KG 아이티뱅크

파이썬

PYTHON

리스트

리스트(List)

❖ **연관된 여러 개의 값들을 하나의 변수에 넣기 위한 자료형**

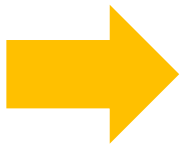
- 일종의 변수의 묶음이며, 이를 하나로 묶어 사용하는 것
- 자료형이며 동시에 값의 의미도 가지고 있음

❖ **하나의 이름으로 묶어 반복되는 규칙을 이용해 활용하는 것**

- 변수는 별개의 값에 이름을 붙여 쓰기 때문에 연결되지 않음
- 리스트보다는 내부에 있는 값의 확인과 사용이 중요함

❖ 코드

```
lst1= [1, 2, 3, 'A']          #생성
print( lst1 )                 #확인
print( lst1[0] )              #사용
lst1[1]=6                     #사용
print( lst1 )                 #확인
```



❖ 결과

```
[1, 2, 3, 'A']
1
[1, 6, 3, 'A']
```

리스트(List)

❖ **리스트 자료형은 아래와 같은 특징을 가지고 있음**

1) 연산이 가능함 : 리스트끼리 더하거나 정수와 곱하기 가능
   - ➢ **더하면 합쳐지며 새로운 리스트를 생성**
   - ➢ **곱하면 반복되며 새로운 리스트를 생성**
2) 인덱싱이 가능함 : 값에 대한 인덱스번호가 부여됨
   - ➢ 이를 이용한 **값의 수정, 사용이 가능**함
3) 슬라이싱이 가능함 : 인덱스 번호로 슬라이싱이 가능
   - ➢ 이를 이용한 **특정 구간의 수정, 사용이 가능**함
4) 메서드를 보유함 : **원본을 갱신**하는 방식으로 작동

❖ 코드

```
lst1= [1,2]
lst2= [3,4]
print( lst1 + lst2 )
print( lst1 * 2 )
print( lst1[0:1] )
```

❖ 결과

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

리스트(List)

< 파일이름 : listEX1.py >

❖ 실습예제1. 아래의 값들을 이용해 리스트를 생성하고 출력하고, 사용해보세요.

아래의 값을 사용

```
10, 20, 30, “A”, “ABC”, 3.1466
```

결과

```
- 리스트 출력 -
[10, 20, 30, ‘A’, ‘ABC’, 3.1466]
- 리스트 사용 -
0번 인덱스 : 10
1번 인덱스 : 20
4번 인덱스 : ABC
5번 인덱스 : 3.15
```

리스트(List)

< 파일이름 : listEX2.py >

❖ **실습예제2. 입력을 받은 임의의 정수 3개로 리스트를 생성하고, 출력하고, 사용해보세요.**

조건

입력을 받는 방법은 직접 구성합니다.
입력을 받은 정수의 총합과 평균을 구합니다.

결과

```
- 리스트 출력 -
[(값1), (값2), (값3)]
- 리스트에 저장된 값의 연산결과 -
값들의 총합 : (결과1)
값들의 평균 : (결과2)
```

< 파일이름 : listEX3.py >

❖ 실습예제3. 조건을 만족하는 코드를 구성하세요.

조건

입력을 받는 방법은 직접 구성합니다.
단, 값의 자료형은 정수값으로 통일합니다.

1. 값을 2개 입력을 받아 리스트에 저장합니다.
2. 저장된 값으로 합을 구하고 리스트에 추가합니다.
3. 리스트에 저장된 값들의 평균을 구하고 리스트에 추가합니다.

결과

- 최초 상태 : [(값1),(값2)]
- 합 추가 : [(값1),(값2),(합)]
- 평균 추가 : [(값1),(값2),(합),(평균)]

< 파일이름 : listEX4.py >

❖ **실습예제4. 아래의 리스트를 이용해 조건을 해결하세요.**

아래의 값을 사용

```
[1, 2, 3], ['A', 'B', 'C']
```

조건

슬라이싱을 이용해 결과처럼 출력합니다.

결과

```
[1, 2, 'A', 'B', 'C', 3]
[1, 2, 3, 'A', 'B', 'C']
['A', 'B', 'C', 1, 2, 3]
['A', 'B', 1, 2, 3, 'C']
```

리스트(List)

< 파일이름 : listEX5.py >

❖ **실습예제5. 아래의 리스트로 조건을 달성하세요.**

아래의 값을 사용

```
['PEN', 'APPLE', 'PINEAPPLE']
```

조건

리스트에서 APPLE을 삭제합니다.
삭제한 리스트의 중간에 WOW를 넣습니다.

결과

```
['PEN', 'APPLE', 'PINEAPPLE']
['PEN', 'PINEAPPLE']
['PEN', 'WOW', 'PINEAPPLE']
```

리스트(List)

❖ 리스트에서 사용할 수 있는 메서드(내장함수)

- 일부를 제외하고 원본을 갱신하는 방식으로 작동
- 연산, 인덱싱, 슬라이싱으로 충분히 구현이 가능한 기능
  - ➢ 좀 더 쉽게 알려진 연산을 사용할 수 있음
  - ➢ 연산을 단어로 바꿔 가독성을 늘리는 것이 목적

❖ 목록

1) **append** / insert / extend
2) **pop** / remove / clear
3) **count**
4) **index**
5) **sort** / reverse

리스트(List)

< 파일이름 : listEX6.py >

❖ 실습예제6. 아래의 값을 이용해 조건을 해결하세요.

아래의 값을 사용

[] # 비어 있는 리스트

조건

1. 메서드를 이용해 정수를 3번 입력을 받아 추가합니다.
2. 입력된 값 중에 정수 5가 있는지 여부를 출력합니다.
3. 입력된 값 중에 정수 7이 있는지 확인하고
   있으면 새로운 정수를 입력을 받아 7이 있는 위치에
   다시 저장합니다.

결과

- 최초 리스트 : [(값1), (값2), (값3)]
- 5가 있는지 여부 : 있다 / 없다

7이 있을 경우에만 출력합니다.

- 값이 바뀐 리스트 : [(값1),(값2),(값4)]

리스트(List)

< 파일이름 : listEX7.py >

❖ **실습예제7. 아래의 값을 이용해 조건을 해결하세요.**

아래의 값을 사용

```
[] # 비어 있는 리스트
```

조건

1. 임의의 문자열을 입력을 받습니다.
2. 입력을 받은 문자열을 split()을 이용하여 리스트로 쪼갭니다.
3. 새로운 문자열을 입력을 받고, 해당 문자열이 리스트에 있다면 제거합니다.

결과 (A B C 를 입력하였을 경우)

```
- 최초리스트 : ['A','B','C']
# 새로 입력을 받은 문자열이 있으면
- 제거된 리스트 : ['A','C']
# 없으면 제거하지 않고 그대로 출력
- 제거가 안된 리스트 : ['A','B','C']
```