

리스트, 딕셔너리



소프트웨어융합대학
교수 진혜진

목차

1. 리스트
2. 딕셔너리

- 리스트는 정수, 문자열, 실수 등 서로 다른 자료형을 하나로 묶을 수 있다.

File Edit Format Run Options Window Help	File Edit Shell Debug Options Window Help
<pre>list=[] score=[90,89,60,70] major=["소프트웨어", "정보공학", "정보융합"] print(list) print(score) print(major)</pre>	<pre>= RESTART: C:/Users/hjjin/AppData/ [] [90, 89, 60, 70] ['소프트웨어', '정보공학', '정보융합'] >>> </pre>

File Edit Format Run Options Window	>>>
<pre>list=["진콩이",10,3.1] print(list)</pre>	<pre>= RESTART: C:/User ['진콩이', 10, 3.1] >>> </pre>

■ 빈 리스트의 생성과 항목 추가

```
File Edit Format Run Optio >>>  
list=[]  
list.append(7)  
list.append(17)  
list.append(27)  
list.append(37)  
print(list)  
= RESTART: C:/U  
[7, 17, 27, 37]  
>>> |
```

```
list=[]  
for i in range(0,10):  
    list.append("★")  
print(len(list))  
print(list)  
= RESTART: C:/Users/hjjin/AppData/Local/Programs/Pyt  
10  
['★', '★', '★', '★', '★', '★', '★', '★', '★', '★']  
>>> |
```

■ 리스트 항목 접근하기

score=[90,89,60,70]	= RESTAP
print(score[0])	90
print(score[1])	89
print(score[2])	60
print(score[3])	70
print(score[-1])	70
print(score[-2])	60
print(score[-3])	89

■ 슬라이싱

- 리스트에서 한 번에 여러 개의 항목을 추출하는 기법
- 리스트에 접근할 때 콜론(:)을 사용해 범위를 지정

```
score=[90,89,60,70] = RESTART: (
print(score[0:2])    [90, 89]
print(score[2:4])    [60, 70]
>>> |
```

- 콜론의 앞이나 뒤 숫자 생략

```
File Edit Format Run Options Win
score=[90,89,60,70] = RESTART: C:/Us
print(score[2:])    [60, 70]
print(score[:4])    [90, 89, 60, 70]
>>> |
```

■ 리스트끼리 덧셈, 곱셈 연산

```
File Edit Format Run Options Window Help
a=[1,2,3]
b=[4,5,6]
print(a+b)
print(a*4)
name=["진혜진","진선미","진콩이"]
city=["서울","부산","대구"]
print(name+city)
print(name*2)
```

```
>>>
= RESTART: C:/Users/hjjin/AppData/Local/Programs/Python/Python38-32/
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
['진혜진', '진선미', '진콩이', '서울', '부산', '대구']
['진혜진', '진선미', '진콩이', '진혜진', '진선미', '진콩이']
>>>
```

■ 두 번째에 위치한 값을 변경하는 방법

```
a=[1,2,3]
b=[4,5,6]
name=["진혜진","진선미","진콩이"]
city=["서울","부산","대구"]
```

```
a[2]=30
name[1]="진달래"
```

```
print(a)
print(name)
```

```
>>>
>>>
= RESTART: C:/Users/hjjin/AppD
[1, 2, 30]
['진혜진', '진달래', '진콩이']
>>>
>>>
>>>
```

- 두 번째인 `a[1]`의 항목 삭제
- 두 번째인 `a[1]`에서 네 번째인 `a[3]`까지 삭제

```
a=[10,20,30,40,50] [10]
del(a[1])          >>>
print(a)           = RESTART: C:/Use
a[1:4]=[]          [10, 30, 40, 50]
print(a)           [10]
                   >>> |
```


■ 리스트 조작 함수

함수	설명	사용법
append()	리스트 맨 뒤에 항목을 추가한다.	리스트명.append(값)
pop()	리스트 맨 뒤의 항목을 빼낸다	리스트명.pop()
sort()	리스트의 항목을 정렬한다.	리스트명.sort()
reverse()	리스트 항목의 순서를 역순으로 만든다.	리스트명.reverse()
insert()	지정된 위치에 값을 삽입한다.	리스트명.insert(위치, 값)

```

a=[50,20,30,40,10]
a.append(7)
print(a)
a.pop()
print(a)
a.sort()
print(a)
a.reverse()
print(a)
a.insert(2,"진콩이")
print(a)
    
```

```

File Edit Shell Debug Options Window Help
>>>
= RESTART: C:/Users/hjjin/AppD
[50, 20, 30, 40, 10, 7]
[50, 20, 30, 40, 10]
[10, 20, 30, 40, 50]
[50, 40, 30, 20, 10]
[50, 40, '진콩이', 30, 20, 10]
    
```

■ 리스트 조작 함수

함수	설명	사용법
del()	리스트에서 해당 위치의 항목을 삭제한다.	del(리스트명[위치])
len()	리스트에 포함된 전체 항목의 개수를 센다.	len(리스트명)
copy()	리스트의 내용을 새로운 리스트에 복사한다.	새 리스트=리스트명.copy()
sorted()	리스트의 항목을 정렬해서 새로운 리스트에 대입한다.	새 리스트=sorted(리스트)

```

a=[50,20,30,40,10]
del(a[2])
print(a)
print(len(a))
new=a.copy()
print(new)
new=sorted(a)
print(new)
    
```

```

>>>
= RESTART: C:/User
[50, 20, 40, 10]
4
[50, 20, 40, 10]
[10, 20, 40, 50]
>>> |
    
```

■ 쌍 2개가 하나로 묶인 자료구조

예

- ‘kiwi:키위’처럼 의미 있는 두 값을 연결해 구성

- 다른 프로그래밍 언어에서는 해시(Hash), 연관 배열(Associative Array)이라 함
- **중괄호 {}**로 묶어 구성, **키(Key)**와 **값(Value)**의 **쌍**으로 구성

딕셔너리변수 = {키1:값1, 키2:값2, 키3:값3, ...}

■ 딕셔너리 생성

```
File Edit Format Run Options Window Help
info={"진혜진":20,"진콩이":10}
print(info)

= RESTART: C:/Users/hjjin/AppData/Local/Programs/Python/Python38-64/Python.exe
{'진혜진': 20, '진콩이': 10}
>>> |
```

■ 여러 정보의 딕셔너리 표현

키	값
학번	201901
이름	진혜진
학과	소프트웨어

```
File Edit Format Run Options Window Help
stu={"학번":201901,"이름":"진혜진","학과":"소프트웨어"}
print(stu)

= RESTART: C:/Users/hjjin/AppData/Local/Programs/Python/Python38-64/Python.exe
{'학번': 201901, '이름': '진혜진', '학과': '소프트웨어'}
>>> |
```

■ Stu 딕셔너리 연락처 추가

```
stu={"학번":201901,"이름":"진혜진","학과":"소프트웨어"}  
stu["연락처"]="010-000-1234"  
print(stu)
```

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

>>>

= RESTART: C:/Users/hjjin/AppData/Local/Programs/Python/Python37-32/list.py
{'학번': 201901, '이름': '진혜진', '학과': '소프트웨어', '연락처': '010-000-1234'}

■ 학과 수정

```
stu={"학번":201901,"이름":"진혜진","학과":"소프트웨어"}  
stu["연락처"]="010-000-1234"  
print(stu)  
stu["학과"]="컴퓨터학과"  
print(stu)
```

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

= RESTART: C:/Users/hjjin/AppData/Local/Programs/Python/Python37-32/list.py =
{'학번': 201901, '이름': '진혜진', '학과': '소프트웨어', '연락처': '010-000-1234'}
{'학번': 201901, '이름': '진혜진', '학과': '컴퓨터학과', '연락처': '010-000-1234'}

■ 학과 삭제

```
stu={"학번":201901,"이름":"진혜진","학과":"소프트웨어"}  
print(stu)  
del(stu["학과"])  
print(stu)
```

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

```
= RESTART: C:/Users/njjin/AppData/Local/Programs/Py  
{'학번': 201901, '이름': '진혜진', '학과': '소프트웨어'}  
{'학번': 201901, '이름': '진혜진'}
```

■ 키로 값에 접근하는 코드

```
stu={"학번":201901,"이름":"진혜진","학과":"소프트웨어"}  
print(stu["학번"])  
print(stu["이름"])
```

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

```
= RESTART: C:/Users/njjin/AppData/Local/Programs/Py  
201901  
진혜진
```

- `딕셔너리명.get(키)` 함수를 사용해 키로 값에 접근
- `딕셔너리명.keys()`는 **딕셔너리의 모든 키 반환**
- 출력 결과의 `dict_keys`가 보기 싫으면 `list(딕셔너리명.keys())` 함수 사용
- `딕셔너리명.values()` 함수는 딕셔너리의 모든 값을 리스트로 만들어 반환

```
stu={"학번":201901,"이름":"진혜진","학과":"소프트웨어"}  
print(stu.get("이름"))  
print(stu.keys())  
print(list(stu.keys()))  
print(stu.values())
```

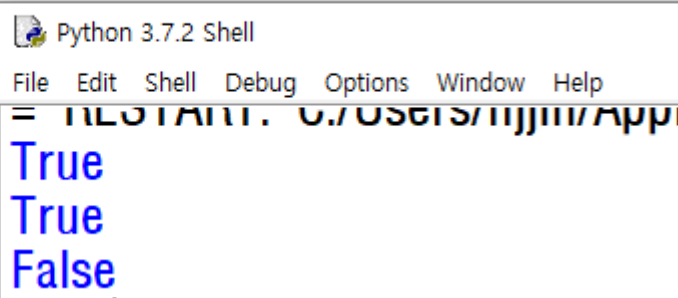
Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

```
= RESTART: C:/Users/hjjin/AppData/Local/Programs/Python/Python37-64/Python.exe  
진혜진  
dict_keys(['학번', '이름', '학과'])  
['학번', '이름', '학과']  
dict_values([201901, '진혜진', '소프트웨어'])
```

- 딕셔너리 안에 해당 키가 있는지 없는지는 **in**을 사용해 확인
- 딕셔너리에 키가 있다면 True를 반환하고, 없다면 False를 반환

```
stu={"학번":201901,"이름":"진혜진","학과":"소프트웨어"}  
print("이름" in stu)  
print("학과" in stu)  
print("학점" in stu)
```



Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

True
True
False