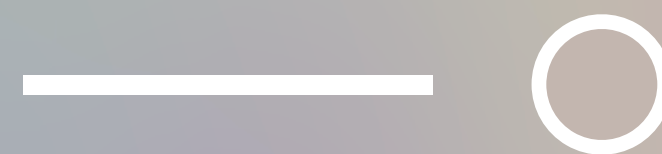


AI TRACK

PYTHON 기초



조건문 — 0.1

반복문 — 0.2



강연자 이름
남궁규민



중앙대학교
산업보안학과 교육봉사동아리

23/02/09 *Thur*

목차

01 소프트웨어 교육과 파이썬

02 PYTHON 기초

03 Colab 사용법

04 조건문의 활용

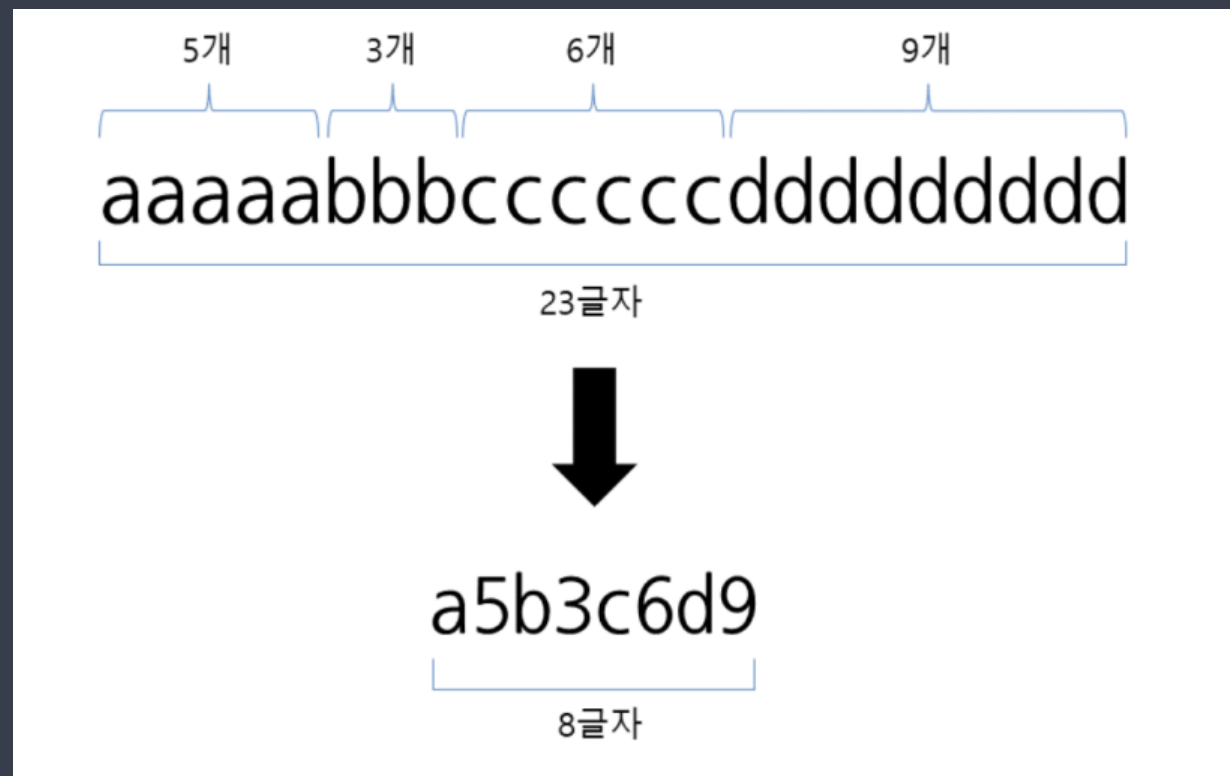
05 반복문의 활용

06 과제

압축 알고리즘

: 휴대폰이나 카메라를 가진 사람들은 모두가 이용하는 알고리즘

휴대폰이나 카메라로 사진을 찍으면 JPG(압축 알고리즘을 구현한 포맷)라는 확장자로 저장 -> 많은 사진 보관 가능



'aaaaabbbbccccccddddddddd' 문자열 같은 문자의 반복

-> 압축

코딩

* 알고리즘

일정한 패턴을 발견하고, 패턴을 토대로 문제를 해결하는 절차

: 알고리즘을 코드로 표현하는 행동

패턴

- 같은 문자가 여러 번 반복되는 패턴을 발견

문제를 해결하는 절차(알고리즘)

- 반복되는 문자를 세기
- 문자가 반복되는 횟수를 적어줌



코드

```
data = 'aaaaabbbccccccddddddddd'
encoded = ''

count = 1
for i in range(1, len(data)):
    if data[i] == data[i - 1]:
        count += 1
    else:
        encoded += data[i - 1] + str(count)
        count = 1

if i == len(data) - 1:
    encoded += data[i] + str(count)
```

PYTHON

: 네덜란드의 귀도 반 로섬(Guido van Rossum)이 개발한 프로그래밍 언어

파이썬의 특징

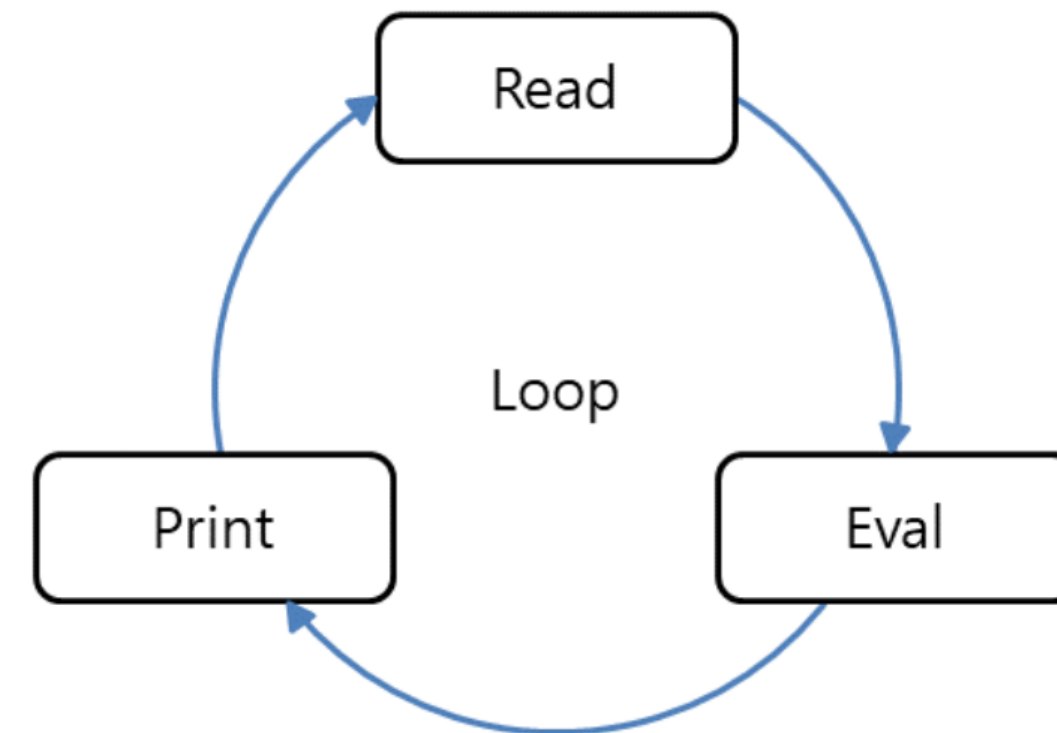
- 쉽고 간단한 문법, 배우기 쉬움
- 객체 지향
- 다양한 패키지
- 오픈 소스, 무료



PRINT문

```
>>> print('Hello, world!')  
Hello, world!  
>>>
```

=> Hello, world! 출력



변수

```
x=10 #x에 숫자 저장
print(x)
y='Hello, world!' #y에 문자열 저장
print(y)
```

```
10
Hello, world!
```

변수 이름의 규칙

1. 영문 문자와 숫자를 사용할 수 있습니다.
2. 대소문자를 구분합니다.
3. 문자부터 시작해야 하며 숫자부터 시작하면 안 됩니다.
4. _(밑줄 문자)로 시작할 수 있습니다.
5. 특수 문자(+, -, *, /, \$, @, &, % 등)는 사용할 수 없습니다.
6. 파이썬의 키워드(if, for, while, and, or 등)는 사용할 수 없습니다.

불과비교연산자

참(True), 거짓(False)을 나타내는 불(boolean)

```
print(True)  
print(False)
```

```
print(3>1)  
print(10==4)  
print('python' != 'python')
```

```
True  
False  
True  
False  
False
```


논리연산자

```
#논리 연산자 and  
print(True and True)  
print(True and False)  
print(False and False)
```

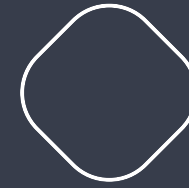
True
False
False

```
#논리 연산자 or  
print(True or True)  
print(True or False)  
print(False or False)
```

True
True
False

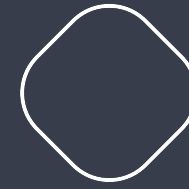
```
#논리연산자 not  
print(not True)  
print(not False)
```

False
True



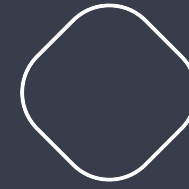
A or B

A와 B 둘중에 하나만 참이어도 참이다.



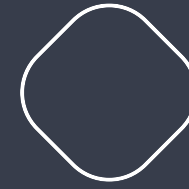
A and B

A와 B 모두 참이어야 참이다.



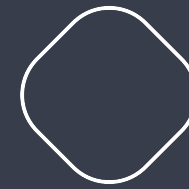
not A

A가 거짓이면 참이다.



x in A

x가 A에 포함되어 있으면 참이다
(A: 리스트, 튜플, 문자열)



x not in A

x가 A에 포함되어 있지 않으면 참이다.

논리 연산자와 비교연산자

```
>>> 10 == 10 and 10 != 5      # True and True
True
>>> 10 > 5 or 10 < 3          # True or False
True
>>> not 10 > 5                 # not True
False
>>> not 1 is 1.0               # not False
True
```

<연산자 우선순위>

비교 연산자로 비교한 결과를 논리 연산자로 다시 판단

이때는 비교 연산자(is, is not, ==, !=, <, >, <=, >=)를 먼저 판단하고 논리 연산자(not, and, or)를 판단

Google Colab

: 구글 colaboratory 서비스의 줄임말

<Colab의 특징>

1

파이썬

파이썬 설치 필요 X

브라우저에서 python을 작성하고
실행 가능합니다.

2

데이터분석

패키지 기본 설치

데이터에 분석 사용되는 Tensor Flow, Keras, matplotlib, scikit-learn, pandas와 같은 패키지가 기본적으로 설치되어있습니다.

3

깃허브

공유 및 협업

깃과 연동이 가능하여 사람들과
협업하여 코딩이 가능합니다.

구글 로그인
검색창에 colab 검색

01

02 코드 입력 가능
왼쪽 상단에서 "새 노트"를 클릭

구글 드라이브에 저장
ctrl+s 혹은 파일 -> 저장을 클릭하여
저장을 해준 뒤 구글 드라이브를 접속

03

04 "Colab Notebooks" 폴더
해당 폴더에 들어가면 저장한 파일 존재

실행 관련 단축키

1. Ctrl + Enter = 해당 셀을 실행하고 커서를 해당 셀에 두는 경우 (결과 값만 보고자 할 때)
2. Shift + Enter = 해당 셀을 실행하고 커서를 다음 셀로 넘기는 경우
(여러가지 값을 빠르게 출력할 때)
3. Alt + Enter = 해당 셀을 실행하고 셀을 삽입한 후 커서를 삽입한 셀로 넘기는 경우
(다음 작업 공간이 없을 때)

* Ctrl + M H = 단축키 모음 (단축키 생각 아닐 때)

셀 삽입/삭제

Ctrl + M A = 코드 셀 위에 삽입

Ctrl + M B = 코드 셀 아래 삽입

Ctrl + M D = 셀 지우기

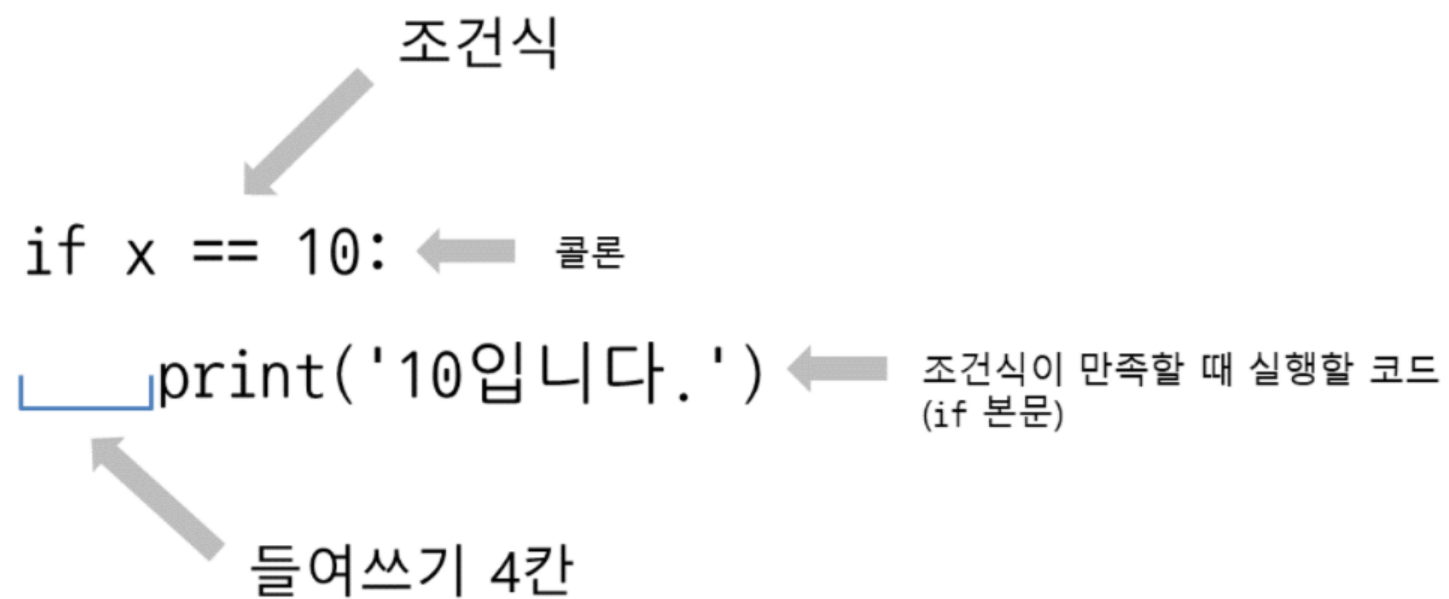
Ctrl + M Y = 코드 셀로 변경

Ctrl + M M = 마크다운 셀로 변경

Ctrl + M Z = 실행 취소

if 조건문

if 조건문은 if에 조건식을 지정하고 :(콜론)을 붙이며 다음 줄에 실행할 코드가 옵니다. 이때 실행할 코드는 반드시 들여쓰기를 해야 합니다.



```
if x == 10:
    print('10입니다.')
```

The diagram shows the following components of the code:

- 조건식** (Condition): Points to the expression `x == 10`.
- 콜론** (Colon): Points to the colon `:` at the end of the first line.
- 조건식이 만족할 때 실행할 코드 (if 본문)** (Code to execute when the condition is satisfied (if body)): Points to the `print('10입니다.')` statement.
- 들여쓰기 4칸** (Indentation 4 spaces): Points to the indentation of the `print` statement.

변수 x에 10을 할당한 뒤 if 조건문으로 x가 10과 같은지 검사하였습니다. 조건식은 `x == 10`과 같은 형식으로 지정해주는데 `==`은 두 값이 "같은 때" 라는 뜻입니다.

즉, `if x == 10:`은 x가 10과 같은지 비교한 뒤 같으면 다음에 오는 코드를 실행하라는 뜻이 됩니다. 따라서 x는 10이고 조건식을 만족하므로 그다음 줄의 `print`가 실행되어 '10입니다.'가 출력됩니다.

조건문 연습문제

```

age = 18
money = 3000
if age <=20 or money >= 4000:    # true
    print("버스 이용가능")

if age <=20 and money >= 4000:    # false
    print("택시 이용가능")

A = [1, 2, 3, 4]
if 2 in A:                        # true
    print("2 is in A!")
if 2 not in A:                    # false
    print("2 not in A")

```

```

S = "I have an apple"

if 'a' in S:                      # true
    print('a is in S!')
if 'ap' in S:                     # true
    print('ap is in S!')

x = 1 > 2                         # x = false
if x:                             # false
    print("True")

if not x:                         # true
    print("false")

```

출력 결과

```

# 버스 이용가능
# 2 is in A!
# a is in S!
# ap is in S!
# false

```

[조건]에 들어가는 문장은 반드시 참 또는 거짓을 판별할 수 있는 문장이 들어가야 한다.
 예를 들어서 $2 < 3$ 과 같은 문장은 항상 참이고, $2 == 3$ 은 항상 거짓이다.
 이렇게 $<$, $==$ 등과 같이 참 또는 거짓을 갖도록 하는 연산자들이 다양하게 있다.

else 사용

if에 else를 사용하면 조건식이 만족할 때와 만족하지 않을 때 각각 다른 코드를 실행할 수 있습니다. 즉, 프로그램이 두 방향으로 분기하는 것이죠.

```
if x == 10:
    print('10입니다.')
else:
    print('10이 아닙니다.')
```

조건식

if x == 10: ← 콜론

print('10입니다.') ← 조건식이 만족할 때 실행할 코드(if 본문)

else: ← 콜론

print('10이 아닙니다.') ← 조건식이 만족하지 않을 때 실행할 코드(else 본문)

들여쓰기 4칸

```
>>> x = 5
>>> if x == 10:
...     print('10입니다.')
... else:
...     print('10이 아닙니다.')
...
10이 아닙니다.
```

else는 if의 조건식이 만족하지 않을 때 코드를 실행합니다. 여기서는 x에 5가 들어있어서 x == 10을 만족하지 않으므로 else의 print가 실행되어 '10이 아닙니다.'가 출력됩니다.

A 기업의 입사 시험은 필기 시험 점수가 80점 이상이면서 코딩 시험을 통과해야 합격이라고 정했다.
(코딩 시험 통과 여부는 True, False로 구분).

```
written_test = 75
coding_test = True

if written_test >= 80 and coding_test == True:
    print('합격')
else:
    print('불합격')
```

합격 조건이 필기 시험 점수가 80점 이상이면서 코딩 시험을 통과해야 한다고 했으므로 두 조건이 모두 만족해야 합니다. 따라서 if 조건문을 만들고 `written_test >= 80 and coding_test == True`와 같이 필기 시험 점수가 80점 이상인지 검사하는 조건식과 코딩 시험이 통과했는지 판단하는 조건식을 만듭니다. 여기서 필기 시험 점수, 코딩 시험 통과 여부가 모두 만족해야 하므로 `and` 연산자로 연결해줍니다.

elif 사용

elif는 else인 상태에서 조건식을 지정할 때 사용하며 else if라는 뜻입니다. 물론 if, else와 마찬가지로 조건식 끝에 :(콜론)을 붙여야 하고, elif 단독으로 사용할 수 없습니다.

```
x = 30

if x == 10:           # x가 10일 때
    print('10입니다.')
elif x == 20:         # x가 20일 때
    print('20입니다.')
else:                 # 앞의 조건식에 모두 만족하지 않을 때
    print('10도 20도 아닙니다.')
```

if, elif의 조건식이 모두 거짓일 때만 else의 코드가 실행됩니다. 여기서는 x가 30이라 if, elif의 조건식에 모두 만족하지 않습니다. 따라서 마지막 else의 '10도 20도 아닙니다.'가 출력됩니다.

실행 결과

10도 20도 아닙니다.

```
a = 10
b = 20

if a > b:
    print('a is greater than b')
elif a == b:
    print('a is equal to b')
elif a < b:
    print('a is less than b')
else:
    print("error")
```

$a > b$ 는 false이므로 다음 elif의 조건인 $a == b$ 를 확인하고, $a == b$ 도 false이므로 $a < b$ 를 확인한다. 이때, $a < b$ 는 true이므로 `print('a is less than b')`를 출력하고 조건문에서 빠져나온다.

for와 range 사용

for 변수 in range(횟수):
반복할 코드

for 반복문은 range에 반복할 횟수를 지정하고 앞에 in과 변수를 입력합니다. 그리고 끝에 :(콜론)을 붙인 뒤 다음 줄에 반복할 코드를 넣습니다.

```
for i in range(100):  
    print('Hello, world!')
```

숫자를 하나씩 꺼냄

숫자 100개 생성
0, 1, 2, 3, 4 ... 97, 98, 99

숫자를 꺼낼 때마다
코드 실행

range(100)과 같이 지정하면 0부터 99까지 숫자 100개를 생성합니다. 그리고 for는 in으로 숫자를 하나씩 꺼내서 변수 i에 저장하고, print를 실행합니다. 즉, range(100)에서 숫자를 100번 꺼내면서 print를 실행하므로 'Hello, world!'가 100번 출력되는 것이죠. 이처럼 for 반복문은 반복 횟수가 정해져 있을 때 주로 사용합니다.

1. 시작하는 숫자와 끝나는 숫자 지정

```
>>> for i in range(5, 12):    # 5부터 11까지 반복
...     print('Hello, world!', i)
...
Hello, world! 5
Hello, world! 6
Hello, world! 7
Hello, world! 8
Hello, world! 9
Hello, world! 10
Hello, world! 11
```

- `for 변수 in range(시작, 끝):`

2. 증가폭 사용

```
>>> for i in range(0, 10, 2):    # 0부터 8까지 2씩 증가
...     print('Hello, world!', i)
...
Hello, world! 0
Hello, world! 2
Hello, world! 4
Hello, world! 6
Hello, world! 8
```

- `for 변수 in range(시작, 끝, 증가폭):`

3. 숫자 순서 뒤집기

```
>>> for i in reversed(range(10)):
...     print('Hello, world!', i)
...
Hello, world! 9
Hello, world! 8
Hello, world! 7
... (생략)
Hello, world! 1
Hello, world! 0
```

- for 변수 in reversed(range(횟수))
- for 변수 in reversed(range(시작, 끝))
- for 변수 in reversed(range(시작, 끝, 증가폭))

4. 횟수대로 반복하기

```
count = int(input('반복할 횟수를 입력하세요: '))

for i in range(count):
    print('Hello, world!', i)
```

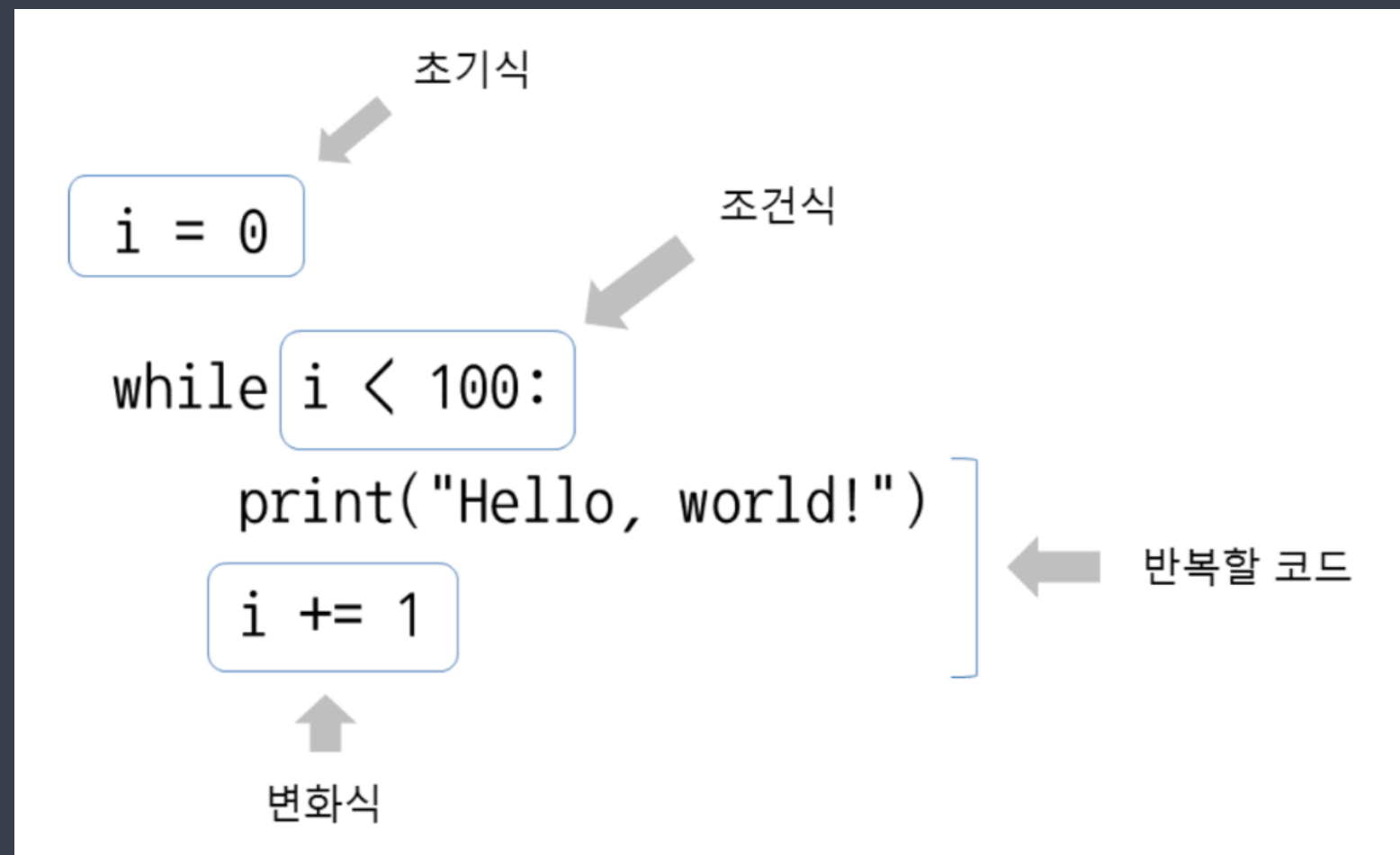
```
반복할 횟수를 입력하세요: 3 (입력)
Hello, world! 0
Hello, world! 1
Hello, world! 2
```

while 반복문 사용

초기식

while 조건식 :
반복할 코드
변화식

while 반복문은 조건식을 지정하고 끝에 :(콜론)을 붙인 뒤 다음 줄에 반복할 코드와 변화식을 넣습니다. 초기식은 특별한 것이 없고 보통 변수에 값을 저장하는 코드입니다.



$i < 100$ 과 같이 조건식을 지정하여 i 가 100 미만일 때만 반복하고, i 가 100이 되면 반복을 끝내도록 만들었습니다. 그리고 반복할 코드의 변화식에는 $i += 1$ 로 i 를 1씩 증가시켰으므로 i 가 0부터 99까지 증가하면서 100번 반복하게 됩니다. 물론 변화식 $i += 1$ 을 풀어서 $i = i + 1$ 로 만들어도 동작은 같습니다.

1. 초깃값 1부터 시작

```
>>> i = 1
>>> while i <= 100:
...     print('Hello, world!', i)
...     i += 1
...
Hello, world! 1
Hello, world! 2
Hello, world! 3
... (생략)
Hello, world! 99
Hello, world! 100
```

2. 초깃값 감소시키기

```
>>> i = 100
>>> while i > 0:
...     print('Hello, world!', i)
...     i -= 1
...
Hello, world! 100
Hello, world! 99
Hello, world! 98
... (생략)
Hello, world! 2
Hello, world! 1
```


3. 입력한 횟수대로 반복하기

```
count = int(input('반복할 횟수를 입력하세요: '))

i = 0
while i < count:      # i가 count보다 작을 때 반복
    print('Hello, world!', i)
    i += 1
```

```
반복할 횟수를 입력하세요: 3 (입력)
Hello, world! 0
Hello, world! 1
Hello, world! 2
```

4. 입력받은 초깃값만큼 출력

```
count = int(input('반복할 횟수를 입력하세요: '))

while count > 0:      # count가 0보다 클 때 반복
    print('Hello, world!', count)
    count -= 1         # count를 1씩 감소시킴
```

```
반복할 횟수를 입력하세요: 3 (입력)
Hello, world! 3
Hello, world! 2
Hello, world! 1
```

Thank you

강의 열심히 들어주셔서 감사합니다 :)



강연자 이름
남궁규민



중앙대학교
산업보안학과 교육봉사동아리

23/02/09 THUR