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고

extractExt(123)	AssertionError	AssertionError	
extractExt(123.4)	AssertionError	AssertionError	
extractExt(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
extractExt('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (파일명에 확장자가 포함되어 있는지) 확인

입력 (호출)	예상결과	실제 결과	비고
extractExt('a.txt')	'.txt'	'.txt'	마지막 '.' 뒤에 확장자 존재
extractExt('a.')	''	'.'	마지막 '.' 뒤에 확장자 미존재 (오류)
extractExt('a.123')	'.123'	'.123'	마지막 '.' 뒤에 확장자 존재
extractExt('a.abc')	'.abc'	'.abc'	마지막 '.' 뒤에 확장자 존재
extractExt('a.()')	'.()'	'.()'	마지막 '.' 뒤에 확장자 존재
extractExt('a.....')	''	'.'	'.' 뒤에 '.'은 확장자가 아님 (오류)
extractExt('a...')	''	'.'	'.' 뒤에 '.'은 확장자가 아님 (오류)
extractExt('a')	''	''	마지막 '.' 뒤에 확장자 미존재
extractExt('a.abc.a')	'.a'	'.a'	마지막 '.' 뒤에 확장자 존재
extractExt('a.ab.ab')	'.ab'	'.ab'	마지막 '.' 뒤에 확장자 존재
extractExt('a.ab.ab.ab')	'.ab'	'.ab'	마지막 '.' 뒤에 확장자 존재
extractExt('a_._._')	'._'	'._'	마지막 '.' 뒤에 확장자 존재
extractExt('..a')	'.a'	'.a'	마지막 '.' 뒤에 확장자 존재
extractExt('.a')	'.a'	'.a'	마지막 '.' 뒤에 확장자 존재
extractExt('...a')	'.a'	'.a'	마지막 '.' 뒤에 확장자 존재
extractExt('a...a')	'.a'	'.a'	마지막 '.' 뒤에 확장자 존재
extractExt('a(1)..)	''	'.'	'.' 뒤에 '.'은 확장자가 아님 (오류)

extractExt('a(1).t')	'.t'	'.t'	마지막 '.' 뒤에 확장자 존재
extractExt('a(1).(1).t')	'.t'	'.t'	마지막 '.' 뒤에 확장자 존재

2.2.10 extractExt 메소드 #2 (FileMoverToRoot class) – 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

마지막에 연속된 '.' 에 대하여 올바르게 확장자가 추출되도록 수정

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, 예외 없이 언제나 "확장자('.' 포함)" 를 문자열 값으로 리턴 해야함

존재하지 않을 경우 빈 문자열('')을 반환함

검사방법

FileMoverToRoot 모듈(FileMoverToRoot.py)을 Pythonshell에 로드한 후 프롬프트에 extractExt 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
extractExt('(1)')	TypeError 없음	TypeError 없음	
extractExt('(1)', '(2)')	TypeError	TypeError	
extractExt('(1)', '(2)', '(3)')	TypeError	TypeError	
extractExt('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
extractExt(1)	AssertionError	AssertionError	assert문이 우선시됨
extractExt()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
extractExt(123)	AssertionError	AssertionError	
extractExt(123.4)	AssertionError	AssertionError	
extractExt(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
extractExt('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (파일명에 확장자가 포함되어 있는지) 확인

입력 (호출)	예상결과	실제 결과	비고
extractExt('a.txt')	'.txt'	'.txt'	마지막 '.' 뒤에 확장자 존재
extractExt('a.')	''	''	마지막 '.' 뒤에 확장자 미존재
extractExt('a.123')	'.123'	'.123'	마지막 '.' 뒤에 확장자 존재
extractExt('a.abc')	'.abc'	'.abc'	마지막 '.' 뒤에 확장자 존재
extractExt('a.()')	'.()'	'.()'	마지막 '.' 뒤에 확장자 존재
extractExt('a.....')	''	''	'.' 뒤에 '.'은 확장자가 아님
extractExt('a...')	''	''	'.' 뒤에 '.'은 확장자가 아님
extractExt('a')	''	''	마지막 '.' 뒤에 확장자 미존재
extractExt('a.abc.a')	'.a'	'.a'	마지막 '.' 뒤에 확장자 존재
extractExt('a.ab.ab')	'.ab'	'.ab'	마지막 '.' 뒤에 확장자 존재
extractExt('a.ab.ab.ab')	'.ab'	'.ab'	마지막 '.' 뒤에 확장자 존재
extractExt('a_._._')	'._'	'._'	마지막 '.' 뒤에 확장자 존재
extractExt('..a')	'.a'	'.a'	마지막 '.' 뒤에 확장자 존재
extractExt('.a')	'.a'	'.a'	마지막 '.' 뒤에 확장자 존재
extractExt('...a')	'.a'	'.a'	마지막 '.' 뒤에 확장자 존재

extractExt('a...a')	'.a'	'.a'	마지막 '.' 뒤에 확장자 존재
extractExt('a(1)..)	''	''	'.' 뒤에 '.'은 확장자가 아님
extractExt('a(1).t')	'.t'	'.t'	마지막 '.' 뒤에 확장자 존재
extractExt('a(1).(1).t')	'.t'	'.t'	마지막 '.' 뒤에 확장자 존재

2.2.11 uniqFileNameParser 메소드 #1 (FileMoverToRoot class) – 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, 예외 없이 언제나 "라벨을 제외한 파일 명" 을 문자열 값으로 리턴 해야함

검사방법

FileMoverToRoot 모듈(FileMoverToRoot.py)을 Pythonshell에 로드한 후 프롬프트에 uniqFileNameParser 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
uniqFileNameParser('(1)')	TypeError 없음	TypeError 없음	
uniqFileNameParser('(1)', '(2)')	TypeError	TypeError	
uniqFileNameParser('(1)', '(2)', '(3)')	TypeError	TypeError	
uniqFileNameParser('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
uniqFileNameParser(1)	AssertionError	AssertionError	assert문이우선시됨
uniqFileNameParser()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
uniqFileNameParser(123)	AssertionError	AssertionError	
uniqFileNameParser(123.4)	AssertionError	AssertionError	
uniqFileNameParser(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
uniqFileNameParser('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (라벨을 제외한 파일 명을 반환) 확인

입력 (호출)	예상결과	실제 결과	비고
uniqFileNameParser('a.txt')	'.txt'	'.txt'	라벨이 존재X => 그대로
uniqFileNameParser('a.')	'a.'	'a.'	라벨이 존재X => 그대로
uniqFileNameParser('a.123')	'.123'	'.123'	라벨이 존재X => 그대로
uniqFileNameParser('a.abc')	'a.abc'	'a.abc'	라벨이 존재X => 그대로
uniqFileNameParser('a.()')	'a.()'	'a.()'	라벨이 존재X => 그대로
uniqFileNameParser('a.....')	'a.....'	'a.....'	라벨이 존재X => 그대로
uniqFileNameParser('a...')	'a...'	'a...'	라벨이 존재X => 그대로
uniqFileNameParser('a')	'a'	'a'	라벨이 존재X => 그대로
uniqFileNameParser('a.abc.a')	'a.abc.a'	'a.abc.a'	라벨이 존재X => 그대로
uniqFileNameParser('a.ab.ab')	'a.ab.ab'	'a.ab.ab'	라벨이 존재X => 그대로
uniqFileNameParser('a.ab.ab.ab')	'a.ab.ab.ab'	'a.ab.ab.ab'	라벨이 존재X => 그대로
uniqFileNameParser('a_._._')	'a_._._'	'a_._._'	라벨이 존재X => 그대로
uniqFileNameParser('..a')	'..a'	'..a'	라벨이 존재X => 그대로
uniqFileNameParser('.a')	'.a'	'.a'	라벨이 존재X => 그대로
uniqFileNameParser('...a')	'...a'	'...a'	라벨이 존재X => 그대로
uniqFileNameParser('a...a')	'a...a'	'a...a'	라벨이 존재X => 그대로

uniqFileNameParser('a(1)..')	'a..'	'a..'	라벨이 존재X => 그대로
uniqFileNameParser('a(1).t')	'a.t'	'a.t'	라벨이 존재X => 그대로
uniqFileNameParser('a(1).(1).t')	'a.(1).t'	'a.(1).t'	라벨이 존재 => 라벨 제거
uniqFileNameParser('a1).txt')	'a1).txt'	'a1).txt'	라벨이 존재X => 그대로
uniqFileNameParser('a(1.txt')	'a(1.txt'	'a(1.txt'	라벨이 존재X => 그대로
uniqFileNameParser('a(1).')	'a(1).'	'a.'	라벨이 존재 => 라벨 제거
uniqFileNameParser('(1).txt')	'(1).txt'	'.txt'	라벨이 존재 => 라벨 제거
uniqFileNameParser('a( ).txt')	'a( ).txt'	'a( ).txt'	라벨이 존재X => 그대로
uniqFileNameParser('a((1)).txt')	'a((1)).txt'	'a((1)).txt'	라벨이 존재X => 그대로

2.2.12 isDuplicatedName 메소드 #1 (FileMoverToRoot class) – 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, 예외 없이 언제나 "인자로 넣어진 파일이름이 parsedFileNameDict 리스트 안에 존재하는가" 을 진리 값으로 리턴 해야함

검사방법

FileMoverToRoot 모듈(FileMoverToRoot.py)을 Pythonshell에 로드한 후 프롬프트에 isDuplicatedName 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
isDuplicatedName('(1)')	TypeError 없음	TypeError 없음	
isDuplicatedName('(1)', '(2)')	TypeError	TypeError	

isDuplicatedName('(1)', '(2)', '(3)')	TypeError	TypeError	
isDuplicatedName('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
isDuplicatedName(1)	AssertionError	AssertionError	assert문이우선시됨
isDuplicatedName()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
isDuplicatedName(123)	AssertionError	AssertionError	
isDuplicatedName(123.4)	AssertionError	AssertionError	
isDuplicatedName(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
isDuplicatedName('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (라벨을 제외한 파일 명을 반환) 확인

가정: parsedFileNameDict 리스트에 'a', 'a.txt' 가 존재한다고 가정함

입력 (호출)	예상결과	실제 결과	비고
isDuplicatedName('a.txt')	True	True	리스트 안에 존재
isDuplicatedName('a.')	False	False	리스트 안에 존재하지 않음
isDuplicatedName('a.123')	False	False	리스트 안에 존재하지 않음
isDuplicatedName('a.abc')	False	False	리스트 안에 존재하지 않음
isDuplicatedName('a.()')	False	False	리스트 안에 존재하지 않음
isDuplicatedName('a.....')	False	False	리스트 안에 존재하지 않음
isDuplicatedName('a...')	False	False	리스트 안에 존재하지 않음
isDuplicatedName('a.txt')	True	True	리스트 안에 존재

2.3FileDataCollector.py

2.3.1 __init__ 메소드 #1 (FileDataCollector class) – 최종

첫번째 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

두번째 매개변수는 가변 매개변수(튜플)로 받으며, ('FILE_SIZE', 'FILE_NAME', 'CREATED_DATE', 'MODIFIED_DATE', 'EXTENSION') 중 최소한 1개의 값과 일치 하지 않는 것에 대해 assert문을 제시함

부분 설계

정확히 두개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

두번째 인자는 0개이상 5개이하인 가변 매개변수 값으로 받으며, ('FILE_SIZE', 'FILE_NAME', 'CREATED_DATE', 'MODIFIED_DATE', 'EXTENSION') 중 최소한 1개의 값과 일치 하지 않으면 AssertionError 메시지를 출력해야 함

검사방법

FileDataCollector 모듈(FileDataCollector.py)을 Pythonshell에 로드한 후 프롬프트에 __init__ 메소드를 직접 호출하고 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 2개)확인

입력 (호출)	예상결과	실제 결과	비고
FileDataCollector ('(1)')	AssertionError	AssertionError	assert문이우선시됨
FileDataCollector ('(1)', '(2)')	TypeError 없음	TypeError 없음	
FileDataCollector ('(1)', '(2)', '(3)')	TypeError 없음	TypeError 없음	
FileDataCollector ('(1)', '(2)', '(3)', '(4)')	TypeError 없음	TypeError 없음	
FileDataCollector (1)	AssertionError	AssertionError	assert문이우선시됨
FileDataCollector ()	TypeError	TypeError	
목적: 첫번째 인자조건 (문자열) 확인			
입력 (호출)	예상결과	실제 결과	비고

FileDataCollector (123, 'FILE_SIZE')	AssertionError	AssertionError	
FileDataCollector (123.4, 'FILE_SIZE')	AssertionError	AssertionError	
FileDataCollector(!!!!!, 'FILE_SIZE')	SyntaxError	SyntaxError	assert문조건보다 우선
FileDataCollector('abc', 'FILE_SIZE')	AssertionError 없음	AssertionError 없음	

목적: 두번째 인자조건 (유효한 문자열) 확인

지면상, FileDataCollector 을 FDC 로 대체, AssertionError 을 AE 로 대체

입력 (호출)	예상결과	실제 결과	비고
FileDataCollector ('path', 'FILE_SIZE')	AE 없음	AE 없음	
FileDataCollector ('path', 'FILE_NAME')	AE 없음	AE 없음	
FileDataCollector ('path', 'CREATED_DATE')	AE 없음	AE 없음	
FileDataCollector ('path', 'MODIFIED_DATE')	AE 없음	AE 없음	
FileDataCollector ('path', 'EXTENSION')	AE 없음	AE 없음	
FileDataCollector ('path', '')	AE	AE	유효한 mode X
FileDataCollector ('path', '1')	AE	AE	유효한 mode X
FileDataCollector ('path', 123)	AE	AE	유효한 mode X
FileDataCollector ('path', '가')	AE	AE	유효한 mode X
FileDataCollector ('path', 'a')	AE	AE	유효한 mode X
FileDataCollector ('path', 'FILE_SIZE', 'FILE_NAME')	AE 없음	AE 없음	
FileDataCollector ('path', 'FILE_SIZE', 'FILE_NAME', 'CREATED_DATE', 'MODIFIED_DATE', 'EXTENSION')	AE 없음	AE 없음	
FileDataCollector ('path', 'FILE_SIZE', 'FILE_NAME', 'CREATED_DATE', 'MODIFIED_DATE', 'EXTENSION', 'aaa')	AE	AE	두번째 가변 매개변수가 5개 초과함

2.3.2 isVaildMode 메소드 #1 (FileDataCollector class) – 최종

하나의 매개변수에 assert 문을 통하여, 튜플이라는 조건만 건 상태

그리고, 튜플의 원소는 1개 이상 5개이하여야 함.

또한, 튜플의 원소는 ('FILE_SIZE', 'FILE_NAME', 'CREATED_DATE', 'MODIFIED_DATE', 'EXTENSION') 중 최소한 1개의 값과 일치하여야 함.

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 튜플 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자는 반드시 ('FILE_SIZE', 'FILE_NAME', 'CREATED_DATE', 'MODIFIED_DATE', 'EXTENSION') 중 최소한 1개의 값과 일치하여야 함. 만일 위배될 경우 AssertionError 메시지를 출력.

검사방법

FileDataCollector 모듈(FileDataCollector.py)을 Pythonshell에 로드한 후 프롬프트에 isVaildMode 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
isVaildMode('(1)')	TypeError 없음	TypeError 없음	
isVaildMode('(1)', '(2)')	TypeError	TypeError	
isVaildMode('(1)', '(2)', '(3)')	TypeError	TypeError	
isVaildMode('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
isVaildMode(1)	AssertionError	AssertionError	assert문이 우선시됨
isVaildMode()	TypeError	TypeError	

목적: 첫번째 인자조건 (튜플) 확인

입력 (호출)	예상결과	실제 결과	비고

isVaildMode(123)	AssertionError	AssertionError	
isVaildMode(123.4)	AssertionError	AssertionError	
isVaildMode(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
isVaildMode('abc')	AssertionError	AssertionError	
isVaildMode(('abc',))	AssertionError 없음	AssertionError 없음	

목적: 첫번째 인자조건 (튜플 원소들의 값이 유효) 확인

('FILE_SIZE', 'FILE_NAME', 'CREATED_DATE', 'MODIFIED_DATE', 'EXTENSION')

입력 (호출)	예상결과	실제 결과	비고
isVaildMode((123,))	AssertionError	AssertionError	
isVaildMode((123.4,))	AssertionError	AssertionError	
isVaildMode((!!!!!,))	AssertionError	AssertionError	
isVaildMode(('abc',))	AssertionError	AssertionError	
isVaildMode(('FILE_SIZE',))	AssertionError 없음	AssertionError 없음	
isVaildMode(('FILE_NAME',))	AssertionError 없음	AssertionError 없음	
isVaildMode(('CREATED_DATE',))	AssertionError 없음	AssertionError 없음	
isVaildMode(('MODIFIED_DATE',))	AssertionError 없음	AssertionError 없음	
isVaildMode(('EXTENSION',))	AssertionError 없음	AssertionError 없음	
isVaildMode(('MODIFIED_DATE', 'FILE_NAME'))	AssertionError 없음	AssertionError 없음	
isVaildMode(('MODIFIED_DATE', 'FILE_NAME', 'CREATED_DATE'))	AssertionError 없음	AssertionError 없음	
isVaildMode(('FILE_SIZE', 'FILE_NAME', 'CREATED_DATE', 'MODIFIED_DATE', 'EXTENSION'))	AssertionError 없음	AssertionError 없음	

isVaildMode(('FILE_SIZE', 'FILE_NAME', 'CREATED_DATE', 'MODIFIED_DATE', 'EXTENSION', 'a'))	AssertionError AssertionError

2.3.3 getFileSize 메소드 #1 (FileDataCollector class) – 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, "제시된 경로에 존재하는 파일의 크기" 를 double 값으로 리턴 해야함

만약, 유효하지 않은 경로, 파일에 대해서는 아무런 값도 반환을 하지 않음.

검사방법

FileDataCollector 모듈(FileDataCollector.py)을 Pythonshell에 로드한 후 프롬프트에 getFileSize 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
getFileSize('(1)')	TypeError 없음	TypeError 없음	
getFileSize('(1)', '(2)')	TypeError	TypeError	
getFileSize('(1)', '(2)', '(3)')	TypeError	TypeError	
getFileSize('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
getFileSize(1)	AssertionError	AssertionError	assert문이 우선시됨
getFileSize()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
getFileSize(123)	AssertionError	AssertionError	

getFileSize(123.4)	AssertionError	AssertionError	
getFileSize(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
getFileSize('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (제시된 경로에 존재하는 파일의 크기) 확인

조건: 경로 'C:/aaa.txt'에 존재하는 1MB 크기의 txt 파일

입력 (호출)	예상결과	실제 결과	비고
getFileSize('')	리턴 X	리턴 X	유효하지 않은 경로임
getFileSize('asd')	리턴 X	리턴 X	유효하지 않은 경로임
getFileSize('D:/')	리턴 X	리턴 X	유효하지 않은 경로임
getFileSize('C:/a')	리턴 X	리턴 X	유효하지 않은 경로임
getFileSize('C:/aaa.txt")	1.0	1.0	파일이 존재함
getFileSize(C:/a.txt')	리턴 X	리턴 X	존재하지 않는 파일임

2.3.4 getCreatedDate 메소드 #1 (FileDataCollector class) – 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, "제시된 경로에 존재하는 파일이 생성된 날짜" 를 (월)-(달) 형태를 지닌 문자열 값으로 리턴 해야함

만약, 유효하지 않은 경로, 파일에 대해서는 아무런 값도 반환을 하지 않음.

검사방법

FileDataCollector 모듈(FileDataCollector.py)을 PythonshelI에 로드한 후 프롬프트에 getCreatedDate 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
getCreatedDate('(1)')	TypeError 없음	TypeError 없음	
getCreatedDate('(1)', '(2)')	TypeError	TypeError	
getCreatedDate('(1)', '(2)', '(3)')	TypeError	TypeError	
getCreatedDate('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
getCreatedDate(1)	AssertionError	AssertionError	assert문이 우선시됨
getCreatedDate()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
getCreatedDate(123)	AssertionError	AssertionError	
getCreatedDate(123.4)	AssertionError	AssertionError	
getCreatedDate(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
getCreatedDate('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (제시된 경로에 존재하는 파일의 크기) 확인

조건: 경로 'C:/aaa.txt'에 존재하고, 2019 년 10 월에 생성된 txt 파일

입력 (호출)	예상결과	실제 결과	비고
getCreatedDate('')	리턴 X	리턴 X	유효하지 않은 경로임
getCreatedDate('asd')	리턴 X	리턴 X	유효하지 않은 경로임
getCreatedDate('D:/')	리턴 X	리턴 X	유효하지 않은 경로임
getCreatedDate('C:/a')	리턴 X	리턴 X	유효하지 않은 경로임
getCreatedDate('C:/aaa.txt")	2019-10	2019-10	파일이 존재함

getCreatedDate(C:/a.txt')	리턴 X	리턴 X	존재하지 않는 파일임

2.3.5 getModifiedDate 메소드 #1 (FileDataCollector class) – 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, "제시된 경로에 존재하는 파일이 변경된 날짜" 를 (월)-(달) 형태를 지닌 문자열 값으로 리턴 해야함

만약, 유효하지 않은 경로, 파일에 대해서는 아무런 값도 반환을 하지 않음.

검사방법

FileDataCollector 모듈(FileDataCollector.py)을 Pythonshell에 로드한 후 프롬프트에 getModifiedDate 메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
getModifiedDate('(1)')	TypeError 없음	TypeError 없음	
getModifiedDate('(1)', '(2)')	TypeError	TypeError	
getModifiedDate('(1)', '(2)', '(3)')	TypeError	TypeError	
getModifiedDate('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
getModifiedDate(1)	AssertionError	AssertionError	assert문이 우선시됨
getModifiedDate()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고

getModifiedDate(123)	AssertionError	AssertionError	
getModifiedDate(123.4)	AssertionError	AssertionError	
getModifiedDate(!!!!!)	SyntaxError	SyntaxError	assert문 조건보다 우선
getModifiedDate('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (제시된 경로에 존재하는 파일의 크기) 확인

조건: 경로 'C:/aaa.txt'에 존재하고, 2019 년 10 월에 변경된 txt 파일

입력 (호출)	예상결과	실제 결과	비고
getModifiedDate('')	리턴 X	리턴 X	유효하지 않은 경로임
getModifiedDate('asd')	리턴 X	리턴 X	유효하지 않은 경로임
getModifiedDate('D:/')	리턴 X	리턴 X	유효하지 않은 경로임
getModifiedDate('C:/a')	리턴 X	리턴 X	유효하지 않은 경로임
getModifiedDate('C:/aaa.txt")	2019-10	2019-10	파일이 존재함
getModifiedDate(C:/a.txt')	리턴 X	리턴 X	존재하지 않는 파일임

2.3.6 getExtension 메소드 #1 (FileDataCollector class) – 최종

하나의 매개변수에 assert 문을 통하여, 문자열이라는 조건만 건 상태

부분 설계

정확히 한개의 인자를 받아야함. 만일 인자의 개수가 틀릴 경우 TypeError 메시지를 출력해야 함.

첫번째 인자는 반드시 문자열 이어야함. 만일 위배될 경우 AssertionError 메시지를 출력해야 함.

첫번째 인자에 대해서, "확장자가 포함된 파일이름" 을 확장자만('.' 포함) 지닌 문자열 값으로 리턴 해야함

만약, 유효하지 않은 경로, 파일에 대해서는 아무런 값도 반환을 하지 않음.

검사방법

FileDataCollector 모듈(FileDataCollector.py)을 PythonshellI에 로드한 후 프롬프트에 getExtension

메소드를 직접 호출하고 리턴 값 및 오류 메시지 확인

Test Case 및 결과 정리

목적: 인자 개수 조건(정확히 1개)확인

입력 (호출)	예상결과	실제 결과	비고
getExtension('(1)')	TypeError 없음	TypeError 없음	
getExtension('(1)', '(2)')	TypeError	TypeError	
getExtension('(1)', '(2)', '(3)')	TypeError	TypeError	
getExtension('(1)', '(2)', '(3)', '(4)')	TypeError	TypeError	
getExtension(1)	AssertionError	AssertionError	assert문이 우선시됨
getExtension()	TypeError	TypeError	

목적: 첫번째 인자조건 (문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
getExtension(123)	AssertionError	AssertionError	
getExtension(123.4)	AssertionError	AssertionError	
getExtension(!!!!!) SyntaxError	SyntaxError	assert문 조건보다 우선	
getExtension('abc')	AssertionError 없음	AssertionError 없음	

목적: 결과조건 (제시된 경로에 존재하는 파일의 크기) 확인

조건: 경로 'C:/' 에 'a.txt', 'a.e', 'a', '.a' 파일이 존재함

입력 (호출)	예상결과	실제 결과	비고
getExtension('')	리턴 X	리턴 X	존재하지 않는 파일임
getExtension('asd')	리턴 X	리턴 X	존재하지 않는 파일임
getExtension('!@')	리턴 X	리턴 X	존재하지 않는 파일임

getExtension('a')	''	''	확장자가 존재 하지 않음
getExtension('aaa.txt")	리턴 X	리턴 X	존재하지 않는 파일임
getExtension('a.txt')	'.txt'	'.txt'	
getExtension('a.e')	'.e'	'.e'	
getExtension('.a')	'.a'	'.a'	

2.4FileClassifier.py

2.4.1 isHangul 함수 #1 - 수정 전

re모듈의 findall 함수를 통해 인자가 한글인 경우에 한해서만 리스트로 출력

부분 설계

정확히 한 개의 인자를 받아야 함. 만일 인자의 개수가 틀릴 경우 TypeError 발생.

인자는 문자열이어야 한다. 만일 위배될 경우 TypeError 발생.

인자로 문자열을 받은 경우, 그 인자에서 한글(자음 또는 모음일 수 있으며, 온전한 글자 형태일 수 있음)부분을 리스트로 반환하는지 즉, 한글이 포함되는지 여부를 진리값으로 확인.

검사방법

FileClassifier 모듈(FileClassifier.py)을 PyCharm IDE에 로드한 후 isHangul 함수를 직접 호출하여 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 인자 개수 조건(1개) 확인

입력 (호출)	예상결과	실제 결과	비고
isHangul()	TypeError	TypeError	
isHangul('가')	TypeError없음	TypeError없음	
isHangul('가','나')	TypeError	TypeError	
isHangul('가','나','다')	TypeError	TypeError	

목적: 인자 조건(문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
isHangul('가')	TypeError없음	TypeError없음	
isHangul('abc')	TypeError없음	TypeError없음	
isHangul('123')	TypeError없음	TypeError없음	
isHangul(123)	TypeError	TypeError	
isHangul(True)	TypeError	TypeError	
isHangul(['가'])	TypeError	TypeError	리스트
isHangul(('가'))	TypeError없음	TypeError없음	투플
isHangul(Dict['a'])	TypeError없음	TypeError없음	Dict['a'] = '가'
isHangule('가'+'나')	TypeError없음	TypeError없음	

목적: 결과조건(인자에 한글이 포함되는지 여부) 확인

입력 (호출)	예상결과	실제 결과	비고
isHangul('가')	True	True	
isHangul('abc')	False	False	
isHangul('123')	False	False	
isHangul('123가')	True	True	
isHangul('ab가')	True	True	
isHangul('123ab')	False	False	
isHangul('묛')	False	False	
isHangul('@')	False	False	
isHangul('가'+'나')	True	True	
isHangul('가'+'@')	True	True	

2.4.2 isHangul 함수 #2 - 최종

인자가 한글인지 판별하는 함수. text 인자를 넘겨받아 str 자료형 판별 조건을 추가함.

부분 설계

• text 인자가 str 이외의 다른 자료형인 경우, TypeError 메시지를 출력해야 함.

검사방법

FileClassfier_modify.py를 파이참(PyCharm) IDE에서 실행한 상태로, isHangul() 함수에 직접 문자열 인자를 넘겨주어 호출하면서 Error 발생 여부 및 반환 값(True or False)을 확인함.

Test Case 및 결과 정리

목적: text 인자가 str 자료형인지 판별

입력 (호출)	예상결과	실제	결과비고
isHangul('한')	TypeError 없음	TypeError 없음	
isHangul('안녕')	TypeError 없음	TypeError 없음	
isHangul('1')	TypeError 없음	TypeError 없음	
isHangul('125')	TypeError 없음	TypeError 없음	
isHangul('A')	TypeError 없음	TypeError 없음	
isHangul('play')	TypeError 없음	TypeError 없음	
isHangul('安')	TypeError 없음	TypeError 없음	
isHangul('!!*&^')	TypeError 없음	TypeError 없음	
isHangul(['가','나'])	TypeError	TypeError	
isHangul({'가':'나'})	TypeError	TypeError	
isHangul(-1)	TypeError	TypeError	
isHangul(10.3)	TypeError	TypeError	
isHangul(3/5)	TypeError	TypeError	
isHangul(10//2)	TypeError	TypeError	

isHangul(57%13)	TypeError	TypeError
isHangul(57|13)	TypeError	TypeError
isHangul(57&13)	TypeError	TypeError

목적: isHangul() 함수의 반환 값 확인

입력 (호출)	예상결과	실제	결과비고
isHangul('한')	True	True	
isHangul('안녕')	True	True	
isHangul('1')	False	False	
isHangul('125')	False	False	
isHangul('A')	False	False	
isHangul('play')	False	False	
isHangul('安')	False	False	
isHangul("$^&@")	False	False	

2.4.3 extractExt 함수 #1 - 수정 전

부분 설계

정확히 한 개의 인자를 받아야 함. 만일 인자의 개수가 틀릴 경우 TypeError 발생.

인자는 문자열이어야 함. 만일 위배될 경우 AssertionError 발생.

인자로 문자열을 받은 경우 문자열 내의 확장자를 반환한다. 확장자가 없는 경우는 반환하지 않는다.

검사방법

FileClassifier 모듈(FileClassifier.py)을 PyCharm IDE에 로드한 후 extractExt 함수를 직접 호출하여 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 인자 개수 조건(1개) 확인

입력 (호출)	예상결과	실제 결과	비고
extractExt()	TypeError	TypeError	
extractExt('a.jpg')	TypeError없음	TypeError없음	
extractExt('a.jpg','b.png')	TypeError	TypeError	
extractExt('a.jpg','b.png' ,'c.gif','d.bmp')	TypeError	TypeError	

목적: 인자 조건(문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
extractExt('abc')	AssertionError없음	AssertionError없음	
extractExt('abc.jpg')	AssertionError없음	AssertionError없음	
extractExt('abc.jpeg')	AssertionError없음	AssertionError없음	
extractExt('abc.png')	AssertionError없음	AssertionError없음	
extractExt('abc.bmp')	AssertionError없음	AssertionError없음	
extractExt('abc.gif')	AssertionError없음	AssertionError없음	
extractExt(123)	AssertionError	AssertionError	
extractExt(True)	AssertionError	AssertionError	
extractExt(['abc.jpg'])	AssertionError	AssertionError	리스트
extractExt(('abc.jpg'))	AssertionError없음	AssertionError없음	투플
extractExt(Dict['a'])	AssertionError없음	AssertionError없음	Dict['a'] = 'abc.jpg'

목적: 결과 조건(확장자 반환 여부) 확인

입력 (호출)	예상결과	실제 결과	비고

extractExt('abc.jpeg')	반환 O	반환 O
extractExt('abc.jpg')	반환 O	반환 O
extractExt('abc.png')	반환 O	반환 O
extractExt('abc.bmp')	반환 O	반환 O
extractExt('abc.gif')	반환 O	반환 O
extractExt('123.jpeg')	반환 O	반환 O
extractExt('@@.jpeg')	반환 O	반환 O
extractExt('abc')	반환 X	반환 X
extractExt('뭇')	반환 X	반환 X

2.4.4 extractExt 함수 #2 – 최종

이전 extractExt() 함수와 동일함. fn 인자를 넘겨받아 str 자료형인지 판별함. 파일이 확장자를 추출하는 함수.

부분 설계

• fn 인자가 str 이외의 다른 자료형인 경우, AssertionError 메시지를 출력해야 함.

검사방법

FileClassfier_modify.py를 파이참(PyCharm) IDE에서 실행한 상태로, extractExt() 함수에 직접 파일명 인자를 넘겨주어 호출하면서 Error 발생 여부를 확인함.

Test Case 및 결과 정리

목적: fn 인자가 str 자료형인지 판별

입력 (호출)	예상결과	실제	결과비고
입력 (호출)	예상결과	실제	결과비고
extractExt('hello.txt')	AssertionError 없음	AssertionError 없음	
extractExt('hello.jpeg")	AssertionError 없음	AssertionError 없음	

extractExt('hello.gif")	AssertionError 없음	AssertionError 없음
extractExt('hello.png")	AssertionError 없음	AssertionError 없음
extractExt('hello.jpg")	AssertionError 없음	AssertionError 없음
extractExt('hello.bmp")	AssertionError 없음	AssertionError 없음
extractExt('한')	AssertionError 없음	AssertionError 없음
extractExt('안녕')	AssertionError 없음	AssertionError 없음
extractExt('1')	AssertionError 없음	AssertionError 없음
extractExt('125')	AssertionError 없음	AssertionError 없음
extractExt('A')	AssertionError 없음	AssertionError 없음
extractExt('play')	AssertionError 없음	AssertionError 없음
extractExt('安')	AssertionError 없음	AssertionError 없음
extractExt('$^&@')	AssertionError 없음	AssertionError 없음
extractExt(['가','나'])	AssertionError	AssertionError
extractExt({'가':'나'})	AssertionError	AssertionError
extractExt(-1)	AssertionError	AssertionError
extractExt(10.3)	AssertionError	AssertionError
extractExt(3/5)	AssertionError	AssertionError
extractExt(10//2)	AssertionError	AssertionError
extractExt(57%13)	AssertionError	AssertionError
extractExt(57|13)	AssertionError	AssertionError
extractExt(57&13)	AssertionError	AssertionError

2.4.5 MoveToETC 함수 #1 - 수정 전

부분 설계

클래스 생성시 인자로 받은 작업폴더 경로는 문자열이어야 함. 문자열이 아닐 경우 생성자에 있는 assert문에 의해 AssertionError 발생.

클래스 생성시 인자로 받은 작업폴더 경로는 실제로 존재해야 함. 존재하지 않을시 오류메세지(OSError) 발생.

작업폴더 안의 모든 파일들에 대하여 파일이름에 지정된 확장자를 포함하지 않을 경우에만 ETC폴더로 분류한다.

검사방법

FileClassifier 모듈(FileClassifier.py)을 PyCharm IDE에 로드한 후 MoveToETC 함수를 직접 호출하여 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 클래스 생성시 인자로 받은 작업폴더 경로 조건(문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
FileClassifier('abc')	AssertionError없음	AssertionError없음	
FileClassifier('C:₩Users')	AssertionError없음	AssertionError없음	
FileClassifier(123)	AssertionError	AssertionError	
FileClassifier(True)	AssertionError	AssertionError	
FileClassifier('가')	AssertionError없음	AssertionError없음	
FileClassifier(['가'])	AssertionError	AssertionError	리스트
FileClassifier(('가'))	AssertionError없음	AssertionError없음	투플
FileClassifier(Dict['a'])	AssertionError없음	AssertionError없음	Dict['a'] = '가'
FileClassifier('a'&'b')	AssertionError	AssertionError	

목적: 클래스 생성시 인자로 받은 작업폴더 경로 조건(문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
FileClassifier('abc')	AssertionError없음	AssertionError없음	
FileClassifier(123)	AssertionError	AssertionError	
FileClassifier(True)	AssertionError	AssertionError	
FileClassifier('가')	AssertionError없음	AssertionError없음	
FileClassifier(['가'])	AssertionError	AssertionError	리스트
FileClassifier(('가'))	AssertionError없음	AssertionError없음	투플
FileClassifier(Dict['a'])	AssertionError없음	AssertionError없음	Dict['a'] = '가'

2.4.6 MoveToETC 함수 #2– 최종

이전 MoveToETC() 함수와 동일함. FileClassfier 클래스 생성자를 호출할 때 자동으로 호출됨.

작업 폴더에서 최초로 사진 파일과 이외의 파일을 분류해주는 함수

부분 설계

- FileClassfier 클래스 생성자를 호출할 때 넘겨준 작업 폴더 경로가 실제로 존재하지 않는 경우, OSError 메시지를 출력해야 함
- FileClassfier 클래스 생성자를 호출할 때 넘겨준 작업 폴더 경로의 자료형이 str이 아닌 경우, AssertionError 메시지를 출력해야 함

검사방법

FileClassfier_modify.py를 파이참(PyCharm) IDE에서 실행한 상태로, FileClassifier 생성자를 직접 호출함. 호출 시 작업 폴더 경로를 넘겨주면서 Error 발생 여부를 확인함.

Test Case 및 결과 정리

목적: 인자로 넘어온 작업 폴더 경로가 실제로 존재하는지 확인

입력 (호출)	예상결과	실제 결과	비고

FileClassifier('D:/project/test')	OSError 없음	OSError 없음
FileClassifier('C:/project/test/whoareu')	OSError 없음	OSError 없음

부연설명 : OSError를 발생시키려면, 작업 폴더가 프로그램이 돌아가는 도중에 사용자가 인위로 삭제하는 방법밖에 없음. 본 검사에서는 이러한 상황은 없다고 가정하고 진행함.

목적: 인자로 넘어온 작업 폴더 경로가 str 자료형인지 확인

입력 (호출)	예상결과	실제 결과	비고
FileClassifier('D:/project/test')	AssertionError 없음	AssertionError 없음	
FileClassifier('C:/project/test/whoareu')	AssertionError 없음	AssertionError 없음	
FileClassifier('1')	AssertionError 없음	AssertionError 없음	
입력 (호출)	예상결과	실제 결과	비고
FileClassifier('hello')	AssertionError 없음	AssertionError 없음	
FileClassifier('안녕')	AssertionError 없음	AssertionError 없음	
FileClassifier('安')	AssertionError 없음	AssertionError 없음	
FileClassifier('*^*&')	AssertionError 없음	OSError	
FileClassifier(['가','나'])	AssertionError	AssertionError	
FileClassifier({'가':'나'})	AssertionError	AssertionError	
FileClassifier(1)	AssertionError	AssertionError	
FileClassifier(-0.5)	AssertionError	AssertionError	
FileClassifier(3/5)	AssertionError	AssertionError	
FileClassifier(13//5)	AssertionError	AssertionError	
FileClassifier(13%5)	AssertionError	AssertionError	
FileClassifier(13&5)	AssertionError	AssertionError	
FileClassifier(13|5)	AssertionError	AssertionError	

부연설명 : FileClassifier('*^*&')은 AssertionError를 출력할 것으로 예상했으나, 폴더나 파일명은 특수문자가 불가능하므로 OSError를 출력했음

2.4.7 createDirBySize 함수 #1 - 수정 전

부분 설계

정확히 한 개의 인자를 받아야 함. 만일 인자의 개수가 틀릴 경우 TypeError 발생.

Assert 문에 의해 인자는 문자열이어야 함. 만일 위배될 경우 AssertionError 발생.

인자로 경로를 받을 경우 그 해당 경로에 Size이름으로 폴더를 생성하고, 각 폴더 경로를 포함하는 리스트를 반환한다.

검사방법

FileClassifier 모듈(FileClassifier.py)을 PyCharm IDE에 로드한 후 createDirBySize 함수를 직접 호출하여 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 인자 개수 조건(1개) 확인

입력 (호출)	예상결과	실제 결과	비고
createDirBySize()	TypeError	TypeError	
createDirBySize('a')	TypeError 없음	TypeError 없음	
createDirBySize('a','b')	TypeError	TypeError	
createDirBySize('a','b','c','d','e')	TypeError	TypeError	

목적: 인자 조건(문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
createDirBySize('a')	AssertionError없음	AssertionError없음	
createDirBySize('C:₩Users')	AssertionError없음	AssertionError없음	
createDirBySize(123)	AssertionError	AssertionError	

createDirBySize(True)	AssertionError	AssertionError	
createDirBySize(['a'])	AssertionError	AssertionError	리스트
createDirBySize(('a'))	AssertionError없음	AssertionError없음	투플
createDirBySize(Dict['a'])	AssertionError없음	AssertionError없음	Dict['a'] = 'C:₩Users'

목적: 결과 조건(Size별 폴더를 생성하여 그 경로들을 리스트로 반환하는지 여부) 확인

입력 (호출)	예상결과	실제 결과	비고
createDirBySize('C:/')	True	True	입력 경로에 Size별 폴더 생성
createDirBySize('abc')	False	False	경로가 아닌 다른 문자열 입력
createDirBySize('123')	False	False	경로가 아닌 다른 문자열 입력
createDirBySize('C:/HI')	False	False	존재하지 않는 경로 입력

2.4.8 createDirBySize 함수 #2 – 최종

이전 createDirBySize() 함수와 동일함.. 작업 폴더 또는 작업 폴더의 자식(자손) 폴더에 존재하는 사진 파일에 대하여, 고정 기준(1MB 미만, 1MB이상 4MB미만, 4MB 이상)에 따라 폴더를 생성해주는 함수.

부분 설계

- createDirBySize() 함수를 호출할 때 넘겨준 작업 폴더 경로가 실제로 존재하지 않는 경우,

  OSError 메시지를 출력해야 함

- FileClassfier 클래스 생성자를 호출할 때 넘겨준 작업 폴더 경로의 자료형이 str이 아닌 경우,

  AssertionError 메시지를 출력해야 함

검사방법

FileClassfier_modify.py를 파이참(PyCharm) IDE에서 실행한 상태로, createDirBySize() 함수를 직접 호출함. 호출 시 폴더 경로를 넘겨주면서 Error 발생 여부를 확인함.

Test Case 및 결과 정리

목적: 인자로 넘어온 폴더 경로가 str 자료형인지 확인

입력 (호출)	예상결과	실제 결과	비고
createDirBySize('D:/project/test')	AssertionError 없음	AssertionError 없음	
createDirBySize('한')	AssertionError 없음	AssertionError 없음	
createDirBySize('안녕')	AssertionError 없음	AssertionError 없음	
createDirBySize('1')	AssertionError 없음	AssertionError 없음	
createDirBySize('123')	AssertionError 없음	AssertionError 없음	
createDirBySize('A')	AssertionError 없음	AssertionError 없음	
createDirBySize('abc')	AssertionError 없음	AssertionError 없음	
createDirBySize('安')	AssertionError 없음	AssertionError 없음	
createDirBySize('%&#')	AssertionError 없음	AssertionError 없음	
createDirBySize('['가','나']')	AssertionError	AssertionError	
createDirBySize({"가":"나"})	AssertionError	AssertionError	
createDirBySize(1)	AssertionError	AssertionError	
createDirBySize(-0.5)	AssertionError	AssertionError	
createDirBySize(-0.5)	AssertionError	AssertionError	
createDirBySize(13/5)	AssertionError	AssertionError	
createDirBySize(13*5)	AssertionError	AssertionError	
createDirBySize(13&5)	AssertionError	AssertionError	
createDirBySize(13|5)	AssertionError	AssertionError	

목적: 인자로 넘어온 폴더 경로가 실제로 존재하는지(유효한지) 확인

입력 (호출)	예상결과	실제 결과	비고
createDirBySize('D:/project/test')	OSError 없음	OSError 없음	

createDirBySize('D:/project12/test')	OSError	OSError	실제로는 없는 경로
createDirBySize('한')	OSError	OSError 없음	코드 실행은 안됨
createDirBySize('안녕')	OSError	OSError 없음	코드 실행은 안됨
createDirBySize('1')	OSError	OSError 없음	코드 실행은 안됨
createDirBySize('123')	OSError	OSError 없음	코드 실행은 안됨
createDirBySize('A')	OSError	OSError 없음	코드 실행은 안됨
createDirBySize('abc')	OSError	OSError 없음	코드 실행은 안됨
createDirBySize('安')	OSError	OSError 없음	코드 실행은 안됨
createDirBySize('%&#')	OSError	OSError 없음	코드 실행은 안됨

부연설명 : 비고란에 코드 실행이 안되는 case들은 OSError 메시지를 출력하지 않으나 실제로 폴더 생성하는 코드는 실행하지 않음. 폴더 경로명이 실제 윈도우 상에 이용되는 절대 경로 형태로 입력되고, 그 경로가 실제로 존재하지 않는 경우에만 OSError 메시지를 출력함.

2.4.9 createDirByName 함수 #1 - 수정 전

부분 설계

정확히 두 개의 인자를 받아야 함. 만일 인자의 개수가 틀릴 경우 TypeError 발생.

첫 번째 인자인 작업폴더 경로와 두 번째 인자인 2차 분류기준(한글/영어/숫자 중 하나)을 입력받아서 작업폴더 경로에 있는 모든 파일이름의 첫글자로 폴더를 생성한다. (해당파일이 2차 분류기준에 맞지 않는 경우 ETC 폴더로 이동시킨다.)

검사방법

FileClassifier 모듈(FileClassifier.py)을 PyCharm IDE에 로드한 후 createDirBySize 함수를 직접 호출하여 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 인자 개수 조건(2개) 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByName()	TypeError	TypeError	인자 0 개 입력
createDirByName('C:₩')	TypeError	TypeError	인자 1개 입력(path)
createDirByName('한글')	TypeError	TypeError	인자 1개 입력(mode)
createDirByName('C:/test','한글')	TypeError없음	TypeError없음	인자 2개 입력
createDirByName('C:/','한글','영어')	TypeError	TypeError	인자 3개 입력

인자 개수에 대한 조건은 존재하지만, 각 인자별 입력조건은 존재하지 않는다.

목적: 결과 조건(작업폴더 경로에 있는 모든 파일이름의 첫글자로 폴더 생성 여부) 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByName('C:/test','한글')	Error없음	Error없음	
createDirByName('C:/test','한')	Error없음	Error없음	
createDirByName('C:/test','kor')	Error없음	Error없음	
createDirByName('C:/test','korean')	Error없음	Error없음	
createDirByName('C:/test','Korean')	Error없음	Error없음	
createDirByName('C:/test','Kor')	Error없음	Error없음	
createDirByName('C:/test','KOREAN')	Error없음	Error없음	
createDirByName('C:/test','KOR')	Error없음	Error없음	
createDirByName('C:/test','영어')	Error없음	Error없음	
createDirByName('C:/test','영')	Error없음	Error없음	
createDirByName('C:/test','eng')	Error없음	Error없음	
createDirByName('C:/test','english')	Error없음	Error없음	
createDirByName('C:/test','Eng')	Error없음	Error없음	
createDirByName('C:/test','English')	Error없음	Error없음	
createDirByName('C:/test','ENGLISH')	Error없음	Error없음	

createDirByName('C:/test','ENG')	Error없음	Error없음
createDirByName('C:/test','숫자')	Error없음	Error없음
createDirByName('C:/test','num')	Error없음	Error없음
createDirByName('C:/test','number')	Error없음	Error없음
createDirByName('C:/test','NUM')	Error없음	Error없음
createDirByName('C:/test','NUMBER')	Error없음	Error없음
createDirByName('C:/test','Num')	Error없음	Error없음
createDirByName('C:/test','Number')	Error없음	Error없음

2.4.10 createDirByName 함수 #2 – 최종

사용자가 지정한 분류 기준에 맞게 사진 파일을 저장할 폴더를 만들어주는 함수. 함수의 인자인 폴더 경로명, 파일 분류 기준(2차 분류기준)에 대해 str 자료형 및 유효성을 검사 하는 코드를 추가하였음.

부분 설계

- createDirByName() 함수에 인자로 들어가는 폴더 경로명(__path 파라미터) 형식이 str이 아닌 경우, TypeError 메시지를 출력해야 함.
- createDirByName() 함수에 인자로 들어가는 폴더 경로명이 유효하지 않은 경우, OSError 메시지를 출력해야 함.
- createDirByName() 함수에 인자로 들어가는 파일 분류기준(mode 파라미터) 형식이 str이 아닌 경우, TypeError 메시지를 출력해야 함.
- createDirByName() 함수에 인자로 들어가는 파일 분류기준(mode 파라미터) 값이 2차 분류 기준 범주에 속하지 않은 경우, AssertionError 메시지를 출력해야 함.

1. 2차 분류 기준이 이름인 경우

1-1) 파일 이름 분류 기준이 한글인 경우

→ [한글, 한, kor, Korean, Korean, Kor, KOREAN, KOR]

1-2) 파일 이름 분류 기준이 영어인 경우

→ [영어, 영, eng, english, Eng, English, ENGLISH, ENG]

1-3) 파일 이름 분류 기준이 숫자인 경우

→ [숫자, num, number, NUM, NUMBER, Num, Number]

2. 2차 분류 기준이 파일 생성날짜 또는 수정날짜인 경우

2-1) 파일을 월별로 분류하는 경우

→ ['월', '달', 'month', 'Month', 'm', 'MONTH', 'M']

2-2) 파일을 연도별로 분류하는 경우

→ ['년', '년도', 'Y', 'Year', 'y', 'year', 'YEAR']

검사방법

FileClassfier_modify.py를 파이참(PyCharm) IDE에서 실행한 상태로, createDirByName() 함수를 직접 호출함. 호출 시 폴더 경로와 파일 분류 기준을 넘겨주면서 Error 발생 여부를 확인함.

Test Case 및 결과 정리

목적: 폴더 경로명(__path 파라미터)이 str 자료형인지 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByName('D:/project/test','한")	TypeError 없음	TypeError 없음	
createDirByName (1, '한')	TypeError	TypeError	
createDirByName (-0.5, '한')	TypeError	TypeError	
입력 (호출)	예상결과	실제 결과	비고
createDirByName (13/5, '한')	TypeError	TypeError	
createDirByName (13*5, '한')	TypeError	TypeError	
createDirByName (13&5, '한')	TypeError	TypeError	
createDirByName (13|5, '한')	TypeError	TypeError	
createDirByName('['가','나']' ,'한')	TypeError	TypeError	
createDirByName({"가":"나"},'한')	TypeError	TypeError	

부연설명 : 목적이 폴더 경로명의 자료형만 확인하는 것이므로, 파일 분류기준은 유효한 값('한')으로 고정시킴.

목적: 폴더 경로명(__path 파라미터)이 유효한지(실제로 존재하는지) 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByName('D:/project/test','한')	OSError 없음	OSError 없음	
createDirByName('D:/project12/test','한')	OSError	OSError	실제로는 없는 경로
createDirByName('한' ,'한')	OSError	OSError	
createDirByName('안녕' ,'한')	OSError	OSError	
createDirByName('1' ,'한')	OSError	OSError	
createDirByName('123' ,'한')	OSError	OSError	
createDirByName('A' ,'한')	OSError	OSError	
createDirByName('abc' ,'한')	OSError	OSError	
createDirByName('安' ,'한')	OSError	OSError	
createDirByName ('%&#' ,'한')	OSError	OSError	

부연설명 : 목적이 폴더 경로명의 유효성만 확인하는 것이므로, 파일 분류기준은 유효한 값('한')으로 고정시킴.

목적: 파일 분류기준(mode 파라미터)값이 str 형식인지 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByName('D:/project','한')	TypeError 없음	TypeError 없음	
createDirByName('D:/project','영')	TypeError 없음	TypeError 없음	
createDirByName('D:/project','숫자')	TypeError 없음	TypeError 없음	
createDirByName('D:/project', '월')	TypeError 없음	TypeError 없음	
createDirByName('D:/project', '달')	TypeError 없음	TypeError 없음	

createDirByName('D:/project, '안녕')	TypeError 없음	TypeError 없음
createDirByName('D:/project', '123')	TypeError 없음	TypeError 없음
createDirByName('D:/project', 'ABcd')	TypeError 없음	TypeError 없음
createDirByName ('D:/project', '安')	TypeError 없음	TypeError 없음
createDirByName ('D:/project', '$$#')	TypeError 없음	TypeError 없음
createDirByName ('D:/project', ['가','나'])	TypeError	TypeError
createDirByName ('D:/project', {"가":"나"])	TypeError	TypeError
createDirByName ('D:/project', 1)	TypeError	TypeError
createDirByName ('D:/project', -5)	TypeError	TypeError
createDirByName ('D:/project', 0.7)	TypeError	TypeError
createDirByName ('D:/project', 13/5)	TypeError	TypeError
createDirByName ('D:/project', 13//5)	TypeError	TypeError
createDirByName ('D:/project', 13%5)	TypeError	TypeError
createDirByName ('D:/project', 13&5)	TypeError	TypeError
createDirByName ('D:/project', 13|5)	TypeError	TypeError

부연설명 : mode 파라미터가 str 자료형인지 확인하는 것이므로, 폴더 경로명은 'D:/project'로 고정 시킴. 또한, 2차 분류 기준에 대한 유효성 검사 표를 따로 만들었으므로, 여기서는 일부의 2차 분류 기준만 테스트 함.

목적: 파일 분류기준(mode 파라미터)값이 2차 분류 기준 범주(유효)에 포함되었는지 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByName('D:/project','한')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project','한글')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project','kor')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project', 'Korean')	AssertionError 없음	AssertionError 없음	

createDirByName('D:/project', 'Kor')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'KOREAN')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'KOR')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, '영어')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, '영')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'eng')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'english')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'Eng')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'English')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'ENGLISH')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'ENG')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, '숫자')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'num')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'number')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'NUM')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'NUMBER')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'Num')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'Number')	AssertionError 없음	AssertionError 없음	
입력 (호출)	예상결과	실제 결과	비고
createDirByName('D:/project, '월')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, '달')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'month')	AssertionError 없음	AssertionError 없음	
createDirByName('D:/project, 'Month')	AssertionError 없음	AssertionError 없음	

createDirByName('D:/project, 'm')	AssertionError 없음	AssertionError 없음
createDirByName('D:/project, 'Month')	AssertionError 없음	AssertionError 없음
createDirByName('D:/project, 'M')	AssertionError 없음	AssertionError 없음
createDirByName('D:/project, '년')	AssertionError 없음	AssertionError 없음
createDirByName('D:/project, '년도')	AssertionError 없음	AssertionError 없음
createDirByName('D:/project, 'Y')	AssertionError 없음	AssertionError 없음
createDirByName('D:/project, 'Year')	AssertionError 없음	AssertionError 없음
createDirByName('D:/project, 'year')	AssertionError 없음	AssertionError 없음
createDirByName('D:/project, 'YEAR')	AssertionError 없음	AssertionError 없음
createDirByName('D:/project', '123')	AssertionError	AssertionError
createDirByName('D:/project', 'a')	AssertionError	AssertionError
createDirByName('D:/project', 'ABcd')	AssertionError	AssertionError
createDirByName('D:/project', '안')	AssertionError	AssertionError
createDirByName('D:/project', '1+1')	AssertionError	AssertionError
createDirByName ('D:/project', '安')	AssertionError	AssertionError
createDirByName ('D:/project', '$$#')	AssertionError	AssertionError

부연설명 : 목적이 파일 분류기준의 유효성만 확인하는 것이므로, 폴더 경로명은 'D:/project'로 고정 시킴.

※ Testcase 점검 중 OSError가 발생했으나 검사표에 추가시키지 않은 부분: 폴더를 이름을 영문 한 글자로 만드는 경우, os.makedirs(), os.mkdir()함수는 대/소문자를 구분하지 않음.

(예를 들어, 이름이 a인 폴더가 있는 상태에서, A 폴더를 생성하려고 할 때, 이미 존재하는 폴더로 판단하여 os에서 자체적으로 OSError를 던짐.) 따라서, 본 코드 상에서 파일이름이 영어 대문자로 시작하는 경우, 소문자 폴더에 파일이 저장되도록 구현하였음.

2.4.11 createDirByCD 함수 #1 - 수정 전

부분 설계

정확히 두 개의 인자를 받아야 함. 만일 인자의 개수가 틀릴 경우 TypeError 발생.

첫 번째 인자는 문자열이어야 함. 문자열이 아닐 경우 TypeError 발생.

첫 번째 인자인 작업폴더 경로와 두 번째 인자인 2차 분류기준(년도/월 중 하나)을 입력 받아서 작업폴더 경로에 있는 모든 파일의 수정날짜에서 2차 분류기준에 해당하는 값으로 폴더를 생성한 후 그 경로들을 리스트로 반환.

검사방법

FileClassifier 모듈(FileClassifier.py)을 PyCharm IDE에 로드한 후 createDirByCD 함수를 직접 호출하여 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 인자 개수 조건(2개) 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByCD()	TypeError	TypeError	인자 0 개 입력
createDirByCD('C:₩')	TypeError	TypeError	인자 1개 입력(path)
createDirByCD('년도')	TypeError	TypeError	인자 1개 입력(mode)
createDirByCD('C:/test','년도')	TypeError없음	TypeError없음	인자 2개 입력
createDirByCD('C:/','년도','월')	TypeError	TypeError	인자 3개 입력
createDirByCD('C:/','년도','년도')	TypeError	TypeError	인자 3개 입력(중복)

목적: 첫번째 인자 조건(문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
CreateDirByCD('C:/test','년도')	TypeError없음	TypeError없음	
CreateDirByCD('abc','년도')	TypeError없음	TypeError없음	
CreateDirByCD(123,'년도')	TypeError	TypeError	

CreateDirByCD(True,'년도')	TypeError	TypeError	
CreateDirByCD(('abc'),'년도')	TypeError없음	TypeError없음	
CreateDirByCD(['abc'],'년도')	TypeError	TypeError	
CreateDirByCD(Dict['a'],'년도')	TypeError없음	TypeError없음	Dict['a'] = 'abc'

목적: 결과 조건(작업폴더 경로에 있는 파일의 생성날짜에서 2차 분류기준에 해당하는 값으로 폴더생성 여부) 확인

입력 (호출)	예상결과	실제 결과	비고
CreateDirByCD('C:/test','년도')	생성 O	생성 O	
CreateDirByCD('C:/test','년')	생성 O	생성 O	
CreateDirByCD('C:/test','Y')	생성 O	생성 O	
CreateDirByCD('C:/test','Year')	생성 O	생성 O	
CreateDirByCD('C:/test','y')	생성 O	생성 O	
CreateDirByCD('C:/test','year')	생성 O	생성 O	
CreateDirByCD('C:/test','YEAR')	생성 O	생성 O	
CreateDirByCD('C:/test','월')	생성 O	생성 O	
CreateDirByCD('C:/test','달')	생성 O	생성 O	
CreateDirByCD('C:/test','month')	생성 O	생성 O	
CreateDirByCD('C:/test','Month')	생성 O	생성 O	
CreateDirByCD('C:/test','m')	생성 O	생성 O	
CreateDirByCD('C:/test','MONTH')	생성 O	생성 O	
CreateDirByCD('C:/test','M')	생성 O	생성 O	
CreateDirByCD('C:/testA','M')	생성 O	생성 O	존재하지 않는 경로
CreateDirByCD('abc','M')	생성 X	생성 X	경로가 아닌 다른 문자열

CreateDirByCD('C:/test','일')	생성 X	생성 X	유효하지 않은 2차 분류기준

2.4.12 createDirByCD 함수 #2 – 최종

사진 파일의 생성날짜에 따라 폴더를 생성해주는 함수. 함수의 인자인 폴더 경로명, 파일 분류 기준(2차 분류기준)에 대해 str 자료형이 맞는지 검사 및 유효성을 검사 하는 코드를 추가하였음.

부분 설계

- createDirByCD() 함수에 인자로 들어가는 폴더 경로명(__path 파라미터) 형식이 str이 아닌 경우, TypeError 메시지를 출력해야 함.
- createDirByCD() 함수에 인자로 들어가는 폴더 경로명이 유효하지 않은 경우, OSError 메시지를 출력해야 함.
- createDirByCD() 함수에 인자로 들어가는 파일 분류기준(mode 파라미터) 형식이 str이 아닌 경우, TypeError 메시지를 출력해야 함.
- createDirByCD() 함수에 인자로 들어가는 파일 분류기준(mode 파라미터) 값이 2차 분류 기준 범주에 속하지 않은 경우, AssertionError 메시지를 출력해야 함.

(2차 분류 기준 범주는 createDirByName() 함수 부분 설계 part를 참고.)

검사방법

FileClassfier_modify.py를 파이참(PyCharm) IDE에서 실행한 상태로, createDirByCD() 함수를 직접 호출함. 호출 시 폴더 경로와 파일 분류 기준을 넘겨주면서 Error 발생 여부를 확인함.

Test Case 및 결과 정리

목적: 폴더 경로명(__path 파라미터)이 str 자료형인지 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByCD('D:/project/test','월')	TypeError 없음	TypeError 없음	
createDirByCD(1, '월')	TypeError	TypeError	
createDirByCD(-0.5, '월')	TypeError	TypeError	
입력 (호출)	예상결과	실제 결과	비고
createDirByCD(13/5, '월')	TypeError	TypeError	

createDirByCD(13*5, '월')	TypeError	TypeError
createDirByCD(13&5, '월')	TypeError	TypeError
createDirByCD(13|5, '월')	TypeError	TypeError
createDirByCD('['가','나']' , '월')	TypeError	TypeError
createDirByCD({"가":"나"},'월')	TypeError	TypeError

부연설명 : 목적이 폴더 경로명의 자료형만 확인하는 것이므로, 파일 분류기준은 유효한 값('월')으로 고정시킴.

목적: 폴더 경로명(__path 파라미터)이 유효한지(실제로 존재하는지) 확인

입력 (호출)	예상 결과	실제 결과	비고
createDirByCD('D:/project/test', '월')	OSError 없음	OSError 없음	
createDirByCD('D:/project12/test', '월')	OSError	OSError	실제로는 없는 경로
createDirByCD('한' , '월')	OSError	OSError	
createDirByCD('안녕하세요' , '월')	OSError	OSError 없음	원인 파악 불가능
createDirByCD('1' , '월')	OSError	OSError 없음	원인 파악 불가능
createDirByCD('123' ,'월')	OSError	OSError	
createDirByCD('A' ,'월')	OSError	OSError	
createDirByCD('abc' ,'월')	OSError	OSError	
createDirByCD('安' ,'월')	OSError	OSError	
createDirByCD('%&#' ,'월')	OSError	OSError	

부연설명 : 목적이 폴더 경로명의 유효성만 확인하는 것이므로, 파일 분류기준은 유효한 값('월')으로 고정시킴. 같은 문자열 형식을 입력했으나, 일부 case에서 실제로 OSError가 출력되지 않았음.

동일 성격의 문자열이라 처리 방식도 같기 때문에, 현재 정확한 원인을 발견하지 못했음.

목적: 파일 분류기준(mode 파라미터)값이 str 형식인지 확인

입력 (호출)	예상결과	실제 결과	비고

createDirByCD('D:/project','월')	TypeError 없음	TypeError 없음	
createDirByCD('D:/project','달')	TypeError 없음	TypeError 없음	
createDirByCD('D:/project','년도')	TypeError 없음	TypeError 없음	
createDirByCD('D:/project', '년')	TypeError 없음	TypeError 없음	
createDirByCD('D:/project, '안녕')	TypeError 없음	TypeError 없음	AssertionError
createDirByCD('D:/project', '123')	TypeError 없음	TypeError 없음	AssertionError
createDirByCD('D:/project', 'ABcd')	TypeError 없음	TypeError 없음	AssertionError
createDirByCD('D:/project', '安')	TypeError 없음	TypeError 없음	AssertionError
createDirByCD('D:/project', '$$#')	TypeError 없음	TypeError 없음	AssertionError
createDirByCD('D:/project', ['가','나'])	TypeError	TypeError	
createDirByCD('D:/project', {"가":"나"])	TypeError	TypeError	
createDirByCD('D:/project', 1)	TypeError	TypeError	
createDirByCD('D:/project', -5)	TypeError	TypeError	
createDirByCD('D:/project', 0.7)	TypeError	TypeError	
createDirByCD('D:/project', 13/5)	TypeError	TypeError	
createDirByCD('D:/project', 13//5)	TypeError	TypeError	
createDirByCD('D:/project', 13%5)	TypeError	TypeError	
createDirByCD('D:/project', 13&5)	TypeError	TypeError	
createDirByCD('D:/project', 13|5)	TypeError	TypeError	

부연설명 : mode 파라미터가 str 자료형인지 확인하는 것이므로, 폴더 경로명은 'D:/project'로 고정 시킴. 또한, 2차 분류 기준에 대한 유효성 검사 표를 따로 만들었으므로, 여기서는 일부의 2차 분류 기준만 테스트 함.

목적: 파일 분류기준(mode 파라미터)값이 2차 분류 기준 범주(유효)에 포함되었는지 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByCD('D:/project','월')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project','달')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project','month')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project', 'Month')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project', 'm')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project, 'MONTH')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project, 'M')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project, '년')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project, '년도')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project, 'Y')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project, 'Year')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project, 'y')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project, 'year')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project, 'YEAR')	AssertionError 없음	AssertionError 없음	
createDirByCD('D:/project', '123')	AssertionError	AssertionError	
createDirByCD('D:/project', 'a')	AssertionError	AssertionError	
createDirByCD('D:/project', 'ABcd')	AssertionError	AssertionError	
createDirByCD('D:/project', '안')	AssertionError	AssertionError	
createDirByCD('D:/project', '1+1')	AssertionError	AssertionError	
createDirByCD('D:/project', '安')	AssertionError	AssertionError	
createDirByCD('D:/project', '$$#')	AssertionError	AssertionError	

부연설명 : 목적이 파일 분류기준의 유효성만 확인하는 것이므로, 폴더 경로명은 'D:/project'로 고정 시킴.

2.4.13 createDirByMD 함수 #1 - 수정 전

부분 설계

정확히 두 개의 인자를 받아야 함. 만일 인자의 개수가 틀릴 경우 TypeError 발생.

첫 번째 인자는 문자열이어야 함. 문자열이 아닐 경우 TypeError 발생.

첫 번째 인자인 작업폴더 경로와 두 번째 인자인 2차 분류기준(년도/월 중 하나)을 입력 받아서 작업폴더 경로에 있는 모든 파일의 수정날짜에서 2차 분류기준에 해당하는 값으로 폴더를 생성한 후 그 경로들을 리스트로 반환.

검사방법

FileClassifier 모듈(FileClassifier.py)을 PyCharm IDE에 로드한 후 createDirByMD 함수를 직접 호출하여 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 인자 개수 조건(2개) 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByMD()	TypeError	TypeError	인자 0 개 입력
createDirByMD('C:/')	TypeError	TypeError	인자 1개 입력(path)
createDirByMD('년도')	TypeError	TypeError	인자 1개 입력(mode)
createDirByMD('C:/test','년도')	TypeError없음	TypeError없음	인자 2개 입력
createDirByMD('C:/','년도','월')	TypeError	TypeError	인자 3개 입력
createDirByMD('C:/','년도','년도')	TypeError	TypeError	인자 3개 입력(중복)

목적: 첫번째 인자 조건(문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
CreateDirByMD('C:/test','년도')	TypeError없음	TypeError없음	
CreateDirByMD('abc','년도')	TypeError없음	TypeError없음	
CreateDirByMD(123,'년도')	TypeError	TypeError	

CreateDirByMD(True,'년도')	TypeError	TypeError	
CreateDirByMD(('abc'),'년도')	TypeError없음	TypeError없음	
CreateDirByMD(['abc'],'년도')	TypeError	TypeError	
CreateDirByMD(Dict['a'],'년도')	TypeError없음	TypeError없음	Dict['a'] = 'abc'

목적: 결과 조건(작업폴더 경로에 있는 파일의 수정날짜에서 2차 분류기준에 해당하는 값으로 폴더생성 여부) 확인

입력 (호출)	예상결과	실제 결과	비고
CreateDirByMD('C:/test','년도')	생성 O	생성 O	
CreateDirByMD('C:/test','년')	생성 O	생성 O	
CreateDirByMD('C:/test','Y')	생성 O	생성 O	
CreateDirByMD('C:/test','Year')	생성 O	생성 O	
CreateDirByMD('C:/test','y')	생성 O	생성 O	
CreateDirByMD('C:/test','year')	생성 O	생성 O	
CreateDirByMD('C:/test','YEAR')	생성 O	생성 O	
CreateDirByMD('C:/test','월')	생성 O	생성 O	
CreateDirByMD('C:/test','달')	생성 O	생성 O	
CreateDirByMD('C:/test','month')	생성 O	생성 O	
CreateDirByMD('C:/test','Month')	생성 O	생성 O	
CreateDirByMD('C:/test','m')	생성 O	생성 O	
CreateDirByMD('C:/test','MONTH')	생성 O	생성 O	
CreateDirByMD('C:/test','M')	생성 O	생성 O	
CreateDirByMD('C:/testA','M')	생성 O	생성 O	존재하지 않는 경로
CreateDirByMD('abc','M')	생성 X	생성 X	경로가 아닌 다른 문자열

CreateDirByMD('C:/test','일')	생성 X	생성 X	유효하지 않은 2차 분류기준

2.4.14 createDirByMD 함수 #2 – 최종

사진 파일의 수정날짜에 따라 폴더를 생성해주는 함수. 함수의 인자인 폴더 경로명, 파일 분류 기준(2차 분류기준)에 대해 str 자료형이 맞는지 검사 및 유효성을 검사 하는 코드를 추가하였음.

부분 설계

- createDirByMD() 함수에 인자로 들어가는 폴더 경로명(__path 파라미터) 형식이 str이 아닌 경우, TypeError 메시지를 출력해야 함.
- createDirByMD() 함수에 인자로 들어가는 폴더 경로명이 유효하지 않은 경우, OSError 메시지를 출력해야 함.
- createDirByMD() 함수에 인자로 들어가는 파일 분류기준(mode 파라미터) 형식이 str이 아닌 경우, TypeError 메시지를 출력해야 함.
- createDirByMD() 함수에 인자로 들어가는 파일 분류기준(mode 파라미터) 값이 2차 분류 기준 범주에 속하지 않은 경우, AssertionError 메시지를 출력해야 함.

  (2차 분류 기준 범주는 createDirByName() 함수 부분 설계 part를 참고.)

검사방법

FileClassfier_modify.py를 파이참(PyCharm) IDE에서 실행한 상태로, createDirByMD() 함수를 직접 호출함. 호출 시 폴더 경로와 파일 분류 기준을 넘겨주면서 Error 발생 여부를 확인함.

Test Case 및 결과 정리

목적: 폴더 경로명(__path 파라미터)이 str 자료형인지 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByMD('D:/project/test','월')	TypeError 없음	TypeError 없음	
createDirByMD(1, '월')	TypeError	TypeError	
createDirByMD(-0.5, '월')	TypeError	TypeError	
createDirByMD(13/5, '월')	TypeError	TypeError	
createDirByMD(13*5, '월')	TypeError	TypeError	

createDirByMD(13&5, '월')	TypeError	TypeError	

입력 (호출)	예상결과	실제 결과	비고
createDirByMD(13|5, '월')	TypeError	TypeError	
createDirByMD('['가','나']' , '월')	TypeError	TypeError	
createDirByMD({"가":"나"},'월')	TypeError	TypeError	

부연설명 : 목적이 폴더 경로명의 자료형만 확인하는 것이므로, 파일 분류기준은 유효한 값('월')으로 고정시킴.

목적: 폴더 경로명(__path 파라미터)이 유효한지(실제로 존재하는지) 확인

입력 (호출)	예상 결과	실제 결과	비고
createDirByMD('D:/project/test', '월')	OSError 없음	OSError 없음	
createDirByMD('D:/project12/test', '월')	OSError	OSError	실제로는 없는 경로
createDirByMD('한' , '월')	OSError	OSError	
createDirByMD('안녕하세요' , '월')	OSError	OSError 없음	원인 파악 불가능
createDirByMD('1' , '월')	OSError	OSError 없음	원인 파악 불가능
createDirByMD('123' ,'월')	OSError	OSError	
createDirByMD('A' ,'월')	OSError	OSError	
createDirByMD('abc' ,'월')	OSError	OSError	
createDirByMD('安' ,'월')	OSError	OSError	
createDirByMD('%&#' ,'월')	OSError	OSError	

부연설명 : 목적이 폴더 경로명의 유효성만 확인하는 것이므로, 파일 분류기준은 유효한 값('월')으로 고정시킴. createDirByCD() 함수 호출 시 사용했던 case를 그대로 적용하였으나, 똑같은 case에서 실제로 OSError가 출력되지 않았음. 동일 성격의 문자열이라 처리 방식도 같기 때문에, 현재 정확한 원인을 발견하지 못했음.

목적: 파일 분류기준(mode 파라미터)값이 str 형식인지 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByMD('D:/project','월')	TypeError 없음	TypeError 없음	
createDirByMD('D:/project','달')	TypeError 없음	TypeError 없음	
createDirByMD('D:/project','년도')	TypeError 없음	TypeError 없음	
createDirByMD('D:/project', '년')	TypeError 없음	TypeError 없음	
createDirByMD('D:/project, '안녕')	TypeError 없음	TypeError 없음	AssertionError
createDirByMD('D:/project', '123')	TypeError 없음	TypeError 없음	AssertionError
createDirByMD('D:/project', 'ABcd')	TypeError 없음	TypeError 없음	AssertionError
createDirByMD('D:/project', '安')	TypeError 없음	TypeError 없음	AssertionError
createDirByMD('D:/project', '$$#')	TypeError 없음	TypeError 없음	AssertionError
createDirByMD('D:/project', ['가','나'])	TypeError	TypeError	
createDirByMD('D:/project', {"가":"나"])	TypeError	TypeError	
createDirByMD('D:/project', 1)	TypeError	TypeError	
createDirByMD('D:/project', -5)	TypeError	TypeError	
createDirByMD('D:/project', 0.7)	TypeError	TypeError	
createDirByMD('D:/project', 13/5)	TypeError	TypeError	
createDirByMD('D:/project', 13//5)	TypeError	TypeError	
createDirByMD('D:/project', 13%5)	TypeError	TypeError	
createDirByMD('D:/project', 13&5)	TypeError	TypeError	
createDirByMD('D:/project', 13|5)	TypeError	TypeError	

부연설명 : mode 파라미터가 str 자료형인지 확인하는 것이므로, 폴더 경로명은 'D:/project'로 고정 시킴. 또한, 2차 분류 기준에 대한 유효성 검사 표를 따로 만들었으므로, 여기서는 일부의 2차 분류 기준만 테스트 함.

목적: 파일 분류기준(mode 파라미터)값이 2차 분류 기준 범주(유효)에 포함되었는지 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByMD('D:/project','월')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project','달')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project','month')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project', 'Month')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project', 'm')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project, 'MONTH')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project, 'M')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project, '년')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project, '년도')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project, 'Y')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project, 'Year')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project, 'y')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project, 'year')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project, 'YEAR')	AssertionError 없음	AssertionError 없음	
createDirByMD('D:/project', '123')	AssertionError	AssertionError	
createDirByMD('D:/project', 'a')	AssertionError	AssertionError	
createDirByMD('D:/project', 'ABcd')	AssertionError	AssertionError	
createDirByMD('D:/project', '안')	AssertionError	AssertionError	
createDirByMD('D:/project', '1+1')	AssertionError	AssertionError	
createDirByMD('D:/project', '安')	AssertionError	AssertionError	
createDirByMD('D:/project', '$$#')	AssertionError	AssertionError	

부연설명 : 목적이 파일 분류기준의 유효성만 확인하는 것이므로, 폴더 경로명은 'D:/project'로 고정 시킴.

2.4.15 createDirByExt 함수 #1 - 수정 전

assert문으로 인자에 대해 문자열이어야하는 조건이 존재.

부분 설계

정확히 한 개의 인자를 받아야 함. 만일 인자의 개수가 틀릴 경우 TypeError 발생.

Assert 문에 의해 인자는 문자열이어야 함. 만일 위배될 경우 AssertionError 발생.

인자로 경로를 받을 경우 그 경로에 해당하는 디렉토리에 'bmp', 'gif', 'png', 'jpg' 확장자 이름으로 폴더를 생성한 후 각각의 경로를 리스트로 반환

검사방법

FileClassifier 모듈(FileClassifier.py)을 PyCharm IDE에 로드한 후 createDirByExt 함수를 직접 호출하여 리턴값 및 오류 메시지 확인.

Test Case 및 결과 정리

목적: 인자 개수 조건(1개) 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByExt()	TypeError	TypeError	인자 0개인 경우
createDirByExt("C:/")	TypeError	TypeError	인자 1개인 경우
createDirByExt('C:/','C:/test')	TypeError	TypeError	인자 2개인 경우
createDirByExt('C:/','C:/test','C:/testA')	TypeError	TypeError	인자 3개인 경우

목적: 인자 조건(문자열) 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByExt(”C:/”)	AssertionError없음	AssertionError없음	
createDirByExt(”abc’)	AssertionError없음	AssertionError없음	
createDirByExt(123)	AssertionError	AssertionError	
createDirByExt(True)	AssertionError	AssertionError	
createDirByExt([’C:/’])	AssertionError	AssertionError	리스트
createDirByExt((’C:/’))	AssertionError없음	AssertionError없음	투플
createDirByExt((Dict[’a’]))	AssertionError없음	AssertionError없음	Dict[’a’]=’C:/’

목적: 결과 조건(’bmp’, ’gif’, ’png’, ’jpg’ 확장자 이름으로 폴더를 생성하는지 여부) 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByExt(’C:/test’)	생성 성공	생성 성공	
createDirByExt(’C:/testA’)	생성 성공	생성 성공	존재하지 않는 경로
createDirByExt(’abc’)	생성 실패	생성 실패	경로가 아닌 문자열
createDirByExt(’#’)	생성 실패	생성 실패	
createDirByExt(’C:/Users/obh/Desktop’)	생성 성공	생성 성공	

2.4.16 createDirByExt 함수 #2 – 최종

사진 파일의 확장자에 따라 폴더를 생성하는 함수. Jpeg 확장자에 대해서도 코드가 적용되도록 추가하였고, 인자로 넘어오는 폴더 경로명이 유효성(실제로 존재하는지) 검사 코드를 추가함.

부분 설계

- createDirByExt() 함수에 인자로 들어가는 폴더 경로명(path 파라미터) 형식이 str이 아닌 경우, TypeError 메시지를 출력해야 함.
- createDirByExt() 함수에 인자로 들어가는 폴더 경로명이 유효하지 않은 경우, OSError 메시지를 출력해야 함.

검사방법

FileClassfier_modify.py를 파이참(PyCharm) IDE에서 실행한 상태로, createDirByExt() 함수를 직접 호출함. 호출 시 폴더 경로를 넘겨주면서 Error 발생 여부를 확인함.

Test Case 및 결과 정리

목적: 인자로 넘어온 폴더 경로가 str 자료형인지 확인

입력 (호출)	예상결과	실제 결과	비고
createDirByExt('D:/project/test')	AssertionError 없음	AssertionError 없음	
createDirByExt ('1')	AssertionError 없음	AssertionError 없음	
createDirByExt ('hello')	AssertionError 없음	AssertionError 없음	
createDirByExt ('안녕')	AssertionError 없음	AssertionError 없음	
createDirByExt ('安')	AssertionError 없음	AssertionError 없음	
createDirByExt ('^*$#')	AssertionError 없음	AssertionError 없음	
createDirByExt (['가','나'])	AssertionError	AssertionError	
createDirByExt ({'가':'나'})	AssertionError	AssertionError	
createDirByExt (1)	AssertionError	AssertionError	
createDirByExt (-0.5)	AssertionError	AssertionError	
createDirByExt (3/5)	AssertionError	AssertionError	
createDirByExt (13//5)	AssertionError	AssertionError	
createDirByExt (13%5)	AssertionError	AssertionError	
입력 (호출)	예상결과	실제 결과	비고
createDirByExt (13&5)	AssertionError	AssertionError	
createDirByExt (13|5)	AssertionError	AssertionError	

목적: 폴더 경로명(path 파라미터)이 유효한지(실제로 존재하는지) 확인

입력 (호출)	예상 결과	실제 결과	비고
createDirByExt('D:/project/test')	OSError 없음	OSError 없음	
createDirByExt('D:/project12/test')	OSError	OSError	실제로는 없는 경로
createDirByExt('한')	OSError	OSError	
createDirByExt('안녕하세요')	OSError	OSError 없음	원인 파악 불가능
createDirByExt('1')	OSError	OSError 없음	원인 파악 불가능
createDirByExt('123')	OSError	OSError	
createDirByExt('A')	OSError	OSError	
createDirByExt('abc')	OSError	OSError	
createDirByExt('安')	OSError	OSError	
createDirByExt('%&#')	OSError	OSError	

부연설명 : createDirByCD() 함수 호출 시 사용했던 case를 그대로 적용하였으나, 똑같은 case에서 실제로 OSError가 출력되지 않았음. 동일 성격의 문자열이라 처리 방식도 같기 때문에, 현재 정확한 원인을 발견하지 못했음.

2.4.17 moveFileToSrc 함수 #1 - 수정 전

경로와 2차 분류기준을 입력 받아, 그 기준에 따라 실제로 파일을 이동시켜주는 함수.

TestCase

'수정 전 part'에서는 인터페이스 part 에서 폴더 경로와, 2 차 분류 기준이 항상 올바르게 넘어온다는 것을 가정하였음. 그러므로 수정 전 파일에서는 파라미터에 대한 조건 검사코드가 없어서 test case 를 다루지 않음. 대신 경로와 딕셔너리 파라미터에 대해서, 자료형 및 유효성 점검코드를 추가한 '수정 후 part'에서 test case 에 대한 검사 내용을 다루겠음

2.4.18 moveFileToSrc 함수 #2 – 최종

사용자가 2차 분류기준을 선택한 경우, 실제로 선택된 기준에 맞춰서 사진 파일을 재귀적으로 생성된 폴더로 이동시켜주는 함수. 최종 함수에서는 아래와 같은 코드를 추가함.

1. 함수의 인자로 넘어오는 작업 폴더 경로명의 자료형 점검 코드(str형인지 아닌지)
2. 작업 폴더 경로명의 유효성 점검 코드
3. 2차 분류 기준 자료형 점검 코드(dict형인지 아닌지)
4. 2차 분류 기준 딕셔너리 길이 측정 코드(정상 값은 정수 0, 1, 2, 3임. 2차 분류 기준은 생성날짜, 수정날짜, 파일이름 3가지임.)
5. 딕셔너리의 value 값의 자료형 점검 코드(str형인지 아닌지)
6. Value 값이 2차 분류 기준 범주에 포함되는지 점검하는 코드

부분 설계

- moveFileToSrc() 함수에 인자로 들어가는 작업 폴더 경로명(srcPath 파라미터) 형식이 str이 아닌 경우, TypeError 메시지를 출력해야 함.
- moveFileToSrc() 함수에 인자로 들어가는 작업 폴더 경로명이 유효하지 않은 경우, OSError 메시지를 출력해야 함.
- moveFileToSrc() 함수에 인자로 들어가는 딕셔너리(second_sort_dict 파라미터) 형식이 dict가 아닌 경우, TypeError 메시지를 출력해야 함.
- moveFileToSrc() 함수에 인자로 들어가는 딕셔너리 길이가 3보다 큰 경우, AssertionError 메시지를 출력해야 함(길이가 0보다 작을 수는 없으므로 이 조건은 제외).
- moveFileToSrc() 함수에 인자로 들어가는 딕셔너리 value가 2차 분류 기준 범주에 속하지 않는 경우, AssertionError 메시지를 출력해야 함.
- moveFileToSrc() 함수에 인자로 들어가는 딕셔너리의 value 자료형이 str이 아닌 경우, TypeError 메시지를 출력해야 함.

(2차 분류 기준 범주는 createDirByName() 함수 부분 설계 part를 참고.)

검사방법

FileClassfier_modify.py를 파이참(PyCharm) IDE에서 실행한 상태로, moveFileToSrc() 함수를 직접 호출함. 호출 시 작업 폴더 경로와 딕셔너리를 넘겨주면서 Error 발생 여부를 확인함.

Test Case 및 결과 정리

목적: 작업 폴더 경로명(srcPath 파라미터) 자료형이 str인지 확인

입력 (호출)	예상결과	실제결과	비고

moveFileToSrc('D:/project/test',{'FILE_NAME':"영"})	TypeError 없음	TypeError 없음
moveFileToSrc('1',{'FILE_NAME':"영"})	TypeError 없음	TypeError 없음
moveFileToSrc('hello',{'FILE_NAME':"영"})	TypeError 없음	TypeError 없음
moveFileToSrc('안녕',{'FILE_NAME':"영"})	TypeError 없음	TypeError 없음

입력 (호출)	예상결과	실제결과	비고
moveFileToSrc('安',{'FILE_NAME':"영"})	TypeError 없음	TypeError 없음	
moveFileToSrc('%&!',{'FILE_NAME':"영"})	TypeError 없음	TypeError 없음	
moveFileToSrc(['가','나'] ,{'FILE_NAME':"영"})	TypeError	TypeError	
moveFileToSrc(1 ,{'FILE_NAME':"영"})	TypeError	TypeError	
moveFileToSrc(0.5 ,{'FILE_NAME':"영"})	TypeError	TypeError	
moveFileToSrc(-1 ,{'FILE_NAME':"영"})	TypeError	TypeError	
moveFileToSrc(3/5 ,{'FILE_NAME':"영"})	TypeError	TypeError	

부연설명 : 작업 폴더 경로명을 점검하는 코드이므로 딕셔너리는 유효한 값으로 고정함.

목적: 작업 폴더 경로명(srcPath 파라미터)이 유효한(실제로 존재) 경로인지 확인

입력 (호출)	예상결과	실제결과	비고
moveFileToSrc('D:/project/test',{'FILE_NAME':"영"})	OSError 없음	OSError 없음	
moveFileToSrc('D:/project/test/12',{'FILE_NAME':"영"})	OSError	OSError	실제로 존재하지 않는 폴더
moveFileToSrc('hello',{'FILE_NAME':"영"})	OSError	OSError	
moveFileToSrc('안녕',{'FILE_NAME':"영"})	OSError	AssertionError	
moveFileToSrc('安',{'FILE_NAME':"영"})	OSError	OSError	
moveFileToSrc('%&!',{'FILE_NAME':"영"})	OSError	OSError	

부연설명 : 경로명이 '안녕'인 case 같은 경우, os.path.exists() 코드 값을 True 로 반환하여 OSError 를 출력하지 않았는데, 정확한 원인을 파악하지 못했음.

목적: 2차 분류기준 딕셔너리(second_sort_dict 파라미터) 자료형이 dict인지 확인

입력 (호출)	예상결과	실제결과	비고
moveFileToSrc('D:/project/test',{'FILE_NAME':"영"})	TypeError 없음	TypeError 없음	
moveFileToSrc('D:/project/test',{'CREATED_DATE':"월"})	TypeError 없음	TypeError 없음	
moveFileToSrc('D:/project/test',{'MODIFIED_DATE':"월"})	TypeError 없음	TypeError 없음	
moveFileToSrc('D:/project/test','한')	TypeError	TypeError	
moveFileToSrc('D:/project/test','안녕')	TypeError	TypeError	
moveFileToSrc('D:/project/test','a')	TypeError	TypeError	
moveFileToSrc('D:/project/test','AB')	TypeError	TypeError	
moveFileToSrc('D:/project/test',['가','나'])	TypeError	TypeError	
moveFileToSrc('D:/project/test', 1)	TypeError	TypeError	
moveFileToSrc('D:/project/test',0.5)	TypeError	TypeError	
moveFileToSrc('D:/project/test',-1)	TypeError	TypeError	
moveFileToSrc('D:/project/test',3/5)	TypeError	TypeError	

부연설명 : 작업 폴더 경로명은 'D:/project/test'로 고정시켜놓고 검사함.

목적: 2차 분류기준 딕셔너리 길이가 0 이상 3 이하인지 확인

입력 (호출)	예상결과	실제결과	비고
moveFileToSrc('D:/project/test',{'FILE_NAME':"영"})	AssertionError없음	AssertionError 없음	
moveFileToSrc('D:/project/test',{'CREATED_DATE':"월"})			

AssertionError없음	AssertionError 없음	
moveFileToSrc('D:/project/test',{'MODIFIED_DATE':"월"})		
AssertionError없음	AssertionError 없음	
moveFileToSrc('D:/project/test',{})	AssertionError없음	AssertionError 없음
moveFileToSrc('D:/project/test',{'1':'1','2':'2'})	AssertionError없음	**AssertionError**
입력 (호출)	예상결과	실제결과
moveFileToSrc('D:/project/test',{'1':'1','2':'2','3':'3'})	AssertionError없음	**AssertionError**
moveFileToSrc('D:/project/test',{'1':'1','2':'2','3':'3','4':'4'})		
AssertionError	AssertionError	

부연설명 : 예상결과에서는 AssertionError 없음, 실제결과에서는 AssertionError 가 발생한 case 는 2 차 분류 딕셔너리 key 가 ['FILE_NAME', 'CREATED_DATE', 'MODIFIED_DATE']에 속하지 않았기 때문임.

※ moveFileToSrc() 함수에 인자로 들어가는 딕셔너리 value가 2차 분류 기준 범주에 속하지 않는 경우, AssertionError 메시지를 출력해야 함.

moveFileToSrc() 함수에 인자로 들어가는 딕셔너리의 value 자료형이 str이 아닌 경우, TypeError 메시지를 출력해야 함.

위 두 부분설계 조건에 대한 검사는 바로 다음에 나오는 mainClassifier() 함수 part에서 실행함.

2.4.19 mainClassifier 함수 #1 - 수정 전

1,2차 분류기준을 입력 받아, 그 기준에 따라 폴더를 생성하고, 파일들을 해당 폴더로 이동시켜주는 함수.

TestCase

'수정 전 part'에서는 인터페이스 part 에서 폴더 1,2 차 분류 기준이 항상 올바르게 넘어온다는 것을 가정하였음. 그러므로 수정 전 파일에서는 파라미터에 대한 조건 검사코드가 없어서 test case 를 다루지 않음. 대신 경로와 딕셔너리 파라미터에 대해서, 자료형 및 유효성 점검코드를 추가한 '수정 후 part'에서 test case 에 대한 검사 내용을 다루겠음

2.4.20 mainClassifier 함수 #2 – 최종

파일을 사용자가 입력한 1차 분류 기준과 2차 분류 기준(선택된 경우)에 따라 폴더를 재귀적으로 생성하고, 파일을 조건에 맞는 폴더로 이동시켜주는 함수

부분 설계

- mainClassifier() 함수에 인자로 들어가는 1차 분류 기준 리스트(first_sort_list 파라미터) 형식이 list가 아닌 경우, TypeError 메시지를 출력해야 함.
- mainClassifier() 함수에 인자로 들어가는 리스트의 길이가 5보다 큰 경우, AssertionError 메시지를 출력해야 함(길이가 0보다 작을 수는 없으므로 이 조건은 제외).
- mainClassifier() 함수에 인자로 들어가는 리스트의 요소 자료형이 int가 아닌 경우, TypeError 메시지를 출력해야 함.
- mainClassifier() 함수에 인자로 들어가는 리스트의 값이 [1, 2, 3, 4, 5]에 속하지 않은 경우, AssertionError 메시지를 출력해야 함.
- mainClassifier() 함수에 인자로 들어가는 2차 분류 기준 딕셔너리(second_sort_dict 파라미터) 형식이 dict가 아닌 경우, TypeError 메시지를 출력해야 함.

(2차 분류 기준 범주는 createDirByName() 함수 부분 설계 part를 참고.)

- mainClassifier() 함수에 인자로 들어가는 딕셔너리의 길이가 3보다 큰 경우, AssertionError 메시지를 출력해야 함(길이가 0보다 작을 수는 없으므로 이 조건은 제외).
- mainClassifier() 함수에 인자로 들어가는 딕셔너리의 요소 자료형이 str가 아닌 경우, TypeError 메시지를 출력해야 함.
- mainClassifier() 함수에 인자로 들어가는 딕셔너리의 key가 ['FILE_NAME', 'CREATED_DATE', 'MODIFIED_DATE']에 속하지 않은 경우, KeyError 메시지를 출력해야 함.
- mainClassifier() 함수에 인자로 들어가는 딕셔너리의 value 값이 2차 분류 기준 범주에 속하지 않은 경우, AssertionError 메시지를 출력해야 함.
- mainClassifier() 함수에 인자로 들어가는 리스트에 대응하는 딕셔너리 key가 존재하지 않은 경우, KeyError 메시지를 출력해야 함
- **1차 분류 기준 : 파일 크기(1번), 파일 이름(2번), 파일 생성날짜(3번), 파일 수정날짜(4번), 파일 확장자(5번)**
- **2차 분류 기준 key : 'FILE_NAME'(파일 이름에 대응), 'CREATED_DATE'(파일 생성날짜에 대응), ' MODIFIED_DATE'(파일 수정날짜에 대응)**

이 때, 1차 분류 기준에 대응하는 2차 분류 기준이 없다면 KeyError가 발생함.

(예: 입력 리스트가 [1, 2, 3]이고 입력 딕셔너리가 {"CREATED_DATE":"월"}인 경우, 2번에 대응하는 'FILE_NAME'이 없으므로 KeyError가 발생함.)

검사방법

FileClassfier_modify.py를 파이참(PyCharm) IDE에서 실행한 상태로, mainClassifier() 함수를 직접 호출함. 호출 시 리스트(1차 분류 기준)과 딕셔너리(2차 분류 기준)을 넘겨주면서 Error 발생 여부를 확인함.

Test Case 및 결과 정리

목적: 1차 분류 기준 리스트(first_sort_list 파라미터) 형식이 list인지 확인

입력 (호출)	예상결과	실제 결과
mainClassifier([1, 2, 3], {'FILE_NAME':'영'})	TypeError 없음	TypeError 없음
mainClassifier([3, 5, 1, 2], {'FILE_NAME':'영'})	TypeError 없음	TypeError 없음
mainClassifier([], {'FILE_NAME':'영'})	TypeError 없음	TypeError 없음
mainClassifier([5, 4, 3, 2, 1], {'FILE_NAME':'영'})	TypeError 없음	TypeError 없음
mainClassifier(1, {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier(-1, {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier(0.5, {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier(13/5, {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier(13&5, {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier('한', {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier('ABC', {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier('安', {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier('$&#', {'FILE_NAME':'영'})	TypeError	TypeError

부연설명 : 리스트 자료형을 점검하는 부분이므로, 2차 분류기준 딕셔너리는 {'FILE_NAME':'영'}로 고정함.

목적: 1차 분류 기준 리스트 길이가 0부터 5사이의 올바른 값인지 확인

입력 (호출)	예상결과	실제 결과
mainClassifier([1, 2], {'FILE_NAME':'영'})	AssertionError 없음	AssertionError 없음
mainClassifier([3, 5, 1, 2], {'FILE_NAME':'영'})	AssertionError 없음	AssertionError 없음
mainClassifier([], {'FILE_NAME':'영'})	AssertionError 없음	AssertionError 없음
mainClassifier([5, 4, 3, 2, 1], {'FILE_NAME':'영'})	AssertionError 없음	AssertionError 없음
mainClassifier([5, 4, 3, 2, 1, 1], {'FILE_NAME':'영'})	AssertionError	AssertionError
mainClassifier([1, 2, 5, 2, 4, 1, 2], {'FILE_NAME':'영'})	AssertionError	AssertionError

부연설명 : 리스트 자료형을 점검하는 부분이므로, 2차 분류기준 딕셔너리는 {'FILE_NAME':'영'}로 고정함.

목적: 1차 분류 기준 리스트 요소의 자료형이 int형 인지 확인

입력 (호출)	예상결과	실제 결과
mainClassifier([1, 2, 3], {'FILE_NAME':'영'})	TypeError 없음	TypeError 없음
mainClassifier([3, 5, 1, 2], {'FILE_NAME':'영'})	TypeError 없음	TypeError 없음
mainClassifier([], {'FILE_NAME':'영'})	TypeError 없음	TypeError 없음
mainClassifier([5, 4, 3, 2, 1], {'FILE_NAME':'영'})	TypeError 없음	TypeError 없음
mainClassifier(['hi'], {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier(['123'], {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier(['안녕하세유'], {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier(['$%#'], {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier(['安志鎬'], {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier([1, 'hi', 2], {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier([1, '123', 2], {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier([5, 1, '안녕', 2], {'FILE_NAME':'영'})	TypeError	TypeError

mainClassifier([1, '안녕', '!!'], {'FILE_NAME':'영'})	TypeError	TypeError
mainClassifier(['abc', '안녕', '!!'], {'FILE_NAME':'영'})	TypeError	TypeError

부연설명 : 리스트 자료형을 점검하는 부분이므로, 2차 분류기준 딕셔너리는 {'FILE_NAME':'영'}로 고정함.

목적: 1차 분류 기준 리스트 요소가 0부터 5사이의 올바른 값을 가지는지 확인

입력 (호출)	예상결과	실제 결과
mainClassifier([1, 2], {'FILE_NAME':'영'})	AssertionError 없음	AssertionError 없음
mainClassifier([3, 5, 1, 2], {'FILE_NAME':'영', 'CREATED_DATE' : '월'})	AssertionError 없음	AssertionError 없음
mainClassifier([3, 4, 2], {'FILE_NAME':'영', 'CREATED_DATE' : '월', 'MODIFIED_DATE':'월'})	AssertionError 없음	AssertionError 없음

입력 (호출)	예상결과	실제 결과
mainClassifier([3, 6, 4, 2], {'FILE_NAME':'영', 'CREATED_DATE' : '월', 'MODIFIED_DATE':'월'})	AssertionError	AssertionError
mainClassifier([6, 2], {'FILE_NAME':'영'})	AssertionError	AssertionError
mainClassifier([3, 5, 1, 7, 2], {'FILE_NAME':'영', 'CREATED_DATE' : '월'})	AssertionError	AssertionError

목적: 2차 분류 기준 딕셔너리(second_sort_dict 파라미터) 형식이 dict인지 확인

입력 (호출)	예상결과	실제 결과
mainClassifier([1,2], {'FILE_NAME':'영'})	TypeError 없음	TypeError 없음
mainClassifier([3], {'MODIFIED_DATE':'년'})	TypeError 없음	TypeError 없음
mainClassifier([4], {'CREATED_DATE':'년'})	TypeError 없음	TypeError 없음
mainClassifier([5,4,3,2,1], {'FILE_NAME':'영', 'MODIFIED_DATE':'년', 'CREATED_DATE':'월'})	TypeError 없음	TypeError 없음

mainClassifier([1,2], 'hi')	TypeError	TypeError
mainClassifier([4], '안녕하세요')	TypeError	TypeError
mainClassifier([4], '安')	TypeError	TypeError
mainClassifier([4], '#$#')	TypeError	TypeError
mainClassifier([5,4,3,2,1], 1)	TypeError	TypeError
mainClassifier([4], -1)	TypeError	TypeError
mainClassifier([5,2], 3/5)	TypeError	TypeError
mainClassifier([5,2], ['1','가'])	TypeError	TypeError

목적: 2차 분류 기준 딕셔너리 길이가 0부터 3사이의 올바른 값인지 확인

입력 (호출)	예상결과	실제 결과
mainClassifier([1], {})	AssertionError 없음	AssertionError 없음
mainClassifier([3], {'CREATED_DATE":'년'})	AssertionError 없음	AssertionError 없음
mainClassifier([4], {'MODIFIED_DATE':'년'})	AssertionError 없음	AssertionError Error 없음
mainClassifier([3, 4], {'CREATED_DATE":'년' ,'MODIFIED_DATE':'년'})	AssertionError 없음	AssertionError Error 없음
mainClassifier([1, 3, 4, 2], {'CREATED_DATE":'년' ,'MODIFIED_DATE':'년','FILE_NAME':'한'})	AssertionError 없음	AssertionError Error 없음
mainClassifier([5,4,3,2,1], {'FILE_NAME':'영', 'MODIFIED_DATE':'년', 'CREATED_DATE':'월'})	AssertionError 없음	AssertionError 없음
mainClassifier([5,4,3,2,1],{'FILE_NAME':'영','MODIFIED_DATE':'년','CREATED_DATE':'월','1':'1'})	AssertionError	AssertionError
mainClassifier([5,4,3,2,1],{'FILE_NAME':'영','MODIFIED_DATE':'년','CREATED_DATE':'월','1':1,'2':2})		

AssertionError AssertionError

목적: 2차 분류 기준 딕셔너리 요소의 자료형이 str인지 확인

입력 (호출)	예상결과	실제 결과
mainClassifier([3], {'CREATED_DATE":'년'})	TypeError 없음	TypeError 없음
mainClassifier([4], {'MODIFIED_DATE':'년'})	TypeError 없음	TypeError 없음
mainClassifier([3, 4], {'CREATED_DATE":'년' ,'MODIFIED_DATE':'년'}) TypeError 없음 TypeError 없음		
mainClassifier([5,4,3,2,1], {'FILE_NAME':'영', 'MODIFIED_DATE':'년', 'CREATED_DATE':'월'}) TypeError 없음 TypeError 없음		
mainClassifier([3], {'CREATED_DATE":1})	TypeError	TypeError
입력 (호출)	예상결과	실제 결과
mainClassifier([4], {'MODIFIED_DATE':-1})	TypeError	TypeError
mainClassifier([4], {'MODIFIED_DATE':0.5})	TypeError	TypeError
mainClassifier([2],{'FILE_NAME':15%3}	TypeError	TypeError
mainClassifier([2],{'FILE_NAME':15/3}	TypeError	TypeError
mainClassifier([3], {'CREATED_DATE":15&3})	TypeError	TypeError

목적: 2차 분류 기준 딕셔너리의 key가 올바른 문자열인지 확인

입력 (호출)	예상결과	실제 결과
mainClassifier([3], {'CREATED_DATE":'년'})	KeyError 없음	KeyError 없음
mainClassifier([4], {'MODIFIED_DATE':'년'})	KeyError 없음	KeyError 없음
mainClassifier([3, 4], {'CREATED_DATE":'년' ,'MODIFIED_DATE':'년'}) KeyError 없음 KeyError 없음		

mainClassifier([5,4,3,2,1], {'FILE_NAME':'영', 'MODIFIED_DATE':'년', 'CREATED_DATE':'월'})		
KeyError 없음	KeyError 없음	
mainClassifier([3], {'DATE":"월"})	KeyError	KeyError
mainClassifier([4], {'MODIFIED':"년"})	KeyError	KeyError
mainClassifier([4], {'MODIFIED_DATA':"월"})	KeyError	KeyError
mainClassifier([2],{'FIL_NAME':"영"}	KeyError	KeyError
mainClassifier([5,4,3,2,1], {'FILE_NAME':'영', 'MODIFIED':'년', 'CREATE':'월'})		
KeyError	KeyError	

목적: 2차 분류 기준 딕셔너리 요소 값이 분류 기준 범주에 속하는지 확인

입력 (호출)	예상결과	실제 결과
mainClassifier([1, 2], {'FILE_NAME':'영'})	AssertionError 없음	AssertionError 없음
mainClassifier([3, 5, 1, 2], {'FILE_NAME':'영','CREATED_DATE':'월'})		
AssertionError 없음	AssertionError 없음	
mainClassifier([], {'FILE_NAME':'영'})	AssertionError 없음	AssertionError 없음
mainClassifier([5,4,3,2,1],{'FILE_NAME':'영','CREATED_DATE':'달','MODIFIED_DATE':'월'})		
AssertionError 없음	AssertionError 없음	
mainClassifier([1, 2], {'FILE_NAME':'영어!'})	AssertionError	AssertionError
mainClassifier([1, 2, 3], {'FILE_NAME':'영','CREATED_DATE':'월월'})		
AssertionError	AssertionError	
mainClassifier([5,4,3,2,1],{'FILE_NAME':'영','CREATED_DATE':'달달','MODIFIED_DATE':'월'})		
AssertionError	AssertionError	

목적: 1차 리스트 요소와 2차 딕셔너리 key가 서로 대응하는지 확인

입력 (호출)	예상결과	실제 결과
mainClassifier([1, 2], {'FILE_NAME':'영'})	KeyError 없음	KeyError 없음
mainClassifier([3], {'CREATED_DATE':'월'})	KeyError 없음	KeyError 없음
mainClassifier([4], {'MODIFIED_DATE':'월'})	KeyError 없음	KeyError 없음
mainClassifier([3, 5, 1, 2], {'FILE_NAME':'영','CREATED_DATE':'월'})		
KeyError 없음	KeyError 없음	
mainClassifier([5,4,3,2,1],{'FILE_NAME':'영','CREATED_DATE':'달','MODIFIED_DATE':'월'})		
KeyError 없음	KeyError 없음	
mainClassifier([1, 2], {'MODIFIED_DATE':'영어!'}	KeyError	KeyError
mainClassifier([3], {'FILE_NAME':'월'})	KeyError	KeyError
mainClassifier([4], {'FILE_NAME':'월'})	KeyError	KeyError
mainClassifier([1, 2, 3], {'MODIFIED_DATE':'영','CREATED_DATE':'월'})		
KeyError	KeyError	
mainClassifier([5,4,3,2,1],{'MODIFIED_DATE':'영','FILE_NAME':'달','MODIFIED_DATE':'월'})		
KeyError	KeyError	

3. 통합검사

사용자가 입력을 하거나 조작할 수 있는 각 상황마다 다음과 같이 검사했습니다.

3.1검사방법

Pycharm에서 프로그램을 직접 돌리며 콘솔창에 각각의 경우를 입력함.

3.2Test Cases 및 결과 정리

3.2.1 사용자 입력상황 1 : 작업폴더명을 입력할 시

목적: 파일 이름이 50자를 초과했을 때 반응 확인

예상 결과: 파일 이름 길이 제한을 초과했다는 오류 메시지 출력과 함께 다시 입력받을 수 있도록 입력문구 띄움.

실제 입력 및 실제 결과:

```
작업 폴더를 생성합니다.
작업 폴더명을 입력하여 주십시오 : abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz
파일명 지정은 50자이내 입니다.(파일명 미지정도 불가합니다.) 다시 설정해주세요.

작업 폴더명을 입력하여 주십시오 :
```

목적: 파일 아무 입력 없이 ENTER만 눌렀을 때 반응 확인

예상 결과: 파일명 미지정이 불가능하다는 오류 메시지 출력과 함께 다시 입력받을 수 있도록 입력문구 띄움.

실제 입력 및 실제 결과:

```
작업 폴더명을 입력하여 주십시오 :
파일명 지정은 50자이내 입니다.(파일명 미지정도 불가합니다.) 다시 설정해주세요.

작업 폴더명을 입력하여 주십시오 :
```

목적: 파일 이름에 유효하지 않은 특수문자(/, |, *, ", ?, <, >, ₩, : , " , *)가 포함되었을 때 반응 확인

예상 결과: 유효하지 않은 특수문자 리스트를 보여주며 파일명에 포함될 수 없음을 알리는 오류 메시지 출력과 함께 다시 입력 받을 수 있도록 입력문구 띄움.

실제 입력 및 실제 결과: (",*,₩,/,|만 캡쳐본 올림, 돌리긴 다 돌려봤고 정상작동됨)

```
작업 폴더를 생성합니다.
작업 폴더명을 입력하여 주십시오 : abc"
 {/ , \ , : , . , | , < , > , ? , *, "} 는 파일명에 포함될 수 없습니다. 다시 지정해주세요.

작업 폴더명을 입력하여 주십시오 :
```

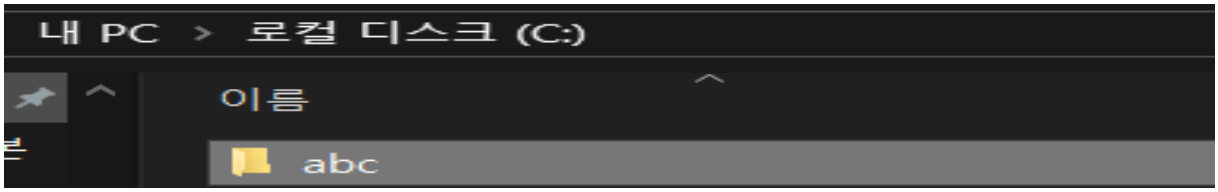
```
작업 폴더명을 입력하여 주십시오 : a*bc
 {/ , \ , : , . , | , < , > , ? , *, "} 는 파일명에 포함될 수 없습니다. 다시 지정해주세요.

작업 폴더명을 입력하여 주십시오 :
```

```
작업 폴더명을 입력하여 주십시오 : a\/|b
 {/ , \ , : , . , | , < , > , ? , *, "} 는 파일명에 포함될 수 없습니다. 다시 지정해주세요.

작업 폴더명을 입력하여 주십시오 :
```

목적: C 드라이브 바로 아래에 사용자가 입력한 폴더명과 같은 폴더가 이미 존재할 경우 반응 확인

예상 결과: 파일명이 중복되었음을 알리는 오류 메시지 출력과 함께 다시 입력받을 수 있도록 입력문구 띄움.

실제 입력 및 실제 결과:



```
작업 폴더를 생성합니다.
작업 폴더명을 입력하여 주십시오 : abc
파일명이 중복되었습니다. 다시 지정해주세요.

작업 폴더명을 입력하여 주십시오 :
```

목적: 파일명에 공백이 존재할 경우 반응 확인

예상 결과: 파일명에 공백이 허용되지 않음을 알리는 오류 메시지 출력과 함께 다시 입력받을 수 있도록 입력문구 띄움.

실제 입력 및 실제 결과:

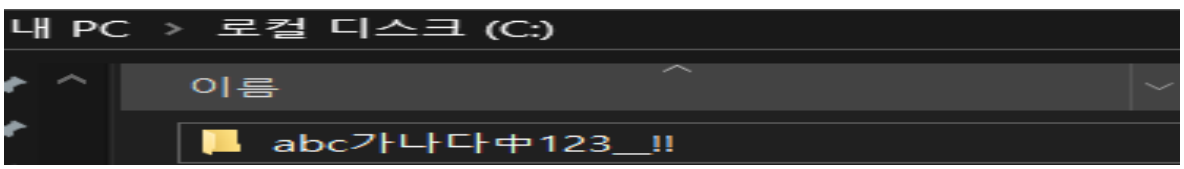
```
작업 폴더명을 입력하여 주십시오 : a                bc
파일명에 공백이 존재하면 안됩니다
작업 폴더명을 입력하여 주십시오 :
```

목적: 한글, 영어, 숫자, 유효하지 않은 특수문자이외의 특수문자, 한자를 합쳐 파일명으로 지정했을 때 반응 확인—(정상작동 되는거 합쳐놓은거임.)

예상 결과: 정상적으로 C 드라이브 바로 아래에 작업 폴더 생성

실제 입력 및 실제 결과:

```
작업 폴더를 생성합니다.
작업 폴더명을 입력하여 주십시오 : abc가나다中123__!!
작업 폴더가 abc가나다中123__!! 이라는 이름으로 생성되었습니다.
사진파일을 작업 폴더에 넣어주세요.
```



3.2.2 사용자 입력상황 2 : 작업 폴더에 파일 넣을 시

검사방법

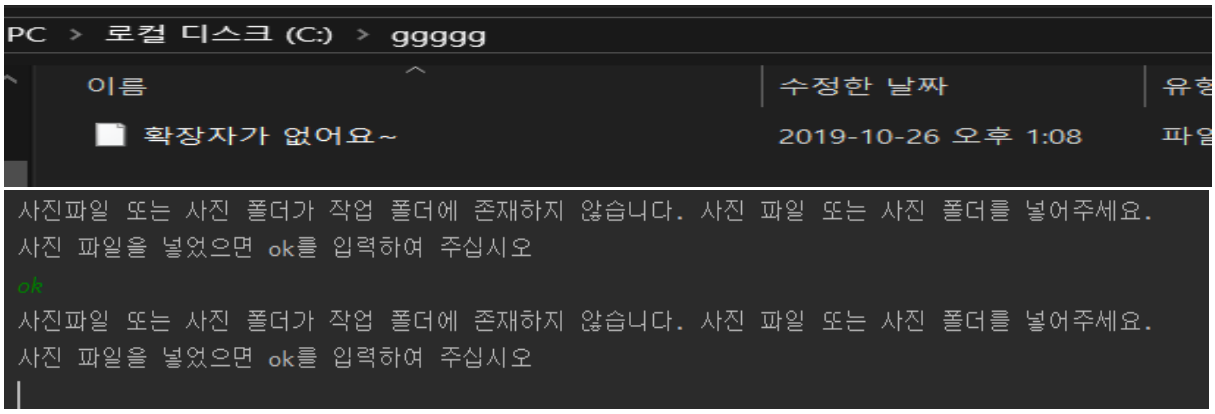
Pycharm에서 프로그램을 직접 돌려 작업 폴더를 생성한 후 여러 종류의 확장자를 가진 파일들을 넣어봄.

Test Cases 및 결과 정리

목적: 작업 폴더에 확장자가 없는 파일만 넣었을 경우 반응 확인

예상 결과: 작업 폴더에 사진파일이 존재하지 않으므로 폴더에 사진을 넣으라는 메시지 출력

실제 입력 및 실제 결과:



목적: 작업 폴더에 확장자가 없는 파일과 프로그램에서 사진파일로 취급해주기로 한 { jpeg, jpg, png, bmp, gif } 중 한 파일을 같이 넣었을 경우 반응 확인

예상 결과: 사진 파일이 확인 되었다는 문구 출력

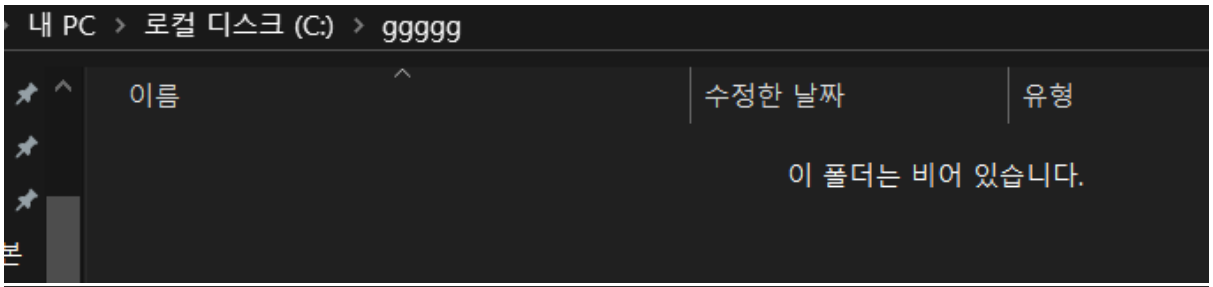
실제 입력 및 실제 결과:

```
사진파일 또는 사진 폴더가 작업 폴더에 존재하지 않습니다. 사진 파일 또는 사진 폴더를 넣어주세요.
사진 파일을 넣었으면 ok를 입력하여 주십시오
ok
사진 파일이 확인되었습니다.
```

목적: 작업 폴더에 아무 파일도 넣지 않은 채로 다음 단계인 ok를 제대로 입력할 경우 반응 확인

예상 결과: 작업 폴더에 사진파일이 존재하지 않으므로 폴더에 사진을 넣으라는 메시지 출력

실제 입력 및 실제 결과:




```
작업 폴더를 생성합니다.
작업 폴더명을 입력하여 주십시오 : ggggg
작업 폴더가 ggggg 이라는 이름으로 생성되었습니다.
사진파일을 작업 폴더에 넣어주세요.

사진파일 또는 사진 폴더가 작업 폴더에 존재하지 않습니다. 사진 파일 또는 사진 폴더를 넣어주세요.
사진 파일을 넣었으면 ok를 입력하여 주십시오
ok
사진파일 또는 사진 폴더가 작업 폴더에 존재하지 않습니다. 사진 파일 또는 사진 폴더를 넣어주세요.
사진 파일을 넣었으면 ok를 입력하여 주십시오
```

3.2.3 사용자 입력상황 3 : OK 입력 시

검사방법

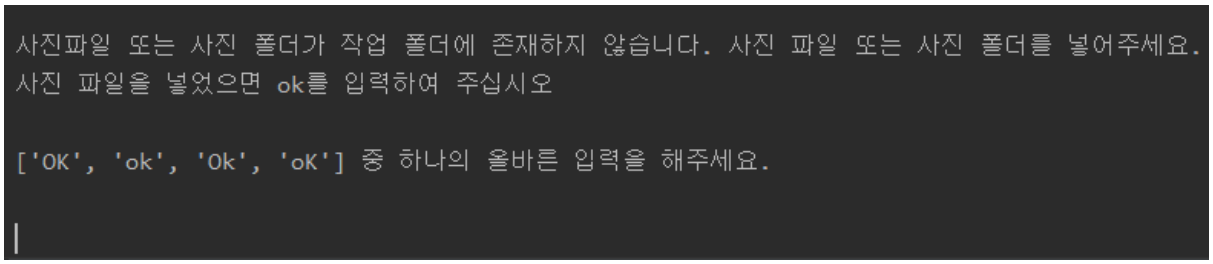
Pycharm에서 프로그램을 직접 돌려 작업 폴더를 생성, 폴더 안에 사진 파일들을 넣은 후 여러 상황들을 입력해봄.

Test Cases 및 결과 정리

목적: 아무 입력 없이 ENTER만 눌렀을 때 반응 확인

예상 결과: 올바른 OK의 예를 보여주며 이 중 하나를 입력하라는 문구 출력과 함께 다시 받음.

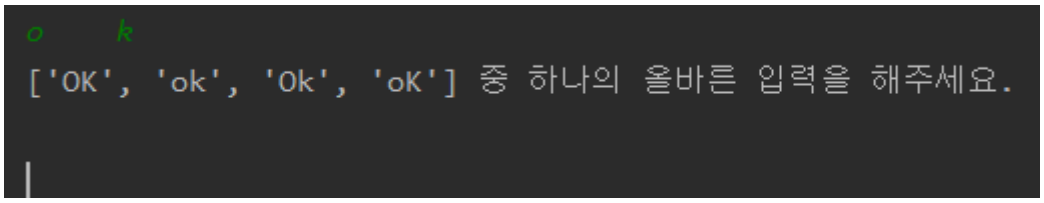
실제 입력 및 실제 결과:



목적: 공백을 포함한 ok를 입력했을 때 반응 확인

예상 결과: 올바른 OK의 예를 보여주며 이 중 하나를 입력하라는 문구 출력과 함께 다시 받음.

실제 입력 및 실제 결과:



목적: ['OK', 'ok', 'Ok', oK'] 중 하나를 올바르게 입력하고 특수문자를 붙였을 때 반응 확인

예상 결과: 올바른 OK의 예를 보여주며 이 중 하나를 입력하라는 문구 출력과 함께 다시 받음.

실제 입력 및 실제 결과:

```
사진파일 또는 사진 폴더가 작업 폴더에 존재하지 않습니다. 사진 파일 또는 사진 폴더를 넣어주세요.
사진 파일을 넣었으면 ok를 입력하여 주십시오
ok!
['OK', 'ok', 'Ok', 'oK'] 중 하나의 올바른 입력을 해주세요.

```

목적: ['OK', 'ok', 'Ok', oK'] 중 여러 개를 입력했을 때 반응 확인

예상 결과: 올바른 OK의 예를 보여주며 이 중 하나를 입력하라는 문구 출력과 함께 다시 받음.

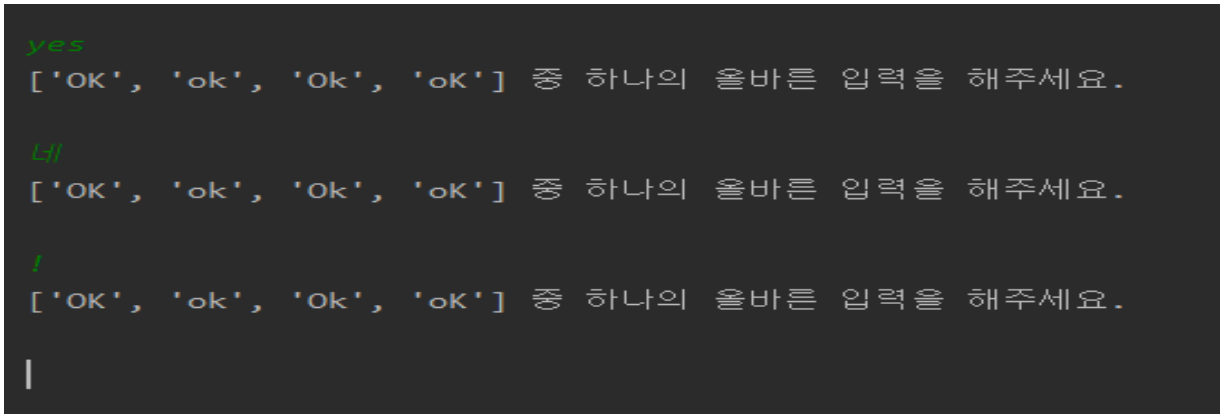
실제 입력 및 실제 결과:

```
사진파일 또는 사진 폴더가 작업 폴더에 존재하지 않습니다. 사진 파일 또는 사진 폴더를 넣어주세요.
사진 파일을 넣었으면 ok를 입력하여 주십시오
okOk
['OK', 'ok', 'Ok', 'oK'] 중 하나의 올바른 입력을 해주세요.

ok, OK
['OK', 'ok', 'Ok', 'oK'] 중 하나의 올바른 입력을 해주세요.

```

목적: ['OK', 'ok', 'Ok', oK'] 이외의 값을 입력 했을 때 반응 확인

예상 결과: 올바른 OK의 예를 보여주며 이 중 하나를 입력하라는 문구 출력과 함께 다시 받음.

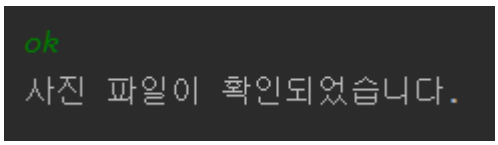
실제 입력 및 실제 결과:

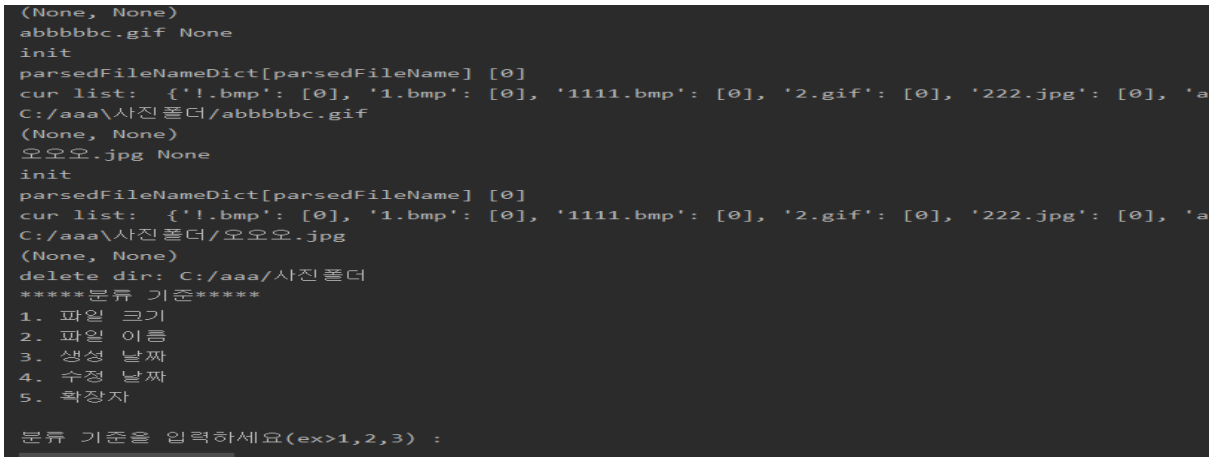


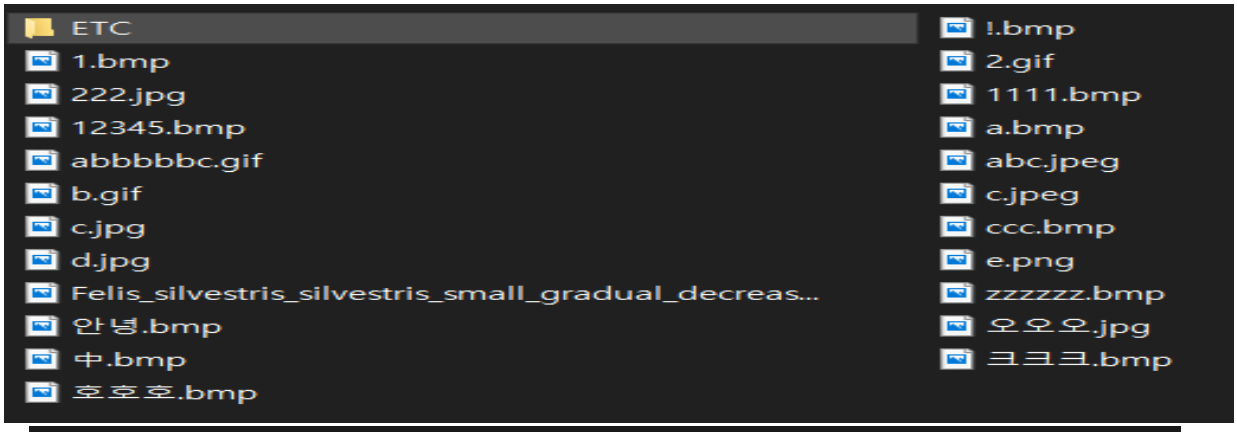
목적: ['OK', 'ok', 'Ok', oK'] 중 하나를 올바르게 입력했을 때 반응 확인 ---정상작동

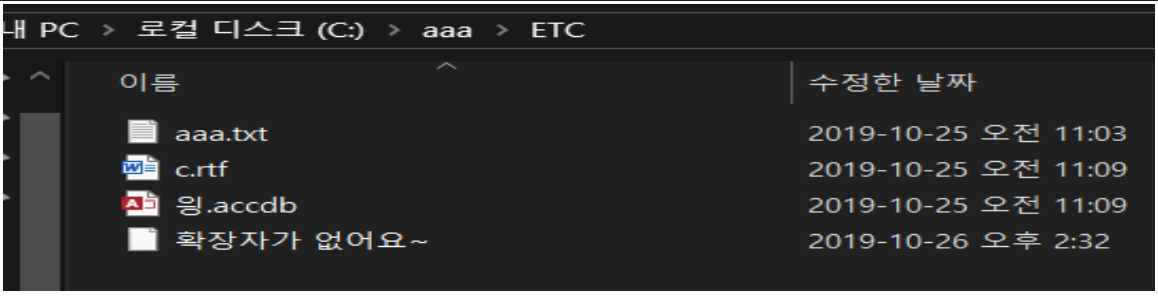
예상 결과: 사진 파일이 확인 됐다는 메시지 출력과 함께 파일이나 사진폴더의 각각의 정보가 출력됨. 폴더 속의 파일들은 ROOT폴더로 옮겨지며 폴더는 삭제되고 지정된 확장자가 아닌 확장자를 가지고 있는 파일들은 ETC폴더로 분류됨.

실제 입력 및 실제 결과:






ETC
1.bmp
222.jpg
12345.bmp
abbbbbc.gif
b.gif
c.jpg
d.jpg
Felis_silvestris_silvestris_small_gradual_decreas...
안녕.bmp
中.bmp
호호호.bmp
!.bmp
2.gif
1111.bmp
a.bmp
abc.jpeg
c.jpeg
ccc.bmp
e.png
zzzzzz.bmp
오오오.jpg
크크크.bmp


내 PC > 로컬 디스크 (C:) > aaa > ETC
이름
수정한 날짜
aaa.txt 2019-10-25 오전 11:03
c.rtf 2019-10-25 오전 11:09
잉.accdb 2019-10-25 오전 11:09
확장자가 없어요~ 2019-10-26 오후 2:32

3.2.4 사용자 입력상황 4 : 분류 기준 입력 시

->결과 화면 두개인건 처음 오류났을 때

검사방법

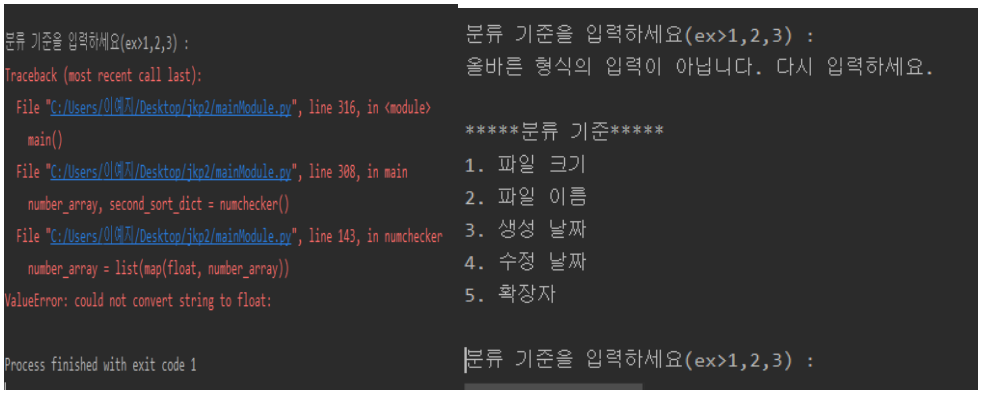
Pycharm에서 프로그램을 직접 돌려 분류기준을 입력받는 콘솔창에 여러 상황을 입력해 봄.

Test Cases 및 결과 정리

목적: 아무 입력 없이 ENTER만 눌렀을 때 반응 확인

예상 결과: 올바른 형식의 입력이 아니라는 메시지 출력과 함께 다시 입력받음.

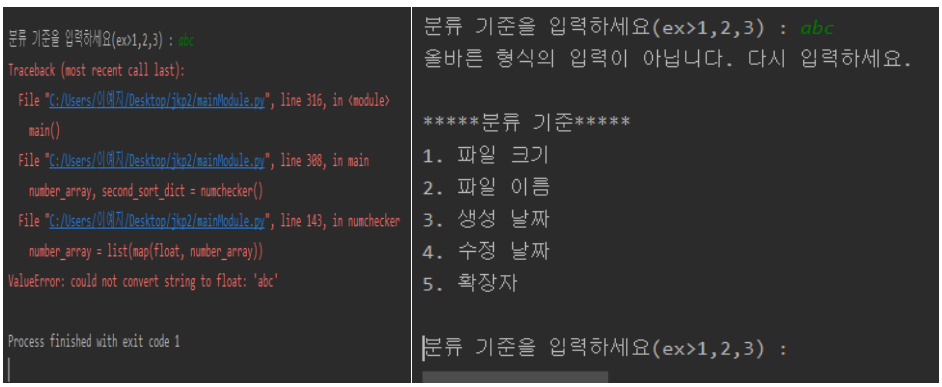
실제 입력 및 실제 결과:



목적: {1,2,3,4,5}의 원소가 아닌 문자 또는 문자열을 값으로 입력했을 때 반응 확인

예상 결과: 유효한 값이 아니라는 메시지 출력과 함께 다시 입력받음.

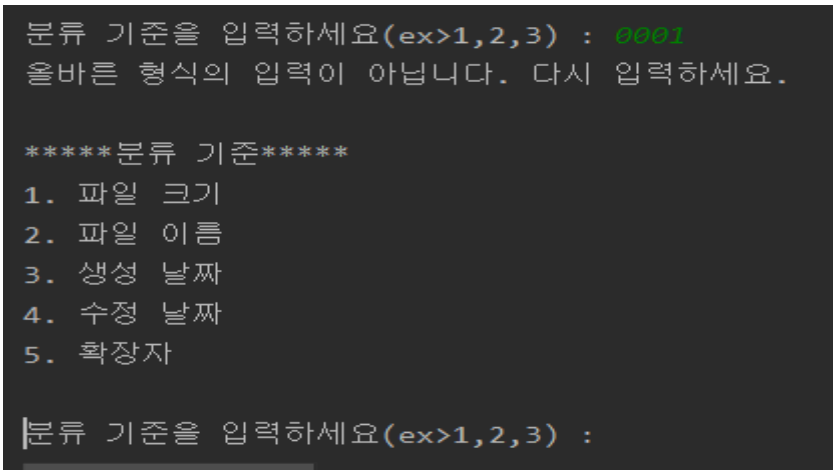
실제 입력 및 실제 결과:



목적: 선행 0을 포함한 숫자를 입력했을 때 반응 확인

예상 결과: 올바른 형식이 아니므로 다시 입력하라는 오류메세지 출려과 함께 다시 받음.

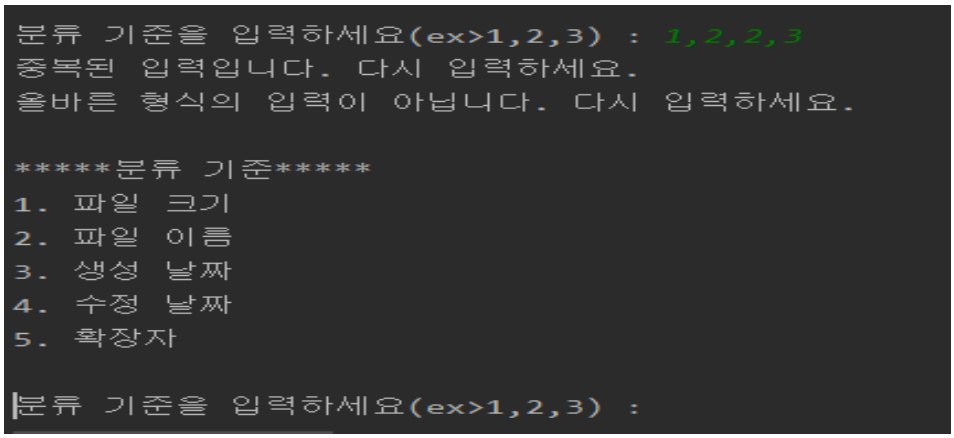
실제 입력 및 실제 결과:



목적: 중복된 숫자를 입력했을 때 반응 확인

예상 결과: 중복된 값이라는 오류 메시지 출력과 함께 다시 입력받음.

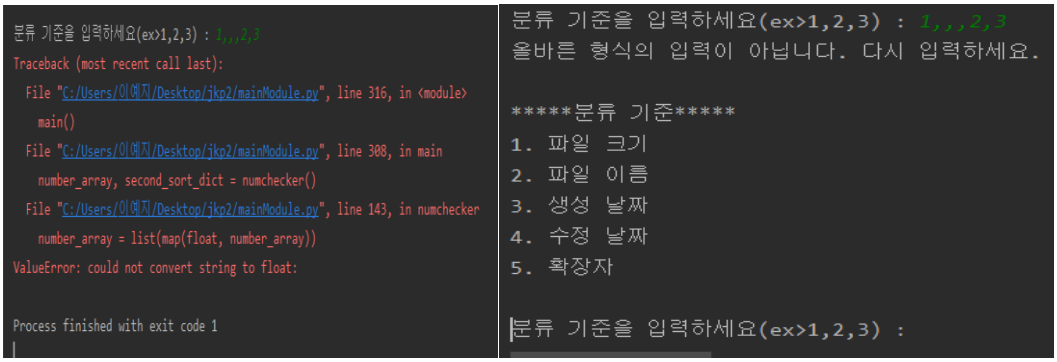
실제 입력 및 실제 결과:



목적: 숫자를 여러 개 입력할 때, 숫자와 숫자 사이에 콤마(,)를 2개 이상 입력한 경우 반응 확인

예상 결과: 올바른 형식이 아니므로 다시 입력하라는 오류메세지 출려과 함께 다시 받음.

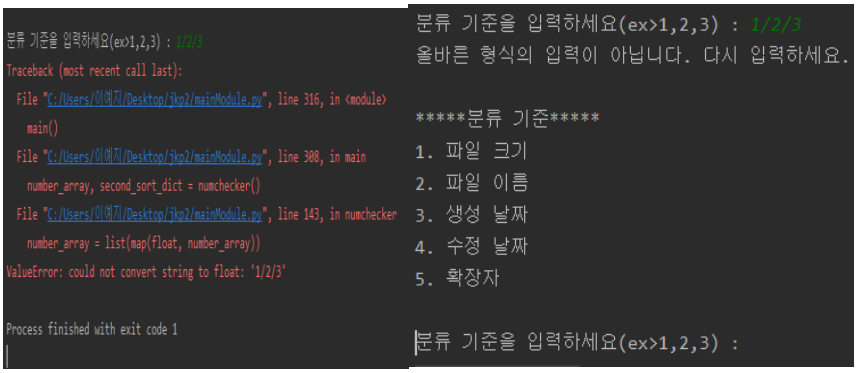
실제 입력 및 실제 결과:



목적: 숫자 구획 문자로 콤마(,)가 아닌 다른 구획 문자를 입력했을 때 반응 확인

예상 결과: 올바른 형식이 아니므로 다시 입력하라는 오류메세지 출려과 함께 다시 받음.

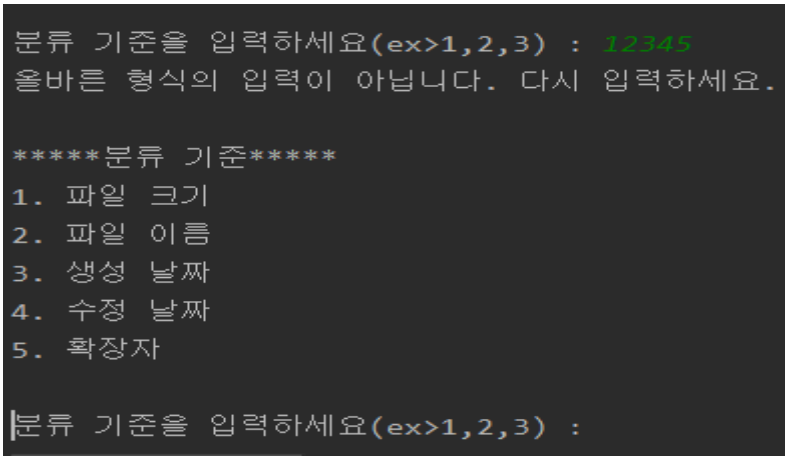
실제 입력 및 실제 결과:



목적: 구획 문자(,)없이 숫자만 입력했을 때 반응 확인

예상 결과: 올바른 형식이 아니므로 다시 입력하라는 오류메세지 출려과 함께 다시 받음.

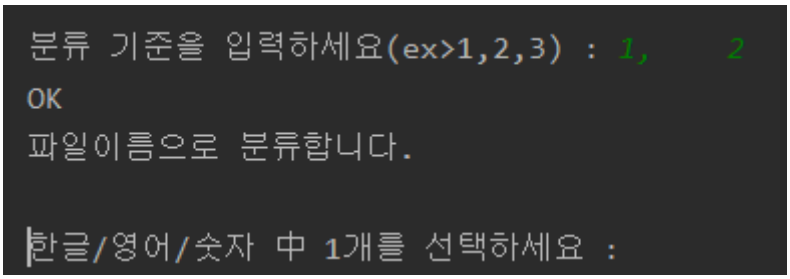
실제 입력 및 실제 결과:



목적: 숫자와 구획 문자(,) 사이에 공백이 있을 때 반응 확인

예상 결과: 정상적으로 받아짐.

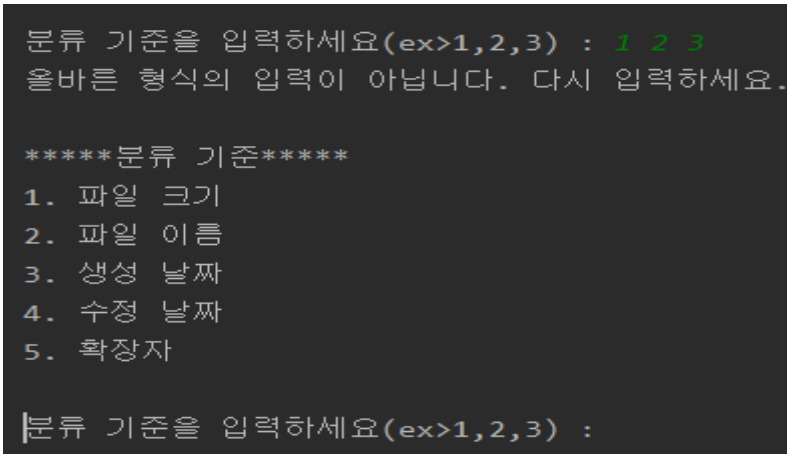
실제 입력 및 실제 결과:



목적: 구획 문자 대신 공백이 있을 때 반응 확인

예상 결과: 올바른 형식이 아니므로 다시 입력하라는 오류메세지 출려과 함께 다시 받음.

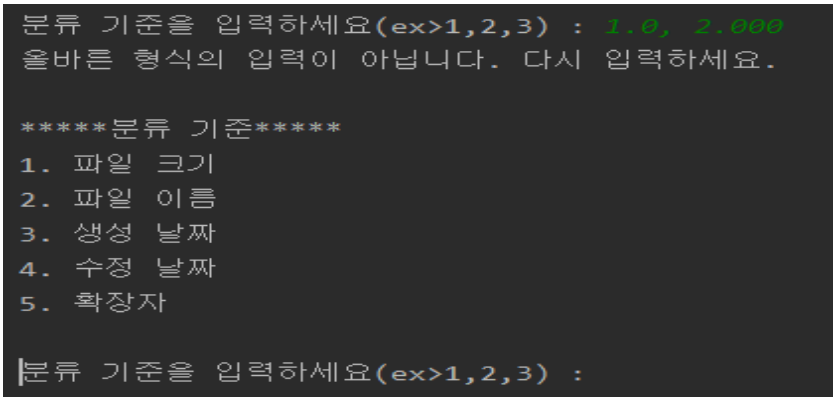
실제 입력 및 실제 결과:



목적: 입력값을 정수가 아닌 실수로 입력했을 때 반응 확인

예상 결과: 올바른 형식이 아니므로 다시 입력하라는 오류메세지 출려과 함께 다시 받음.

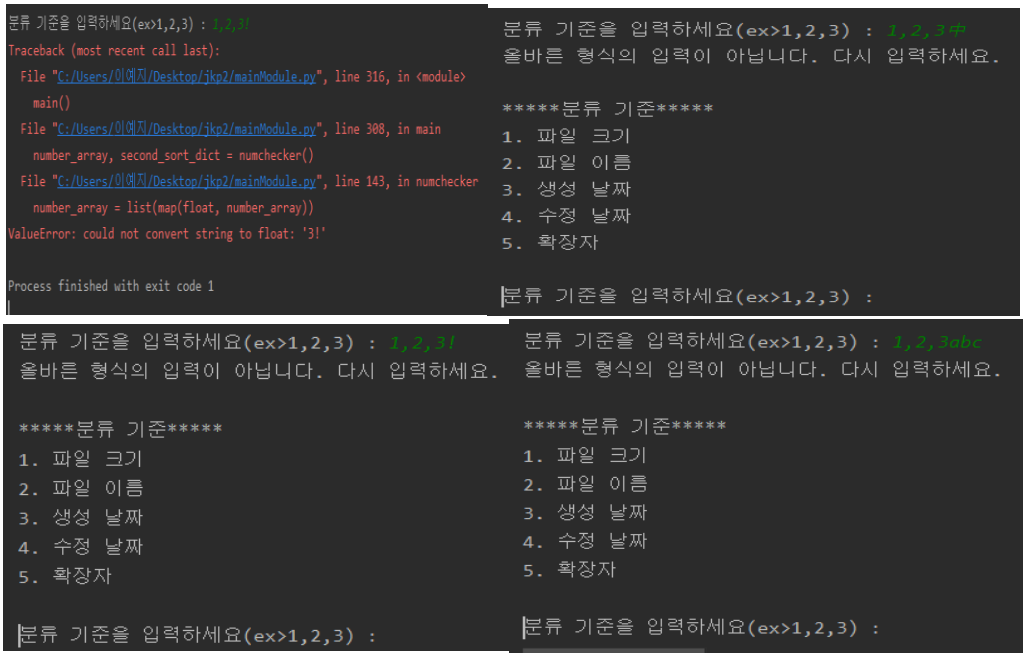
실제 입력 및 실제 결과:



목적: 올바른 입력 후 특수문자나 한자나 문자를 같이 입력했을 때 반응 확인

예상 결과: 올바른 형식이 아니므로 다시 입력하라는 오류메세지 출력과 함께 다시 받음.

실제 입력 및 실제 결과:

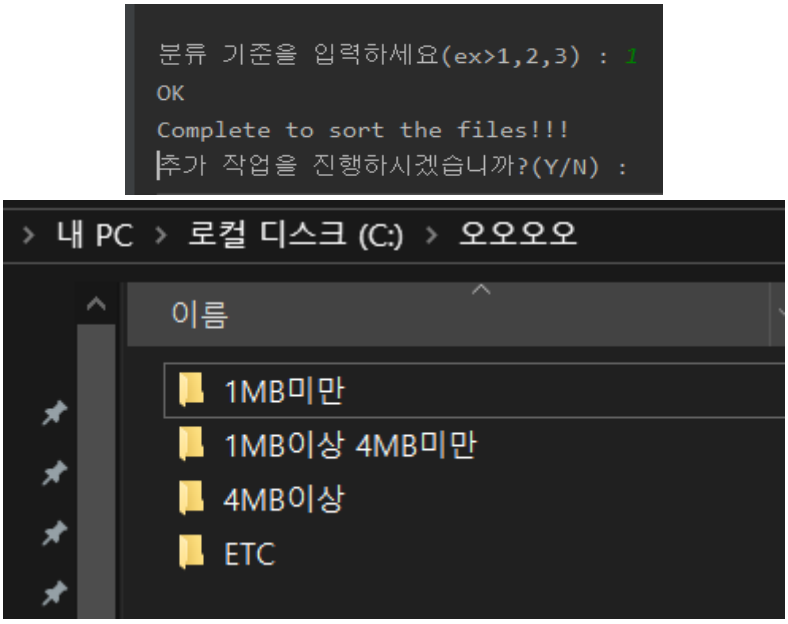


<분류 기준을 1개만 입력 했을 경우>

목적: 1 차 분류기준으로 1 번 선택 시 반응 확인

예상 결과 : 사진 크기에 따라 분류될것이라 예측.

실제 입력 및 실제 결과:



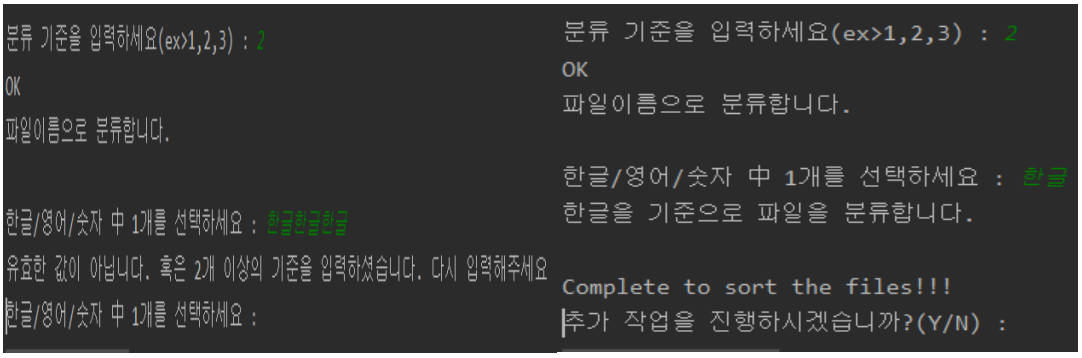
목적: 1차 분류기준으로 2번 선택 후-한글 문자 기준 선택했을 때 (한글/한/kor/korean/Korean/Kor/KOREAN/KOR) 이외의 값 입력했을 때와 이 안의 값 입력했을 때

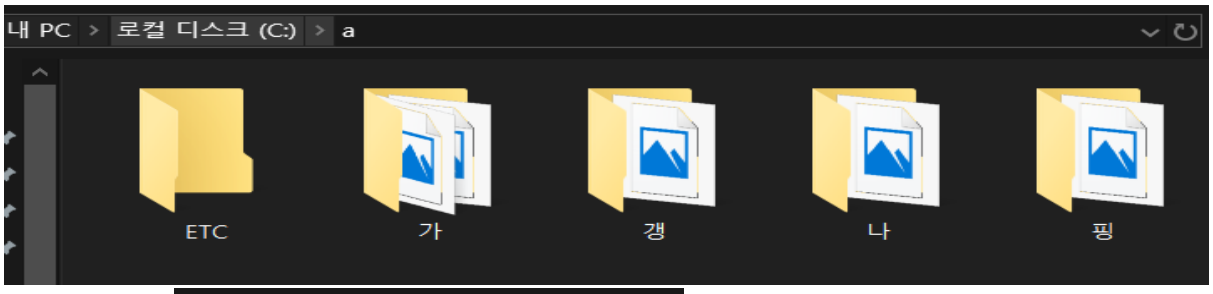
예상 결과:

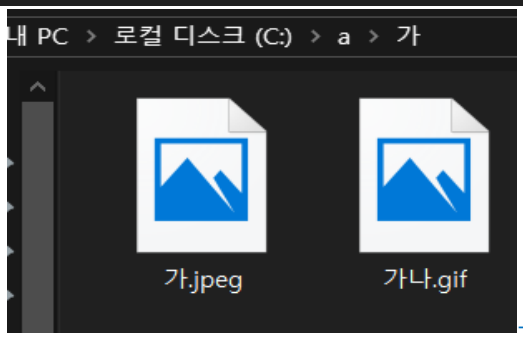
- (한글/한/kor/korean/Korean/Kor/KOREAN/KOR) 이외의 값 입력했을 때 : 유효한 값이 아니라는 오류 메시지 출력과 함께 다시 입력받음.

- (한글/한/kor/korean/Korean/Kor/KOREAN/KOR) 안의 값 입력했을 때 : 한글을 기준으로 파일 분류함.

실제 입력 및 실제 결과:







-->결과 중 '가'폴더 안 캡쳐

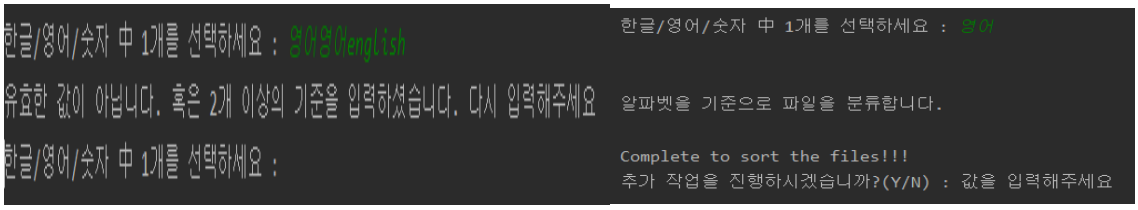
목적: 1 차 분류기준으로 2 번 선택 후-알파벳 기준 선택했을 때 (영어/영/eng/english/Eng/English/ENGLISH/ENG)이외의 값 입력했을 때와 이 안의 값 입력했을 때

예상 결과:

-(영어/영/eng/english/Eng/English/ENGLISH/ENG)이외의 값 입력했을 때 : 유효한 값이 아니라는 오류 메시지 출력과 함께 다시 입력받음.

-(영어/영/eng/english/Eng/English/ENGLISH/ENG)안의 값 입력했을 때 : 알파벳을 기준으로 분류함.

실제 입력 및 실제 결과:



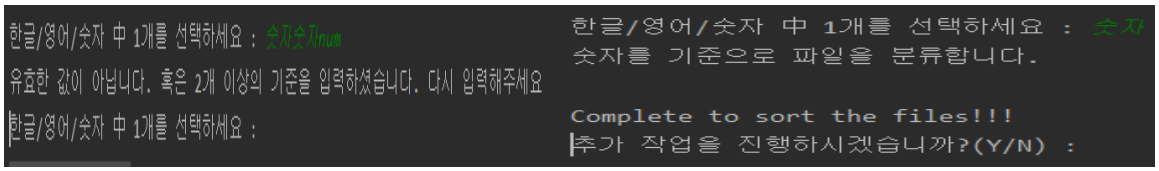
목적: 1차 분류기준으로 2번 선택 후-숫자 기준 선택했을 때 (숫자/num /number/Number/Num/ NUM/NUMBER)이외의 값 입력했을 때와 이 안의 값 입력했을 때

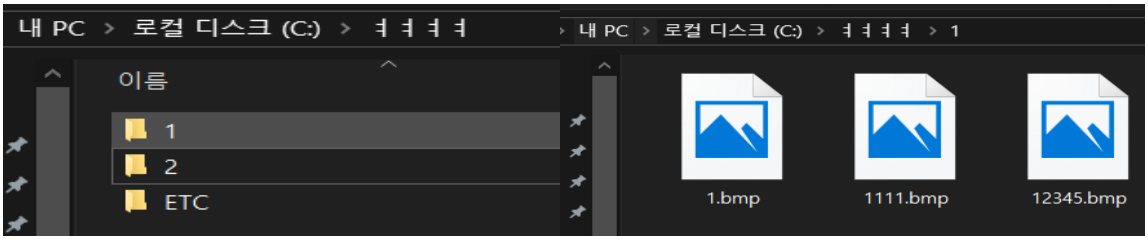
예상 결과:

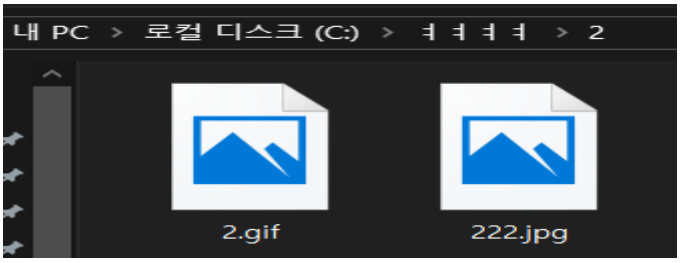
-(숫자/num /number/Number/Num/NUM/NUMBER)이외의 값 입력했을 때 : 유효한 값이 아니라는 오류 메시지 출력과 함께 다시 입력받음.

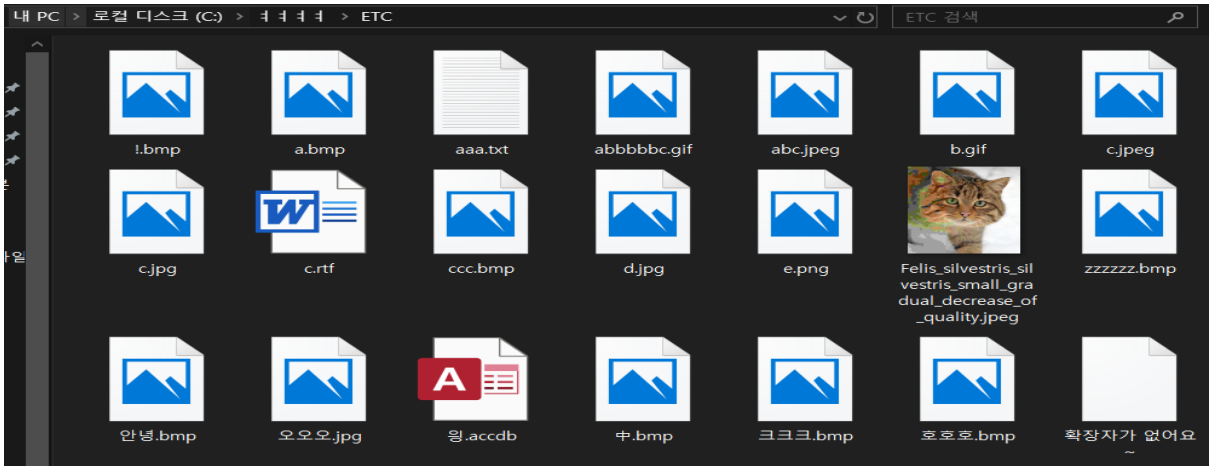
-(숫자/num /number/Number/Num/NUM/NUMBER)안의 값 입력했을 때 : 숫자를 기준으로 분류함.

실제 입력 및 실제 결과:









목적: 1차 분류기준으로 2번 선택 후-한글,알파벳,숫자 기준 중 2개 이상의 기준을 선택할 경우

예상 결과: 2개 이상의 기준을 입력했다는 오류 메시지 출력과 함께 다시 입력받음.

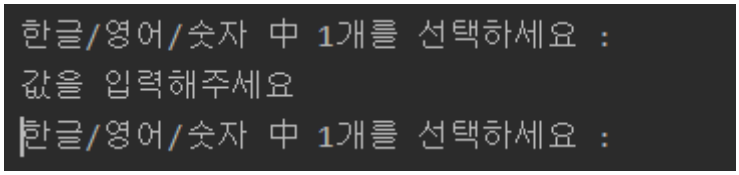
실제 입력 및 실제 결과:

```
한글/영어/숫자 中 1개를 선택하세요 : 한글, 영어
유효한 값이 아닙니다. 혹은 2개 이상의 기준을 입력하셨습니다. 다시 입력해주세요
한글/영어/숫자 中 1개를 선택하세요 : 한글 영어 숫자
유효한 값이 아닙니다. 혹은 2개 이상의 기준을 입력하셨습니다. 다시 입력해주세요
한글/영어/숫자 中 1개를 선택하세요 :
```

목적: 1차 분류 기준으로 2번 선택 후- 분류 기준을 선택하지 않은 상태로 ENTER를 했을 경우

예상 결과: 값을 입력하라는 오류 메시지 출력과 함께 다시 입력받음.

실제 입력 및 실제 결과:



목적: 1차 분류 기준으로 3번이나 4번 중 한 개를 선택 후- 년도를 기준으로 선택했을 때 (년/년도/Y/Year/y/year/YEAR)이외의 값을 입력했을 때와 이 안의 값을 입력했을 때

예상 결과:

-(년/년도/Y/Year/y/year/YEAR)이외의 값을 입력했을 때 : 유효한 값이 아니라는 오류 메시지 출력과 함께 다시 입력받음.

-(년/년도/Y/Year/y/year/YEAR)안의 값을 입력했을 때 : 년도를 기준으로 분류함.

실제 입력 및 실제 결과:

목적: 1차 분류 기준으로 3번이나 4번 중 한 개를 선택 후- 월을 기준으로 선택했을 때 (월/달/M/Month/m/MONTH)이외의 값을 입력했을 때와 이 안의 값을 입력했을 때

예상 결과:

-(월/달/M/Month/m/MONTH))이외의 값을 입력했을 때 : 유효한 값이 아니라는 오류 메시지 출력과 함께 다시 입력받음.

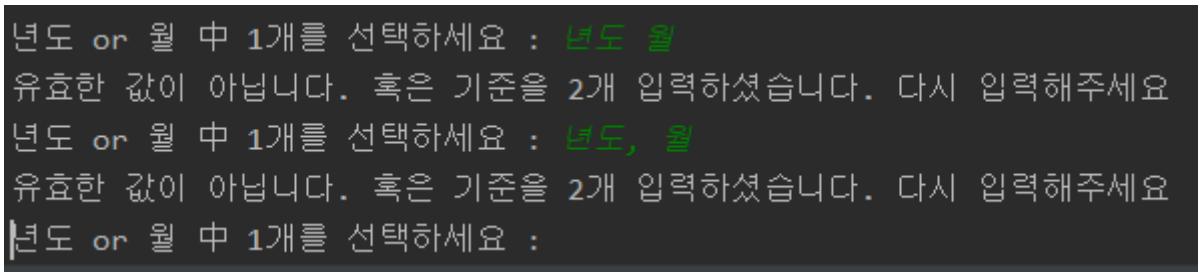
-(월/달/M/Month/m/MONTH))안의 값을 입력했을 때 : 월을 기준으로 분류함.

실제 입력 및 실제 결과:

목적: 1 차 분류기준으로 3 번이나 4 번중 한 개를 선택 후-월,별 기준 중 2 개의 기준을 선택할 경우

예상 결과: 2개 이상의 기준을 입력했다는 오류 메시지 출력과 함께 다시 입력받음.

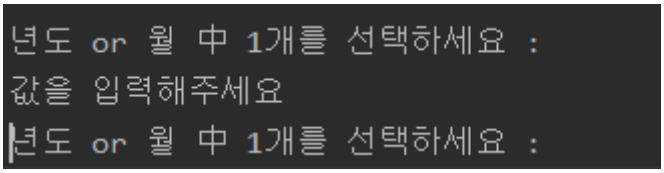
실제 입력 및 실제 결과:



목적: 1 차 분류 기준으로 3 번이나 4 번 중 한 개를 선택 후- 분류 기준을 선택하지 않은 상태로 ENTER 를 했을 경우

예상 결과: 값을 입력하라는 오류 메시지 출력과 함께 다시 입력받음.

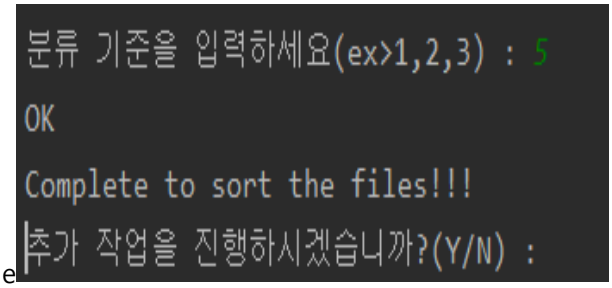
실제 입력 및 실제 결과:

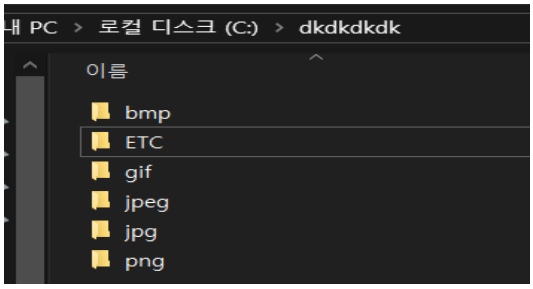


목적: 1 차 분류기준으로 5 번 선택 시 반응 확인

예상 결과 : 확장자에 따라 분류될것이라 예측.

실제 입력 및 실제 결과:





e

<분류 기준을 여러 개 입력했을 경우>

목적: 1차 분류 기준으로 여러 기준을 선택 했을 때 반응 확인(대표 케이스로 1,2,3,4,5 모두 입력 했을 때로 테스트 함.-2번 기준:한글/3번 기준:년도/4번 기준:월)

예상 결과: 입력한 순서대로 분류가 진행 될 것이라 예측.

실제 입력 및 실제 결과:

```
분류 기준을 입력하세요(ex>1,2,3) : 1,2,3,4,5
OK
파일이름으로 분류합니다.

한글/영어/숫자 中 1개를 선택하세요 : 허안
유효한 값이 아닙니다. 혹은 2개 이상의 기준을 입력하셨습니다. 다시 입력해주세요
한글/영어/숫자 中 1개를 선택하세요 : englishes
유효한 값이 아닙니다. 혹은 2개 이상의 기준을 입력하셨습니다. 다시 입력해주세요
한글/영어/숫자 中 1개를 선택하세요 : numbers
유효한 값이 아닙니다. 혹은 2개 이상의 기준을 입력하셨습니다. 다시 입력해주세요
한글/영어/숫자 中 1개를 선택하세요 : 한글
한글을 기준으로 파일을 분류합니다.

날짜를 기준으로 분류합니다.
년도 or 월 中 1개를 선택하세요 : 이어
유효한 값이 아닙니다. 혹은 기준을 2개 입력하셨습니다. 다시 입력해주세요
년도 or 월 中 1개를 선택하세요 : 먼스
유효한 값이 아닙니다. 혹은 기준을 2개 입력하셨습니다. 다시 입력해주세요
년도 or 월 中 1개를 선택하세요 : 년
년도를 기준으로 파일을 분류합니다.

날짜를 기준으로 분류합니다.
년도 or 월 中 1개를 선택하세요 : 월
월을 기준으로 파일을 분류합니다.

Complete to sort the files!!!
추가 작업을 진행하시겠습니까?(Y/N) :
```

예상대로 1,2,3,4,5 순서대로 분류가 되었음.

목적: 1 차 분류 기준으로 여러 개를 입력할 때 기준의 순서를 바꿔서 입력했을 때 반응 확인(대표 케이스로 5,1,2,4,3 을 입력했을 때로 테스트 함.-2 번 기준: 한글/ 3 번 기준: 월/ 4 번 기준: 년도)

예상 결과: 기준의 순서를 바꿔서 입력하면 입력한 순서대로 분류될 것이라 예측.

실제 입력 및 실제 결과:

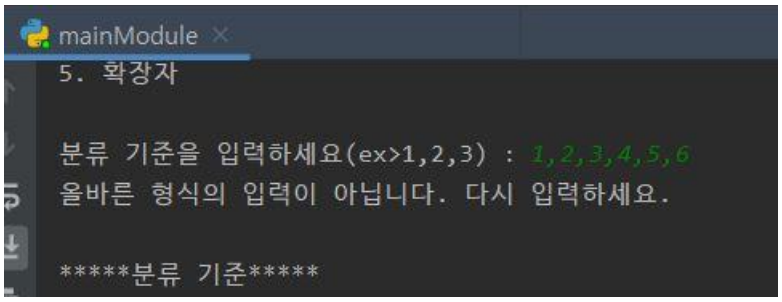
```
*****분류 기준*****
1. 파일 크기
2. 파일 이름
3. 생성 날짜
4. 수정 날짜
5. 확장자

분류 기준을 입력하세요(ex>1,2,3) : 5, 1, 2,    4, 3
OK
파일이름으로 분류합니다.

한글/영어/숫자 中 1개를 선택하세요 : kkorea
유효한 값이 아닙니다. 혹은 2개 이상의 기준을 입력하셨습니다. 다시 입력해주세요
한글/영어/숫자 中 1개를 선택하세요 : korea
유효한 값이 아닙니다. 혹은 2개 이상의 기준을 입력하셨습니다. 다시 입력해주세요
한글/영어/숫자 中 1개를 선택하세요 : korean
한글을 기준으로 파일을 분류합니다.

날짜를 기준으로 분류합니다.
년도 or 월 中 1개를 선택하세요 : m
월을 기준으로 파일을 분류합니다.

날짜를 기준으로 분류합니다.
년도 or 월 中 1개를 선택하세요 : y
년도를 기준으로 파일을 분류합니다.

Complete to sort the files!!!
추가 작업을 진행하시겠습니까?(Y/N) :
```

- 기준의 순서를 바꿔서 입력하면 입력한 순서대로 분류되는 것이 아닌 순서대로 분류를 진행함

목적: 1차 분류 기준으로 여러 개를 입력할 때 5를 초과한 숫자를 입력하거나 5개를 초과한 숫자를 입력했을 때 반응 확인

예상 결과: 올바른 형식의 입력이 아니라는 오류 메시지 출력과 함께 다시 입력 받음.

실제 입력 및 실제 결과:



3.2.5 사용자 입력상황 5 : 추가 작업 여부 입력 시

검사방법

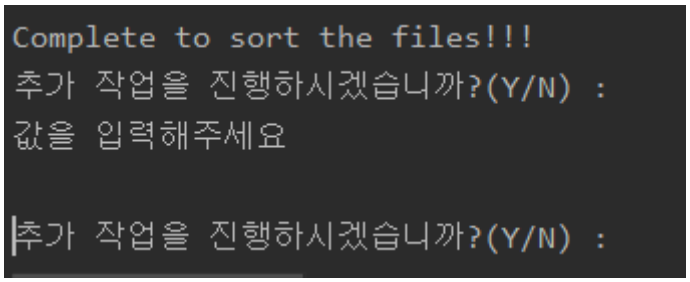
Pycharm에서 프로그램을 직접 돌려 추가 작업을 입력받는 콘솔창에 여러 상황을 입력해 봄.

Test Cases 및 결과 정리

목적: 아무 입력 없이 ENTER만 눌렀을 때 반응 확인

예상 결과: 값을 입력하라는 오류 메시지 출력과 함께 다시 입력받을 수 있도록 입력문구 출력함.

실제 입력 및 실제 결과:



목적: 공백을 포함한 문자를 입력했을 때 반응 확인

예상 결과: ['Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO'] 중 1개의 값을 입력하라는 오류 메시지 출력과 함께 다시 입력 받음.

실제 입력 및 실제 결과:

```
추가 작업을 진행하시겠습니까?(Y/N) : y     es
'Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO' 중 1개의 값을 입력해 주세요

추가 작업을 진행하시겠습니까?(Y/N) :
```

목적: [N, n, No, no, NO, 아니오, 아니요, Y, y, Yes, yes, 네, 예, YES] 중 여러 개를 입력했을 때 반응 확인

예상 결과: ['Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO'] 중 1개의 값을 입력하라는 오류 메시지 출력과 함께 다시 입력 받음.

실제 입력 및 실제 결과:

```
추가 작업을 진행하시겠습니까?(Y/N) : y,yes
'Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO' 중 1개의 값을 입력해 주세요

추가 작업을 진행하시겠습니까?(Y/N) : n,no
'Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO' 중 1개의 값을 입력해 주세요

추가 작업을 진행하시겠습니까?(Y/N) : y,n
'Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO' 중 1개의 값을 입력해 주세요

추가 작업을 진행하시겠습니까?(Y/N) : y yes
'Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO' 중 1개의 값을 입력해 주세요

추가 작업을 진행하시겠습니까?(Y/N) : n no
'Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO' 중 1개의 값을 입력해 주세요

추가 작업을 진행하시겠습니까?(Y/N) :
```

목적: [N, n, No, no, NO, 아니오, 아니요, Y, y, Yes, yes, 네, 예, YES] 이외의 다른 값을 입력했을 때 반응 확인---특수문자, 한자, 숫자, 영어 다른 값, 한글 다른 값

예상 결과: ['Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO'] 중 1개의 값을 입력하라는 오류 메시지 출력과 함께 다시 입력 받음.

실제 입력 및 실제 결과:

```
추가 작업을 진행하시겠습니까?(Y/N) : !
'Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO' 중 1개의 값을 입력해 주세요

추가 작업을 진행하시겠습니까?(Y/N) : 中
'Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO' 중 1개의 값을 입력해 주세요

추가 작업을 진행하시겠습니까?(Y/N) : 123
'Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO' 중 1개의 값을 입력해 주세요

추가 작업을 진행하시겠습니까?(Y/N) : oh yes
'Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO' 중 1개의 값을 입력해 주세요

추가 작업을 진행하시겠습니까?(Y/N) : 오 네
'Y', 'y', 'Yes', 'yes', '네', '예', 'YES', 'N', 'n', 'No', 'no', '아니오', '아니요', 'NO' 중 1개의 값을 입력해 주세요

추가 작업을 진행하시겠습니까?(Y/N) :
```

목적: [N, n, No, no, NO, 아니오, 아니요] 중 1 개를 입력했을 때 반응 확인 ----no 정상작동

예상 결과: 정상적으로 종료됨.

실제 입력 및 실제 결과:

```
추가 작업을 진행하시겠습니까?(Y/N) : N
프로그램을 종료하겠습니다.

Process finished with exit code 0
```

목적: [Y, y, Yes, yes, 네, 예, YES] 중 1개를 입력했을 때 반응 확인 ---yes 정상작동

예상 결과: 다시 새롭게 작업 폴더명을 입력 받으며 분류과정 재시작.

실제 입력 및 실제 결과: