Lesson 7

Collection 자료형
리스트(List)

목차

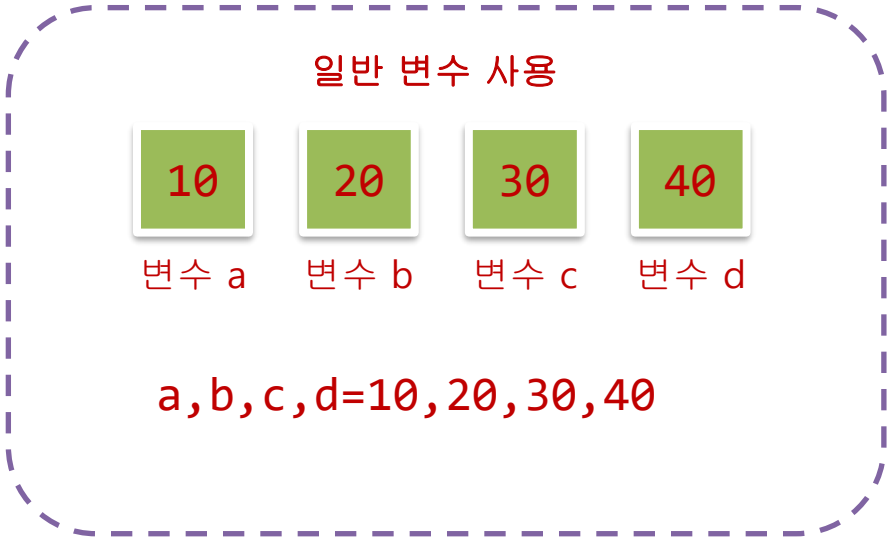
Collection 자료형

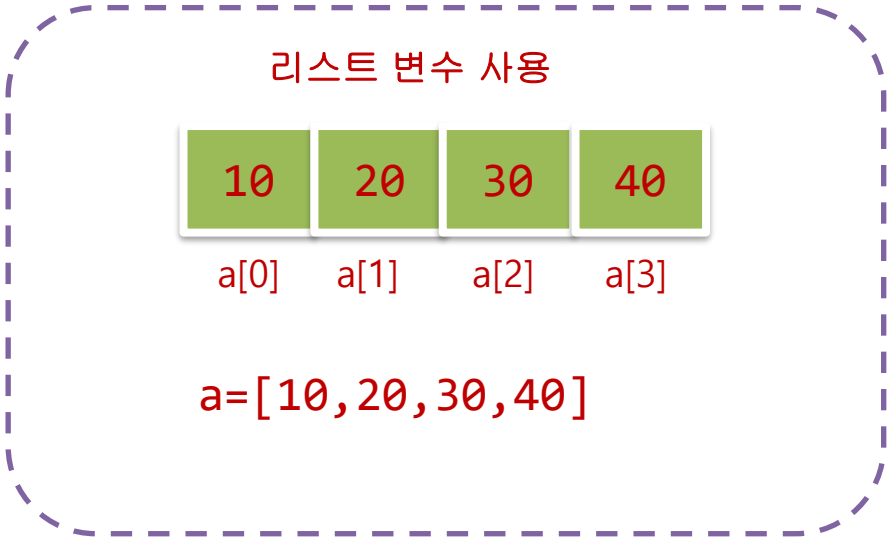
Collection

Python

- 여러 값을 하나의 이름으로 묶어서 관리하는 자료형
- 리스트(list), 튜플(tuple), 세트(set), 딕셔너리(dict)가 있다.

지금까지 변수 한 개에 한 개의 값만 저장했다.
만약 10명의 학생이 있고 이들의 중간 고사 성적을
저장해서 처리하려면 변수 10개가 필요...

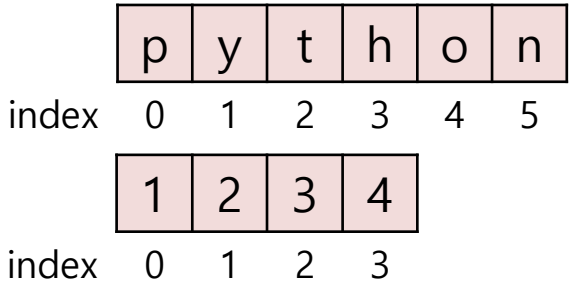




시퀀스(sequence)형과 비시퀀스형(nonsequence)

Python

- 시퀀스(sequence)형
  - 문자열: ‘python’
  - 리스트: [1,2,3,4]
  - 튜플: (1,2,3,4)



시퀀스형: 순서가 있는 자료형
비시퀀스형: 순서가 없는 자료형

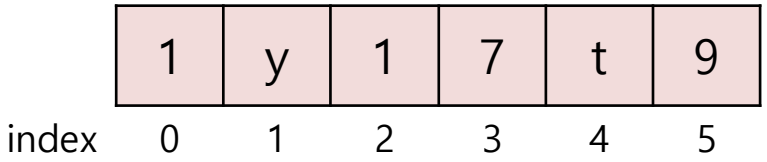
- 비시퀀스(nonsequence)형
  - 셋: {1,2,3}
  - 딕셔너리: {'a':'apple','b':'banana','c':'cherry'}

리스트(list)

리스트(list)

Python

- 순서가 있는 자료를 관리해야 할 때 편리하다.
- 저장할 값의 자료형이 달라도 저장할 수 있다.
- **순서O, 중복O, 수정O, 삭제O**



리스트의 생성

Python

- 리스트 생성
  - **대괄호 [ ] 또는 list( )함수로 생성**
  - 변수=[값1, 값2,...]
  - 변수=list(반복가능한객체)

코드

```
li=[]   # li=list()와 동일
li=[1, 3.14, "python", [5,6,7]]
print(li)
```

```
[1, 3.14, "python",[5,6,7]]
```

리스트 연산(+, *)

Python

- [리스트]+[리스트] => 연결
- [리스트]*숫자 => 리스트 반복
- in, not in => 포함 여부 확인

코드

```
li1=[1,2]
li2=[3,4,5,6]
print(li1+li2)
print(li1*3)
print([0]*5)
print(3 in li2)
```

```
[1, 2, 3, 4, 5, 6]
[1, 2, 1, 2, 1, 2]
[0,0,0,0,0]
True
```

리스트의 인덱싱, 슬라이싱

Python

- 리스트는 순서가 있으므로 인덱싱과 슬라이싱이 가능함.
- 문자열의 인덱싱, 슬라이싱과 동일함
  - 인덱싱 : 변수명[인덱스]
  - 슬라이싱 : 변수명[시작 인덱스 : 종료 인덱스 : 증감값]

```
li=[1, 3.14, 'hi~', 10, 20, 30]
```

1	3.14	hi~	10	20	30
li[0]	li[1]	li[2]	li[3]	li[4]	li[5]
li[-6]	li[-5]	li[-4]	li[-3]	li[-2]	li[-1]

리스트의 인덱싱, 슬라이싱

Python

코드

```
li=[1, 3.14, 'hi~', 10, 20, 30]

print(li[0])
print(li[-1])
print(li[0:3])
print(li[:2])
print(li[2:])
print(li[:])
```

```
1
30
[1, 3.14, 'hi~’]
[1, 3.14]
['hi~', 10, 20, 30]
[1, 3.14, 'hi~', 10, 20, 30]
```

중첩 리스트 인덱싱, 슬라이싱

Python

코드

```
li=[[1,2,3],
    [4,5,6],
    [7,8,9]
   ]
print(li[0][0])    # 1
print(li[1][0])    # 4
print(li[2][0])    # 7
print(li[0][1:])   # [2,3]
```

리스트 요소 추가와 삭제

Python

- 요소 추가
  - **append()**
  - insert(인덱스, 값)

✓ append()는 요소를 마지막에 추가.
✓ insert()는 요소를 지정한 인덱스 위치에 추가.

- 요소 삭제
  - **pop()**
  - **remove(삭제할 값)**

✓ pop() : 리스트 마지막 요소를 제거.
✓ pop(인덱스) : 지정한 인덱스 위치의 요소를 제거.
✓ remove(값) : 지정한 요소를 제거

리스트 추가

Python

코드

```
li=[1,2,3]
li.append(4)
print(li)       # [1,2,3,4]
li.insert(0,5)
print(li)       # [5,1,2,3,4]
```

```
print(li1.append(4))    # None
```

리턴값이 None인 함수는 원래 데이터를 제자리(in-place)에서 수정한다. 즉, **원본 데이터 자체를 수정한다.**

리스트 요소 추가

li.append(4)
li+=[4]

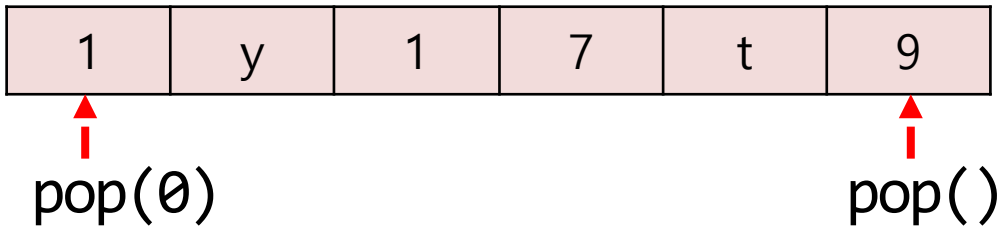
리스트 삭제

Python

코드

```
li=[1,2,3,4,5]
li.pop()
print(li)      # [1,2,3,4]
li.pop(0)
print(li)      # [2,3,4]
li.remove(2)
print(li)      # [3,4]
```

pop()은 인덱스를 지정
remove()는 삭제할 요소를 지정



pop()과 append()로 스택 자료구조 구현

Python

코드

```
dishes = ['접시1', '접시2', '접시3']
print(dishes)
dishes.append('접시4')
dishes.append('접시5')
print(dishes)
dishes.pop()
print(dishes)
dishes.pop()
print(dishes)
```

['접시1', '접시2', '접시3']

['접시1', '접시2', '접시3', '접시4', '접시5']

['접시1', '접시2', '접시3', '접시4']

['접시1', '접시2', '접시3']

스택(Stack)

LIFO(Last In First Out)
마지막 들어간 데이터부터 출력되는 자료 구조.

(사용예)
브라우저에서 이전페이지, 다음페이지 이동
에디터에서 실행취소, 다시실행
후위 표기법 수식의 연산 등...

리스트 수정

Python

코드

```
li=[1,2,3]
li[0]=100
print(li)      # [100,2,3]
```

리스트 조작 메소드

Python

메소드 및 함수	기능	사용법
append()	리스트 맨 뒤에 요소를 추가함.	리스트명.append(값)
insert()	지정된 위치에 값를 삽입함.	리스트명.insert(위치, 값)
pop()	리스트 맨 뒤의 요소를 제거함.	리스트명.pop()
remove()	지정한 값을 삭제. 여러 개인 경우 첫 번째 값만 삭제.	리스트명.remove(삭제할 값)
index()	찾을 값의 위치를 반환함.	리스트명.index(찾을 값)
count()	찾는 값의 개수를 반환함.	리스트명.count(찾을 값)
sort()	리스트의 요소를 오름차순 정렬함. 내림차순: reverse=True	리스트명.sort()
reverse()	리스트의 요소를 뒤집어 표시함.(sort() 의미가 아님)	리스트명.reverse()
extend()	리스트 뒤에 리스트를 추가함. + 연산과 기능이 동일함.	리스트명.extend(추가할 리스트)
clear()	리스트의 모든 내용을 삭제함.	리스트명.clear()
copy()	리스트의 내용을 새로운 리스트에 복사함.	새리스트명=리스트명.copy()

리스트 함수(sort, reverse)

Python

```
sort()     : 정렬
reverse()  : 요소를 반대로 출력
```

코드

```
li1=[4,1,3,5,2]
li2=[4,1,3,5,2]
li1.sort()                  # 오름차순으로 정렬
li2.sort(reverse=True)      # 내림차순으로 정렬
print(li1)              # [1, 2, 3, 4, 5]
print(li2)              # [5, 4, 3, 2, 1]

li=[4,1,3,5,2]
li.reverse()            # 요소를 반대로 출력
print(li)               # [2, 5, 3, 1, 4]
```

리스트 함수(index, count)

Python

```
index()    : 찾는 요소의 위치
count()    : 찾는 요소의 개수
```

코드

```
li=[4,1,3,5,2]
print(li.index(3))          # 요소 3의 위치 출력=>2

li=[1,2,3,1]
print(li.count(1))          # 요소 1의 개수. 2
```

리스트 함수(extend)

Python

```
extend()      : 기존 리스트에 신규 리스트를 추가해서 확장
```

코드

```
a=[1,2,3]
a.extend([4,5]) # 리스트 확장(연장)
print(a)        # [1, 2, 3, 4, 5]
```

리스트+리스트와 extend()의 차이점

extend(): 원본 데이터 자체를 수정.

+ : 새로운 리스트를 반환.

문자열->리스트, 리스트->문자열

Python

코드

```
li=['a','c','e]
print(''.join(li))    # ace

word='hello'
word=list(word)
print(word)          # ['h', 'e', 'l', 'l', 'o']
```

입력받은 이름의 데이터를 삭제하고 이름을 오름차순 정렬해서 출력

Python

코드

```
names=['우수한','이민주','송보라','김도현','최도영']
del_name=input('삭제할 학생이름 입력: ')
names.remove(del_name)
names.sort()
print(names)
```

```
삭제할 학생이름 입력: 송보라
['김도현', '우수한', '이민주', '최도영']
```

문자열을 공백으로 구분하고 리스트로 만들기

Python

코드

```
animals='원숭이 닭 개 돼지 쥐 소 호랑이 토끼 용 뱀 말 양'
animals=animals.split()
print(animals)
print(animals[0])
```

```
['원숭이', '닭', '개', '돼지', '쥐', '소', '호랑이', '토끼', '용', '뱀', '말', '양']
```

```
원숭이
```

도전!

[문제] 아래 성적 데이터로 총점, 평균, 최고점수, 최저점수를 출력하기

scores=[90,90,85,50,95,70,30]

단, 평균은 소수 아래 1자리까지 표시

```
총점: 510
평균: 72.9
최고점수: 95
최저점수: 30
```

합계 : **sum()**
평균 : 합계/개수 　 리스트의 개수: **len()**사용. 반올림: **round()**
최대 : **max()**
최소 : **min()**

```
scores = [90, 90, 85, 50, 95, 70, 30]

print("총점: ", sum(scores))
print("평균: ", round(sum(scores) / len(scores), 2))
print("최고점수: ", max(scores))
print("최저점수: ", min(scores))
```