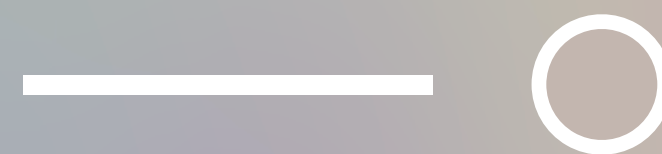


AI TRACK

# PYTHON 기초



조건문 — 0.1

반복문 — 0.2



강연자 이름  
남궁규민



중앙대학교  
산업보안학과 교육봉사동아리

23/02/08 Wed

# 목차

---

01 파이썬

02 Colab 사용법

03 PYTHON 기초

04 조건문의 활용

05 반복문의 활용

06 과제

# 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



# 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

구글 드라이브에 저장 03  
ctrl+s 혹은 파일 -> 저장을 클릭하여  
저장을 해준 뒤 구글 드라이브를 접속

02 코드 입력 가능  
왼쪽 상단에서 "새 노트"를 클릭

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

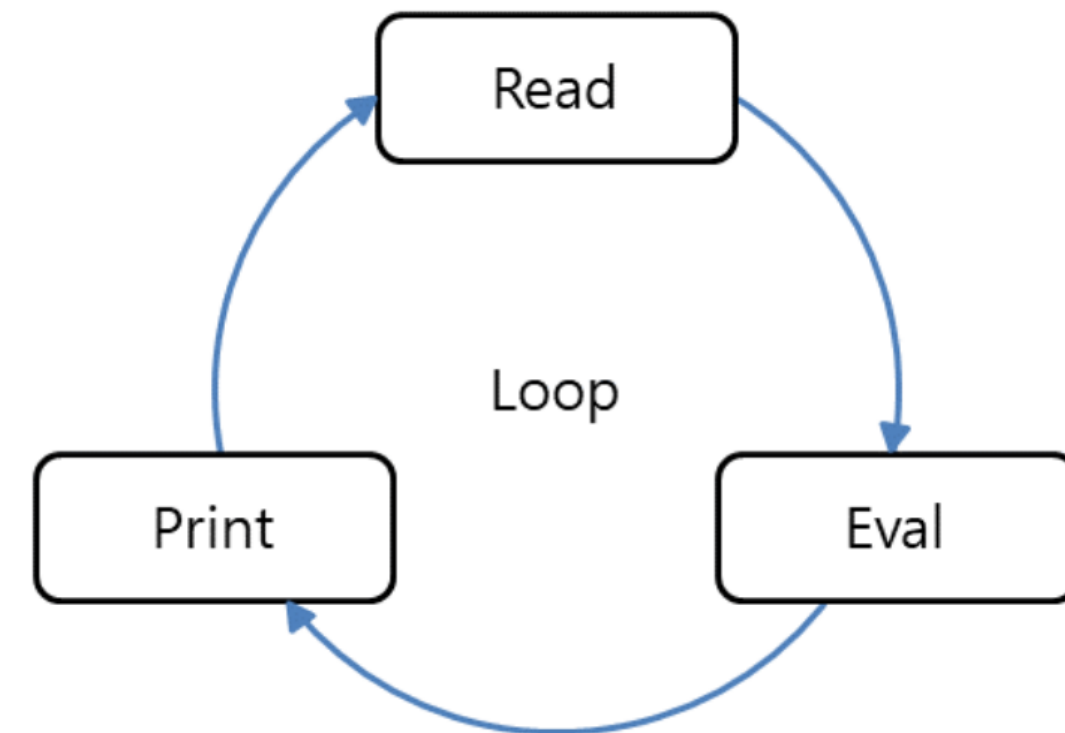
Ctrl + M M = 마크다운 셀로 변경

Ctrl + M Z = 실행 취소

# 출력

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

=> Hello, world! 출력



# 입력

키보드로 문자열을 입력 받을 수 있음

입력된 문자열을 = 를 사용해 변수에 저장 가능

input()으로 입력받은 데이터는 항상 문자열(str) 타입

계산을 하려면 정수(int) 변환 필요

```
name = input("당신의 이름은 무엇입니까? : ")  
print(name, "씨 반가워요. ")
```



# 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 숫자 사용
2. 대소문자 구분
3. 숫자부터 시작 불가
4. \_(밑줄 문자)로 시작 가능
5. 특수 문자(+, -, \*, /, \$, @, &, % 등) 사용 불가
6. 파이썬의 키워드(if, for, while, and, or 등)는 사용 불가

# 산술연산자

산술 연산자	설명	사용 예	예 설명
=	대입 연산자	$a = 3$	정수 3을 a에 대입
+	더하기	$a = 5 + 3$	5와 3을 더한 값을 a에 대입
-	빼기	$a = 5 - 3$	5에서 3을 뺀 값을 a에 대입
*	곱하기	$a = 5 * 3$	5와 3을 곱한 값을 a에 대입
/	나누기	$a = 5 / 3$	5를 3으로 나눈 값을 a에 대입
//	나누기(몫)	$a = 5 // 3$	5를 3으로 나눈 뒤 소수점을 버리고 a에 대입
%	나머지 값	$a = 5 \% 3$	5를 3으로 나눈 뒤 나머지 값을 a에 대입
**	제곱	$a = 5 ** 3$	5의 3제곱을 a에 대입

결과 확인

# 관계연산자

두 값을 비교하는 연산자  
결과는 참(True)나 거짓(False)

관계 연산자	의미	설명
==	같다	두 값이 동일하면 참
!=	같지 않다	두 값이 다르면 참
>	크다	왼쪽이 크면 참
<	작다	왼쪽이 작으면 참
>=	크거나 같다	왼쪽이 크거나 같으면 참
<=	작거나 같다	왼쪽이 작거나 같으면 참

# 관계연산자

두 값을 비교하는 연산자  
결과는 참(True)나 거짓(False)

관계 연산자	의미	설명
==	같다	두 값이 동일하면 참
!=	같지 않다	두 값이 다르면 참
>	크다	왼쪽이 크면 참
<	작다	왼쪽이 작으면 참
>=	크거나 같다	왼쪽이 크거나 같으면 참
<=	작거나 같다	왼쪽이 작거나 같으면 참

# 논리연산자

---

```
a=100  
b=200  
print(a==b, a!=b, a>b, a<b, a>=b, a<=b)
```

코드 결과 확인

# 논리연산자와 관계연산자

```
>>> 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>

관계 연산자로 비교한 결과를 논리 연산자로 다시 판단

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

# 리스트

데이터를 일렬로 연결해서 대괄호[]로 묶어놓은 컨테이너 객체

리스트 내의 데이터는 쉼표로 구분

리스트 내의 데이터에 접근할 때는 0부터 시작하는 번호를 사용하는 데 이것을 인덱스(index)라고 함

리스트이름[인덱스 번호]의 형태로 데이터에 접근 가능

```
리스트이름 = [값1, 값2, 값3, ...]
```

```
aa = [ 10, 20, 30, 40 ]
```

aa



aa[0]

aa[1]

aa[2]

aa[3]

# 리스트

random 모듈을 이용하여, 리스트 내에서 임의의 값을 선택해 출력해보자

```
import random
한식 = ['불고기', '제육볶음', '닭볶음탕', '김치찌개', '해물탕']
print("오늘의 추천음식 : %s" % 한식[random.randint(0, 4)])
```



# if 조건문

if에 조건식을 지정하고

:(콜론)을 붙이며 실행할 코드 작성(들여쓰기 필수)

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

The diagram shows the components of the if statement: '조건식' (Condition) points to 'x == 10'; '콜론' (Colon) points to the ':' after '10'; '들여쓰기 4칸' (Indentation 4 spaces) points to the indentation of the 'print' line; and '조건식이 만족할 때 실행할 코드 (if 본문)' (Code to execute when condition is satisfied (if body)) points to the 'print' line.

변수 x에 숫자를 할당한 뒤 코드 실행

=가 아닌 == 항상 주의!

```
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

# else 사용

if에 else를 사용하면 조건식이 만족할 때와 만족하지 않을 때 각각 다른 코드 실행 가능

```
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

변수 x에 숫자를 할당한 뒤 코드 실행

=가 아닌 == 항상 주의!

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')
```

숫자를 입력받고 짝수인지 홀수인지 구분하여 출력해보기

숫자를 입력하세요: 35  
홀수입니다.

숫자를 입력하세요: 100  
짝수입니다.

힌트: 입력받은 값은 항상 문자열 타입

숫자를 입력받고 짝수인지 홀수인지 구분하여 출력해보기

답안

```
num = int(input("숫자를 입력하세요: "))

if num%2 == 0 :
    print("짝수입니다.")

if num%2 == 1 :
    print("홀수입니다.")
```

# elif 사용

elif는 else인 상태에서 조건식을 지정할 때 사용, else if라는 뜻  
if, else와 마찬가지로 조건식 끝에 :(콜론)을 붙여야 하고, elif 단독으로 사용 불가

```
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 실행

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")
```

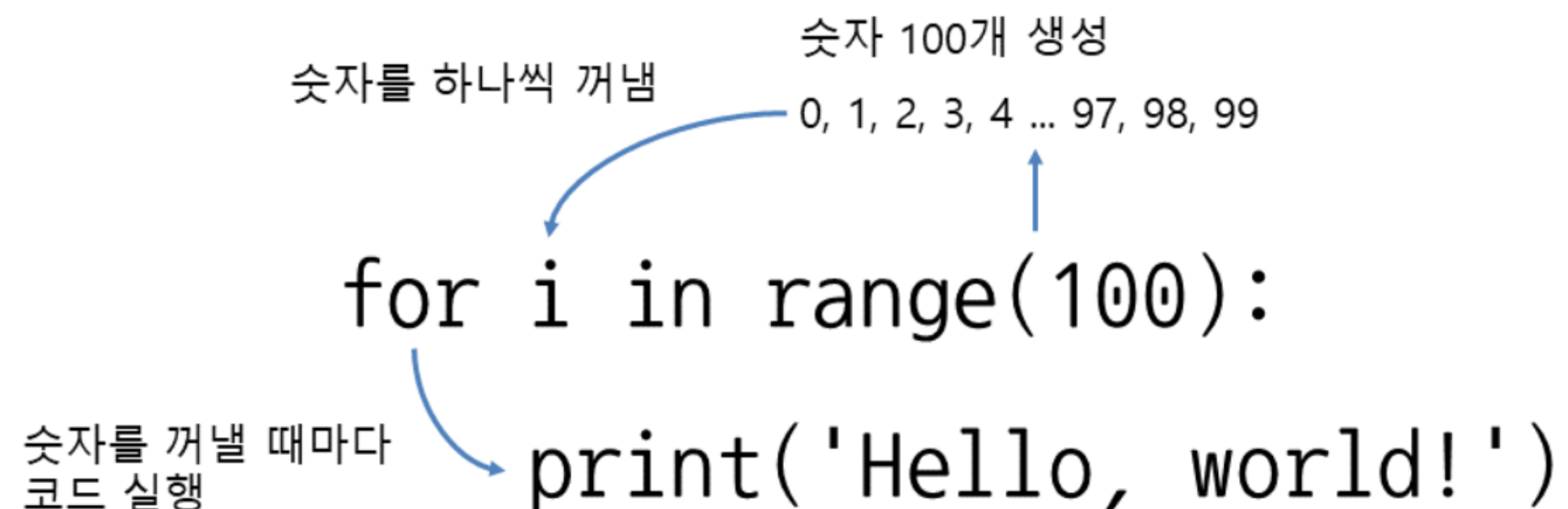
$a > b$ 는 false이므로 다음 elif의 조건인  $a == b$ 를 확인하고,  $a == b$ 도 false이므로  $a < b$ 를 확인한다. 이때,  $a < b$ 는 true이므로 `print('a is less than b')`를 출력하고 조건문에서 빠져나온다.



# for와 range 사용

for 반복문은 range에 반복할 횟수를 지정하고 앞에 in과 변수를 입력  
:(콜론)을 붙인 뒤 다음 줄에 반복할 코드 작성

```
for 변수 in range(횟수):  
    반복할 코드
```



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

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

숫자를 하나씩 꺼냄

숫자를 꺼낼 때마다  
코드 실행

## 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
```

2002~2100년까지 월드컵 개최년도 출력하기

2002  
2006  
2010  
2014  
2018  
2022  
2026  
.  
.  
.  
2082  
2086  
2090  
2094  
2098

2002~2100년까지 월드컵 개최년도 출력하기

답안

```
for i in range(2002, 2101, 4):  
    print(i)
```

range를 사용하지 않고도 데이터 집합을 순회하며 반복 처리 가능  
데이터집합: 리스트와 같이 여러 데이터를 담고있는 iterable 객체

```
for i in ["a", "b", "c", "d"]:  
    print(i)
```



a  
b  
c  
d

```
a = "hello"  
for i in a:  
    print(i)
```



h  
e  
l  
l  
o

문장을 입력받고, 글자 하나를 추가로 입력받아  
문자 내에 해당 글자가 몇 개인지 출력하기

문장을 입력하세요:가가 가가  
어떤 글자를 찾을까요?:가  
입력한 문장에서 가의 갯수는 4개 입니다.

마지막 코드: `print("입력한 문장에서 %s의 갯수는 %d개 입니다." %(글자, count))`

이때 %s는 문자열 타입에 대한 서식 표현, %d는 정수 타입에 대한 서식 표현

문장을 입력받고, 글자 하나를 추가로 입력받아  
문자 내에 해당 글자가 몇 개인지 출력하기

답안

```
문장 = input("문장을 입력하세요:")
글자 = input("어떤 글자를 찾을까요?:")

count = 0
for i in 문장:
    if i == 글자 :
        count += 1

print("입력한 문장에서 %s의 갯수는 %d개 입니다." % (글자, count))
```



# while 반복문 사용

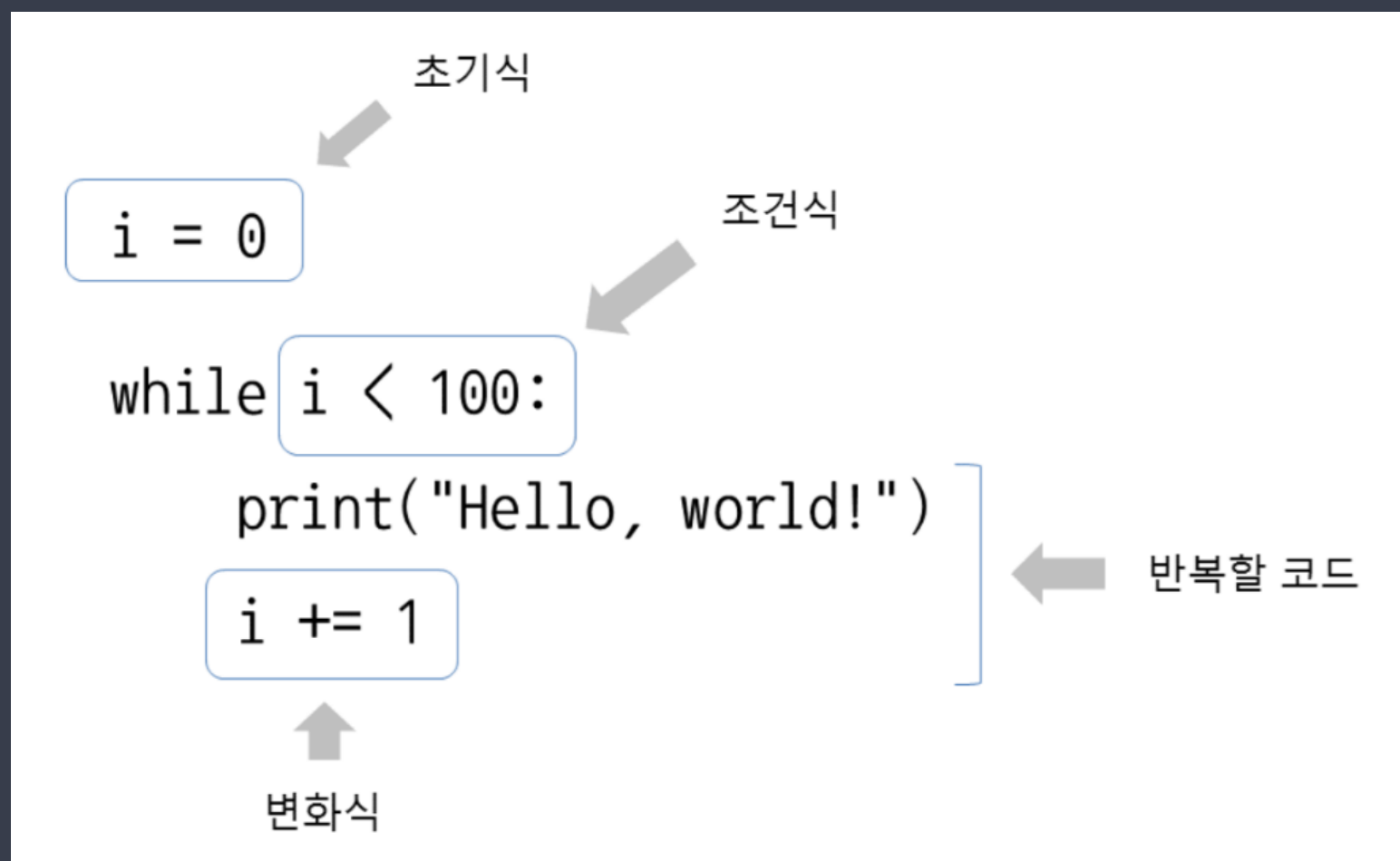
while 반복문은 조건식을 지정하고  
:(콜론)을 붙인 뒤 다음 줄에 반복할 코드와 변화식 작성

초기식

**while** 조건식:

반복할 코드

변화식



초기식과 변화식 필수!

for 반복문은 지정한 범주만큼만 반복

while 반복문은 조건식을 확인하여 값이 참이면 내부 코드를 실행(if와 유사), 이를 반복

## 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
```

2단 출력하기를 수정하여 입력한 숫자에 대한 구구단 출력하기

```
num = 1  
  
while num < 10:  
    print("2 * %d = %d" % (num, 2*num))  
    num = num+1
```

2단 출력하기

몇 단을 출력할까요? 7

7	*	1	=	7
7	*	2	=	14
7	*	3	=	21
7	*	4	=	28
7	*	5	=	35
7	*	6	=	42
7	*	7	=	49
7	*	8	=	56
7	*	9	=	63

2단 출력하기를 수정하여 입력한 숫자에 대한 구구단 출력하기

답안

```
dan = int(input("몇 단을 출력할까요? "))  
  
num = 1  
  
while num<10:  
    print("%d * %d = %d" % (dan, num, dan*num))  
    num = num + 1
```

# Thank you

강의 열심히 들어주셔서 감사합니다 :)



강연자 이름  
남궁규민



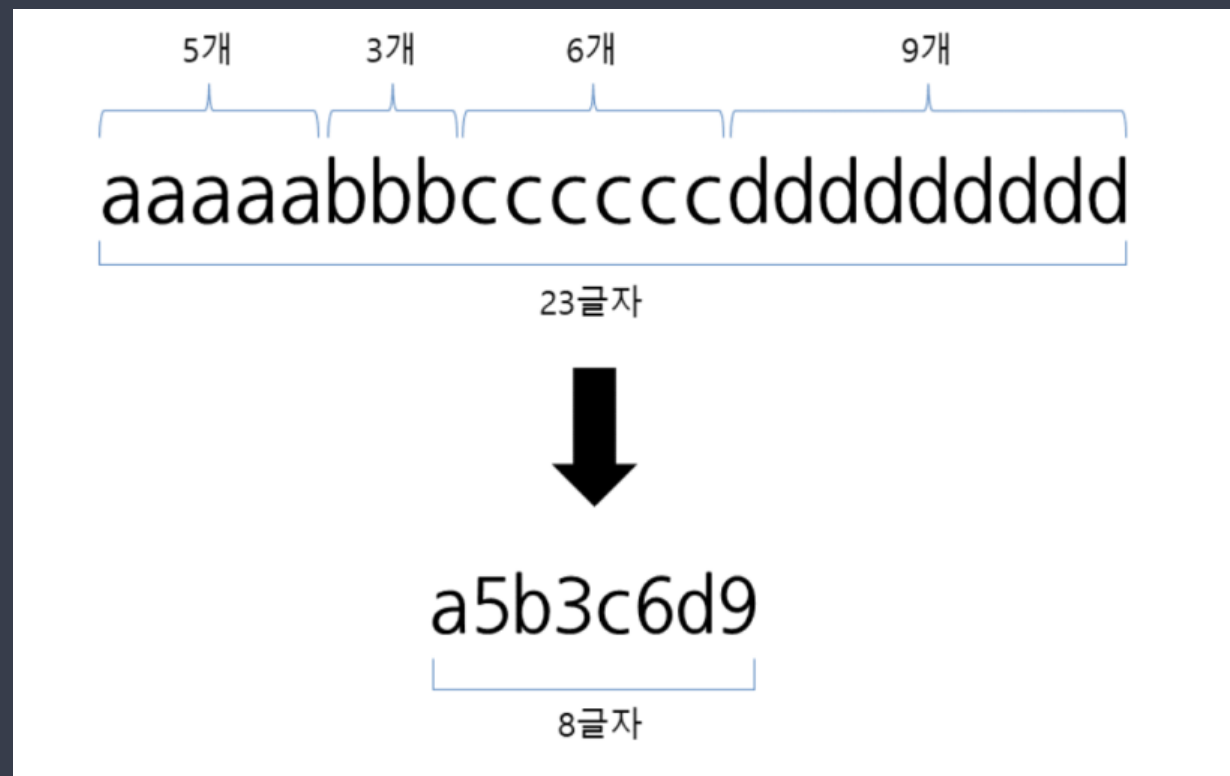
중앙대학교  
산업보안학과 교육봉사동아리

23/02/09 THUR

# 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 = 'aaaaabbbbccccccddddddddd'
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)
```