내장함수 (교재 111)

- 파이썬 내장함수

부성순 bakpak@empas.com

파이썬 내장함수 (1)

분류	함수	설명
자료형 변수 생성/변환	int()	문자열 또는 숫자를 정수로 변환합니다.
	float()	문자열 또는 숫자를 부동 소수점 수로 변환합니다.
	str()	주어진 객체를 문자열로 변환합니다.
	bool()	주어진 객체의 불린(논리적) 값을 반환합니다.
	list()	반복 가능한 객체로부터 리스트를 생성합니다.
	tuple()	반복 가능한 객체로부터 튜플을 생성합니다.
	set()	반복 가능한 객체로부터 세트(집합)를 생성합니다.
	dict()	키-값 쌍으로 딕셔너리를 생성합니다.
수치 데이터 연산	abs()	주어진 숫자의 절대값을 반환합니다.
	sum()	반복 가능한 객체 내 모든 요소의 합을 반환합니다.
	min()	반복 가능한 객체 내 최소값을 반환합니다.
	max()	반복 가능한 객체 내 최대값을 반환합니다.
	round()	숫자를 주어진 자릿수까지 반올림합니다.

파이썬 내장함수(2)

분류	함수	설명
문자/문자열 연산	len()	객체의 길이(요소의 개수)를 반환합니다.
	chr()	주어진 유니코드 코드 포인트에 해당하는 문자를 반환합니다.
	ord()	주어진 문자의 유니코드 코드 포인트를 반환합니다.
반복 가능 자료형 연산	range()	지정된 범위의 숫자 시퀀스를 생성합니다.
	enumerate()	반복 가능한 객체의 인덱스와 값을 쌍으로 반환합니다.
	zip()	여러 반복 가능한 객체들을 짝지어 튜플의 시퀀스로 반환합니다.
	filter()	함수와 반복 가능한 객체를 받아, 함수 조건에 맞는 요소만 걸러내어 반환합니다.
	map()	함수와 반복 가능한 객체를 받아, 각 요소에 함수를 적용한 결과를 반환합니다.
	sorted()	반복 가능한 객체를 정렬하여 리스트로 반환
객체	type()	객체의 타입을 반환합니다.
	id()	객체의 고유 식별자를 반환합니다.
	isinstance()	객체가 주어진 클래스의 인스턴스인지 여부를 반환합니다.

파이썬 내장함수(3)

분류	함수	설명
파일/디렉토리 관리	open()	파일을 열고 파일 객체를 반환합니다.
	os 모듈	파일 시스템을 다루는 여러 함수를 제공합니다 (os 모듈 자체는 내장 함수는 아니지만, 파일 및 디렉토리 관리를 위해 중요합니다).
실행 가능 문자열	eval()	문자열로 표현된 파이썬 식을 평가하고 결과를 반환합니다.
	exec()	문자열로 표현된 파이썬 코드를 실행합니다.
입력/출력	print()	주어진 객체를 표준 출력에 출력합니다.
	input()	사용자 입력을 문자열로 받습니다.
함수	help()	내장 도움말 시스템을 사용하여 함수나 모듈, 클래스 등에 대한 정보를 제공합니다.
	lambda()	이름이 없는 익명 함수 생성

파이썬 내장함수 – map()

➢ map( ): 함수와 반복 가능한 객체를 받아, 각 요소에 함수를 적용한 결과를 반환

기본구조 map(함수, 반복가능한 객체)

```
a=['1','2','3','4']
int_a=list(map(int, a))
print(int_a)
```

리스트(a)에 int함수를 각각 적용

```
[1, 2, 3, 4]
```

```
numbers = [1, 2, 3, 4, 5]
squared=list(map(lambda x:x*x, numbers))
print(squared)
```

```
[1, 4, 9, 16, 25]
```

리스트(numbers)에
익명함수(lambda)에 적용

```
def square(x):
    return x * x
numbers = [1, 2, 3, 4, 5]
squared = map(square, numbers)
print(list(squared))
```

리스트(numbers)에
square함수를 각각 적용

```
[1, 4, 9, 16, 25]
```

파이썬 내장함수 – filter()

➢ filter( ): 함수와 반복 가능한 객체를 받아, 함수 조건에 맞는 요소만 걸러내어 반환

(함수의 반환 값이 True 일 때만 해당 요소를 가져옴)

기본구조 filter(함수, 반복가능한 객체)

```
def func(x):
    return x>0 and x<5

a=[0,1,2,3,4,5,6,7,8,9,10]
result=list(filter(func,a))
print(result)
```

```
[1, 2, 3, 4]
```

- list(filter(func,a) → 리스트(a)에 func함수를 각각 적용해 결과가 True인 것만 해당요소 가져옴

```
a=[0,1,2,3,4,5,6,7,8,9,10]
result=list(filter(lambda x: x>0 and x<5,a))
print(result)
```

```
[1, 2, 3, 4]
```

- list(filter(lambda x:x>0 and x<5,a) → 리스트(a)에 lambda함수를 각각 적용해 결과가 True인 것만 해당요소 가져옴

부성순 bakpak@empas.com

파이썬 내장함수 – zip(), zip(*)

- zip( ): 함수는 여러 반복 가능한(iterable) 객체들의 요소를 튜플로 묶는 데 사용
- zip(*반복가능한 객체) : zip으로 결합된 객체나 이터레이터 앞에 *붙이면 분리 가능(언팩킹)

기본구조 **zip(반복가능한 개체 2개이상)**

```
x=[1,2,3]
y=['a','b','c']
z=[100,200,300]

zip_xyz=list(zip(x,y,z))
print('zip_xyz=',zip_xyz)
```

```
zip_xyz= [(1, 'a', 100), (2, 'b', 200), (3, 'c', 300)]
```

```
d1,d2,d3=zip(*zip_xyz)
print('d1=',d1)
print('d2=',d2)
print('d3=',d3)
```

```
d1= (1, 2, 3)
d2= ('a', 'b', 'c')
d3= (100, 200, 300)
```

list(zip(x, y, z)): x, y, z의 각 요소를 순서대로 묶어서 새로운 튜플 리스트 zip_xyz를 생성

d1, d2, d3 = zip(*zip_xyz)는 zip_xyz의 각 튜플을 언패킹하여 zip() 함수에 다시 전달한 후 원래의 세 리스트 로 분리되어, d1,d2,d3에 각각 할당 됨

파이썬 내장함수 – sorted() (1)

- sorted( ): 반복 가능한(iterable) 객체의 모든 요소를 정렬하여 새로운 리스트로 반환
  원본 데이터를 변경하지 않고, 정렬된 새로운 리스트를 생성

기본구조 sorted(반복가능한 개체, key=None, reverse=False)

```
L=[5,7,2,1,8]
print('sorted(L):',sorted(L))
print('sorted(L,reverse=True):',sorted(L,reverse=True))
```

← L 오름차순 정렬

← L 내림차순 정렬 (reverse=True)

```
sorted(L): [1, 2, 5, 7, 8]
sorted(L,reverse=True): [8, 7, 5, 2, 1]
```

```
TL=[('park',47,60.5),('lee',23,80.2),('kim',30,70.2)]
print('sorted(TL):',sorted(TL))
print('sorted(TL, key=lambda x:x[1]):',sorted(TL, key=lambda x:x[1]))
print('sorted(TL, key=lambda x:x[2]):',sorted(TL, key=lambda x:x[2]))
```

← TL의 첫번째 요소로 정렬

← TL의 두번째 요소로 정렬 (key=lambda x:x[1])

← TL의 두번째 요소로 정렬 (key=lambda x:x[2])

```
sorted(TL): [('kim', 30, 70.2), ('lee', 23, 80.2), ('park', 47, 60.5)]
sorted(TL, key=lambda x:x[1]): [('lee', 23, 80.2), ('kim', 30, 70.2), ('park', 47, 60.5)]
sorted(TL, key=lambda x:x[2]): [('park', 47, 60.5), ('kim', 30, 70.2), ('lee', 23, 80.2)]
```

파이썬 내장함수 – sorted() (2)

➢ sorted()함수를 사용하여 딕셔너리의 키(key)와 값(value)을 다양한 기준으로 정렬하는 하기

```
my_dict={
    'k1':7,
    'k3':4,
    'k2':5
}
print('sorted(my_dict):',sorted(my_dict))❶
print('my_dict.items():',my_dict.items())❷
print('sorted(my_dict.items()):',sorted(my_dict.items()))❸
print('sorted(my_dict.items(),key=lambda x:x[1]):',sorted(my_dict.items(),key=lambda x:x[1]))❹
```

```
sorted(my_dict): ['k1', 'k2', 'k3']
my_dict.items(): dict_items([('k1', 7), ('k3', 4), ('k2', 5)])
sorted(my_dict.items()): [('k1', 7), ('k2', 5), ('k3', 4)]
sorted(my_dict.items(),key=lambda x:x[1]): [('k3', 4), ('k2', 5), ('k1', 7)]
```

❶ my_dict의 key기준으로 오름차순 정렬
❷ my_dict.items()는 딕셔너리의 키-값 쌍을 튜플로 가지는 리스트로 반환
❸ sorted(my_dict.items())는 딕셔너리 아이템(키-값 쌍)을 키를 기준으로 오름차순으로 정렬
❹ sorted(dict.items(), key=lambda x: x[1])는 딕셔너리 값의 오름차순으로 정렬
lambda x: x[1]는 키-값 쌍 튜플에서 값을 기준으로 정렬 기준을 설정

람다함수(교재 127~128)

- 파이썬 람다함수

부성순 bakpak@empas.com

람다 함수(lambda function)

- lambda 함수: 이름이 없는 함수로, 한 줄짜리 코드로 작성
  일회성으로 사용되거나 다른 함수의 인자로 전달될 때 주로 사용 (ex: map(), filter())

기본구조 lambda 매개변수들: 식

```
double= lambda x: x*2
triple= lambda x: x*3

print('dobule(5):', double(5))
print('triple(5):', triple(5))
print('(lambda x:x*4)(5):',(lambda x:x*4)(5))
```

```
dobule(5): 10
triple(5): 15
(lambda x:x*4)(5): 20
```

```
files=['1.txt','2.jpg','4.docx','5.jpg']
jpg_files=list(filter(lambda x: '.jpg' in x,files))
print(jpg_files)
```

```
['2.jpg', '5.jpg']
```

람다 함수(lambda function)

GPT에 '파이썬 람다 함수 사용예 ' 로 다양한 예제를 생성하고 실행합니다.