

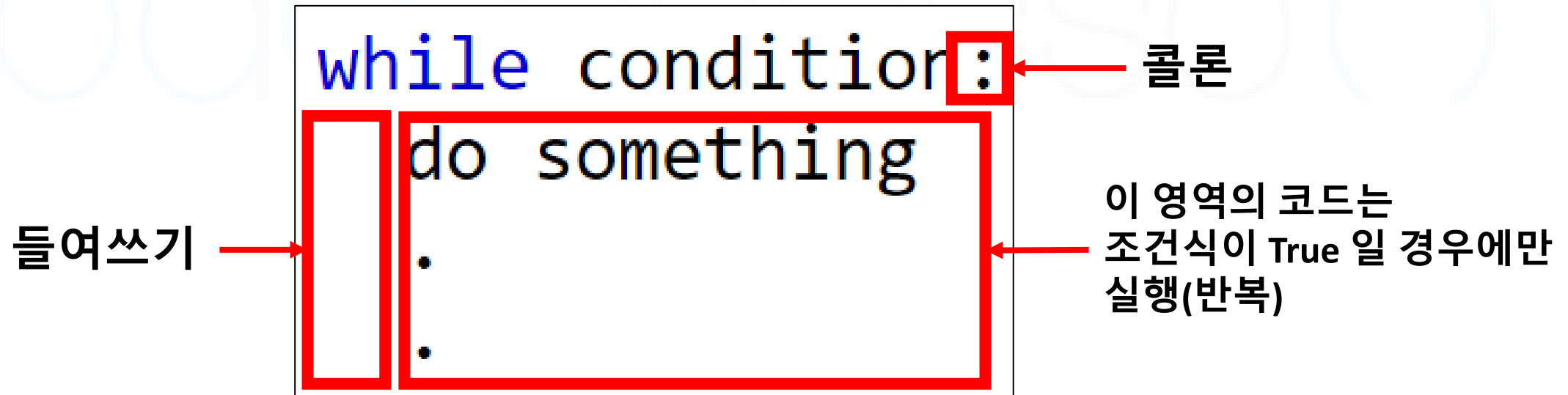
반복문의 개념과 파이썬 반복문 (Iteration Statement)

| 반복문(Iteration Statement)

- 특정 코드 집합(코드 블록)을 반복해서 실행
- 주어진 조건을 만족할 때까지 반복하거나, 일정 회수 만큼 반복
 - Hello Python을 10번 출력
 - 2의 거듭제곱을 반복하여, 결과가 1000이 넘을 때까지 반복
 - 1000보다 작으면 계속 반복
- 프로그래밍 언어에서 반복문을 위해 for, while 문이 일반적으로 사용 됨

|파이썬 반복문

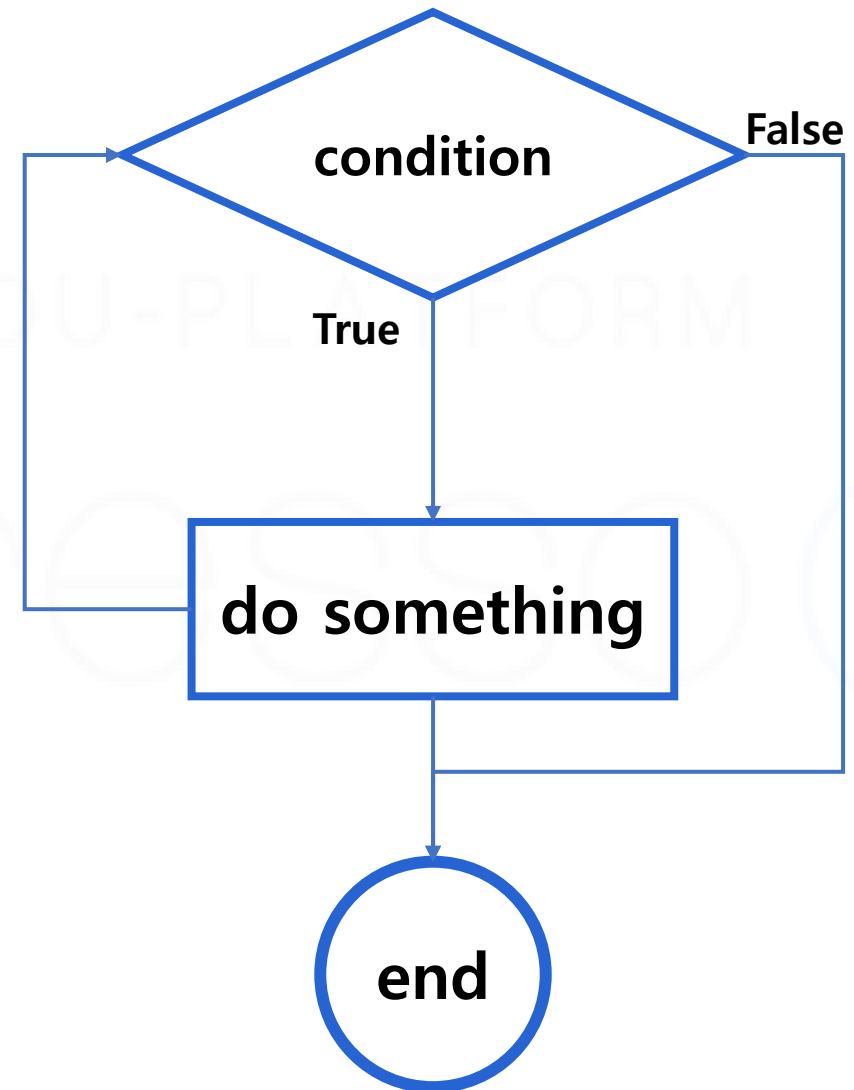
- 반복문을 위해 while문과 for문을 제공
- while 문
 - 조건이 True이면 계속 반복
 - 조건이 False이면 반복 중지



|while 문

```
while condition:  
    do something
```

-
-



|while 문의 활용

- 2의 거듭제곱을 반복하여, 결과가 1000이 넘을 때까지 반복
 - 1000보다 작으면 계속 반복

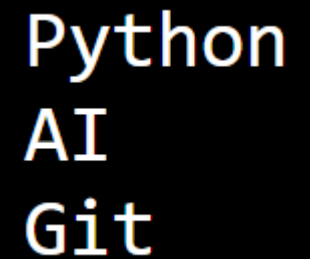
```
result = 1

while result <= 1000:
    result = result * 2
    print(result)
```

```
2
4
8
16
32
64
128
256
512
1024
```

- Sequence 데이터 사용하여 반복
 - 문자열, 리스트, 튜플, ...
 - Sequence 데이터에서 item을 인덱스 순서대로 반복적으로 가져옴
 - range() 함수와 함께 사용하면 주어진 횟수만큼 반복 가능
- for문

```
course_list = ["Python", "AI", "Git"]  
  
for course in course_list:  
    print(course)
```



Python
AI
Git

|for문

리스트에서 꺼낸
값을 저장할 변수

리스트 변수명

```
course_list = ["Python", "AI", "Git"]  
for course in course_list:  
    print(course)
```

콜론

들여쓰기

이 영역의 코드는
리스트에서 남은 item이
있을 경우에만 실행(반복)

|for문의 활용

- 정수 값이 저장되어 있는 리스트가 주어졌을 때, 각각의 값에 100을 곱한 결과를 저장하는 새로운 리스트를 생성
 - [1,2,3,4,5,6,7,8,9,10] → [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]

```
number_list = [1,2,3,4,5,6,7,8,9,10]
new_number_list = []

for number in number_list:
    new_number_list.append(number * 100)

print(new_number_list)
```

```
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]
```


|for문 내에서 if문의 활용

- 정수 값이 저장되어 있는 리스트가 주어졌을 때, 2의 배수만 출력

```
number_list = [1,2,3,4,5,6,7,8,9,10]

for number in number_list:
    ☐ if number % 2 == 0:
        ☐ ☐ print(number)
```

2
4
6
8
10

들여쓰기

break와 continue

| break와 continue

- 특정 조건에 따라 반복문을 제어 - 완전 중단 또는 현재 반복의 중단
- break 문 - 반복문을 완전히 중단
- continue 문 - 현재 반복을 중단하고 다음 반복으로 넘어 감

| break 문

- 반복을 진행중에 break를 만나면 반복문 전체를 중단
- 보통 if문 내부에서 주로 사용 됨
 - 특정 조건이 True 이면 반복문을 중단

```
number_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for number in number_list:
    if number > 5:
        break
    print(number)
```

↑ ↑ ↑ ↑ ↑ break!

1
2
3
4
5

|continue 문

- 반복을 진행중에 continue를 만나면 현재 반복 중단 후 다음 반복으로 넘어 감
- 보통 if문 내부에서 주로 사용 됨
 - 특정 조건이 True 이면 현재 반복을 중단하고 다음 반복으로 이동

```
number_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for number in number_list:
    if number % 3 == 0:
        continue
    print(number)
```

continue! continue! continue!

1
2
4
5
7
8
10

range()와 enumerate()

|for문 에서 range() 함수의 활용

- 특정 회수만큼 반복할 때 사용
- 리스트의 유무와 상관 없이 for문 사용 가능
- for문에서 리스트 위치에 range(n) 사용
 - n은 반복할 회수, 0 부터 n-1까지

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

0
1
2
3
4

|for문 에서 range() 함수의 활용

- 시작 숫자를 지정 가능

- range(start, end) - start를 포함하여 시작하여 end-1까지

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

1
2
3
4

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

0
1
2
3
4

|for문 에서 range() 함수의 활용

```
number_list = [1,2,3,4,5,6,7,8,9,10]
```

```
for number in number_list:  
    if number % 2 == 0:  
        print(number)
```

2
4
6
8
10

=

```
for number in range(1,11):  
    if number % 2 == 0:  
        print(number)
```

|for문 에서 enumerate()의 활용

- for문 사용 시 현재 반복이 몇 번째 반복인지 index 정보 필요한 경우 있음
- enumerate()를 활용하여 현재 반복의 value와, index를 같이 사용 가능

```
course_list = ["Python", "AI", "Git"]  
  
for course in course_list:  
    print(course)
```

```
Python  
AI  
Git
```

```
course_list = ["Python", "AI", "Git"]  
  
for index, course in enumerate(course_list):  
    print(index, course)
```

```
0 Python  
1 AI  
2 Git
```

|range()와 enumerate()의 활용 사례

- 길이가 1,000,000인 리스트가 있을 때
- 100 개 정도의 데이터만 눈으로 확인 하고 싶은 경우

- print() 100번
- range()

```
number_list = [1,2,3,4,5,6,7,8,9,10]

for i in range(3):
    print(number_list[i])
```

1
2
3

- enumerate()

```
number_list = [1,2,3,4,5,6,7,8,9,10]

for i, number in enumerate(number_list):
    if i > 2:
        break
    print(number)
```

1
2
3

반복문의 중첩

| 반복문의 중첩(Nested Iteration Statement)

- 반복문 내에 다른 반복문이 중첩 가능

```
order_list = ["Order 1", "Order 2", "Order 3"]
process_list = ["Checking order", "Cooking", "Packaging", "Delivering"]

for order in order_list:
    for process in process_list:
        print(order, "is in", process, "stage.")
    print("")
```

| 반복문의 중첩 활용

```
order_list = ["Order 1", "Order 2", "Order 3"]
process_list = ["Checking order", "Cooking", "Packaging", "Delivering"]

for order in order_list:
    for process in process_list:
        print(order, "is in", process, "stage.")
    print("")
```

```
Order 1 is in Checking order stage.
Order 1 is in Cooking stage.
Order 1 is in Packaging stage.
Order 1 is in Delivering stage.

Order 2 is in Checking order stage.
Order 2 is in Cooking stage.
Order 2 is in Packaging stage.
Order 2 is in Delivering stage.

Order 3 is in Checking order stage.
Order 3 is in Cooking stage.
Order 3 is in Packaging stage.
Order 3 is in Delivering stage.
```