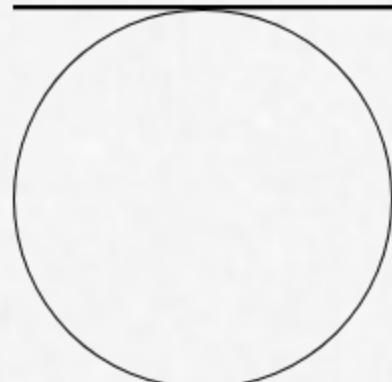
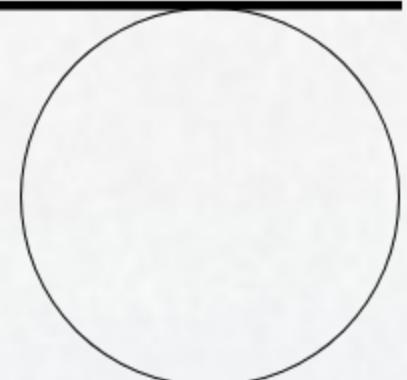
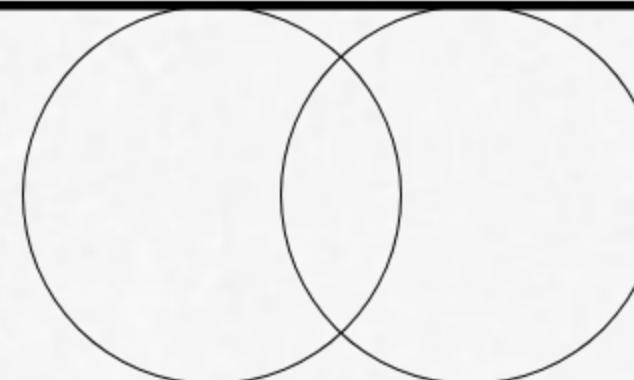


2023.07.13

---

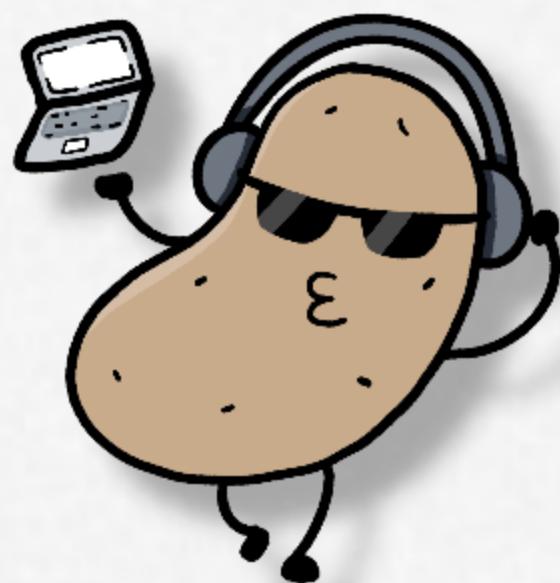
# 코딩공감자

---



# SELF INTRODUCTION

---



코딩공감자

# 지난 시간 복습 Quiz ✨



자료형에는 어떤 것이 있을까?

- 
- 
-

# 지난 시간 복습 Quiz ✨



자료형에는 어떤 것이 있을까?

- 
- 
- 

1. 숫자형
2. 문자열 자료형
3. 리스트 자료형
4. 튜플 자료형
5. 집합 자료형
6. 불 자료형
7. 딕셔너리 자료형

# con

01 — 자료형

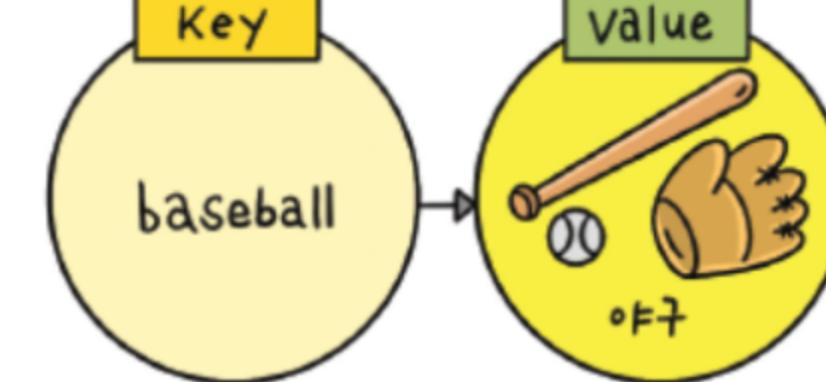
02 — 제어문

# tents



# 7. 딕셔너리 자료형

key 값을 통해 value를 얻음



```
>>> dic = {'name': 'pey', 'phone': '010-9999-1234', 'birth': '1118'}
```

key	value
name	pey
phone	010-9999-1234
birth	1118

# 7. 딕셔너리 자료형

key 값을 통해 value를 얻음



- 딕셔너리에 쌍 추가
- 
- 

```
>>> a = {1: 'a'}  
>>> a[2] = 'b'  
>>> a  
{1: 'a', 2: 'b'}
```

```
>>> a[3] = [1, 2, 3]  
>>> a  
{1: 'a', 2: 'b', 'name': 'pey', 3: [1, 2, 3]}
```

딕셔너리 요소 삭제

```
>>> del a[1]  
>>> a  
{2: 'b', 'name': 'pey', 3: [1, 2, 3]}
```

# 7. 딕셔너리 자료형



## 딕셔너리 사용 방법

- 
- 
- 

```
>>> grade = {'pey': 10, 'julliet': 99}  
>>> grade['pey']  
10  
>>> grade['julliet']  
99
```

```
>>> dic = {'name':'pey', 'phone':'010-9999-1234', 'birth': '1118'}  
>>> dic['name']  
'pey'  
>>> dic['phone']  
'010-9999-1234'  
>>> dic['birth']  
'1118'
```

# 제어문



프로그램의 흐름을 제어하는 경우에 사용하는 실행



1. if 문
2. while 문
3. for 문



# 1. if 문

if 와 else를 사용한 조건문의 기본 구조



if 조건문:

수행할\_문장1

수행할\_문장2

...

else:

수행할\_문장A

수행할\_문장B

...

- 참이면 if 블록 수행  
거짓이면 else 블록 수행

- 들여쓰기 필수!  
안해도, 더해도 ERROR

- 콜론(:) 필수

# 1. if 문



조건문 : 참과 거짓을 판단하는 문장

- 
- 
- 

비교연산자	설명
$x < y$	$x$ 가 $y$ 보다 작다.
$x > y$	$x$ 가 $y$ 보다 크다.
$x == y$	$x$ 와 $y$ 가 같다.
$x != y$	$x$ 와 $y$ 가 같지 않다.
$x >= y$	$x$ 가 $y$ 보다 크거나 같다.
$x <= y$	$x$ 가 $y$ 보다 작거나 같다.

```
>>> x = 3
```

```
>>> y = 2
```

```
>>> x > y
```

```
True
```

```
>>>
```

# 1. if 문 실습



만약 3000원 이상의 돈을 가지고 있으면 택시를 타고, 그렇지 않으면 걸어가라.

```
>>> money = 2000
>>> if money >= 3000:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
걸어가라
>>>
```



# 1. if 문

and, or, not



연산자	설명
x or y	x와 y 둘 중 하나만 참이어도 참이다.
x and y	x와 y 모두 참이어야 참이다.
not x	x가 거짓이면 참이다.

# 1. if 문 실습 Quiz



만약 3000원 이상의 돈을 가지고 있거나 카드가 있으면 택시를 타고, 그렇지 않으면 걸어가라.



연산자	설명
x or y	x와 y 둘 중 하나만 참이어도 참이다.
x and y	x와 y 모두 참이어야 참이다.
not x	x가 거짓이면 참이다.

```
>>> money = 2000
>>> if money >= 3000:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
걸어가라
>>>
```

# 1. if 문 실습 Quiz 정답



만약 3000원 이상의 돈을 가지고 **있거나 카드가 있으면** 택시를 타고, 그렇지 않으면 걸어가라.



```
>>> money = 2000
>>> card = True
>>> if money >= 3000 or card:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
```

...

택시를 타고 가라

```
>>>
```



# 1. if 문

## in, not in

- 
- 
- 

in	not in
x in 리스트	x not in 리스트
x in 튜플	x not in 튜플
x in 문자열	x not in 문자열

```
>>> 1 in [1, 2, 3]
```

```
True
```

```
>>> 1 not in [1, 2, 3]
```

```
False
```

```
>>> 'a' in ('a', 'b', 'c')
```

```
True
```

```
>>> 'j' not in 'python'
```

```
True
```

# 1. if 문 실습 Quiz



만약 ~~3000원 이상의~~ 돈을 가지고 ~~있거나 카드가~~ 있으면 택시를 타고, 그렇지 않으면 걸어가라.

in	not in
x in 리스트	x not in 리스트
x in 튜플	x not in 튜플
x in 문자열	x not in 문자열

```
>>> money = 2000
>>> card = True
>>> if money >= 3000 or card:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
>>>
```

# 1. if 문 실습 Quiz 정답



만약 ~~3000원 이상의~~ 돈을 가지고 ~~있거나 카드가~~ 있으면 택시를 타고, 그렇지 않으면 걸어가라.



```
>>> pocket = ['paper', 'cellphone', 'money']
>>> if 'money' in pocket:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
>>>
```

# 1. if 문



## elif 문

- 
- 
- 

- 조건을 판단하는 부분이 여러군데인 경우 이용
- 다중 조건 판단 가능
- 개수에 제한 없이 사용 가능

**if** 조건문:

수행할\_문장**1**

수행할\_문장**2**

...

**elif** 조건문:

수행할\_문장**1**

수행할\_문장**2**

...

**elif** 조건문:

수행할\_문장**1**

수행할\_문장**2**

...

...

**else:**

수행할\_문장**1**

수행할\_문장**2**

...

# 1. if 문 실습 Quiz



만약 돈을 가지고 있으면 택시를 타고, 돈은 없지만 카드가 있으면 택시를 타고 가고, 둘 다 그렇지 않으면 걸어가라.

```
>>> money = 2000  
>>> card = True  
>>> if money >= 3000 or card:  
...     print("택시를 타고 가라")  
... else:  
...     print("걸어가라")  
...  
택시를 타고 가라
```

```
>>>
```

# 1. if 문 실습 Quiz 정답



만약 돈을 가지고 있으면 택시를 타고, 돈은 없지만 카드가 있으면 택시를 타고 가고, 둘 다 그렇지 않으면 걸어가라.



```
>>> pocket = ['paper', 'cellphone']
>>> card = True
>>> if 'money' in pocket:
...     print("택시를 타고가라")
... else:
...     if card:
...         print("택시를 타고가라")
...     else:
...         print("걸어가라")
...
...
택시를 타고가라
```

&gt;&gt;&gt;

```
>>> pocket = ['paper', 'cellphone']
>>> card = True
>>> if 'money' in pocket:
...     print("택시를 타고가라")
... elif card:
...     print("택시를 타고가라")
... else:
...     print("걸어가라")
...
...
택시를 타고가라
```

# while 문?

- 문장을 반복해서 수행해야 할 경우에 while문을 사용한다!
- while 문은 조건문이 참인 동안 while 문에 속한 문장들이 반복해서 수행

**while** 조건문:

수행할\_문장1

수행할\_문장2

수행할\_문장3

...

# while 문 실습

'열 번 찍어 안 넘어가는 나무 없다'라는 속담을 파이썬 프로그램으로 실습하기

```
>>> treeHit = 0
>>> while treeHit < 10:
...     treeHit = treeHit +1
...     print("나무를 %d번 찍었습니다." % treeHit)
...     if treeHit == 10:
...         print("나무 넘어갑니다.")
...
...
```

# while 문 강제로 빠져나가기(1)

## break 문 사용하여 강제로 while 문 빠져나가기

- 
- 
- 

"자판기 안에 커피가 충분히 있을 때, 동전을 넣으면  
커피가 나온다. 그런데, 만약 커피가 떨어졌다면  
판매를 중단하고 '판매 중지'문구를 사용자에게  
보여주어야 한다. 이렇게 판매를 강제로 멈추게  
하는 것이 바로 break 문이다"



```
>>> coffee = 10
>>> money = 300
>>> while money:
...     print("돈을 받았으니 커피를 줍니다.")
...     coffee = coffee -1
...     print("남은 커피의 양은 %d개입니다." % coffee)
...     if coffee == 0:
...         print("커피가 다 떨어졌습니다. 판매를 중지합니다.")
...         break
...
```

# while 문 강제로 빠져나가기 (2)

실제 자판기는? 커피 값보다 더 큰 금액의 돈을 넣으면,  
커피를 주는 동시에 거스름돈을 돌려준다!

```
# coffee.py
coffee = 10
while True:
    money = int(input("돈을 넣어 주세요: "))
    if money == 300:
        print("커피를 줍니다.")
        coffee = coffee -1
    elif money > 300:
        print("거스름돈 %d를 주고 커피를 줍니다." % (money -300))
        coffee = coffee -1
    else:
        print("돈을 다시 돌려주고 커피를 주지 않습니다.")
        print("남은 커피의 양은 %d개 입니다." % coffee)
    if coffee == 0:
        print("커피가 다 떨어졌습니다. 판매를 중지 합니다.")
        break
```



# while 맨 처음으로 돌아가기(1)

while 문 안의 문장을 수행할 때 입력 조건을 검사해서 조건에 맞지 않으면 while 문을 빠져 나간다. 그런데, while 문을 빠져나가지 않고 while문의 맨 처음 (조건문)으로 다시 돌아가게 만들고 싶은 경우에는 어떻게 해야할까?

--> continue 문 사용



# while 맨 처음으로 돌아가기(2)

- while 문을 사용하여
- 1부터 10까지의 숫자 중에서 홀수만 출력하기
- 힌트) continue문 사용!!



```
>>> a = 0
>>> while a < 10:
...     a = a + 1
...     if a % 2 == 0: continue
...     print(a)
...
1
3
5
7
9
```

# 무한 루프

무한 루프(endless loop) : 무한히 반복한다  
-> 파이썬에서 무한 루프는 while 문으로 구현 가능

무한 루프의 기본 형태?

while True:

수행할\_문장1

수행할\_문장2

...

# 무한 루프의 예

```
>>> while True:  
...     print("Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.")
```

...

Ctrl+C를 눌러야 **while**문을 빠져나갈 수 있습니다.

Ctrl+C를 눌러야 **while**문을 빠져나갈 수 있습니다.

Ctrl+C를 눌러야 **while**문을 빠져나갈 수 있습니다.

....

**위 문장은 영원히 출력된다!!**



# For 문?

- while 문과 비슷한 반복문인 for 문은 문장구조가 한눈에 들어온다는 장점이 있음

**for** 변수 **in** 리스트(또는 튜플, 문자열) :

수행할\_문장**1**

수행할\_문장**2**

...

# 예제를 통해 for 문 이해하기 (1)

## 전형적인 for 문

```
>>> test_list = ['one', 'two', 'three']  
•  
•  
•  
>>> for i in test_list:  
...     print(i)  
...  
one  
two  
three
```

['one', 'two', 'three'] 리스트의 첫 번째 요소인 'one'이 먼저 i 변수에 대입된 후 print(i) 문장을 수행한다. 다음에 두 번째 요소 'two'가 i 변수에 대입된 후 print(i) 문장을 수행하고 리스트의 마지막 요소까지 이것을 반복한다.

# 예제를 통해 for 문 이해하기 (2)

## 다양한 for 문의 사용

```
>>> a = [(1,2), (3,4), (5,6)]  
>>> for (first, last) in a:  
...     print(first + last)  
...  
3  
7  
11
```



a 리스트의 요솟값이 튜플이기 때문에 각각의 요소가 자동으로 (first, last) 변수에 대입된다.

# 예제를 통해 for 문 이해하기 (3)

## for 문의 응용

시험 점수가 60점 이상이면 합격이고 그렇지 않으면 불합격  
이때, 학생 5명의 시험점수에 대해서 합격인지, 불합격인지 결과를 알아보자

```
# marks1.py
marks = [90, 25, 67, 45, 80]      # 학생들의 시험 점수 리스트

number = 0      # 학생에게 붙여 줄 번호
for mark in marks:    # 90, 25, 67, 45, 80을 순서대로 mark에 대입
    number = number +1
    if mark >= 60:
        print("%d번 학생은 합격입니다." % number)
    else:
        print("%d번 학생은 불합격입니다." % number)
```

# for 문과 continue 문

**for 문 안의 문장을 수행하는 도중 continue 문을 만나면 for 문의 처음으로!  
60점 이상인 사람에게는 축하 메시지를 보내고  
나머지 사람에게는 아무런 메시지도 전하지 않는 프로그램**

```
# marks2.py
marks = [90, 25, 67, 45, 80]

number = 0
for mark in marks:
    number = number +1
    if mark < 60:
        continue
    print("%d번 학생 축하합니다. 합격입니다. " % number)
```

점수가 60점 이하인 학생인 경우에는 mark < 60이 참이 되어 **continue** 문이 수행된다. 따라서 축하 메시지를 출력하는 부분인 **print** 문을 수행하지 않고 **for** 문의 처음으로 돌아가게 된다.

C:\doit>python marks2.py

1번 학생 축하합니다. 합격입니다.

3번 학생 축하합니다. 합격입니다.

5번 학생 축하합니다. 합격입니다.

# for 문과 함께 자주 사용하는 range 함수

for 문은 숫자 리스트를 자동으로 만들어 주는 range 함수와 함께 사용하는 경우가 많다!

```
>>> a = range(10)  
  
>>> a  
range(0, 10)
```

range(10)은 0부터 10 미만의 숫자를 포함하는 range 객체를 만들어 준다.  
시작 숫자와 끝 숫자를 지정하려면 range(시작\_숫자, 끝\_숫자) 형태를 사용하는데,  
이때 끝 숫자는 포함되지 않는다.



# range 함수의 예시 살펴보기

for와 range 함수를 사용하면 1부터 10까지 더하는 것을 다음과 같이 쉽게 구현할 수 있다!

```
>>> add = 0  
:  
>>> for i in range(1, 11):  
...     add = add + i  
...  
>>> print(add)  
55
```



range(1, 11)은 숫자 1부터 10까지(1 이상 11 미만)의 숫자를 데이터로 가지는 객체이다.  
따라서 위 예에서 i 변수에 숫자가 1부터 10까지 하나씩 차례로 대입되면서  
add = add + i 문장을 반복적으로 수행하고 add는 최종적으로 55가 된다.

# for와 range를 이용한 구구단

```
>>> for i in range(2,10):          # 1번 for문  
...     for j in range(1, 10):      # 2번 for문  
...         print(i*j, end=" ")  
...     print()  
...  
2 4 6 8 10 12 14 16 18  
3 6 9 12 15 18 21 24 27  
4 8 12 16 20 24 28 32 36  
5 10 15 20 25 30 35 40 45  
6 12 18 24 30 36 42 48 54  
7 14 21 28 35 42 49 56 63  
8 16 24 32 40 48 56 64 72  
9 18 27 36 45 54 63 72 81
```

# for와 range를 이용한 구구단

- 1번 for 문에서 2부터 9까지의 숫자가 차례대로 i에 대입된다. i가 처음 2일 때 2번 for 문을 만나게 된다.
- 2번 for 문에서 1부터 9까지의 숫자가 j에 대입되고 그 다음 문장인 `print(i*j, end=" ")`를 수행한다.
- 따라서 i가 2일 때  $2 * 1, 2 * 2, 2 * 3, \dots 2 * 9$ 까지 차례대로 수행되며 그 값을 출력하게 된다.
- 그다음으로 i가 3일 때 역시 2일 때와 마찬가지로 수행될 것이고 i가 9일 때까지 계속 반복된다.

i가 2일 때			i가 3일 때			i가 4일 때			i가 9일 때		
i	j	i*j									
2	1	2	3	1	3	4	1	4	9	1	9
	2	4		2	6		2	8		2	18
	3	8		3	9		3	12		3	27
	4	12		4	12		4	16		4	36
	5	10		5	15		5	20		5	45
	6	12		6	18		6	24		6	54
	7	14		7	21		7	28		7	63
	8	16		8	24		8	32		8	72
	9	18		9	27		9	36		9	81
②번 for 문 종료			②번 for 문 종료			②번 for 문 종료			전체 for 문 종료		

코딩공감자

# 단체 사진 & 선물 증정

코딩공감자



# Q&A

코딩공감자

We wish you  
success.



( 당신의 성공을 기원합니다. )