



예외 처리와 파일 다루기

Python Programming

국민대학교 경영대학원 AI빅데이터전공
문현실 (hsmoon@kookmin.ac.kr)

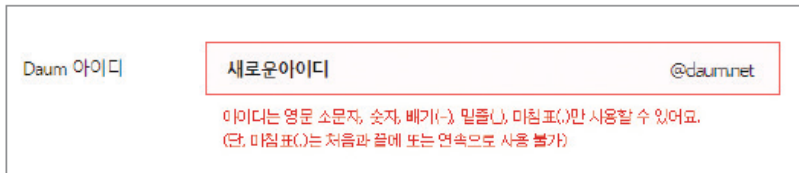




Ch.01 예외 처리

예외(exception)

- 프로그램을 개발하면서 예상하지 못한 상황이 발생
 - 사용자의 입력 오류
 - 갑자기 종료되었을 때를 대비한 자동 저장 기능
 - 참고 : 주피터 노트북에서 자동 저장을 끄고 싶으면 %autosave 0 입력

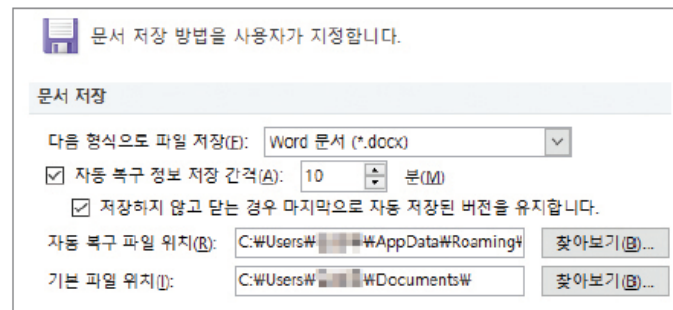


Daum 아이디

새로운아이디 @daumnet

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

(a) 아이디 생성 오류 입력



문서 저장 방법을 사용자가 지정합니다.

문서 저장

다음 형식으로 파일 저장(E): Word 문서 (*.docx)

☒ 자동 복구 정보 저장 간격(A): 10 분(M)

☒ 저장하지 않고 닫는 경우 마지막으로 자동 저장된 버전을 유지합니다.

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

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

(b) 자동 저장 기능

예측 가능한 예외 vs 예측 불가능한 예외

- 예측 가능한 예외
 - 발생 여부를 개발자가 사전에 인지할 수 있는 예외
 - 개발자는 예외를 예측하여 예외가 발생할 때는 어떻게 대응하라고 지정
 - 대표적으로 사용자 입력란에 값이 잘못 들어갔다면,
if문을 사용하여 사용자에게 잘못 입력하였다고 응답하는 방법
 - 매우 쉽게 대응 가능

예측 가능한 예외 vs 예측 불가능한 예외

- 예측 불가능한 예외
 - 대표적으로 매우 많은 파일을 처리할 때 문제가 발생
 - 예측 불가능한 예외가 발생했을 경우, 인터프리터가 자동으로 이것이 예외라고 사용자에게 알려줌
 - 대부분은 예외가 발생하면서 프로그램이 종료되므로 적절한 조치가 필요

예외의 종류와 예외 에러 메시지

- 대표적인 파이썬 내장 예외

예외	내용
IndexError	리스트의 인덱스 범위를 넘어갈 때
NameError	존재하지 않는 변수를 호출할 때
ZeroDivisionError	0으로 숫자를 나눌 때
ValueError	변환할 수 없는 문자나 숫자를 변환할 때
FileNotFoundError	존재하지 않는 파일을 호출할 때

예외 처리 구문

- try-except문

```
try:
```

```
    예외 발생 가능 코드
```

```
except 예외 타입:
```

```
    예외 발생 시 실행되는 코드
```

```
1 for i in range(10):  
2     try:  
3         print(10 / i)  
4     except ZeroDivisionError:  
5         print("Not divided by 0")
```


예외의 종류와 예외 에러 메시지

- 예외 에러 메시지의 사용
 - except 문 마지막에 as e 또는 as 변수명을 입력하여 해당 변수를 출력
 - 에러에 대한 이해를 높임

```
1 for i in range(10):  
2     try:  
3         print(10 / i)  
4     except ZeroDivisionError as e:  
5         print(e)  
6         print("Not divided by 0")
```

예외 처리 구문

- try-except-else문
 - if-else문과 비슷한데, 해당 예외가 발생하지 않을 경우 수행할 코드를 else문에 작성

```
try:  
    예외 발생 가능 코드  
except 예외 타입:  
    예외 발생 시 실행되는 코드  
else:  
    예외가 발생하지 않을 때 실행되는 코드
```

```
1 for i in range(10):  
2     try:  
3         result = 10 / i  
4     except ZeroDivisionError:  
5         print("Not divided by 0")  
6     else:  
7         print(10 / i)
```

예외 발생 구문

- raise문
 - 필요할 때 예외를 발생시키는 코드
 - if문과 함께 많이 사용

raise 예외 타입(예외 정보)

```
1 while True:
2     value =input("변환할 정수값을 입력해 주세요: ")
3     for digit in value:
4         if digit not in "0123456789":
5             raise ValueError("숫자값을 입력하지 않았습니다.")
6     print("정수값으로 변환된 숫자 -", int(value))
```

Wrap-up Exercise

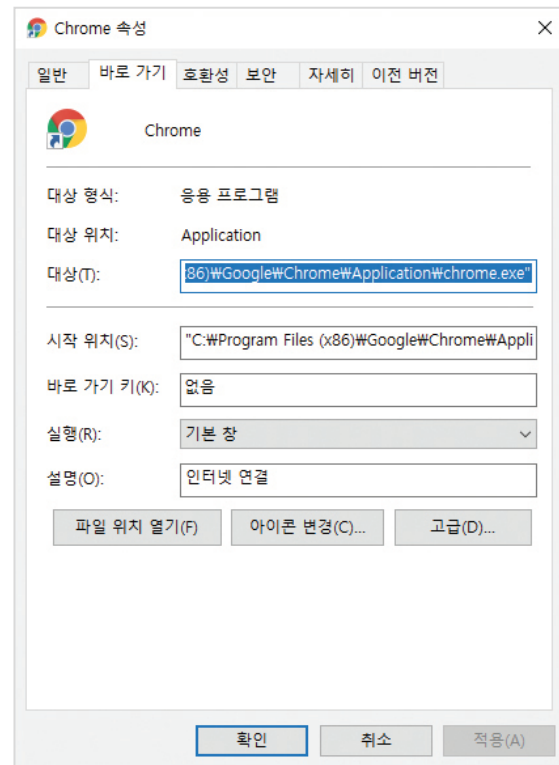
- 구구단 계산기 Ver02
 - 사용자의 입력이 제대로 될때까지!



Ch.02 파일 다루기

파일의 개념

- 컴퓨터를 실행할 때 가장 기본이 되는 단위
 - 컴퓨터에서 정보를 저장하는 가장 논리적인 단위
- GUI 환경에서는 아이콘을 더블 클릭하여 실행
 - 실제로는 아이콘과 연결된 파일을 실행



파일의 개념

- 파일과 디렉토리(폴더)
 - 디렉토리는 파일을 담는 또 하나의 파일로, 여러 파일을 포함할 수 있는 그릇
 - 직접 프로그램을 실행하지는 않지만, 다른 파일들을 구분하고 논리적인 단위로 파일을 분할
- 파일은 파일명과 확장자로 식별
 - 확장자는 파일의 쓰임을 구분하는 글자(예: .ipynb, .html)

파일의 종류

- 바이너리 파일과 텍스트 파일로 구분
 - 텍스트 파일도 사실 컴퓨터가 처리하기 위해 바이너리 형태로 저장
 - 시스템에서 사람이 보기 위해서는 인코딩(Encoding) 방식 사용

바이너리 파일	텍스트 파일
<ul style="list-style-type: none">• 컴퓨터만 이해할 수 있는 형태인 이진(법) 형식으로 저장된 파일• 일반적으로 메모장으로 열면 내용이 깨져 보임(메모장에 해석 불가)• 엑셀 파일, 워드 파일 등	<ul style="list-style-type: none">• 사람도 이해할 수 있는 형태인 문자열 형식으로 저장된 파일• 메모장으로 열면 내용 확인이 가능• 메모장에 저장된 파일, HTML 파일, 파이썬 코드 파일 등

문자열과 메모리 공간

- 숫자를 인식하는 최소 단위는 1Bit
- 문자를 인식하는 최소 단위는 1Byte = 8Bit
 - 컴퓨터는 문자도 숫자로 바꿔서 기억
- 인코딩(Encoding)
 - 컴퓨터가 문자를 처리하기 위해 이진수로 변환되는 표준 규칙
 - 띄어쓰기 개념도 없어 어디서 끊어야 하는지 모호해지는 문제
 - 이 문제를 해결하기 위해 초기 1Byte를 한글자로 인식하여 총 255문자 표현
 - 숫자가 너무 길어짐에 따라 문자를 쓸 때는 16진수 사용 (0x로 시작)

인코딩 예제

- UTF-8의 유니코드

000	(nul)	016	► (dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001	☺ (soh)	017	◄ (dcl)	033	!	049	1	065	A	081	Q	097	a	113	q
002	⦿ (stx)	018	‡ (dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	♥ (etx)	019	‡ (dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	♦ (eot)	020	⌘ (dc4)	036	\$	052	4	068	D	084	T	100	d	116	t
005	♣ (enq)	021	§ (nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	♠ (ack)	022	— (syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	• (bel)	023	‡ (etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	▣ (bs)	024	↑ (can)	040	(056	8	072	H	088	X	104	h	120	x
009	(tab)	025	↓ (em)	041)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	♂ (vt)	027	← (esc)	043	+	059	;	075	K	091	[107	k	123	{
012	♀ (np)	028	L (fs)	044	,	060	<	076	L	092	\	108	l	124	
013	(cr)	029	↔ (gs)	045	-	061	=	077	M	093]	109	m	125	}
014	♫ (so)	030	▲ (rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	✱ (si)	031	▼ (us)	047	/	063	?	079	O	095	_	111	o	127	o

각 나라의 인코딩 방법

- 영어
 - ASCII 인코딩 (첫 64개: 구두점 등의 문자 표시, 65번째부터 알파벳)
- ISO8859
 - 라틴어 등 알파벳 이외의 문자를 ASCII 빈칸에 할당 (서유럽 Latin-1)
- 한글
 - 128Byte로는 표현이 불가능(11,172개 글자)함에 따라 완성형으로 EUC-KR 사용 또는 윈도우 독자적인 CP949 사용(11,172자의 완성형 문자 표현 가능)
 - MBCS (Multi-Byte Character Set) : 국가간 호환이 되지 않는 방식
 - 한글 Encoding 에러시 적용 권장 순서
 - utf-8 -> utf-8-sig -> euc-kr -> mbcs -> cp949
- Unicode : 국제 표준 인코딩 방식 (UTF-8/UTF-16)

파이썬에서 파일 읽기

- open() 함수 사용
 - 파일명의 경로를 입력할 때는 / 기호 사용
 - 절대 경로 vs 상대 경로
 - 상대 경로의 사용을 권장
- close() 함수를 사용해 사용을 완료

```
f = open("파일명", "파일 열기 모드")  
f.close()
```

파이썬에서 파일 읽기

- 파일열기 모드

- r : 파일을 읽기만 할 때 사용(읽기 모드)
- rb : binary 형태로 읽기만 할 때 사용
- w : 파일에 내용을 쓸 때 사용(쓰기 모드)
- wb : 파일에 binary 형태로 내용을 쓸 때 사용
- a : 파일의 마지막에 새로운 내용을 추가할 때 사용 (추가모드)
- ab : 파일의 마지막에 binary 형태로 내용을 추가할 때 사용
- r+ 또는 w+ : 읽으면서 쓰기까지 함께 할 때 사용(읽기+쓰기 모드)
- a+ : 파일을 읽으면서 마지막에 새로운 내용을 추가할 때 사용(읽기+추가모드)

파이썬에서 파일 읽기

- with문과 함께 사용하기
 - 들여쓰기를 사용해 들여쓰기가 있는 코드에서는 open()함수가 유지
 - as문을 사용하여 변수에 할당

```
1 with open("dream.txt","r") as my_file:  
2     contents = my_file.read()  
3     print(type(contents), contents)
```

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

파이썬에서 파일 읽기

- 한 줄씩 읽어 리스트형으로 반환
 - readlines() 메서드 사용
 - 한줄의 기준은 \n 으로 구분

```
1 with open("dream.txt","r") as my_file:
2     content_list = my_file.readlines()           # 파일 전체를 리스트로 반환
3     print(type(content_list))                   # 자료형 확인
4     print(content_list)                         # 리스트값 출력
```

파이썬에서 파일 읽기

- 실행할 때마다 한 줄씩 읽어오기
 - readline() 메서드 사용

```
1 with open("dream.txt", "r") as my_file:
2     i = 0
3     while 1:
4         line = my_file.readline()
5         if not line:
6             break
7         print(str(i)+" === "+line.replace("\n",""))
8         i = i + 1
```


Wrap-up Exercise

- 파일 안 텍스트의 통계 정보를 함께 읽기



A terminal window with a light green title bar and three dots in the top right corner. The window contains three lines of text in a dark blue font:

```
총 글자의 수: 188  
총 단어의 수: 35  
총 줄의 수: 7
```

파이썬에서 파일 쓰기

- 인코딩(Encoding) 방식의 지정이 필요

```
1 f = open("count_log.txt", 'w', encoding = "utf8")
2 for i in range(1,11):
3     data = "%d번째 줄이다.\n"% i
4     f.write(data)
5 f.close()
```

파이썬에서 파일 쓰기

- 파일 추가 모드로 새로운 글 추가하기

```
1 with open("count_log.txt", 'a', encoding = "utf8") as f:  
2     for i in range(1, 11):  
3         data = "%d번째 줄이다.\n"% i  
4         f.write(data)
```

파이썬에서 파일 쓰기

- 디렉토리 생성

- 파이썬 내에서 os모듈을 사용하여 폴더 구조도 함께 다룰 수 있음

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

- 동일한 이름의 디렉토리 생성은 오류가 남에 따라 존재 여부를 확인

```
1 import os
2 os.mkdir("log")
3
4 if not os.path.isdir("log"):
5     os.mkdir("log")
```



Q&A

Thank you for your listening

