09

Loop

Loop

- 시퀀스 자료형(리스트, 튜플, 문자열 등)의 요소를 하나씩 꺼내며 반복
- 변수에는 매 반복마다 시퀀스의 각 요소가 차례대로 대입
- 들여쓰기된 코드 블록은 반복 횟수만큼 실행

```
for 변수 in 리스트(또는 튜플, 문자열):
    실행할 코드
```

- for는 정해진 횟수만큼 반복할 때 적합
- while은 특정 조건이 만족되는 동안 계속 반복할 때 적합

Loop

- fruits 리스트: 3개의 문자열로 구성
- 리스트의 각 요소를 하나씩 꺼내서 fruit 변수에 저장 후 출력

```
fruits = ['apple', 'banana', 'cherry']
for fruit in fruits:
    print(fruit)
```

```
apple
banana
cherry
```

Loop

- fruits 리스트: 3개의 튜플(개수, 과일)로 구성
- 리스트의 각 요소(이 경우 튜플)를 하나씩 꺼내서 각각 count, fruit 변수에 저장 후 출력

```
fruits = [('one', 'apple'), ('two', 'banana'), ('one', 'cherry')]
for (count, fruit) in fruits:
    print(count, fruit)
```

```
one apple
two banana
one cherry
```

Loop

- 튜플 전체를 변수 both에 저장한 후 a, b = both 형태로 분해
- for a, b in fruits 와 동일한 동작

```
fruits = [('one', 'apple'), ('two', 'banana'), ('one', 'cherry')]
for both in fruits:
    a, b = both
    print(a, b)
```

```
one apple
two banana
one cherry
```

Loop

- 키를 변수에 저장해 값은 반복문 안에서 조회
- items()를 활용해 키, 값을 동시에 조회

```
fruits = {'apple': 1, 'banana': 2, 'cherry': 1}
for fruit in fruits:
    print(fruit, '=>', fruits[fruit])
```

```
apple => 1
banana => 2
cherry => 1
```

```
for (key, value) in fruits.items():
    print(key, '=>', value)
```

```
apple => 1
banana => 2
cherry => 1
```

Loop

- 문자열도 문자 하나씩 순회 가능

```
for ch in "hello":
    print(ch)
```

```
h
e
l
l
o
```

Loop

- 리스트 L의 요소 값을 x에 복사
- x += 1은 변수 x만 수정
- 리스트 안의 값은 불변

```
L = [10, 20, 30, 40, 50]

for x in L:
    x += 1                          # x만 변화

print(L)
print(x)
```

```
[10, 20, 30, 40, 50]
51
```

range()

range() 함수란?

- range(n)은 0부터 n-1까지 숫자 생성
- 반복용 숫자 리스트 생성 도구

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

range()

- range(start, end, step)
  - [start, end) 범위의 값을 step을 통해 건너뛰며 반복 가능

```
for i in range(1, 10, 2):
    print(i)
```

```
1
3
5
7
9
```

```
for i in range(5, 0, -1):
    print(i)
```

```
5
4
3
2
1
```

Loop

- range(len(nums))는 0부터 리스트 길이만큼의 인덱스 생성
- 반복문 변수 i는 리스트의 인덱스로 활용

```
nums = [10, 20, 30]

for i in range(len(nums)):
    print(i, nums[i])
```

```
0 10
1 20
2 30
```

Loop

- scores 리스트의 각 점수를 score 변수로 순회
- total += score를 통해 누적 합계 계산
- 반복이 끝난 뒤 total에는 리스트 값들의 총합 저장

```
scores = [80, 90, 100]
total = 0

for score in scores:
    total += score
print(total)
```

```
270
```

Loop

- for 반복문 안에서 if 조건문을 함께 사용하면 특정 조건에 해당하는 값만 선택적으로 처리 가능

```
for n in [1, 2, 3, 4, 5]:
    if n % 2 == 0:
        print(n, "짝수")
```

```
2 짝수
4 짝수
```

Loop

- matrix는 2차원 리스트로, 각 행(row)은 리스트
- 바깥쪽 for는 각 행을 변수 row에 저장
- 안쪽 for는 그 행에서 각 값을 변수 val에 저장하며 순회

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

for row in matrix:
    for val in row:
        print(val)
```

```
1
2
3
4
```

break

- n이 5가 되는 순간 break가 실행되며 반복문이 즉시 종료
- print(n)은 0부터 4까지만 출력 후 중단

```
for n in range(10):
    if n == 5:
        break
    print(n)
```

```
0
1
2
3
4
```

continue

- n이 2일 때 continue가 실행되어 해당 반복을 건너뛰고 다음 반복으로 이동
- print(n)은 n == 2일 때는 실행 X

```
for n in range(5):
    if n == 2:
        continue
    print(n)
```

```
0
1
3
4
```

Loop

- for 반복문이 중간에 break 없이 정상적으로 종료되면 else 블록 실행
- 반복이 정상적으로 모두 끝났음을 알리는 종료 메시지 등에 사용

```
for n in range(3):
    print(n)
else:
    print("끝까지 반복 완료")
```

```
0
1
2
끝까지 반복 완료
```

Loop

- 리스트 a의 각 요소를 3배 한 값을 append()를 이용해 result 리스트에 추가
- 기존 리스트를 기반으로 새로운 리스트 생성 가능

```
a = [1, 2, 3, 4]
result = []

for num in a:
    result.append(num*3)
print(result)
```

```
[3, 6, 9, 12]
```

Summary

항목	while	for
사용 용도	조건 기반 반복	횟수 기반, 시퀀스 순회
제어 방식	True일 때 반복	iterable의 요소 수만큼 반복

9주차 과제

- 09_For_Loop.ipynb