



Python Basic Review III: 모듈과 파일다루기

시빅데이터프로그래밍

Prof.Hyunsil Moon (hsmoon@kookmin.ac.kr)

1.

모듈

모듈? 라이브러리?

- 프로그램에서 독립적인 단위
- import문을 이용해 불러와서 사용할 수 있음

```
import random
```

- 모듈 호출시 이름 지정 가능

```
import pandas as pd
```

1 홀로코스터

4 시간 전

라이브러리하고 프레임워크가 뭐임? 이거 참고해가면서 일하면 얼마나 편함?

👍 0 🗨️ ...

5 Ausfaller

4 시간 전

@홀로코스터 라이브러리만 봐도 좀 긴 함수를 짜놓고 그걸 설정해놓은 단순한 한줄짜리 명령어로 불러올 수 있는데 그런게 여러개인데다 자주 쓰이면 효율성이 장난 아니지

👍 0 🗨️ ...

1 톤스

4 시간 전

@홀로코스터 라이브러리 안썼을때 : 좌우 고관절을 33도로 틀면서 종아리 근육을 20퍼센트의 힘으로 수축시키면서 달팽이관의 기울기에 비례하여 상체의 척추와 등근육을 팽창시키고 안구의 간상세포로 기울기를 보정하며 다리사이로 충분한 높이의 사물을 확인한뒤 뒤통치뼈에 체중을 실어 넘어짐을 방지하면서 무릎을 발목보다 수직 선상앞으로 이동시킨후 체중의 70퍼센트가 뒤로 쏠렸을때 대퇴근을 이완시키며 어찌고저찌고

라이브러리썼을때 : 앓아



2 import의 사용법

- 불러온 모듈의 함수를 모듈.함수 형태로 쓸 수 있음
 - 모듈의 함수 사용은 메서드와 사용법이 비슷

```
random.randint(1,6)
```

2 import의 사용법

- random 모듈 전체를 импорт

```
import random
```

- random 모듈에서 randint 함수만 импорт
 - random.randint가 아닌 randint 형태로 호출

```
from random import randint
```

- from ... import * 형태로 모듈 내 모든 함수를 импорт
 - 여러 모듈을 импорт하면 어떤 함수를 импорт했는지 알 수 없는 문제 발생

```
from random import *
```

3 패키지

- 여러 개의 모듈들로 구성된 덩어리
- pip와 conda를 이용해 설치
 - pip : Python 표준 패키지 설치 프로그램
 - conda : 아나콘다 패키지 설치 프로그램
- conda -> pip 순으로 시도
 - 일반적으로 conda의 버전이 낮은 편

4 TurtleGraphics

- 컴퓨터 프로그래밍의 원리를 가르치기 위한 교육 방법
- 거북이를 움직여서 다양한 그림을 그려보자
 - 패키지 로딩 : `import turtle`
 - 색 변경 : `color('색이름')`
 - 앞으로 이동 : `forward(이동거리)`
 - 왼쪽으로 회전 : `left(각도)`
 - 그리기 종료 : `done()`
 - 거북이 모양으로 바꾸기 : `shape('turtle')`

2.

나만의 함수

1 함수의 개념과 장점

- 여러 명이 프로그램을 함께 개발한다면?
 - 다같이 모여 토론하며 한 줄 한 줄 작성하기
 - 가장 잘하는 사람이 혼자 작성하기
 - 필요한 부분을 나누어 작성한 후 합치기
- 함수, 객체, 모듈

1 함수의 개념과 장점

- 함수(Function)
 - 어떤 일을 수행하는 코드의 덩어리, 또는 코드의 묶음
- 함수, 객체, 모듈
- 장점
 - 필요할 때마다 호출 가능 : 반복적으로 수행해야 하는 업무를 단 한번만 작성하여 필요할 때마다 호출하여 사용
 - 논리적 단위로 분할 가능
 - 코드의 캡슐화 : 인터페이스가 잘 정의된다면 다른 사람이 쉽게 가져가다가 사용할 수 있는 특징

2 함수의 선언과 사용

- 함수의 선언
 - 인자(argument) ≡ 매개변수(parameter)

```
def 함수 이름 (인자1, 인자2, ...):  
    수행문 1  
    수행문 2  
    ...  
    return <반환값>
```

매개변수 유무 반환값 유무	매개변수 없음	매개변수 있음
	매개변수 없음	매개변수 있음
반환값 없음	함수 안 수행문만 수행	매개변수를 사용하여 수행문만 수행
반환값 있음	매개변수 없이 수행문을 수행한 후, 결과값 반환	매개변수를 사용하여 수행문을 수행한 후, 결과값 반환

2 함수의 선언과 사용

- 변수의 사용범위
 - 지역 변수(Local variable) : 함수 안에서만 사용
 - 전역변수(Global variable) : 프로그램 전체에서 사용
 - 함수 내에서 전역 변수를 사용하기 위해서는 global 사용

함수의 인자

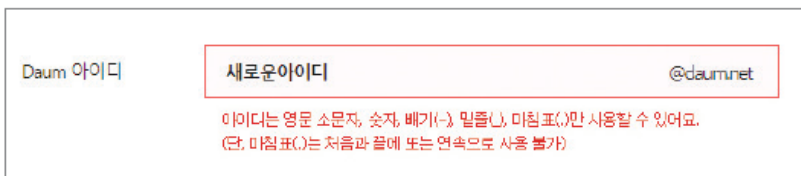
- 키워드 인자
 - 함수의 인터페이스에 지정된 변수명을 사용하여 함수의 인자를 지정
- 디폴트 인자
 - 별도의 인자값이 입력되지 않을 때, 인터페이스 선언에서 지정한 초깃값을 사용
- 가변 인자
 - 함수의 인터페이스에 지정된 변수 이외의 추가 변수를 함수에 입력할 수 있게 지원
- 키워드 가변 인자
 - 매개변수의 이름을 따로 지정하지 않고 입력하는 방법

3.

예외처리 및 파일다루기

1 예외(exception)

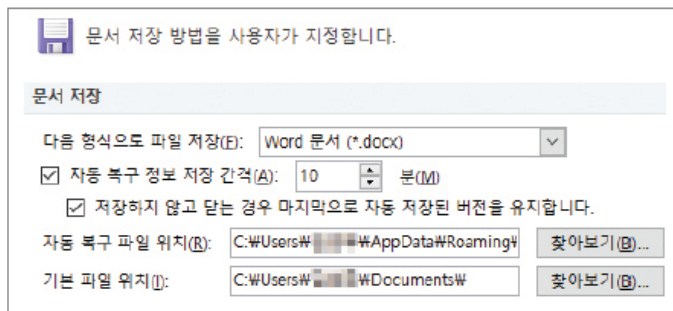
- 프로그램을 개발하면서 예상하지 못한 상황이 발생
 - 사용자의 입력 오류
 - 갑자기 종료되었을 때를 대비한 자동 저장 기능



Daum 아이디 새로운아이디 @daum.net

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

(a) 아이디 생성 오류 입력



문서 저장 방법을 사용자가 지정합니다.

문서 저장

다음 형식으로 파일 저장(E): Word 문서 (*.docx)

☒ 자동 복구 정보 저장 간격(A): 10 분(M)

☒ 저장하지 않고 닫는 경우 마지막으로 자동 저장된 버전을 유지합니다.

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

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

(b) 자동 저장 기능

1 예외(exception)

- 예측 가능한 예외
 - 발생 여부를 개발자가 사전에 인지할 수 있는 예외
 - 개발자는 예외를 예측하여 예외가 발생할 때는 어떻게 대응하라고 지정
 - 매우 쉽게 대응 가능
- 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

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")
```

2 예외 처리 구문

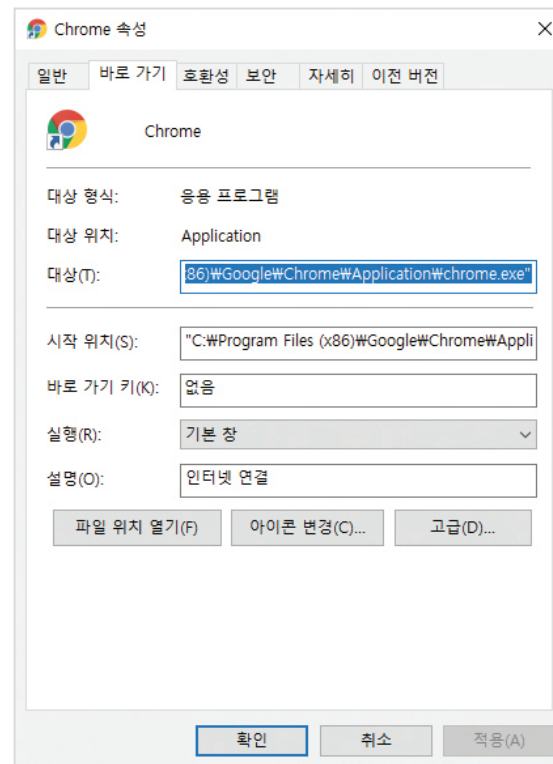
- try-except-else문
 - if-else문과 비슷한데, 해당 예외가 발생하지 않을 경우 수행할 코드를 else문에 작성

```
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)
```

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

- 바이너리 파일과 텍스트 파일
 - 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(메모장에서 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

- 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(Encoding)
 - 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

파일의 개념

- 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)

4 파이썬 기본 파일 다루기

- 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()
```


4 파이썬 기본 파일 다루기

- 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)

4 파이썬 기본 파일 다루기

- 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
```

4 파이썬 기본 파일 다루기

- 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
```

- 여러 변수와 객체는 순간적으로 메모리에 로딩되었다가 종료되면 사라짐
- 사용되는 객체의 저장 필요성 : 영속화(persistence)
- 파이썬에서는 pickle 모듈을 통해 로딩된 객체를 영속화
 - 객체의 속성 값까지 함께 저장함에 따라 향후 그대로 사용이 가능

```
>>> import pickle
>>>
>>> f = open("list.pickle", "wb")
>>> test = [1, 2, 3, 4, 5]
>>> pickle.dump(test, f)
>>> f.close()
```

- 불러오는 프로세스도 저장 프로세스와 동일

```
>>> f = open("list.pickle", "rb")
>>> test_pickle = pickle.load(f)
>>> print(test_pickle)
[1, 2, 3, 4, 5]
>>> f.close()
```



Q&A

Thank you for your listening