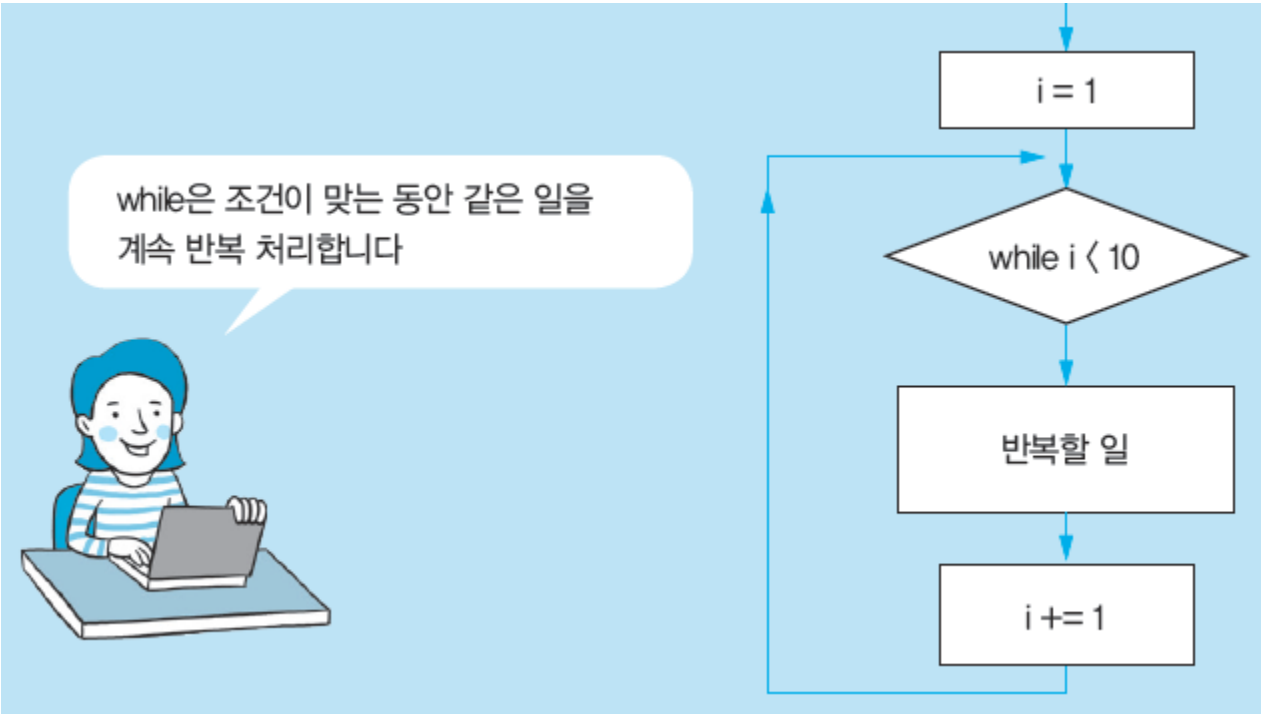
7. while 반복문

1. while 반복 구문
2. while 반복문의 다양한 예
3. 무한 루프와 break
4. else 구문
5. continue 구문
6. 중첩된 반복문
7. 정리

1. while 반복 구문

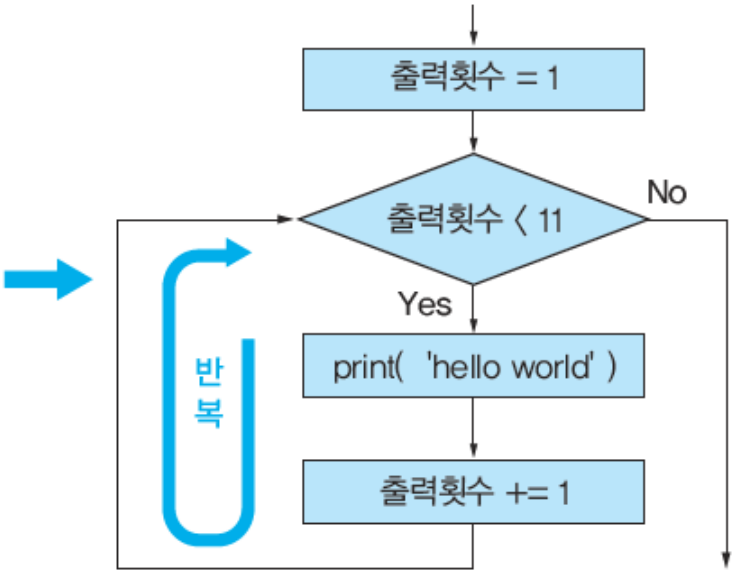
- 컴퓨터는 똑같은 일을 반복 처리하는 작업 능력이 뛰어남.
- 파이썬에서는 while과 for 반복문이 반복 처리를 함.



1. while 반복 구문

◆ ‘hello world’ 10회 출력하기

```
print('hello world')
print('hello world')
print('hello world')
print('hello world')
print('hello world')
print('hello world')
print('hello world')
print('hello world')
print('hello world')
print('hello world')
```

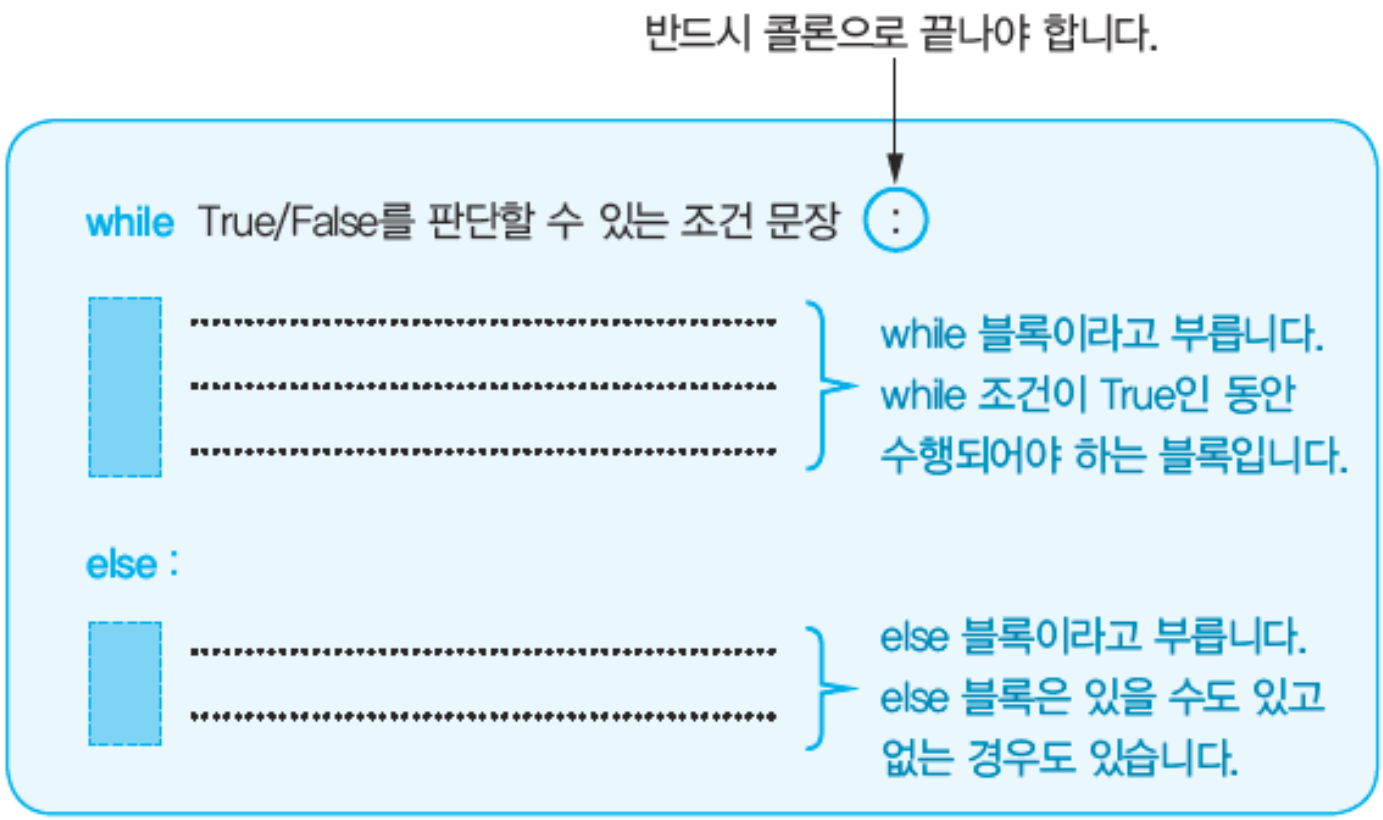


[파이썬 코드]

```
출력횟수 = 1                    # 출력 횟수 1에서 시작함.
while 출력횟수 < 11 :           # 출력 횟수가 11보다 작은 동안 아래 두 줄을 반복함.
    print('hello world')        # ‘hello world’를 출력함.
    출력횟수 += 1               # 출력 횟수를 1 증가시킴.
```

1. while 반복 구문

◆ while 구문 형태

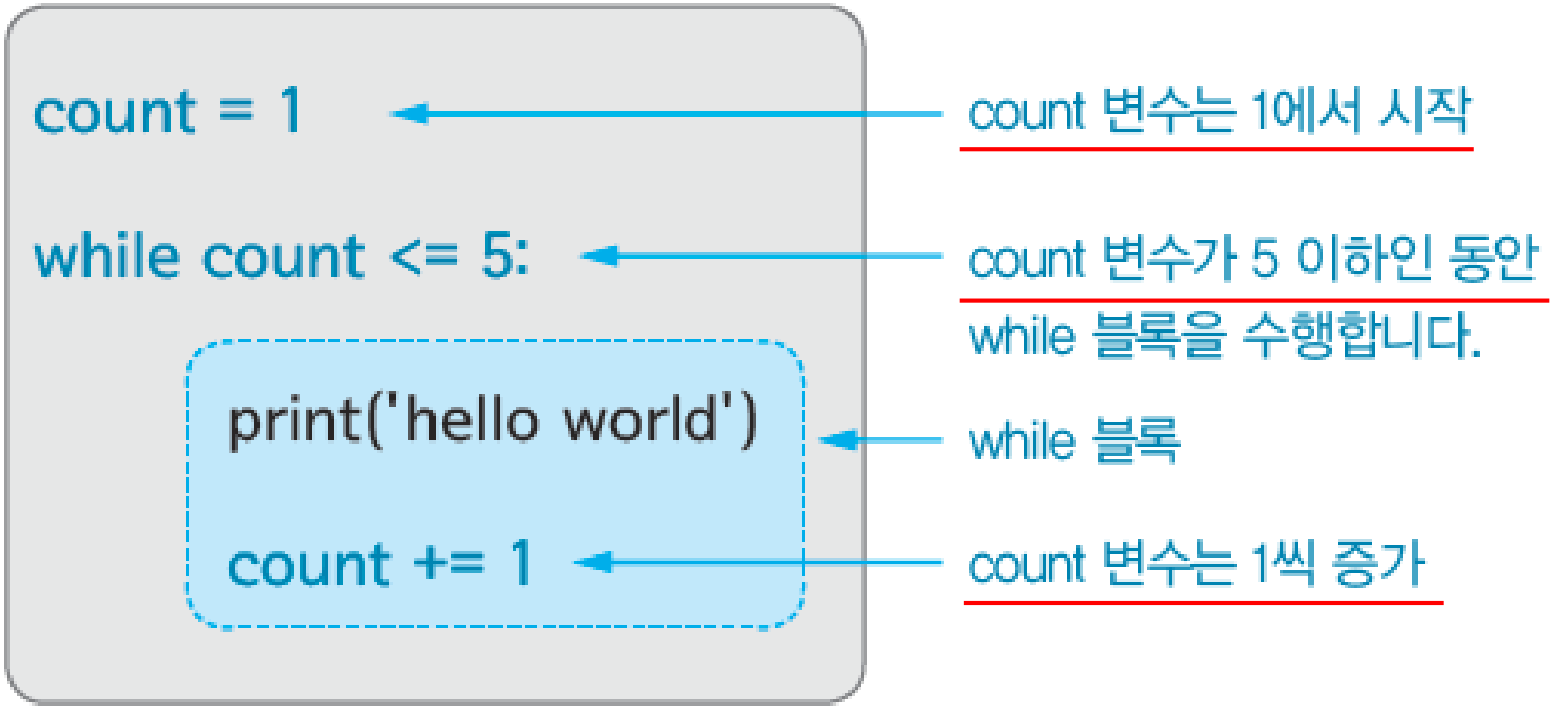


반드시 들여쓰기 되어야 합니다

들여쓰기되지 않았으면 스페이스바 4개 또는 tab키를 이용하여 들여쓰기 합니다

1. while 반복 구문

- ‘hello world’ 5회 출력하는 코드 분석



1. while 반복 구문

- while 반복문을 작성할 때, 무한 루프가 되지 않도록 해야 함

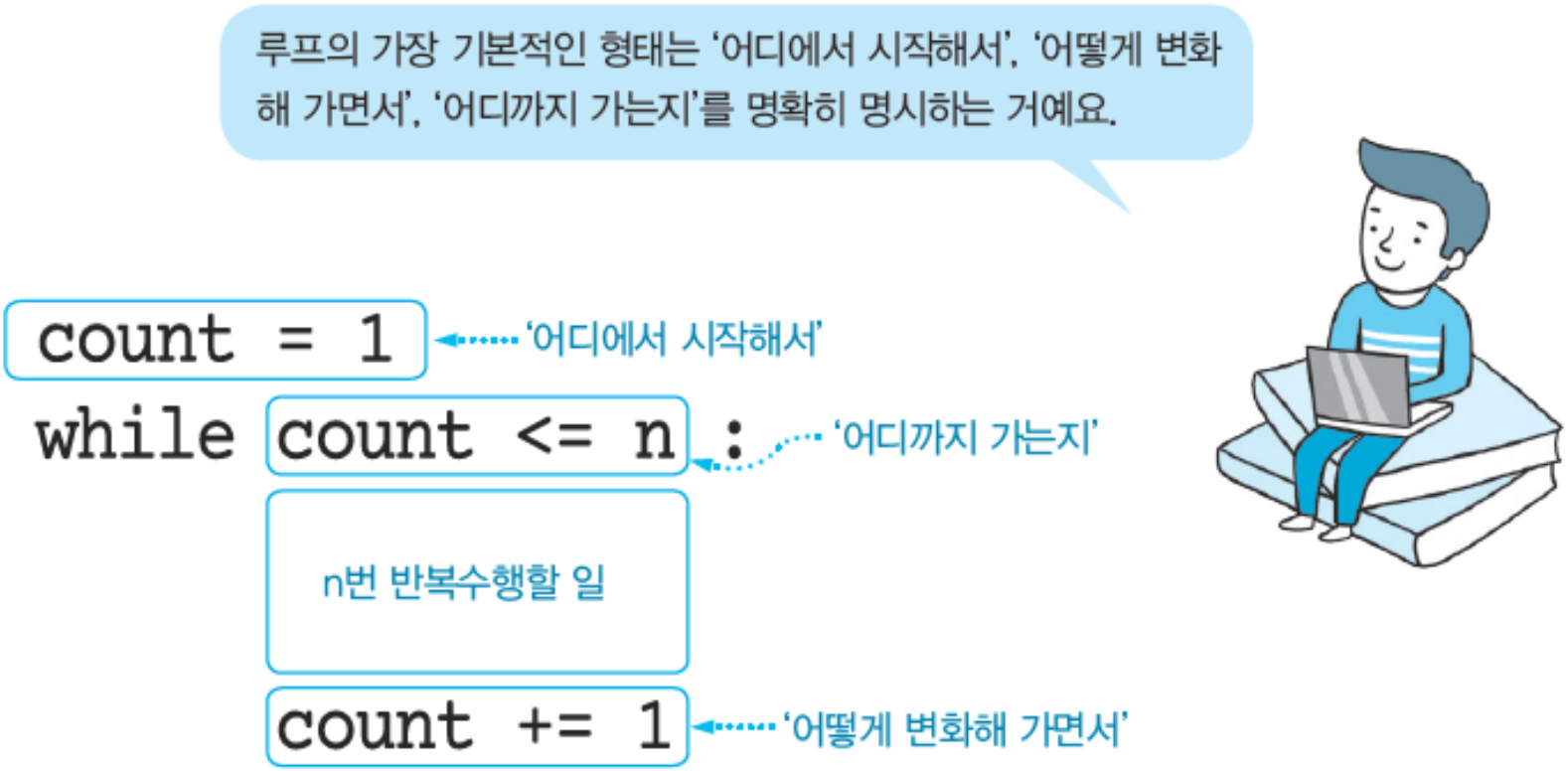
[코드 1]	[코드 2]

count = 1
while count <= 5:
  print(count)
  count += 1 # count가 1씩 증가함
	count = 1
	while count <= 5: # count값이 계속 1이 됨
	  print(count)

[결과 1] 1 2 3 4 5	[결과 2] 1 1 1 1 …… 무한히 1이 출력됩니다.

1. while 반복 구문

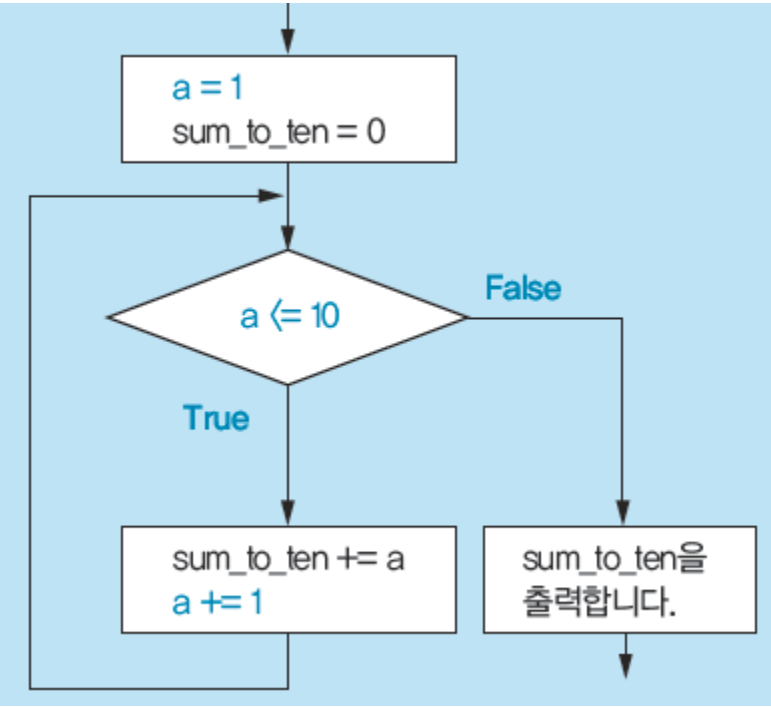
- while 반복문을 작성할 때 중요한 점



2. while 반복문의 다양한 예

◆ while 반복문 기본 예제들

CODE 18 1부터 10까지의 합을 구하는 while 반복문




```
a = 1
sum_to_ten = 0
while a <= 10 :
    sum_to_ten += a
    a += 1
print('1부터 10까지의 합은',
sum_to_ten, '입니다')
```

내장 함수 sum(), range()를 이용하면
위의 작업을 쉽게 처리할 수 있음

```
>>> sum(range(11))
55
```

2. while 반복문의 다양한 예

- 변수 이름을 sum은 사용하지 말 것
  - CODE18에서 변수명 sum_to_ten 대신 sum을 쓰면 어떻게 될까?
  - 파이썬 내장함수 중에 sum() 함수가 있기 때문에 sum을 변수로 사용하게 되면 sum() 함수를 사용할 수 없음

```
〉〉〉 L = [1, 3, 5]
〉〉〉 sum(L)          # sum( ) 내장 함수에 리스트를 넣으면 리스트 원소의 합을 구해 줍니다.
9
〉〉〉 sum = 10        # sum이라는 변수를 만들어서 10을 저장함.
〉〉〉 sum(L)          # sum 변수를 만든 후에 sum( ) 함수를 사용하니까 에러가 발생했어요.
Traceback (most recent call last):
  File "〈pyshell#3〉", line 1, in 〈module〉
    sum(L)
TypeError: 'int' object is not callable
〉〉〉 del sum          # 제가 만든 sum 변수를 삭제했어요.
〉〉〉 sum(L)          # sum 변수를 없앴더니 sum( ) 내장 함수가 다시 동작하네요.
9
```

2. while 반복문의 다양한 예

◆ while 반복문 기본 예제들

CODE 19 하나의 양의 정수를 입력받아서 1부터 그 수까지의 짝수의 합을 구하는 반복문

코드 1	n = int(input('Enter n : '))
	a = 2 # a는 2에서 시작함.
	even_sum = 0
	while a <= n: # a는 입력받은 n까지 가면서 even_sum에 a를 더함.
	  even_sum += a
	  a += 2 # a는 2씩 증가함.
	print('1부터 {}까지의 짝수의 합 : {}'.format(n, even_sum))

코드 2	n = int(input('Enter n : '))
	a = 1 # a는 1에서 시작함.
	even_sum = 0
	while a <= n: # a는 입력받은 n까지 감.
	  if a % 2 == 0: # a가 짝수라면 even_sum에 a를 더함.
	  even_sum += a
	  a += 1 # a는 1씩 증가함.
	print('1부터 {}까지의 짝수의 합 : {}'.format(n, even_sum))

2. while 반복문의 다양한 예

◆ while 반복문 기본 예제들

CODE 20 하나의 양의 정수를 입력받아서, 그 수만큼 숫자를 입력받아서 평균 구하기

```
n = int(input('Enter n : '))        # 정수 n을 입력받음. n은 반복 횟수임.
i = 1
sum_data = 0                        # 변수 sum_data는 0으로 초기화함.
while i <= n:                       # n이 반복 횟수로 이용됨.
    data = int(input('Enter number : '))      # 정수를 n번 입력받음.
    sum_data += data                # sum_data에 입력받는 수 data를 계속 더함.
    i += 1
avg = sum_data / n      # 입력받은 수의 합 sum_data를 n으로 나누어 평균을 구함.
print('average : {:<10.2f}'.format(avg))
```

[결과 1]

```
Enter n : 5
Enter number : 88
Enter number : 93
Enter number : 95
Enter number : 75
Enter number : 82
average : 86.60
```

[결과 2]

```
Enter n : 4
Enter number : 90
Enter number : 88
Enter number : 75
Enter number : 89
average : 85.50
```

[결과 3]

```
Enter n : 3
Enter number : 100
Enter number : 90
Enter number : 80
average : 90.00
```

2. while 반복문의 다양한 예

◆ while 반복문 기본 예제들

CODE 21 정수 10개를 하나씩 입력받아서 10개의 수 중에서 가장 큰 수 출력하기

```
i = 1
prompt = 'Enter number ' + str(i) + ' : '      # str(i)라고 해야 문자열을 연결함.
data = int(input(prompt))                      # 첫 번째 데이터를 입력받음.
max_value = data          # 첫 번째로 입력받는 데이터를 max_value에 저장함.
while i <= 9:             # 남은 9개의 데이터를 입력받아야 함.
    i += 1
    prompt = 'Enter number ' + str(i) + ' : '
    data = int(input(prompt))      # 다음 데이터를 입력받음.
    if data > max_value:           # 현재 max_value보다 더 큰 data가 입력되면,
        max_value = data           # max_value를 data로 바꿈.

# 루프가 끝나고 나서 max_value에 남은 값이 가장 큰 값입니다.
print('The largest value is {}.'.format(max_value))
```

2. while 반복문의 다양한 예

- while 반복문 기본 예제들

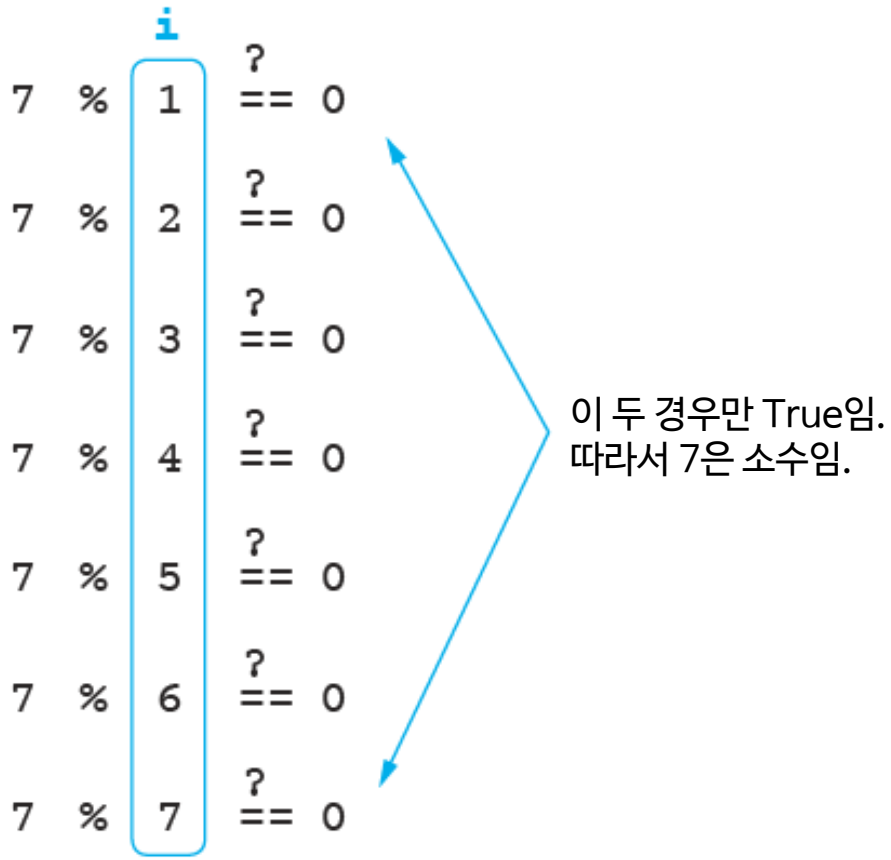
CODE 21 정수 10개를 하나씩 입력받아서 10개의 수 중에서 가장 큰 수 출력하기

[결과 1]	[결과 2]	[결과 3]
Enter number 1 : 4 Enter number 2 : 10 Enter number 3 : 20 Enter number 4 : 5 Enter number 5 : 1 Enter number 6 : 40 Enter number 7 : 22 Enter number 8 : 7 Enter number 9 : 13 Enter number 10 : 9 The largest value is 40.	Enter number 1 : 100 Enter number 2 : 97 Enter number 3 : 88 Enter number 4 : 70 Enter number 5 : 65 Enter number 6 : 55 Enter number 7 : 50 Enter number 8 : 45 Enter number 9 : 33 Enter number 10 : 11 The largest value is 100.	Enter number 1 : 10 Enter number 2 : 12 Enter number 3 : 13 Enter number 4 : 50 Enter number 5 : 55 Enter number 6 : 56 Enter number 7 : 70 Enter number 8 : 77 Enter number 9 : 78 Enter number 10 : 80 The largest value is 80.

2. while 반복문의 다양한 예

◆ while 반복문 기본 예제들

CODE 22 하나의 정수를 입력받아서 소수인지 판단하는 프로그램




```
    n = int(input('Enter n : '))
    count = 0
① i = 1
② while i<= n:
③       if n % i == 0:    # i가 n의 약수인 경우
④           count += 1 # count 증가시킴.
⑤       i += 1

    if count == 2:              # 약수가 2개임.
        print(n,'is prime')
    else:
        print(n, 'is not prime')
```

2. while 반복문의 다양한 예

◆ 시퀀스 자료형에 while 반복문 수행하기 – 문자열

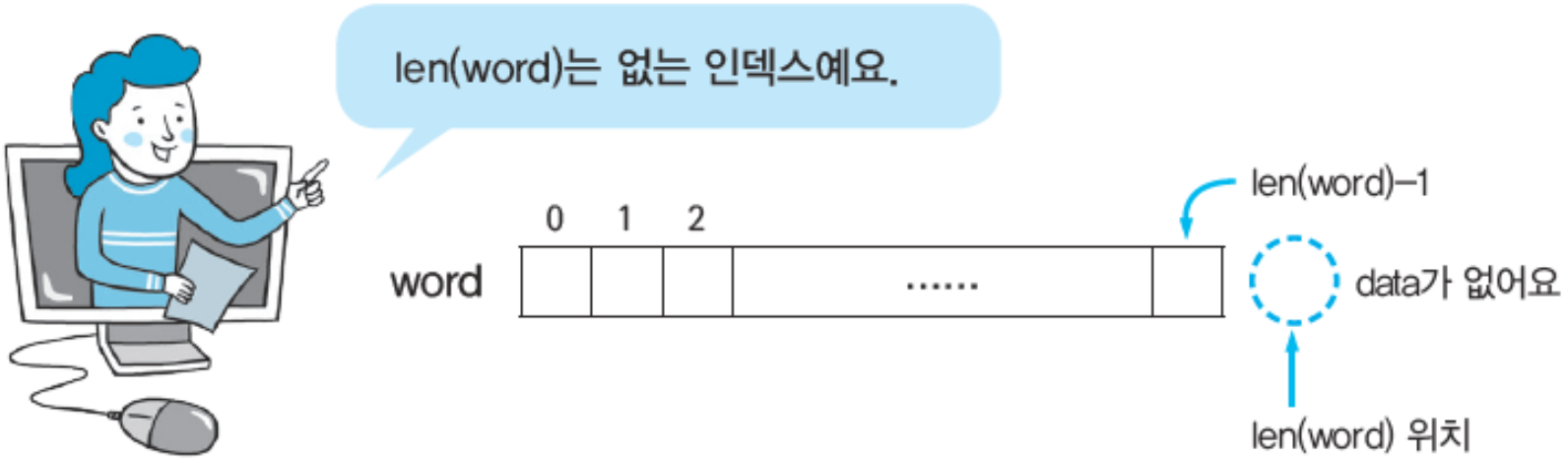
CODE 23 하나의 문자열을 그 안에 모음이 몇 개인지 세는 프로그램

```
word = input('Enter word : ')
i = 0                          # i는 0에서 시작합니다. (문자열의 첫 인덱스는 0입니다)
count = 0                      # count 변수는 모음의 개수를 저장합니다.
while i < len(word) :          # i는 len(word)-1 까지 갑니다.
    if word[i] in 'aeiou' :    # word[i]가 모음인지 판단합니다.
        count += 1             # word[i]가 모음이면 count를 증가시킵니다.
    i += 1                     # i는 1씩 증가합니다.
print('모음의 개수 :', count)
```

[결과 1]	[결과 2]	[결과 3]
Enter word : hello 모음의 개수 : 2	Enter word : python 모음의 개수 : 1	Enter word : looping 모음의 개수 : 3

2. while 반복문의 다양한 예

- 시퀀스 자료형에 while 반복문 수행하기 - 문자열
  - 문자열의 마지막 인덱스는 len(word)-1 임



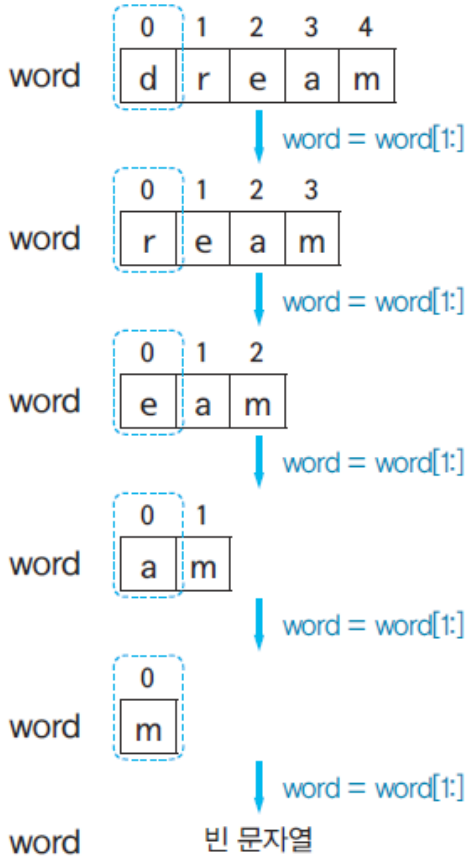
2. while 반복문의 다양한 예

◆ 시퀀스 자료형에 while 반복문 수행하기 – 문자열

CODE 23 하나의 문자열을 그 안에 모음이 몇 개인지 세는 프로그램 (다르게 작성)

```
word = input('Enter word : ')
count = 0
while word :      # word가 빈 문자열이면 False임.
    if word[0] in 'aeiou' :
        count += 1
    word = word[1:]
print('모음의 개수 :', count)
```

word[0]을 떼어내고 나머지를 word에 넣습니다.



2. while 반복문의 다양한 예

◆ 시퀀스 자료형에 while 반복문 수행하기 – 문자열

CODE 24 하나의 문자열을 받아서 대문자, 소문자, 스페이스, 숫자의 개수 세는 프로그램

```
word = input('word : ')      # 문자열을 입력받습니다.
count_upper = 0              # 대문자의 개수를 저장할 변수입니다.
count_lower = 0              # 소문자의 개수를 저장할 변수입니다.
count_space = 0              # 스페이스의 개수를 저장할 변수입니다.
count_digit = 0              # 숫자의 개수를 저장할 변수입니다.
i = 0
while i < len(word):
    if word[i].isupper(): count_upper += 1     # word[i]가 대문자인 경우
    elif word[i].islower(): count_lower += 1   # word[i]가 소문자인 경우
    elif word[i].isspace(): count_space += 1   # word[i]가 스페이스인 경우
    elif word[i].isdigit(): count_digit += 1   # word[i]가 숫자인 경우
    i += 1
print('upper letters :', count_upper)     # 대문자의 개수 출력
print('lower letters :', count_lower)     # 소문자의 개수 출력
print('spaces :', count_space)            # 스페이스의 개수 출력
print('digits :', count_digit)            # 숫자의 개수 출력
```

2. while 반복문의 다양한 예

◆ 시퀀스 자료형에 while 반복문 수행하기 – 문자열

CODE 24 하나의 문자열을 받아서 대문자, 소문자, 스페이스, 숫자의 개수 세는 프로그램

[결과 1]	[결과 2]	[결과 3]
word : hE11pR 9@!-5Y	word : pyTHOn 3 ! !	word : 12 345 !6
upper letters : 3	upper letters : 3	upper letters : 0
lower letters : 2	lower letters : 3	lower letters : 0
spaces : 1	spaces : 2	spaces : 2
digits : 4	digits : 1	digits : 6

2. while 반복문의 다양한 예

◆ 시퀀스 자료형에 while 반복문 수행하기 – 문자열

CODE 25 하나의 문자열을 받아서 그 문자열을 거꾸로 만들어 출력하는 프로그램

```
word = input('Enter word : ')
reversed_word = ''
i = -1    # word 끝에서부터 시작해서 거꾸로 루프를 수행
while i >= -len(word):
    reversed_word += word[i]
    i -= 1

print('reversed word :', reversed_word)
```

[결과 1]
Enter word : school
reversed word : loohcs

[결과 2]
Enter word : programming
reversed word : gnimmargorp

[결과 3]
Enter word : ?
reversed word : ?

[결과 4]
Enter word : hi!
reversed word : !ih

```
>>> word = 'python'
>>> reversed_word = word[::-1]    # 이렇게 해도 되죠. 루프가 필요하지 않아요.
>>> word, reversed_word
('python', 'nohtyp')
```

2. while 반복문의 다양한 예

◆ 시퀀스 자료형에 while 반복문 수행하기 – 리스트

CODE 26 리스트에 저장된 과일 출력하는 코드

```
fruits = ['orange', 'strawberry', 'kiwi', 'pineapple']
i = 0                          # 인덱스 0부터 시작합니다.
while i < len(fruits):         # 마지막 인덱스까지 루프를 돌립니다.
    print(fruits[i])
    i += 1                     # 인덱스를 1씩 증가시킵니다.
```

[결과]

```
orange
strawberry
kiwi
pineapple
```

2. while 반복문의 다양한 예

◆ 시퀀스 자료형에 while 반복문 수행하기 – 리스트

CODE 27 리스트에 같은 이름이 몇 번 저장되어 있는지 확인하는 코드

```
friends = ['Paul', 'Bob', 'Alice', 'Bob', 'David', 'Cindy', 'Alice', 'Bob']
name = input('Enter one name : ')   # 이름을 입력받습니다.
count = 0                           # 친구의 수를 세는 변수로 초기값은 0입니다.
if name not in friends:             # 입력받은 이름이 친구 리스트에 없는 경우입니다.
    print('There is no "{}" in friends list.'.format(name))
else:                               # 입력받은 이름이 친구 리스트에 있으면, 몇 명인지 세어 봅니다.
    i = 0
    while i < len(friends):
        if friends[i] == name:      # name과 같은 이름이 friends 리스트에 있는 경우
            count += 1              # count를 1 증가시킵니다.
        i += 1
if count > 0:                       # count가 0보다 크면 입력한 이름의 친구가 count명만큼 있습니다.
    print('There are {} "{}" in friends list.'.format(count, name))
```

2. while 반복문의 다양한 예

◆ 시퀀스 자료형에 while 반복문 수행하기 – 튜플

CODE 28 튜플에 저장된 토플 성적을 출력하는 코드 (성적을 '*'로 구분하여 한 줄로 출력)

```
toefl = (80, 98, 115, 105, 85, 118, 100, 92, 75, 93)
i = 0
while i < len(toefl):
    if i == len(toefl) - 1:        # 맨 마지막 데이터 다음에는 '*'를 출력하지 않습니다.
        print(toefl[i])
    else:        # 맨 마지막 데이터를 제외한 데이터들은 성적 다음에 '*'를 출력합니다.
        print(toefl[i], '* ', end='')
    i += 1
```

[결과]

```
80 * 98 * 115 * 105 * 85 * 118 * 100 * 92 * 75 * 93
```

2. while 반복문의 다양한 예

◆ 시퀀스 자료형에 while 반복문 수행하기 – 튜플

CODE 29 토플 성적이 100점 이상인 사람의 수 구하는 코드

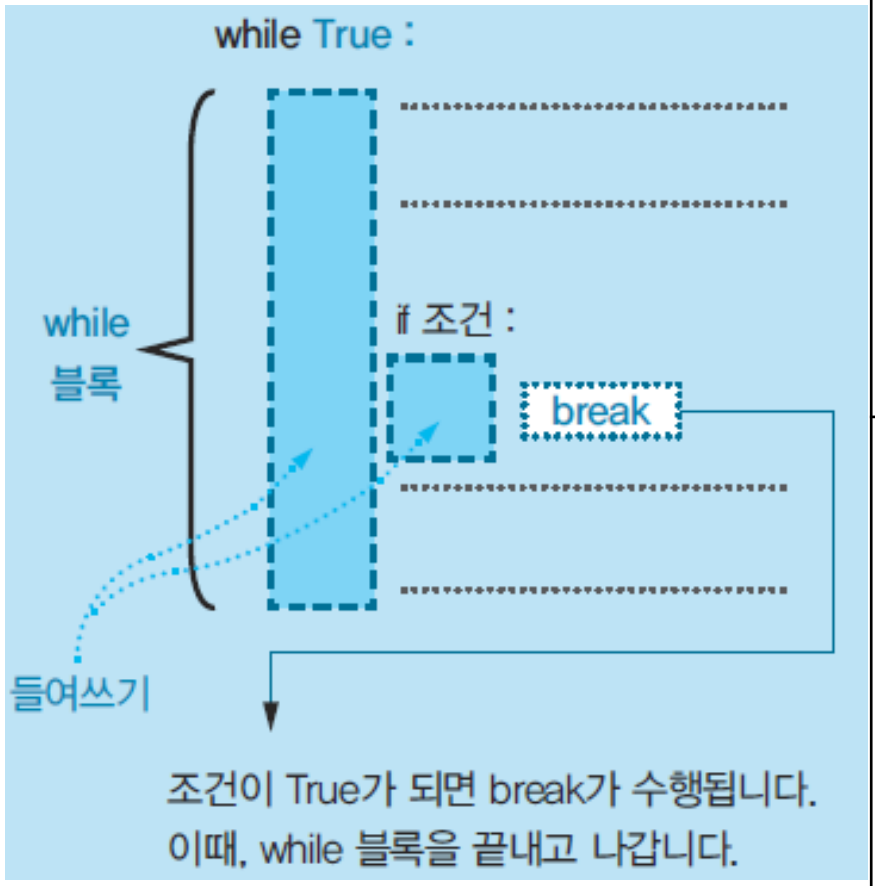
```
toefl = (80, 98, 115, 105, 85, 118, 100, 92, 75, 93)
count = 0                    # 100점 이상인 사람의 수를 세는 변수입니다.
i = 0                        # 0부터 len(toefl)-1까지 루프를 수행합니다.
while i < len(toefl):
    if toefl[i] >= 100:      # toefl[i]가 100 이상이라면, count를 1 증가합니다.
        count += 1
    i += 1
print('There are {} people over 100.'.format(count))
```

[결과]

```
There are 4 people over 100.
```

3. 무한 루프와 break

- 무한 루프는 while True로 작성하고, break가 꼭 있어야 함



[코드]

```
while True:
    friend = input('Who is your friend? ')
    if friend == 'none':
        break
    print('{} is your friend.'.format(friend))
```

[결과]

```
Who is your friend? Alice
Alice is your friend.
Who is your friend? Paul
Paul is your friend.
Who is your friend? David
David is your friend.
Who is your friend? none
```

3. 무한 루프와 break

- 무한 루프는 while True로 작성하고, break가 꼭 있어야 함

```
i = 1
while True :
    print(i)
    if i == 5 : break    # 간단한 if 블록은 콜론 옆에 적어도 됩니다.
    i += 1
```

[결과]

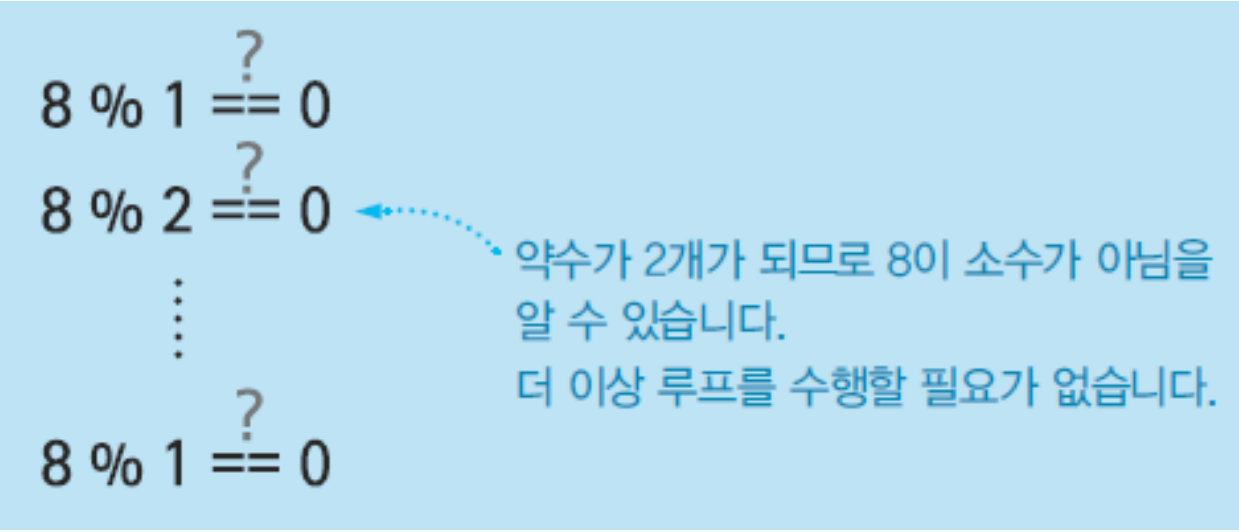
```
1
2
3
4
5
```

참고 IDLE에서 무한 수행을 멈추게 하려면 Ctrl 키를 누른 상태로 C를 눌러주세요(Ctrl+C). 그러면 IDLE에서 무한 루프의 수행을 멈추고 프롬프트를 다시 내 줍니다.

3. 무한 루프와 break

- 무한 루프는 while True로 작성하고, break가 꼭 있어야 함

CODE 30 break를 이용한 소수 판단 코드. 이전 코드에서는 n이 소수인지 판단하려면 루프를 수행했어야 하는데, 이를 break 구문을 이용하여 개선할 수 있음.



3. 무한 루프와 break

◆ 무한 루프는 while True로 작성하고, break가 꼭 있어야 함

```
import sys          # 프로그램을 끝내는 sys.exit(0)를 사용하려면 sys 모듈이 필요합니다.
n = int(input('Enter n : '))
if n == 1:          # 만약에 1을 입력하면 소수가 아니라고 하고 프로그램을 끝냅니다.
    print('1 is not prime')
    sys.exit(0)  # 프로그램을 아주 끝냅니다.
count = 0
i = 1
while True:
    if n % i == 0: count += 1     # i가 n의 약수이면, count 값을 증가시킵니다.
    if count == 2 : break         # count가 2가 되면, 루프를 끝냅니다.
    i += 1
if i < n: print(n,'is not prime')   # count가 2인데, i<n이면, n은 소수가 아니겠죠.
else: print(n, 'is prime')
```

[결과 1]	[결과 2]	[결과 3]
Enter n : 1 1 is not prime	Enter n : 10 10 is not prime	Enter n : 17 17 is prime

3. 무한 루프와 break

◆ 플래그를 이용한 루프 제어

```
flag = True
while flag:                          # flag가 True인 동안 루프가 수행됩니다.
    friend = input('Who is your friend? ')
    if friend == 'none':             # 'none'이 입력되면 flag 값을 False로 바꿉니다.
        flag = False
    else:
        print('{} is your friend.'.format(friend))
```

[결과]

```
Who is your friend? Tom
Tom is your friend.
Who is your friend? Kelly
Kelly is your friend.
Who is your friend? none
```

4. else 구문

◆ while 반복문에 else 구문 사용하기

- while 반복문이 끝나게 되는 두 가지 경우

❶ i = 1 ❷ while i <= 10 : ❸ print(i) ❹ if i == 5 : break # 반복 끝남 ❺ i += 1	❶ i = 1 ❷ while i <= 3 : # 반복 끝남 ❸ print(i) ❹ if i == 5 : break ❺ i += 1

else 구문은 위의 두 코드와 같은 상황과 관련이 있다.

① while의 조건이 False가 되어 루프를 끝내는 경우 else 블록이 있으면 수행됩니다.

② while 블록 안에서 break를 만나서 루프를 끝내는 경우 else 블록이 있어도 수행되지 않습니다.

4. else 구문

- while 반복문에 else 구문 사용하기

```
i = 1
while i <= 5 :     # 루프가 여기서 끝납니다.
    print(i)
    if i == 10 :
        break
    i += 1
else :
    print('else block')
print('outside while')
```

[결과]

```
1
2
3
4
5
else block
outside while
```

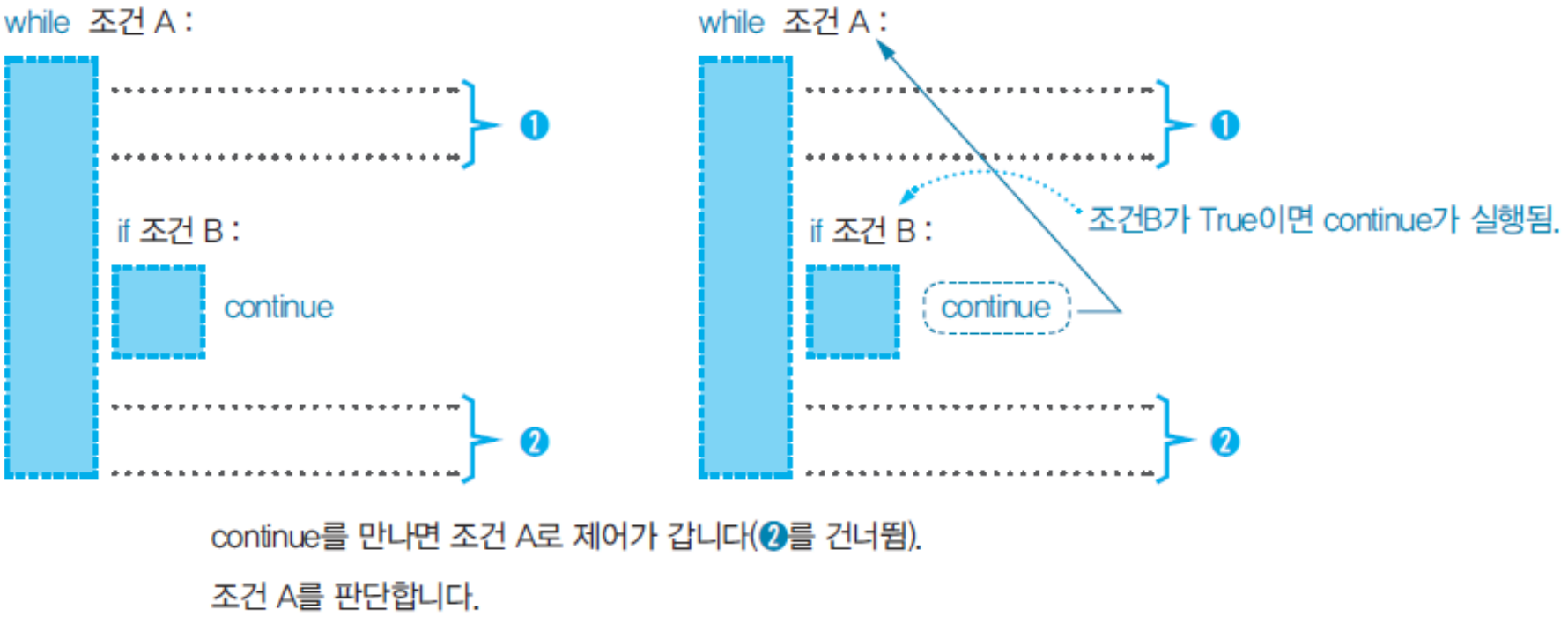
```
i = 1
while True :
    print(i)
    if i == 5 :       # 루프가 여기서 끝납니다.
        break
    i += 1
else :
    print('else block')
print('outside while')
```

[결과]

```
1
2
3
4
5
outside while
```

5. continue 구문

- continue 구문은 루프 안에서만 사용한다
  - while 블록 안에서 continue를 만나면 while 키워드 옆에 조건으로 제어가 간다.



5. continue 구문

- continue 구문은 루프 안에서만 사용한다

```
i = 0
while i < 10 :
    i += 1
    if i % 3 == 0 :
        continue
    print(i)
```

i가 3의 배수인 경우 continue를 만나서 while 조건으로 제어가 갑니다. print(i)가 수행되지 못합니다.

[결과]

```
1
2
4
5
7
8
10
```

5. continue 구문

◆ continue 구문은 루프 안에서만 사용한다

CODE 31 단어를 입력받아서 입력받은 단어가 영어나 숫자로 구성되어 있는 경우에만 출력하는 코드. 이런 단어를 3개 출력한다.

```
i = 1
while i <= 3:
    word = input('Enter word : ')      # 단어를 입력받습니다.
    if not word.isalnum():             # 입력받은 단어가 영문자와 숫자로만 구성되지 않은 경우
        continue
    print('word --> ', word)           # continue가 걸리지 않으면 단어를 출력합니다.
    i += 1
```

[결과 1]

```
Enter word : abcd1234
word --> abcd1234
Enter word : hello?
Enter word : alice@aaa.com
Enter word : friends
word --> friends
Enter word : 112233
word --> 112233
```

[결과 2]

```
Enter word : a1b2c3d4
word --> a1b2c3d4
Enter word : 123ddd
word --> 123ddd
Enter word : hello world
Enter word : python
word --> python
```

6. 중첩된 반복문

- while 반복문 안에 다시 while 반복문이 중첩되는 경우

6. 중첩된 반복문

- while 반복문 안에 다시 while 반복문이 중첩되는 경우
  - 구구단 출력하기

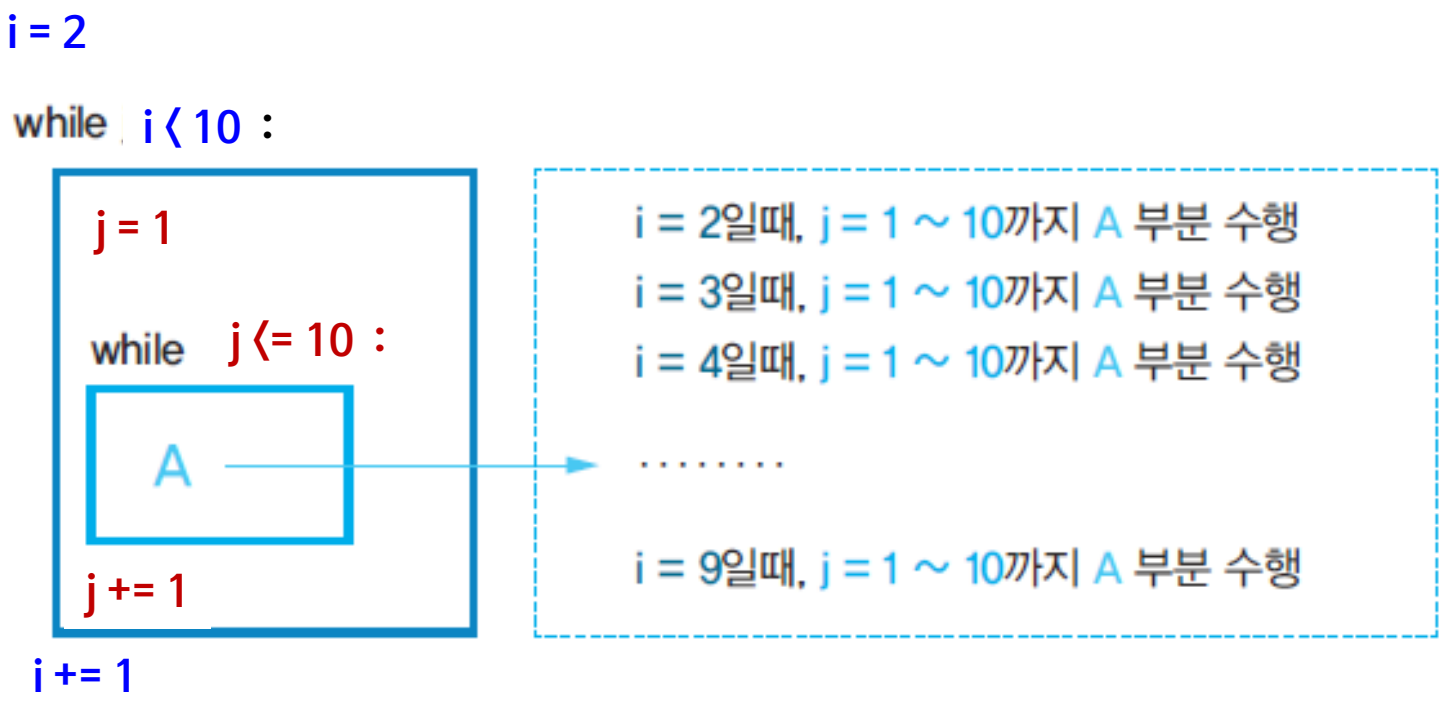
```
i = 2
while i < 10 :
    j = 1
    while j <= 10 :
        k = i * j
        print('%3d' % k, end='')
        j += 1
    i += 1
    print( )
```

j →	1	2	3	4	5	6	7	8	9	10
i = 2	2	4	6	8	10	12	14	16	18	20
i = 3	3	6	9	12	15	18	21	24	27	30
i = 4	4	8	12	16	20	24	28	32	36	40
i = 5	5	10	15	20	25	30	35	40	45	50
i = 6	6	12	18	24	30	36	42	48	54	60
i = 7	7	14	21	28	35	42	49	56	63	70
i = 8	8	16	24	32	40	48	56	64	72	80
i = 9	9	18	27	36	45	54	63	72	81	90

출력 결과

6. 중첩된 반복문

- while 반복문 안에 다시 while 반복문이 중첩되는 경우
  - 구구단 출력하기



6. 중첩된 반복문

- while 반복문 안에 다시 while 반복문이 중첩되는 경우

CODE 32 하나의 양의 정수를 입력받아서 그 수만큼 한 줄에 '*' 출력하는 코드 (중첩 루프 필요하지 않음).

```
b = int(input('Enter b : '))      # 정수 b를 입력받습니다.
i = 1
while i <= b :                    # b번 루프돌면서 '*'를 출력합니다.
    print('*', end='')
    i += 1
```

[결과 1]	[결과 2]	[결과 3]
Enter b : 15 ***************	Enter b : 7 *******	Enter b : 10 **********

사실 위의 코드는 간단히 print('*' * b) 하면 됨.

6. 중첩된 반복문

◆ while 반복문 안에 다시 while 반복문이 중첩되는 경우

CODE 33 두 개의 양의 정수 a와 b를 입력받아서 a행 b열의 행렬에 '*' 출력하는 코드

```
a = int(input('Enter a : '))
b = int(input('Enter b : '))
i = 1
while i <= a:                    # i는 1부터 a까지 1씩 증가하면서 루프를 수행합니다.
    j = 1
    while j <= b:                # j는 1부터 b까지 1씩 증가하면서 '*'를 출력합니다.
        print('*', end='')
        j += 1
    i += 1
    print( )                     # 다음 줄로 넘어갑니다.
```

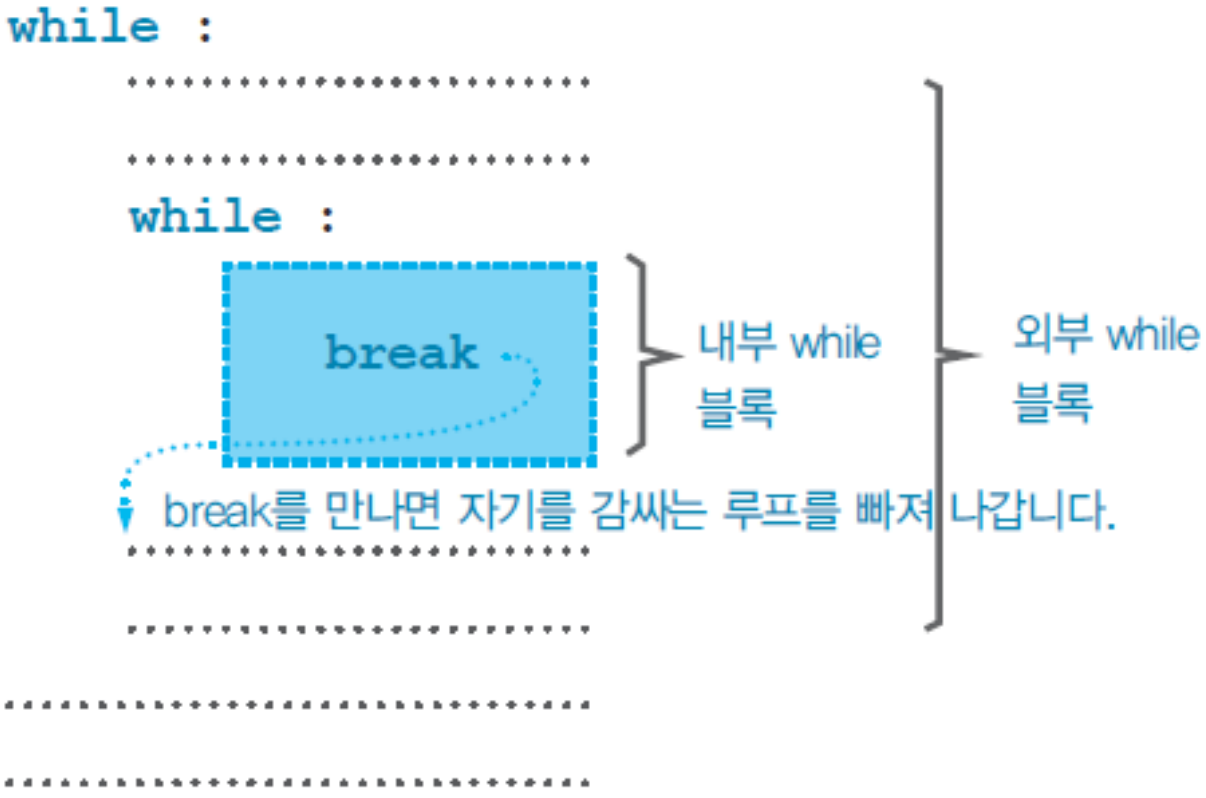
```
Enter a : 5
Enter b : 7
*******
*******
*******
*******
*******
```

```
Enter a : 7
Enter b : 5
*****
*****
*****
*****
*****
*****
*****
```

```
Enter a : 1
Enter b : 10
**********
```

6. 중첩된 반복문

- 중첩 루프에서 break 사용 시에 주의점
  - break는 자신을 감싸고 있는 루프만을 빠져 나감



6. 중첩된 반복문

- 중첩 루프에서 break 사용 시에 주의점
  - break는 자신을 감싸고 있는 루프만을 빠져 나감

```
a = 1
while a <= 3:
    b = 1
    while b <= 5:
        if b == 3:
            break
        print('{} - {}'.format(a,b))
        b += 1
    a += 1
```

[결과]

```
1 - 1
1 - 2
2 - 1
2 - 2
3 - 1
3 - 2
```

7. 정리

- 파이썬의 while 반복문에 대하여 학습하였음.
- 다양한 while 반복문 형태를 학습하였음.
- while 반복문 내의 else 구문을 이해함.
- 반복을 제어하는 break, continue 구문을 이해함.