

# 파이썬 코딩 기초 교육

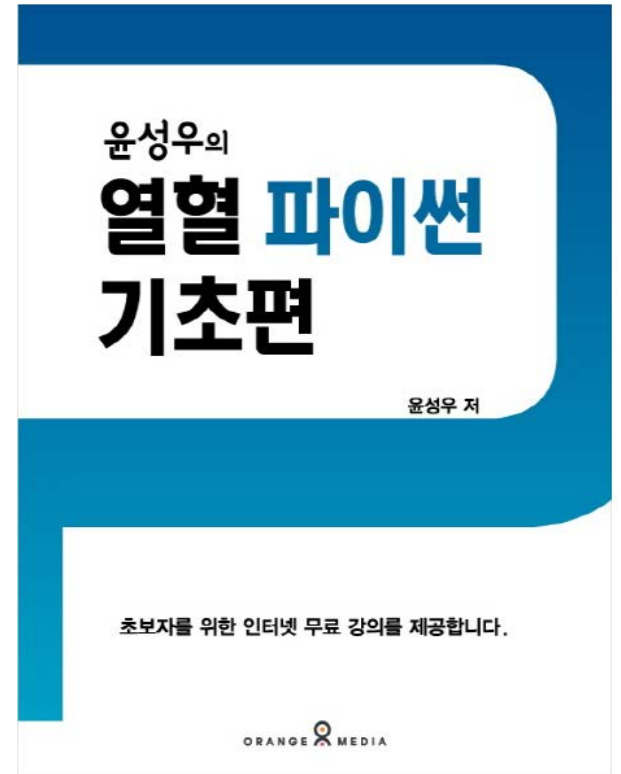
한국환경정책·평가연구원 진대용

# 강사소개

- 이름 : 진대용 (dyjin@kei.re.kr)
  - 광주과학기술원 전기전자컴퓨터공학 박사 (2017)
  - 한국환경정책·평가연구원 환경경제연구실 부연구위원 (2017~ )
- 관심분야
  - 기계학습 및 딥러닝, 자연어처리 및 텍스트 마이닝
  - 계산생물학(Computational Biology), 환경 빅데이터 분석
- 연구 활동
  - 생활밀착형 환경 이슈에 대한 수요반영 개선 연구 : 민원 빅데이터 분석을 중심으로 (2019)
  - 기후환경 이슈분석을 위한 텍스트 마이닝 활용 방안 (2018)
  - 대기이미지를 활용한 미세먼지 오염도 추정 (2018)
  - Jin, Daeyong, and Hyunju Lee. "Prioritizing cancer-related microRNAs by integrating microRNA and mRNA datasets." Scientific reports 6 (2016): 35350.
  - Jin, Daeyong, and Hyunju Lee. "A computational approach to identifying gene-microRNA modules in cancer." PLoS computational biology 11.1 (2015): e1004042

# 교재 소개

- 교재명 : 윤성우의 열혈 파이썬 기초편
- 강의자료
  - <http://github.com/dyjin1217>
- 책 강의자료 및 동영상 강의
  - <https://cafe.naver.com/cstudyjava>



# 강의내용 소개

- 파이썬 개론
- 입출력
- 함수
- 자료형
- 제어문
- 반복문

# 윤성우의 열혈 파이썬 기초편

Chapter 01 파이썬에게 질문하기

01-1.

## 파이썬의 시작 포인트

- 파이썬 공부에 앞서 알고 있어야 할 두 가지  
컴퓨터 프로그램 (소프트웨어)  
프로그래밍 언어 (코딩 언어)

01-2.

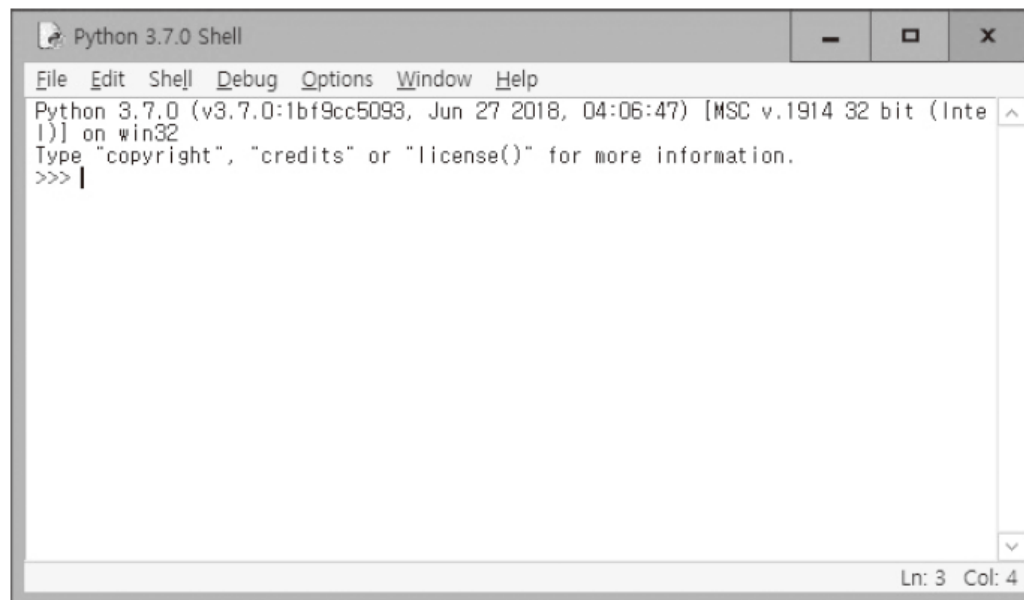
## 파이썬의 특징

- 모두를 위한 파이썬 (Python for everybody)  
Easy  
Large  
Powerful

01-3.

일단 파이썬을  
설치하자.

- <https://www.python.org/>

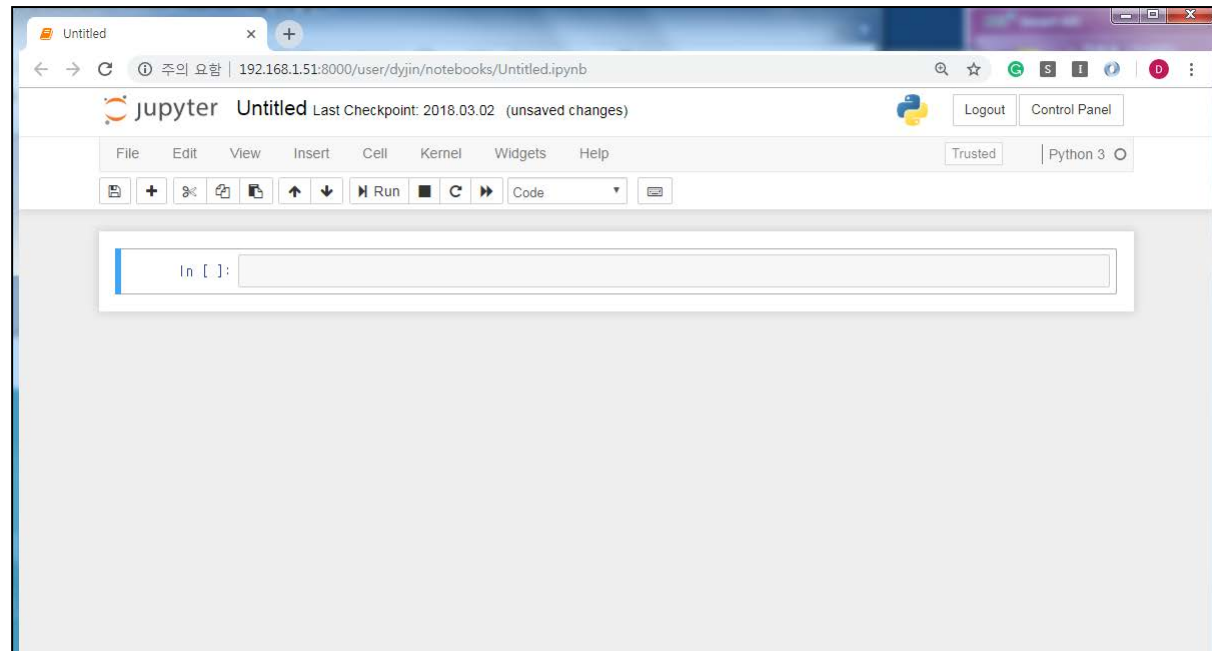




01-3.

일단 파이썬을  
설치하자.

- <https://colab.research.google.com/>
- Jupyter notebook



01-4.

수식을 계산하  
게 하자.

```
>>> 33 + 55  
88
```

```
>>> 33 + 55 =  
SyntaxError: invalid syntax
```

```
>>> 7 * 5 / 2  
17.5
```

```
>>> 9 * (5 - 7) / 3  
-6.0
```

01-5.

헤이 지니!

소리 질러!

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

```
>>> print(5)  
5
```

```
>>> print("5")  
5
```

01-5. (continue)

헤이 지니!

소리 질러!

```
>>> print(3 + 5)
```

```
8
```

```
>>> print("3 + 5")
```

```
3 + 5
```

01-5. (continue)

헤이 지니!

소리 질러!

```
>>> print("3 + 5 =", 8)
3 + 5 = 8
```

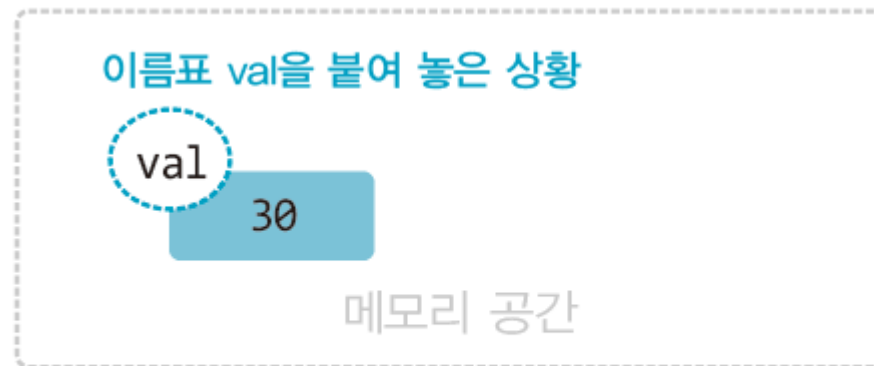
```
>>> print(1, 2, 3, "Hello", "World!")
1 2 3 Hello World!
```

```
>>> print("3 + 5 =", 3 + 5)
3 + 5 = 8
```

01-6.

이거 기억해 뒤

```
>>> val = 30
```



[그림 01-9: 대입의 결과]

```
>>> v1 = 25
```

```
>>> v2 = 30
```

```
>>> print(v1 + v2)
```

```
55
```

01-6. (continue)

이거 기억해 뒀

```
>>> x = 3 * 50
>>> y = x + 120
>>> z = y / 3
>>> print(z)
90.0
```



01-7.

## 이름 가져다 붙이기

```
>>> x = 100
>>> print(x)
100
>>> x = 3.14
>>> print(x)
3.14
>>> x = "Hi~"
>>> print(x)
Hi~
```





01-7. (continue)

# 이름 가져다 붙이기

```
>>> x = 3 * 50  
>>> x = x + 120  
>>> x = x / 3  
>>> print(x)  
90.0
```



01-8.

변수와

대입 연산자

```
>>> v = 25
```

```
>>> v = 30
```

```
>>> v = 25
```



변수 v의 메모리 공간

```
>>> v = 30
```



변수 v의 메모리 공간

이런 상태로 변함,  
저장된 값이 바뀜

[그림 01-17: 변수에 대한 논리적인 이해]

01-9.

어떤 일들이  
벌어질까요?

```
>>> x, y = 0, 797
```

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

```
>>> print (x, y)
```

```
797 797
```

```
>>> x, y = 121, 797
```

```
>>> x, y = y, x
```

```
>>> print(x, y)
```

```
797 121
```

# 윤성우의 열혈 파이썬 기초편

Chapter 02. 간단한 함수 만들기

# 함수란?

- 함수(function)란 하나의 로직을 재실행 할 수 있도록 하는 것 [출저 : 생활코딩]
- 코드를 묶어주는것, 코드의 집합

02-1.

함수 만들기 1:

인자 없는 것

```
>>> def greet():  
    print("반갑습니다.")  
    print("파이썬의 세계로 오신 것을 환영합니다.")
```

```
>>> greet()  
반갑습니다.  
파이썬의 세계로 오신 것을 환영합니다.
```

```
>>> greet()  
반갑습니다.  
파이썬의 세계로 오신 것을 환영합니다.
```

## 02-1. (continue)

# 함수 만들기 1: 인자 없는 것

함수 만든다는 선언

↓  
>>> def greet():

함수의 이름

print("반갑습니다.")

print("파이썬의 세계로 오신 것을 환영합니다.")

함수에 속하는 문장들

함수 만들기 끝낼 때 이 위치에서 엔터키 한번 더 입력!!

>>>

[그림 02-1: 함수의 기본 구조]

02-2.

## 함수 만들기 2: 인자 있는 함수

```
>>> def greet2(name):  
    print("반갑습니다.", name)  
    print(name, "님은 파이썬의 세계로 오셨습니다.")
```

```
>>> greet2("John")  
반갑습니다. John  
John 님은 파이썬의 세계로 오셨습니다.  
>>> greet2("Yoon")  
반갑습니다. Yoon  
Yoon 님은 파이썬의 세계로 오셨습니다.
```




## 02-2. (continue)

# 함수 만들기 2: 인자 있는 함수

함수 호출  
>>> greet2( "Everyone" )

함수 정의  
>>> def greet2( name ):  
    print("반갑습니다.", name)  
    print(name, "님은 파이썬의 세계로 오셨습니다.")

변수 name에 "Everyone" 전달 및 저장

A vertical dashed blue arrow points from the string "Everyone" in the function call to the parameter name in the function definition.

[그림 02-2: 함수 호출 시 값의 전달]

02-3.

함수 만들기 3:

값의 반환이

있는 것

```
>>> def adder2(num1, num2):  
    ar = num1 + num2  
    return ar
```

```
>>> result = adder2(5, 3)  
>>> print(result)  
8
```

02-3. (continue)

함수 만들기 3:  
값의 반환이  
있는 것

```
>>> def adder3(num1, num2):  
        return num1 + num2
```

```
>>> print(adder3(5, 3))
```

8

02-6.

주석을 달자.

```
def adder(num1, num2):    # 함수 adder의 정의
    return num1 + num2    # num1과 num2의 덧셈 결과를 넘김

print(adder(5, 3))        # adder 함수 호출, 그리고 그 결과 출력
```

02-7.

## 이름 달기 규칙 그리고 대소문 자 구분

```
>>> num = 0
>>> Num = 10      # 파이썬은 대소문자 구분
>>> print(num, Num)
0 10
```

```
>>> 2num = 0      # 숫자로 시작하는 것은 불가
SyntaxError: invalid syntax
```

```
>>> return = 2    # 키워드는 이름으로 쓸 수 없음
SyntaxError: invalid syntax
```

## 02-7. (continue)

이름 달기 규칙

그리고 대소문

자 구분

- 변수와 함수의 이름은 **소문자**로 시작한다.
- 둘 이상의 단어를 연결하는 경우 **언더바 \_**를 이용해서 연결한다.

좋은 예                      `my_name`

좋지 않은 예                `myname, MyName, My_Name`

02-8.

main 함수가 있는  
방식으로 예  
제를 작성하자.

```
# adder1.py
def adder(num1, num2):
    return num1 + num2
```

```
print(adder(5, 3))
```

```
# adder2.py
def adder(num1, num2):
    return num1 + num2
```

```
def main():
    print(adder(5, 3))
```

```
main()
```

## 02-9. (continue)

### main 함수가 있는 방식으로 예제를 작성하자.

```
# op4.py
def add(num1, num2):
    return num1 + num2
def min(num1, num2):
    return num1 - num2
def mul(num1, num2):
    return num1 * num2
def div(num1, num2):
    return num1 / num2

def main():
    print(add(5, 3))
    print(min(5, 3))
    print(mul(5, 3))
    print(div(5, 3))

main()
```



# 윤성우의 열혈 파이썬 기초편

Chapter 03. 프로그램 사용자로부터의 입력 그리고 코드의 반복

03-1.

# 프로그램 사용 자로부터 입력 받기

```
>>> str = input("How old are you: ")
```

```
How old are you: 12 years old
```

```
>>> print(str)
```

```
12 years old
```

```
str = input("How old are you: ")
```



사용자가 12를 입력한 이후 상황

```
str = "12"
```

[그림 03-1: input 함수의 동작 방식]

### 03-1. (continue)

프로그램 사용  
자로부터 입력  
받기

```
>>> num = "31" + "24"          # 문자열의 덧셈 결과
```

```
>>> print(num)
```

```
3124
```

03-2.

입력받은 내용  
을 숫자로 바꾸  
려면

```
>>> year = input("This year: ")
This year: 2020
>>> year = eval(year)
>>> year = year + 1
>>> print("Next year:", year)
Next year: 2021
```

## 03-2. (continue)

입력받은 내용을  
숫자로 바꾸  
려면

```
>>> rad = eval(input("radius: "))
radius: 2.5
>>> area = rad * rad * 3.14
>>> print(area)
19.625
```

03-3.

강력한 그러나  
위험할 수 있는  
eval 함수

```
>>> result = eval(input("뭐든 넣어요: "))  
뭐든 넣어요: 2 - 4 * 5 + 3  
>>> print(result)  
-15
```

### 03-3. (continue)

강력한 그러나

위험할 수 있는

eval 함수

```
>>> def ret():  
    return 12  
  
>>> result = eval(input("뭐든 넣어요: "))  
뭐든 넣어요: ret()  
>>> print(result)  
12
```

03-4.

# 정해진 횟수만큼 반복해서 실행시키기

```
>>> for i in [0, 1, 2]:  
    print(i)  
    print("hi~")
```

```
0  
hi~  
1  
hi~  
2  
hi~
```

변수    반복 횟수    정보

```
>>> for i in [0, 1, 2]:  
    [ print(i)  
      print("hi~") ]
```

for 루프에 속하는 문장들

[그림 03-2: for 루프의 구성]



03-4. (continue)

정해진 횟수만

큼 반복해서 실행

시키기

```
>>> sum = 0
>>> for i in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
    sum = sum + i
>>> print(sum)
55
```

03-5.

for.. in 과 range

의 조합

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

## 03-5. (continue) for.. in 과 range 의 조합

```
>>> for i in range(0, 3):      # 3회 반복이 목적  
    print("Happy")
```

Happy

Happy

Happy

```
>>> for i in range(3):      # 첫 번째 0 생략 가능  
    print("Happy")
```

Happy

Happy

Happy

# 윤성우의 열혈 파이썬 기초편

Chapter 04. int형 데이터와 float형 데이터

04-1.

정수의 표현과

실수의 표현

```
>>> num = 2          # 정수는 정확한 값 저장
```

```
>>> print(num)
```

```
2
```

```
>>> num = 1.0000000000000001      # 실수는 오차가 존재
```

```
>>> print(num)
```

```
1.0000000000000001
```

## 04-1. (continue)

정수의 표현과

실수의 표현

```
>>> num1 = 1.0000000000000001
```

```
>>> num2 = 1.1
```

```
>>> print(num1 + num2)
```

```
2.10000000000000014
```

04-2.

## 기본적인 산술 연산

```
>>> type(3)
<class 'int'>
>>> type(3.1)
<class 'float'>
>>> type(3.0)
<class 'float'>
```

## 04-2. (continue)

# 기본적인 산술 연산

```
>>> 3 ** 2
```

```
9
```

```
>>> 5 / 2
```

```
2.5
```

```
>>> 5 // 2    # 나눗셈의 몫을 계산
```

```
2
```

```
>>> 5 % 2     # 나눗셈의 나머지 계산
```

```
1
```

+	덧셈
-	뺄셈
*	곱셈
**	거듭제곱
/	실수형 나눗셈
//	정수형 나눗셈
%	나머지가 얼마?



04-3.

int형 변환

float형 변환

```
>>> num = 10
```

```
>>> num = float(num)
```

```
>>> type(num)
```

```
<class 'float'>
```

```
>>> num = float("3.14")
```

```
>>> type(num)
```

```
<class 'float'>
```

```
>>> height = float(input("키 정보 입력: "))
```

```
키 정보 입력: 165.3
```

```
>>> print(height)
```

```
165.3
```

## 04-3. (continue)

int형 변환

float형 변환

```
>>> num = int(3.14)
```

```
>>> print(num)
```

```
3
```

```
>>> height = int(input("키 정보 cm 단위로 입력: "))
```

```
키 정보 cm 단위로 입력: 178
```

```
>>> print(height)
```

```
178
```

04-4.

복합 대입

연산자

```
>>> num = 10
>>> num += 1      # num = num + 1을 줄인 표현
>>> print(num)
11
```

num = num - 1   vs.   num -= 1

num = num \* 3   vs.   num \*= 3

04-5.

소괄호

```
>>> 3 + 4 / 2
```

```
5.0
```

```
>>> (3 + 4) / 2
```

```
3.5
```

# 윤성우의 열혈 파이썬 기초편

Chapter 05. 리스트와 문자열

05-1.

## print 함수의 복습과 확장

```
>>> for i in [1, 2, 3]:  
        print(i, end = '_')
```

```
1_2_3_
```

```
>>> for i in [1, 2, 3]:  
        print(i, end = ' ')
```

```
1 2 3
```

05-2.

리스트형

데이터

```
>>> 35
```

```
35
```

```
>>> Happy
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#0>", line 1, in <module>
```

```
    Happy
```

```
NameError: name 'Happy' is not defined
```

```
>>> 3.14
```

```
3.14
```

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

```
[1, 2, 3]
```

## 05-2. (continue)

### 리스트형

### 데이터

```
>>> type([1, 2, 3])
```

```
<class 'list'>
```

```
>>> [1, "hello", 3.3]
```

```
[1, 'hello', 3.3]
```

```
>>> [1, 2, [3, 4], ["AAA", "ZZZ"]]
```

```
[1, 2, [3, 4], ['AAA', 'ZZZ']]
```

```
>>> st = [1, "hello", 3.3]
```

```
>>> print(st)
```

```
[1, 'hello', 3.3]
```



## 05-2. (continue)

### 리스트형

### 데이터

- int형 데이터                      ex) 3, 5, 7, 9
- float형 데이터                    ex) 2.2, 4.4, 6.6, 8.8
- 리스트형 데이터                      ex) [3, 5, 7, 9], [2.2, 4.4,  
6.6, 8.8]

05-3.

## 인덱싱 연산

```
>>> [1, 2, 3] + [4, 5]
```

```
[1, 2, 3, 4, 5]
```

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

```
[1, 2, 3, 1, 2, 3]
```

05-3.

## 인덱싱 연산

```
>>> st = [1, 2, 3, 4, 5]
>>> n1 = st[0]
>>> n2 = st[4]
>>> print(n1, n2)
1 5
```

## 05-3. (continue)

### 인덱싱 연산

```
>>> st = [1, 2, 3, 4, 5]
```

```
>>> st[0] = 5
```

```
>>> st[4] = 1
```

```
>>> st
```

```
[5, 2, 3, 4, 1]
```

```
>>> st = [1, 2, 3, 4, 5]
```

```
>>> print(st[0], st[2], st[4])
```

```
1 3 5
```

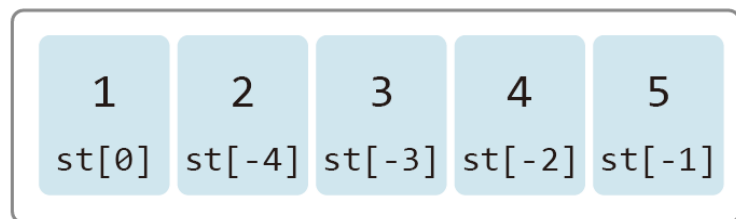
## 05-3. (continue)

### 인덱싱 연산

```
>>> st = [1, 2, 3, 4, 5]
```

```
>>> print(st[-1], st[-2], st[-3])
```

```
5 4 3
```



[그림 05-2: `st = [1, 2, 3, 4, 5]`의 음수 인덱스 값]

05-4.

## 슬라이싱 연산

```
>>> st1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> st2 = st1[2:5]          # st1[2:5]를 꺼내 st2에 저장
>>> st2
[3, 4, 5]
```

05-4. (continue)

슬라이싱 연산

```
>>> st = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> st[2:5] = [0, 0, 0, 0, 0]
```

```
>>> st
```

```
[1, 2, 0, 0, 0, 0, 0, 6, 7, 8, 9]
```

05-5.

# 슬라이싱 연산 에서 생략 가능 한 부분

```
>>> st = [1, 2, 3, 4, 5]
>>> st[0:3] = [0, 0, 0]
>>> st
[0, 0, 0, 4, 5]
```

```
>>> st = [1, 2, 3, 4, 5]
>>> st[:3] = [0, 0, 0]
>>> st
[0, 0, 0, 4, 5]
```



## 05-5. (continue) 슬라이싱 연산 에서 생략 가능 한 부분

```
>>> st = [1, 2, 3, 4, 5]
```

```
>>> st[2:] = [0, 0, 0]
```

```
>>> st
```

```
[1, 2, 0, 0, 0]
```

```
>>> st = [1, 2, 3, 4, 5]
```

```
>>> st[:] = [0, 0, 0, 0, 0]
```

```
>>> st
```

```
[0, 0, 0, 0, 0]
```

## 05-5. (continue) 슬라이싱 연산 에서 생략 가능 한 부분

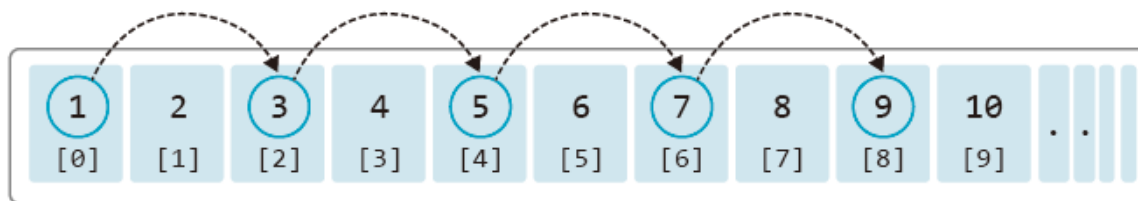
```
>>> st = [1, 2, 3, 4, 5]
>>> st[:] = [0]          # 리스트 전체를 0 하나로 교체
>>> st
[0]
```

```
>>> st = [1, 2, 3, 4, 5]
>>> st[:] = []          # 리스트 전체 내용 삭제
>>> st
[]
```

05-6.

리스트에서 두  
칸씩 뛰면서 저  
장된 값들 꺼내  
기

```
>>> st1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
>>> st2 = st1[0:9:2]      # st1[0] ~ st2[8]까지 2칸씩 뛰면서
>>> st2
[1, 3, 5, 7, 9]
```



[그림 05-3: 두 칸씩 뛰며 값을 뽑아내려면]

## 05-6. (continue)

리스트에서 두  
칸씩 뛰면서 저  
장된 값들 꺼내  
기

```
>>> st1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
>>> st2 = st1[0:9:3]      # st1[0] ~ st2[8]까지 3칸씩 뛰면서
>>> st2
[1, 4, 7]
```

05-7.

스트링형 데이

터: 문자열

```
>>> "Happy birthday to you"  
'Happy birthday to you'
```

```
>>> 'Happy birthday to you'  
'Happy birthday to you'
```

```
>>> type('what1')          # 작은따옴표로 묶은 문자열  
<class 'str'>  
>>> type("what2")          # 큰따옴표로 묶은 문자열  
<class 'str'>
```

## 05-7. (continue) 스트링형 데이터: 문자열

- int형 데이터      ex) 3, 5, 7, 9
- float형 데이터      ex) 2.2, 4.4, 6.6, 8.8
- 리스트형 데이터      ex) [3, 5, 7, 9], [2.2, 4.4, 6.6, 8.8]
- 스트링형 데이터      ex) "I am a boy", 'You are a girl'

## 05-7. (continue) 스트링형 데이터: 문자열

```
>>> [1, 2] + [3, 4]
[1, 2, 3, 4]
>>> "Hello" + "Everybody"
'HelloEverybody'
```

```
>>> [1, 2] * 3
[1, 2, 1, 2, 1, 2]
>>> "AZ" * 3
'AZAZAZ'
```

## 05-7. (continue) 스트링형 데이터: 문자열

```
>>> st = [1, 2, 3, 4, 5]
>>> st[2]      # 세 번째 위치의 값만 뽑아 냄
3
>>> str = "SIMPLE"
>>> str[2]     # 세 번째 위치의 값만(문자만) 뽑아 냄
'M'

>>> st = [1, 2, 3, 4, 5, 6, 7]
>>> st[2:5]
[3, 4, 5]
>>> str = "SIMPLEST"
>>> str[2:5]
'MPL'
```



## 05-7. (continue) 스트링형 데이터: 문자열

```
>>> str = "Happy"
```

```
>>> str[0] = "D"
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#1>", line 1, in <module>
```

```
    str[0] = 'D'
```

```
TypeError: 'str' object does not support item assignment
```

05-8.

# 리스트와 for 루프 그리고 문자열과 for 루프

```
>>> for i in [1, 2, 3]:  
    print(i, end = ' ')
```

1 2 3

```
>>> for i in 'Happy':  
    print(i, end = ' ')
```

H a p p y

05-9.

리스트와 문자

열을 전달받는

함수 len

```
>>> st = [1, 2, 3]
```

```
>>> len(st)
```

```
3
```

```
>>> sr = 'HaHaHa~'
```

```
>>> len(sr)
```

```
7
```

05-10.

리스트와 문자  
열을 인자로 전  
달하고 반환하  
기

```
>>> def so_simple2(st):  
        print(st)
```

```
>>> so_simple2([1, 3, 5])  
[1, 3, 5]
```

05-10. (continue)

리스트와 문자

열을 인자로 전

달하고 반환하

기

```
>>> def so_simple3():  
        st = [1, 2, 3, 4, 5]  
        return st
```

```
>>> r = so_simple3()
```

```
>>> r
```

```
[1, 2, 3, 4, 5]
```

05-10. (continue)

리스트와 문자

열을 인자로 전

달하고 반환하

기

```
>>> def so_simple4(s):  
        print(s)  
        return "Bye~"
```

```
>>> r = so_simple4("Hello")  
'Hello'  
>>> r  
'Bye~'
```

# 윤성우의 열혈 파이썬 기초편

Chapter 06. 리스트와 문자열의 함수들

o6-1.

리스트와

함수들

```
>>> st = [1, 3, 5, 7, 9]
```

```
>>> num = len(st)
```

```
>>> num
```

```
5
```

```
>>> st = [2, 5, 3, 7, 4]
```

```
>>> min(st)
```

```
2
```

```
>>> max(st)
```

```
7
```



## o6-1. (continue)

리스트와

함수들

```
>>> st = [1, 2, 3]
```

```
>>> st.remove(2) # 리스트에서 2를 찾아서 삭제
```

```
>>> st
```

```
[1, 3]
```

## o6-1. (continue)

리스트와

함수들

```
>>> st = [1, 2, 3]
>>> st.append(4)      # st의 끝에 4 추가
>>> st.extend([5, 6]) # st의 끝에 [5, 6]의 내용 추가
>>> st
[1, 2, 3, 4, 5, 6]
```

## o6-1. (continue)

### 리스트와

### 함수들

```
>>> st = [1, 2, 4]
>>> st.insert(2, 3)      # 인덱스 값 2의 위치에 3 저장
>>> st
[1, 2, 3, 4]
>>> st.clear()          # 리스트 내용 전부 삭제
>>> st
[]
```

## o6-1. (continue)

리스트와

함수들

```
>>> st = []          # 빈 리스트 생성
>>> st.append(1)      # 리스트에 1 추가
>>> st.append(9)      # 리스트에 9 추가
>>> st
[1, 9]
```

## o6-1. (continue)

### 리스트와

### 함수들

```
>>> st = [1, 2, 3, 4, 5]
>>> st.pop(0)      # 인덱스 값 0의 위치에 저장된 데이터 삭제
1
>>> st
[2, 3, 4, 5]
>>> st.remove(5)   # 리스트에서 5를 삭제
>>> st
[2, 3, 4]
```

## o6-1. (continue)

리스트와

함수들

```
>>> st = [1, 2, 3, 1, 2]
```

```
>>> st.count(1)      # 1이 몇 번 등장하는지 세어라.
```

```
2
```

```
>>> st.index(2)      # 처음 2가 등장하는 위치의 인덱스 값은?
```

```
1
```

06-2.

## 두 가지 유형의 함수가 갖는 특 징들

- 객체 안에 있는 함수
- 객체 밖에 있는 함수

06-3.

문자열과

함수들

```
>>> str = "YoonSungWoo"
```

```
>>> str.count("o")      # "o"가 몇 번 등장?
```

```
4
```

```
>>> str.count("oo")     # "oo"가 몇 번 등장?
```

```
2
```



## 06-3. (continue)

### 문자열과

### 함수들

```
>>> org = "Yoon"
>>> lcp = org.lower()      # 모든 문자를 소문자로 바꿔서 반환
>>> ucp = org.upper()      # 모든 문자를 대문자로 바꿔서 반환
>>> org                    # 원본은 그대로 존재한다.
'Yoon'
>>> lcp
'yoon'
>>> ucp
'YOON'
```

## o6-3. (continue)

## 문자열과

## 함수들

```
>>> org = " MIDDLE "  
>>> cp1 = org.lstrip()      # 앞쪽에 (왼쪽에) 있는 공백들 모두 제거  
>>> cp2 = org.rstrip()     # 뒤쪽에 (오른쪽에) 있는 공백들 모두 제거  
>>> org  
' MIDDLE '  
>>> cp1  
'MIDDLE '  
>>> cp2  
' MIDDLE '
```

## 06-3. (continue)

문자열과

함수들

```
>>> org = " MIDDLE "  
>>> cpy = org.strip()      # 앞과 뒤에 있는 공백들 모두 제거  
>>> org  
' MIDDLE '  
>>> cpy  
'MIDDLE'
```

## 06-3. (continue)

문자열과

함수들

```
>>> org = "YoonSungWoo"
>>> rps = org.replace("oo", "ee")      # "oo"를 전부 "ee"로 교체
>>> rps
'YeenSungWee'
```

## 06-3. (continue)

문자열과

함수들

```
>>> org = "YoonSungWoo"
>>> rps = org.replace("oo", "ee", 1)    # 첫 번째 "oo"를 "ee"로 교체
>>> rps
'YeenSungWoo'
```

## o6-3. (continue)

문자열과

함수들

```
>>> org = "ab_cd_ef"
>>> ret = org.split('_')      # '_' 기준으로 문자열 쪼개서 리스트에
                              # 담아!
>>> ret
['ab', 'cd', 'ef']
```

06-4.

문자열의 탐색

관련 함수들

```
>>> str = "What is important is that you should choose what is best for you"  
>>> str.find("is")      # "is"가 있는 위치의 인덱스 값은?  
5
```

```
>>> str = "What is important is that you should choose what is best for you"  
>>> str.rfind("is")     # 마지막 "is"가 있는 위치의 인덱스 값은?  
49
```

06-5.

이스케이프

문자

```
>>> str = "escape\ncharacters"
```

```
>>> print(str)
```

```
escape
```

```
characters
```

`\n`

줄 바꿈

`\t`

탭

`\'`

작은따옴표 출력

`\"`

큰따옴표 출력



## 06-5. (continue)

# 이스케이프

# 문자

```
>>> str = "제가 마음속으로 그랬습니다. '이건 아니야.'라고 말이죠."
```

```
>>> print(str)
```

제가 마음속으로 그랬습니다. '이건 아니야.'라고 말이죠.

"제가 마음속으로 그랬습니다. '이건 아니야.'라고 말이죠."

'제가 마음속으로 그랬습니다. "이건 아니야."라고 말이죠.'

'제가 마음속으로 그랬습니다. \'이건 아니야. \'라고 말이죠.'

"제가 마음속으로 그랬습니다. \"이건 아니야.\"라고 말이죠."

o6-6.

함수가 아닌 del

명령

```
>>> st = [1, 2, 3, 4, 5]
```

```
>>> st.clear()          # 리스트의 모든 값 삭제
```

```
>>> st
```

```
[]
```

```
>>> st = [1, 2, 3, 4, 5]
```

```
>>> st[:] = []          # 리스트의 모든 값 삭제
```

```
>>> st
```

```
[]
```

```
>>> st = [1, 2, 3, 4, 5]
```

```
>>> st[2:] = []          # 인덱스 2 이후로 전부 삭제
```

```
>>> st
```

```
[1, 2]
```

## o6-6. (continue)

### 함수가 아닌 del

### 명령

```
>>> st = [1, 2, 3, 4, 5]
>>> del st[:]          # 리스트에 저장된 값 모두 삭제
>>> st
[]
```

```
>>> st = [1, 2, 3, 4, 5]
>>> del st[3:]         # st[3]부터 그 뒤까지 모두 삭제
>>> del st[0]          # st[0] 하나만 삭제
>>> st
[2, 3]
```

o6-6. (continue)

함수가 아닌 del

명령

```
>>> st = [1, 2, 3, 4, 5]
```

```
>>> del st          # 리스트를 통째로 삭제! 리스트 자체를 삭제!
```

# 윤성우의 열혈 파이썬 기초편

Chapter 07. True, False 그리고 if와 그 형제들

07-1.

## 참과 거짓을 의미하는 값

```
>>> True      # True는 그 단어의 의미처럼 '참'을 뜻한다.
```

```
True
```

```
>>> False     # False는 그 단어의 의미처럼 '거짓'을 뜻한다.
```

```
False
```

```
>>> 3 > 10     # 3이 10보다 크니?
```

```
False
```

```
>>> 3 < 10     # 3이 10보다 작으니?
```

```
True
```

## 07-1. (continue)

참과 거짓을

의미하는 값

```
>>> r = 3 < 10      # < 연산의 결과인 True가 변수 r에 저장된다.
```

```
>>> r
```

```
True
```

```
>>> type(True)
```

```
<class 'bool'>
```

```
>>> type(False)
```

```
<class 'bool'>
```

## 07-1. (continue)

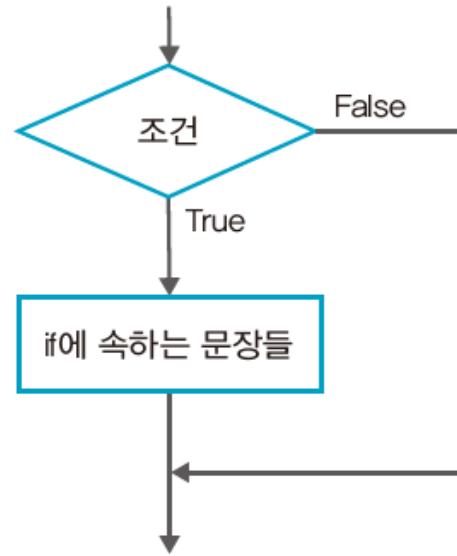
# 참과 거짓을 의미하는 값

- int형 데이터      ex) 3, 5, 7, 9
- float형 데이터      ex) 2.2, 4.4, 6.6, 8.8
- 리스트형 데이터      ex) [3, 5, 7, 9], [2.2, 4.4, 6.6, 8.8]
- 스트링형 데이터      ex) "I am a boy", 'You are a girl'
- 부울형 데이터      ex) True, False



07-3.

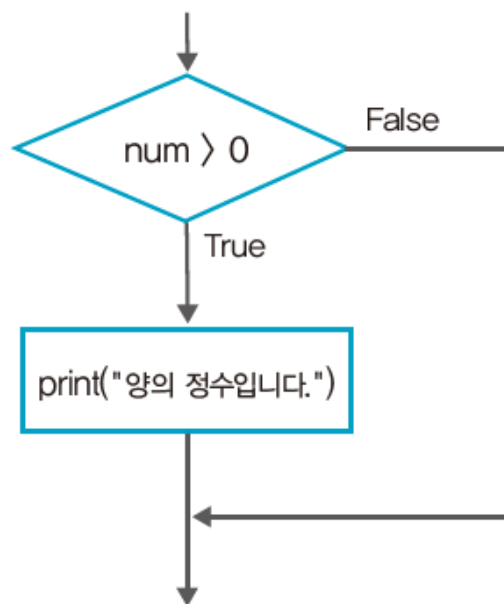
if문: 조건이 맞  
으면 실행을 해  
라.



[그림 07-1: if문의 순서도]

07-3. (continue)  
if문: 조건이 맞  
으면 실행을 해  
라.

```
if num > 0:  
    print("양의 정수입니다.")
```



[그림 07-2: if문 예제의 순서도]

## 07-3. (continue)

if문: 조건이 맞으면 실행을 해라.

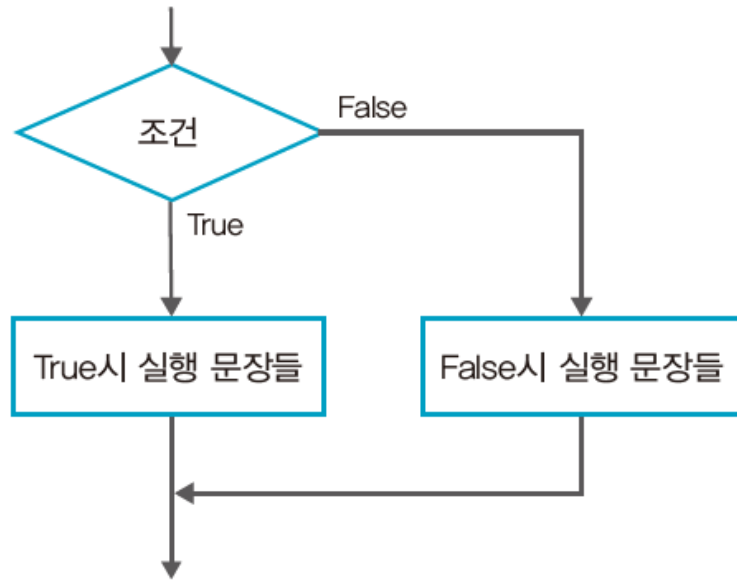
```
num = int(input("정수 입력: "))  
if num > 0:  
    print("양의 정수입니다.")
```

정수 입력: 2  
양의 정수입니다.

```
>>> num = 2  
>>> if num > 0: print("양의 정수입니다.")
```

07-4.

if ~ else문:이 쪽  
길! 아니면 저쪽  
길!



[그림 07-3: if ~ else문의 순서도]

07-4. (continue)

if ~ else문:이 쪽

길! 아니면 저쪽

길!

```
# if_else.py
```

```
def main():
```

```
    num = int(input("정수 입력: "))
```

```
    if num > 0:
```

```
        print("0보다 큰 수입니다.")
```

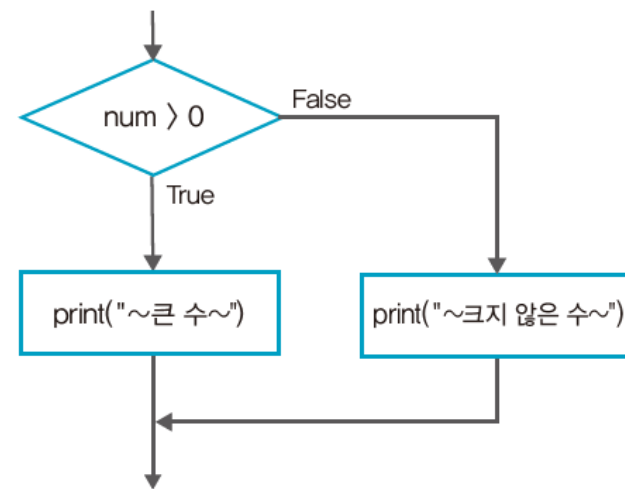
```
    else:
```

```
        print("0보다 크지 않은 수입니다.")
```

```
main()
```

정수 입력: -7

0보다 크지 않은 수입니다.



07-5.

if ~ elif ~ else문:  
여러 길 중에서  
하나의 길만 선택!

```
# if_elif_else.py
def main():
    num = int(input("정수 입력: "))

    if num > 0:
        print("0보다 큰 수입니다.")
    elif num < 0:
        print("0보다 작은 수입니다.")
    else:
        print("0으로 판단이 됩니다.")

main()

정수 입력: 0
0으로 판단이 됩니다.
```

07-6.

## True 또는 False 를 반환하는 연 산들

$A > Z$      A가 Z보다 크면 True, 크지 않으면 False 반환

$A < Z$      A가 Z보다 작으면 True, 작지 않으면 False 반환

$A \geq Z$      A가 Z보다 크거나 같으면 True, 그렇지 않으면 False 반환

$A \leq Z$      A가 Z보다 작거나 같으면 True, 그렇지 않으면 False 반환

$A == Z$      A와 Z가 같으면 True, 같지 않으면 False 반환

$A != Z$      A와 Z가 같지 않으면 True, 같으면 False 반환

## 07-6. (continue) True 또는 False 를 반환하는 연 산들

```
# if_elif_else2.py
def main():
    num = int(input("정수 입력: "))
    if num == 1:
        print("1을 입력했습니다.")
    elif num == 2:
        print("2를 입력했습니다.")
    elif num == 3:
        print("3을 입력했습니다.")
    else:
        print("1, 2, 3 아닌 정수 입력했습니다.")
```

main()

정수 입력: 2

2를 입력했습니다.



## 07-6. (continue) True 또는 False 를 반환하는 연 산들

```
>>> True and True  
True
```

```
>>> True and False  
False
```

```
>>> True or False  
True
```

```
>>> False or False  
False
```

```
>>> not False  
True
```

```
>>> not True  
False
```

## 07-6. (continue) True 또는 False 를 반환하는 연 산들

```
# if_and_if.py
def main():
    num = int(input("2의 배수이면서 3의 배수인 수 입력: "))
    if num % 2 == 0:
        if num % 3 == 0:
            print("OK!")
        else:
            print("NO!")
    else:
        print("NO!")
```

main()

2의 배수이면서 3의 배수인 수 입력: 6

OK!

## 07-6. (continue) True 또는 False 를 반환하는 연 산들

```
if num % 2 == 0:
    if num % 3 == 0:
        print("OK!")
    else:
        print("NO!")
else:
    print("NO!")
```

```
if (num % 2) == 0 and (num % 3) == 0:
    print("OK!")
else:
    print("NO!")
```

07-7.

리스트와 문자

열을 대상으로

도 동작하는

>=, <=, ==, !=

```
>>> 'abc' == 'abc'    # 두 문자열이 같은가?
```

```
True
```

```
>>> 'abc' != 'abc'    # 두 문자열이 다른가?
```

```
False
```

```
>>> [1, 2, 3] == [1, 2]    # 두 리스트가 같은가?
```

```
False
```

```
>>> [1, 2, 3] != [1, 2]    # 두 리스트가 다른가?
```

```
True
```

07-8.

True 또는 False  
로 답하는 함수  
들

```
>>> st1 = "123"
>>> st2 = "OneTwoThree"
>>> st1.isdigit()      # st1은 숫자로만 이뤄져 있나요?
True
>>> st2.isdigit()      # st2는 숫자로만 이뤄져 있나요?
False
```

07-8. (continue)

True 또는 False

로 답하는 함수

들

```
>>> st1 = "123"
>>> st2 = "OneTwoThree"
>>> st1.isalpha()
False
>>> st2.isalpha()
True
```

## 07-8. (continue) True 또는 False 로 답하는 함수 들

```
>>> str = "Supersprint"
>>> str.startswith("Super")      # 문자열이 'Super'로 시작하는가?
True
>>> str.endswith("int")          # 문자열이 'int'로 끝나는가?
True
```

## 07-8. (continue) True 또는 False 로 답하는 함수 들

```
# is_phone_num.py
def main():
    pnum = input("스마트폰 번호 입력: ")
    if pnum.isdigit() and pnum.startswith("010"):
        print("정상적인 입력입니다.")
    else:
        print("정상적이지 않은 입력입니다.")

main()
```

스마트폰 번호 입력: 01077779999  
정상적인 입력입니다.



07-9.

in, not in

```
>>> s = "Tomato spaghetti"
>>> if s.find("ghe") != -1:
    print("있습니다.")
else:
    print("없습니다.")
```

있습니다.

```
>>> if "ghe" in s:
    print("있습니다.")
else:
    print("없습니다.")
```

있습니다.

07-9. (continue)

in, not in

```
>>> 3 in [1, 2, 3]      # 리스트 [1, 2, 3] 안에 3이 있는가?
True
>>> 4 in [1, 2, 3]      # 리스트 [1, 2, 3] 안에 4가 있는가?
False
```

07-9. (continue)

in, not in

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

```
False
```

```
>>> 4 not in [1, 2, 3]
```

```
True
```

```
>>> "he" not in "hello"
```

```
False
```

```
>>> "oo" not in "hello"
```

```
True
```

07-10.

# 수(Number)를 True와 False로 인식하는 방식

```
>>> num = 1  
>>> if num:  
    print("0 아닙니다.")
```

0 아닙니다.

0 오는 경우

False가 온 것으로 간주한다.

0 아닌 수가 오는 경우

True가 온 것으로 간주한다.

## 07-10. (continue) 수(Number)를 True와 False로 인식하는 방식

```
>>> num = 1  
>>> if num != 0:  
    print("num은 0 아닙니다.")
```

num은 0 아닙니다.

```
>>> num = 1  
>>> if num:  
    print("num은 0 아닙니다.")
```

num은 0 아닙니다.

07-10. (continue)

수(Number)를

True와 False로

인식하는 방식

```
>>> bool(5)
```

```
True
```

```
>>> bool("what")
```

```
True
```

```
>>> bool("")
```

```
False
```

```
>>> bool([1, 2, 3])
```

```
True
```

```
>>> bool([])
```

```
False
```

# 윤성우의 열혈 파이썬 기초편

Chapter 08. for 루프와 while 루프

08-1.

for 루프에 대한

복습

```
# for_sum_range.py
def main():
    sum = 0
    for i in range(1, 11):
        sum = sum + i
    print("sum =", sum, end = ' ')

main()

sum = 55
```



o8-2.

True가 될 때까지 반복하는  
while 루프

```
# while_basic.py
def main():
    cnt = 0
    while cnt < 3:
        print(cnt, end = ' ')
        cnt = cnt + 1

main()

0 1 2
```

08-3.

for와 while의

비교

```
for <변수> in <반복 범위>:
```

```
<for에 속하는 문장들>
```

```
while <반복 조건>:
```

```
<조건이 True인 경우 반복 실행할 문장들>
```

## 08-3. (continue) for와 while의 비교

```
# while_sum.py
def main():
    i = 1
    sum = 0
    while i < 11:
        sum = sum + i
        i = i + 1
    print("sum =", sum, end = ' ')

main()

sum = 55
```

## 08-3. (continue) for와 while의 비교

```
# while_over100.py
def main():
    i = 1
    sum = 0
    while sum <= 100:
        sum = sum + i
        i = i + 1
    print(i-1, "더했을 때의 합", sum, end = ' ')
```

main()

14 더했을 때의 합 105

## o8-4. break

```
# while_break.py
def main():
    i = 0
    while i < 100:
        print(i, end = ' ')
        i = i + 1
        if i == 20:
            break
```

```
main()
```

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```

o8-4. (continue)

break

```
# while_over100_break.py
def main():
    i = 1
    sum = 0
    while True:
        sum = sum + i
        if sum > 100:
            print(i, "더했을 때의 합", sum, end = ' ')
            break
        i = i + 1

main()
```

14 더했을 때의 합 105

## o8-5. continue

```
>>> for i in range(1, 11):  
        print(i, end = ' ')
```

```
1 2 3 4 5 6 7 8 9 10
```

```
>>> for i in range(1, 11):  
        if i % 2 == 0:  
            continue  
        print(i, end = ' ')
```

```
1 3 5 7 9
```

o8-5. (continue)

continue

```
>>> i = 0
>>> while i < 10:
    i = i + 1
    print(i, end = ' ')
```

1 2 3 4 5 6 7 8 9 10

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

1 2 4 5 7 8 10



o8-6.

이중 for 루프

```
>>> for i in [1, 2]:  
    for j in ['a', 'b', 'c']:  
        print(j * i, end = ' ')
```

a b c aa bb cc

o8-6. (continue)

이중 for 루프

```
>>> sr = ['father', 'mother', 'brother']
```

```
>>> cnt = 0
```

```
>>> for s in sr:
```

```
    for c in s:
```

```
        if c == 'r':
```

```
            cnt += 1
```

```
>>> cnt
```

```
4
```