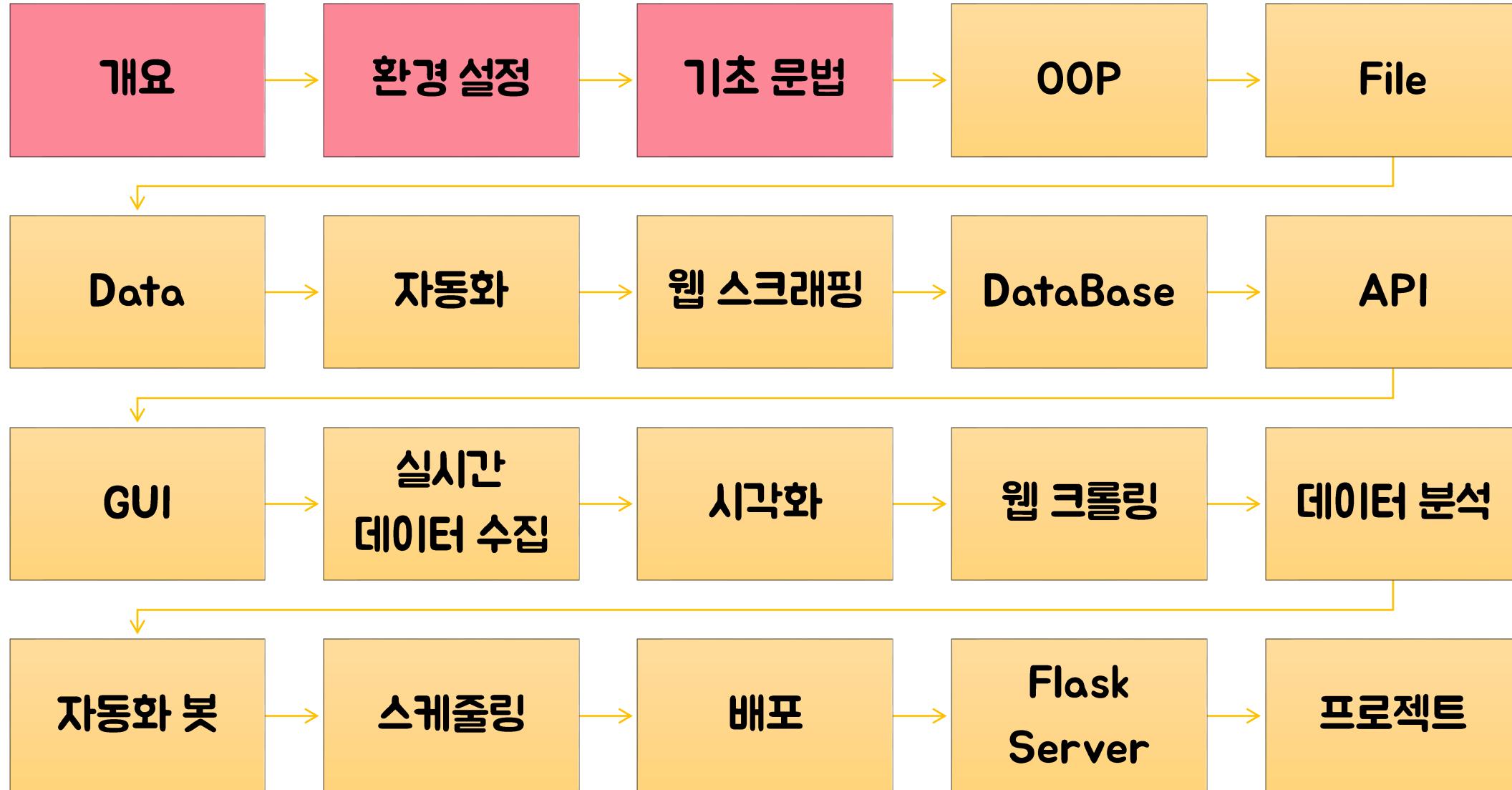


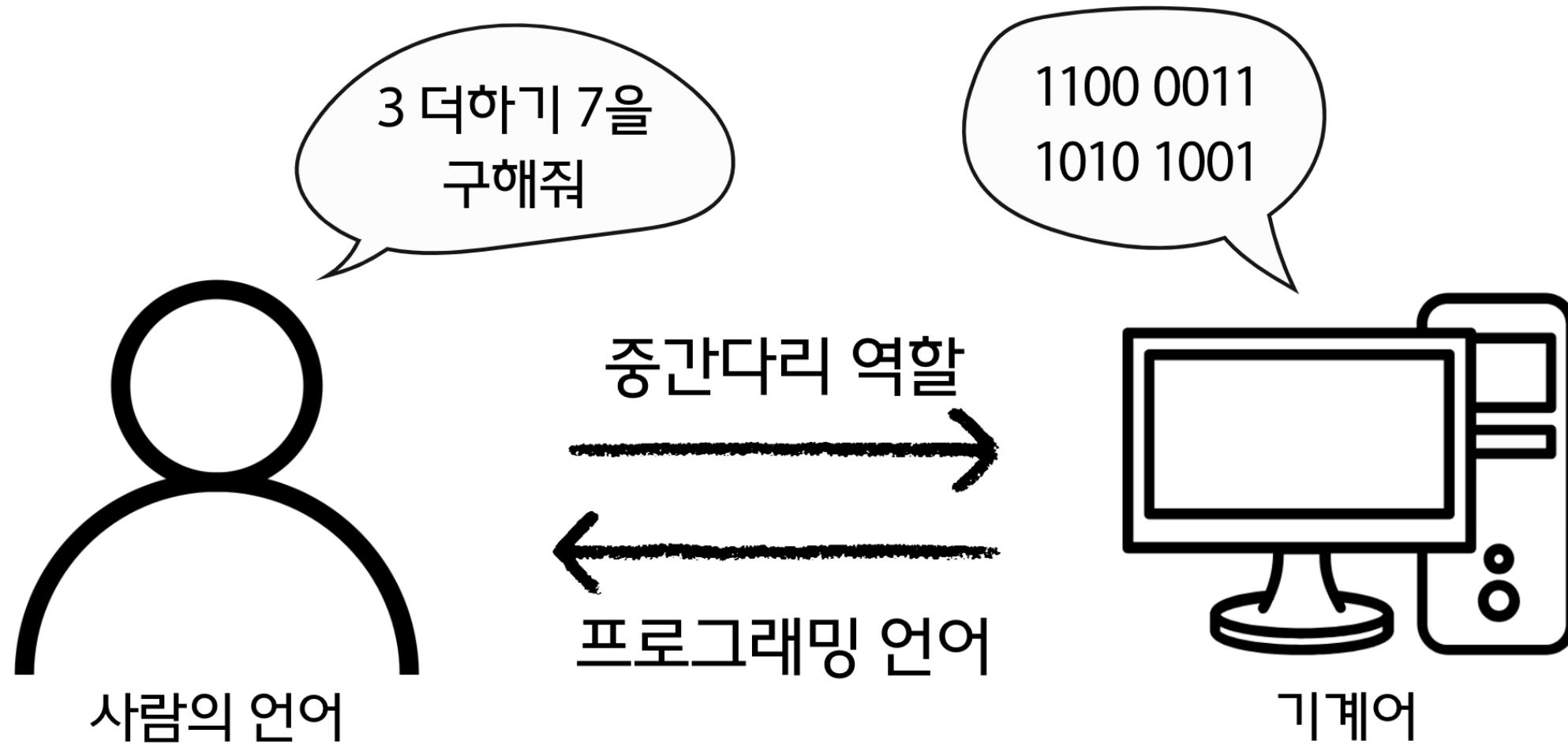


대한상공회의소  
서울기술교육센터

나예호 교수



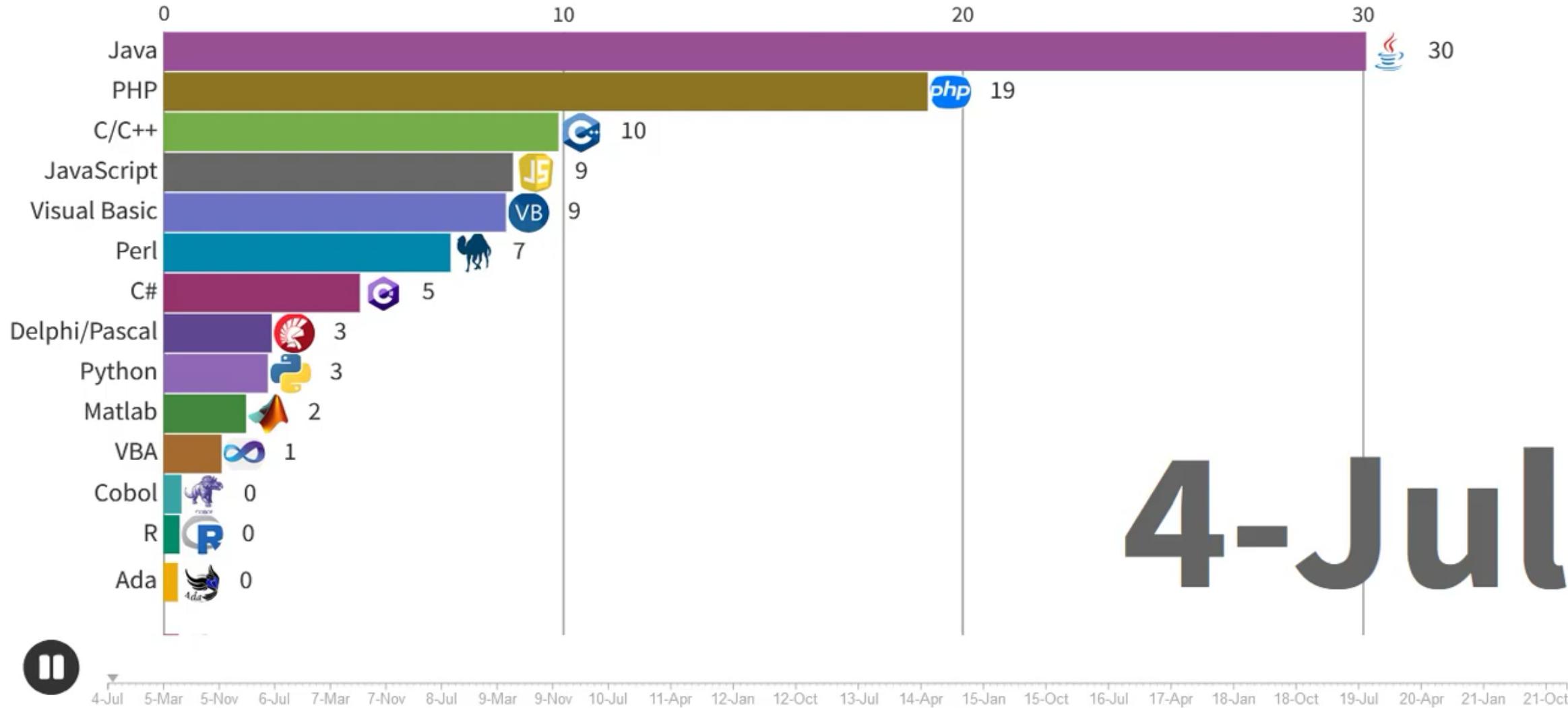
## - 기계와 의사소통을 할 수 있게 해주는 언어



- 기계와 의사소통을 할 수 있게 해주는 언어

`result = a + b`







About us ▾ Knowledge News Coding Standards [TIOBE Index](#) Contact 

Products ▾ Quality Models ▾ Markets ▾ [Schedule a demo](#)

Jan 2025	Jan 2024	Change	Programming Language	Ratings	Change
1	1		 Python	23.28%	+9.32%
2	3		 C++	10.29%	+0.33%
3	4		 Java	10.15%	+2.28%
4	2		 C	8.86%	-2.59%
5	5		 C#	4.45%	-2.71%
6	6		 JavaScript	4.20%	+1.43%
7	11		 Go	2.61%	+1.24%
8	9		 SQL	2.41%	+0.95%
9	8		 Visual Basic	2.37%	+0.77%
10	12		 Fortran	2.04%	+0.94%



NETFLIX



Instagram



- Life is short, you need Python.

Guido van Rossum (1991)

### 장점

직관적인 코드

많은 라이브러리

플랫폼 독립적

오픈 소스

자동 메모리 관리

### 단점

실행속도가 느리다

멀티 스레딩 어려움

모바일 개발 어려움

메모리 사용량이 크다

번거로운 배포



- Life is short, you need Python.

Guido van Rossum (1991)

적합

데이터 분석  
머신 러닝, 딥 러닝

웹 개발(Flask)

업무 자동화

시스템 관리 스크립트

부적합

초고속 연산

모바일 앱 개발

-> Kotlin, Swift

게임 개발

-> Unity, Unreal



**PyCharm**

**Jupyter  
Notebook**

**VS Code**

- 독립적인 프로젝트를 위한 개별적 공간
- '의존성(dependencies)'과 '라이브러리(libraries)' 관리 용이
- `conda env list`
- `conda create --name myenv python=3.9`
- `conda activate myenv`
- `pip3 install jupyter notebook`

## Command Mode



- Enter : Edit Mode로 전환
- a : 위에 셀(Cell) 추가
- b : 아래에 셀(Cell) 추가
- m : Markdown으로 전환
- y : Code로 전환
- dd : 셀(Cell) 삭제

## Edit Mode



- Esc : Command Mode로 전환
- Ctrl + z : 되돌리기
- Ctrl + y : 앞으로 되돌리기

# 공통 단축키

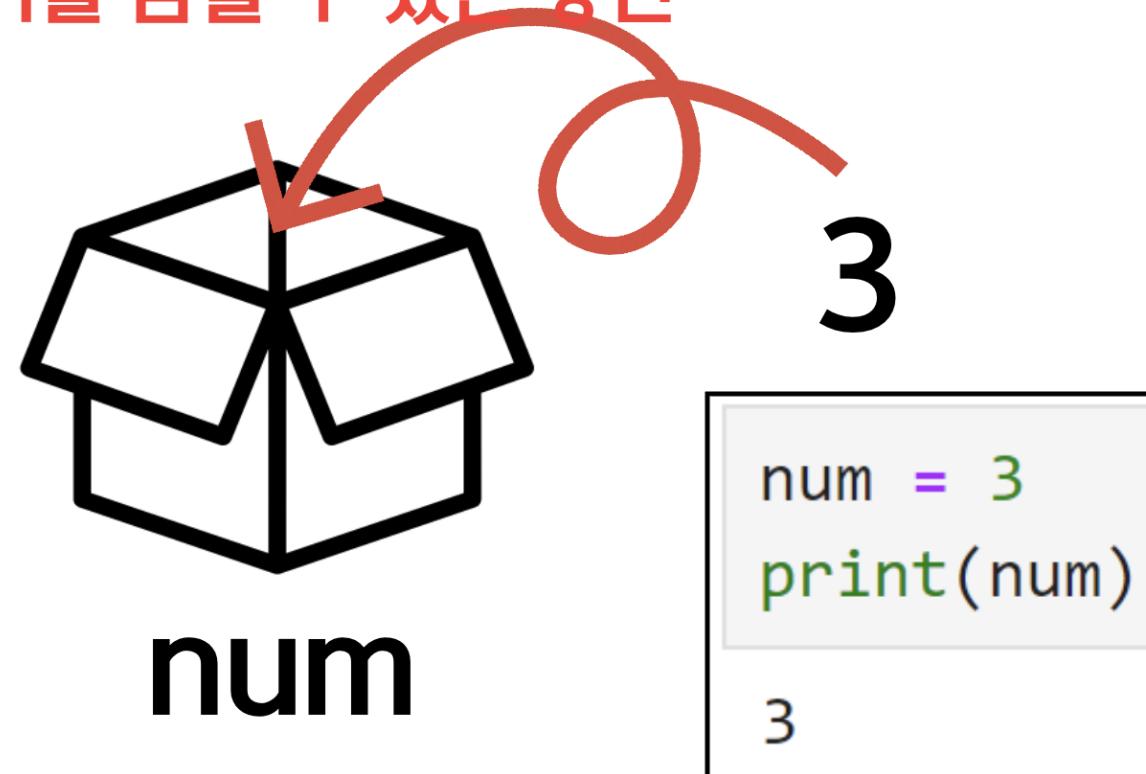
- Ctrl + Enter : 셀(Cell) 실행
- Alt + Enter : 셀(Cell) 실행 후 아래에 셀(Cell) 추가
- Shift + Enter : 셀(Cell) 실행 후 아래로 커서 이동

## 변수(variable)

- 사전적 의미로는 “**변화를 줄 수 있는**” 또는 “**변할 수 있는 수**”
- 프로그래밍에서는 **데이터를 담을 수 있는 공간**

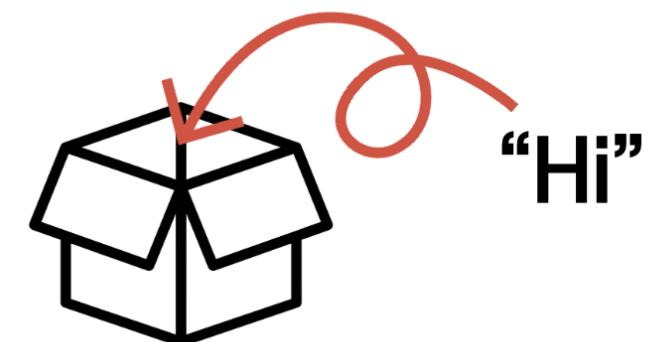
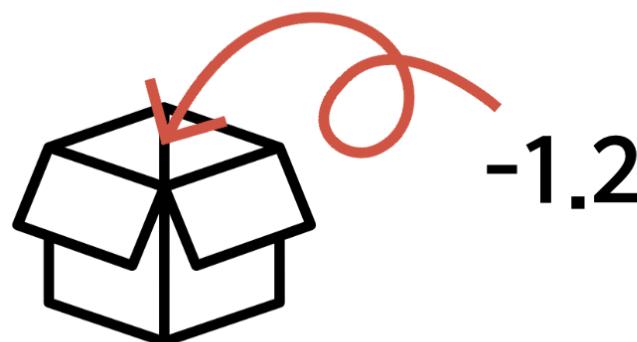
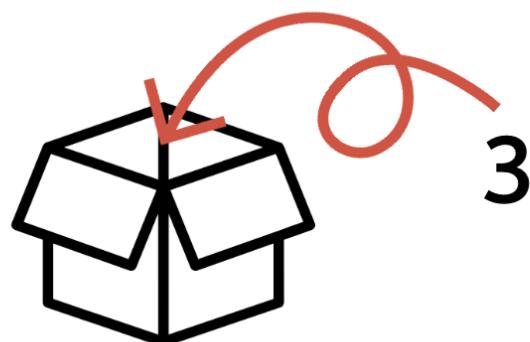
num = 3

변수명      대입      값



- 데이터 저장 : 재사용
- 가독성 : 유지보수 용이
- 동적 처리 가능
- 메모리 관리
- 모듈화

자료형	설명
숫자(정수)	-2, -1, 0, 1, 2
숫자(실수)	-3.2, 3.14, -0.12, 3.0
문자열	'Hello World!', "Hi", "123"
논리	True, False



- num1 변수에 숫자 2을 대입하시오
- num2 변수에 숫자 4를 대입하시오
- num2 변수에 숫자 20을 대입하시오
- num3 변수에 숫자 3.141592를 대입하시오
- num4 변수에 숫자 -1.25를 대입하시오
- num5 변수에 숫자 3.0을 대입하시오

- str1 변수에 문자열 “Hello World!” 를 대입하시오
- str2 변수에 문자열 'Hi Python' 을 대입하시오
- bool1 변수에 True를 대입하시오
- bool2 변수에 False를 대입하시오

# 1. 영문자, 숫자, Underscore(\_)를 사용할 수 있다

- 단, 영문자는 대소문자를 다르게 인식한다

## 2. 숫자로 시작할 수 없다

## 3. 키워드 사용이 불가하다

```
from keyword import kwlist
```

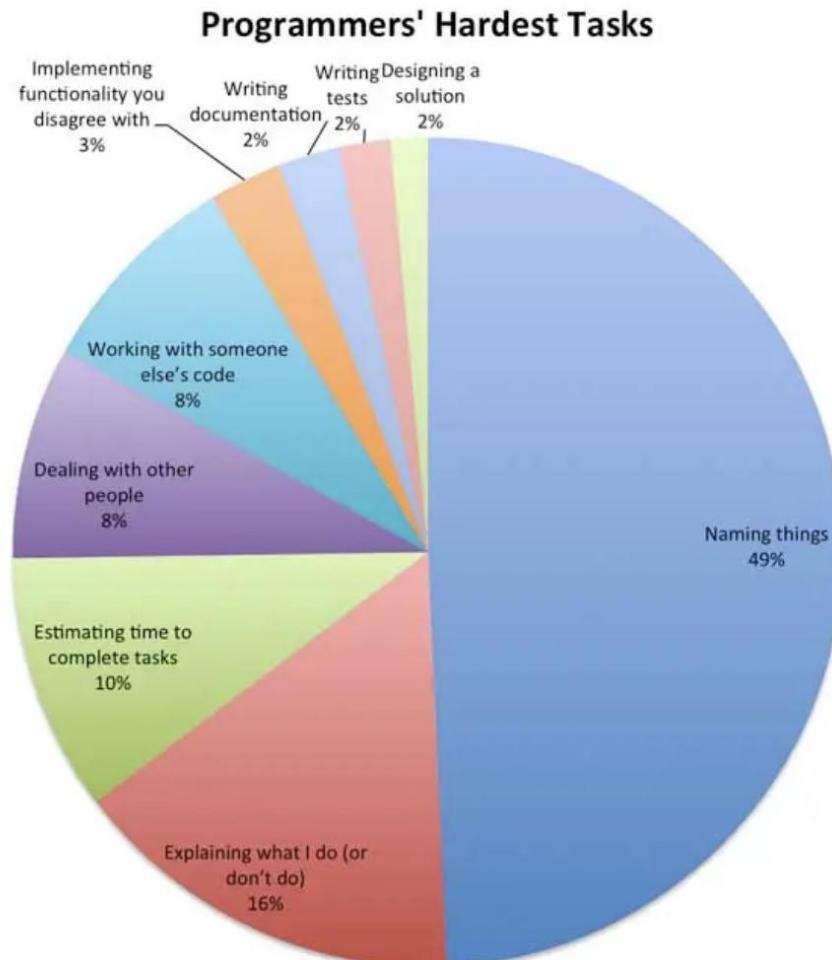
```
print(kwlist)
```

필수는 아니지만 **권장되는** 규칙들

1. 변수 이름의 **첫 글자는 소문자**로 시작한다
2. 여러 단어를 나열할 경우 표기법을 사용한다
  - Camel 표기법 : todayLunchMenu
  - Snake 표기법 : today\_lunch\_menu



# python 변수명 규칙(Variable Naming Rules)



*Data Source: Quora/Ubuntu Forums  
Total Votes: 4,522*

Naming things in code is *really* hard

CREDIT: IMAGE CREDIT: ITWORLD/PHIL JOHNSON

IT WORLD

“I do spend a lot of time as well worrying about the names of anything that can be given a name when I am programming.”  
**willcodejavaforfood**

"I have to agree that naming is an art." Otávio Décio

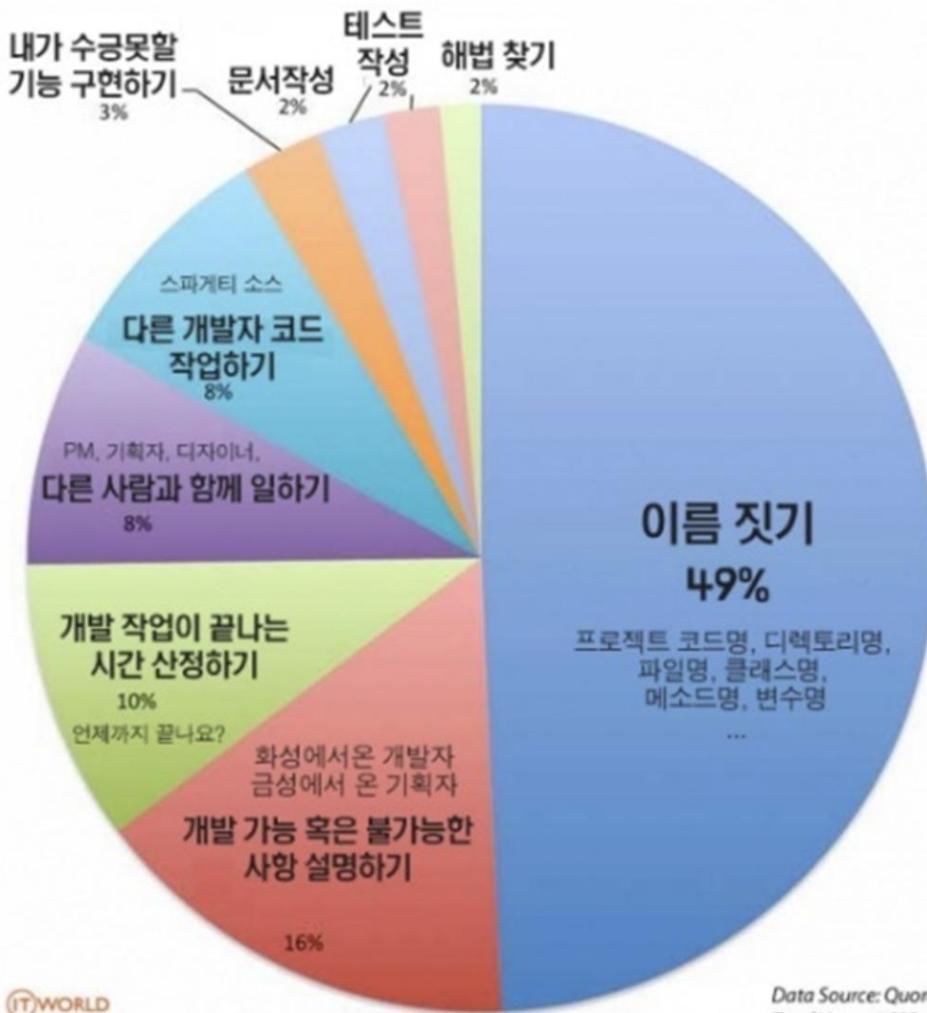
“...its one of the most important things if you want your code to be readable by others.” **Geries Handal**

“Often times, however, the inability to come up with a name may be a hint to something wrong with your design. Does your method have too many responsibilities? Does your class encapsulate a coherent idea?” **Brad Barker**

"One lesson I heave [sic] learned, is that if you can't find a name for a class, there is almost always something wrong with that class: you don't need it, it does too much" **Toon Krijthe**

"It's good that it's difficult. It's forcing you to think about the problem, and what the class is actually supposed to do." JW.

## 프로그래머가 가장 힘들어하는 일은?



"프로그래밍할 때 이름을 지을 수 있는 모든 것의 이름에 대해 걱정하는 데도 많은 시간을 보냅니다." [willcodejavaforfood](#)

"나는 명명이 예술이라는 데 동의해야 합니다." [오타비오 데시오](#)

"...다른 사람들이 코드를 읽을 수 있게 하려면 가장 중요한 것 중 하나입니다." [Geries Handal](#)

"그러나 종종 이름을 생각해 낼 수 없다는 것은 디자인에 문제가 있다는 힌트일 수 있습니다. 메서드에 책임이 너무 많습니까? 클래스가 일관된 아이디어를 캡슐화합니까?" [브래드 바커](#)

"내가 얻은 한 가지 교훈은 클래스 이름을 찾을 수 없다면 거의 항상 그 클래스에 문제가 있다는 것입니다. 필요하지 않은 것이고 너무 많은 일을 하는 것입니다." [Toon Krijthe](#)

"어려운 건 좋은 일이에요. 문제에 대해 생각하게 만들고, 수업이 실제로 무엇을 해야 하는지 생각하게 만드는 거죠." [JW.](#)

- 외부 모듈, 라이브러리를 불러오는 기능

Python 생태계 확장의 1등 공신

연산자	기호
산술 연산자	+ - * / // %
지수 연산자	**
대입(복합) 연산자	= += -= *= /= //= %=
관계(비교) 연산자	> >= < <= == !=
논리 연산자	not and or
삼항 연산자	a if 조건식 else b

문자열 안에 작은따옴표나 큰따옴표를 포함시키고 싶을 때

she's gone

```
s = 'she's gone'  
s
```

```
File "<ipython-input-17-c9d183e05d09>", line 1  
  s = 'she's gone'  
          ^
```

SyntaxError: invalid syntax

문자열 안에 작은따옴표나 큰따옴표를 포함시키고 싶을 때

she's gone

```
s1 = "she's gone"
s2 = "she\'s gone"
print(s1)
print(s2)
```

she's gone
she's gone

## 문자열 안에 작은따옴표나 큰따옴표를 포함시키고 싶을 때

he said that "she is gone"

```
s1 = 'he said that "she is gone"'
s2 = "he said that \"she is gone\""
print(s1)
print(s2)
```

```
he said that "she is gone"
he said that "she is gone"
```

## 여러 줄인 문자열을 변수에 대입하고 싶을 때

```
s = "여러줄로 구성된 \n문자열을 하나로 대입할 때"  
print(s)
```

여러줄로 구성된  
문자열을 하나로 대입할 때

## 이스케이프 코드

- 프로그래밍 할 때 사용할 수 있도록 **미리 정의해둔** “문자 조합”

코드	설명
\n	개행(줄바꿈)
\t	수평 탭
\w	문자 “\w”
\'	단일 인용부호( ' )
\”	이중 인용부호( “ ”)

## 인덱싱(indexing)

- 무엇인가를 '가리킨다'는 의미

## 슬라이싱(Slicing)

- 무엇인가를 '잘라낸다'는 의미

# python 문자열 인덱싱(indexing)과 슬라이싱(Slicing)

```
name = 'My name is YH'
```

M	y		n	a	m	e		i	s		Y	H
0	1	2	3	4	5	6	7	8	9	10	11	12
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

M	y		n	a	m	e		i	s		Y	H
0	1	2	3	4	5	6	7	8	9	10	11	12

```
name = 'My name is YH'  
print(name[0])  
print(name[8])
```

M	y		n	a	m	e		i	s		Y	H
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
name = 'My name is YH'  
print(name[-2])  
print(name[-1])
```

“My”, “name”, “YH” 문자열 가져오기

M	y		n	a	m	e		i	s		Y	H
0	1	2	3	4	5	6	7	8	9	10	11	12

```
name = 'My name is YH'  
print  
print  
print
```

My  
name  
YH

## “My name” 문자열 가져오기

M	y		n	a	m	e		i	s		Y	H
0	1	2	3	4	5	6	7	8	9	10	11	12

```
name = 'My name is YH'  
print(name[0:7])  
print
```

My name  
My name

## “is YH” 문자열 가져오기

M	y		n	a	m	e		i	s		Y	H
0	1	2	3	4	5	6	7	8	9	10	11	12
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
name = 'My name is YH'  
print(name[8:13])  
print(name[8:])  
print(name[-5:])
```

```
is YH  
is YH  
is YH
```

## 모든문자 가져오기

M	y		n	a	m	e		i	s		Y	H
0	1	2	3	4	5	6	7	8	9	10	11	12
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
name = 'My name is YH'  
print
```

My name is YH

yd\_info에 들어있는 문자열을 연도, 월일, 날씨로 구분하여 각각  
year, day, weather에 저장하여 아래와 같이 출력하시오.

**yd\_info = “2025 0205 Sunny”**

**year      day      weather**

결과창 ->

연도 :	2025
월일 :	0205
날씨 :	Sunny

다음과 같은 문자열에서 날짜와 날씨를 출력하시오.

```
s = "2025년 02월 05일의 날씨는 맑음입니다."
```

날짜 : 2025년 02월 05일

날씨 : 맑음

## 나누기, 나머지, 나누기(몫) 구하기

```
num1 = 10
num2 = 7

print(num1/num2)
print(num1%num2)
print(num1//num2)
```

실행결과 =>

## 문자열 더하기

```
str1 = "안녕"  
str2 = "하세요"  
  
print(str1 + str2)
```

안녕하세요

```
str1 = "10"  
str2 = "7"  
  
print(str1 + str2)
```

107

## 숫자, 문자열 더하기

```
num1 = 10
str2 = "7"
print(num1 + str2)
```

```
num1 = 10
str2 = "7"

print(str(num1) + str2)
print(num1 + int(str2))
```

107  
17

다음 코드를 완성하여 다음과 같은 결과를 출력하시오.

```
num1 = 23  
num2 = 3  
?
```

더하기 결과 : 26  
빼기 결과 : 20  
곱하기 결과 : 69  
나누기 결과 : 7.666666666666667

다음 코드에서 변수 num1과 num2를 키보드로 입력 받아서 결과를 계산하시오.

```
num1 = #키보드 입력  
num2 = #키보드 입력  
?
```

더하기 결과 : 26  
빼기 결과 : 20  
곱하기 결과 : 69  
나누기 결과 : 7.666666666666667

# python 키보드로 입력 받는 input() 사용 방법

1

```
num = input("정수를 입력하세요 >> ")
```

정수를 입력하세요 >>

2

```
num = input("정수를 입력하세요 >> ")
```

정수를 입력하세요 >>

3

```
num
```

'133'

문자  
열

## 문자를 숫자로 바꾸는 방법

- `int(문자열)` : 문자열을 정수로 변환
- `float(문자열)` : 문자열을 실수로 변환

```
num = int(input("정수를 입력하세요 >> "))
```

```
num
```

```
정수를 입력하세요 >> 123
```

```
123
```

다음 코드에서 변수 num1과 num2를 키보드로 입력 받아서 결과를 계산하시오.

```
num1 = ? #키보드 입력
num2 = ? #키보드 입력
?
```

```
정수를 입력하세요 >> 3
정수를 입력하세요 >> 7
더하기 결과 : 10
빼기 결과 : -4
곱하기 결과 : 21
나누기 결과 : 0.42857142857142855
```

Python, 머신러닝, 딥러닝 점수를 키보드로 입력 받아 합계와 평균을 출력하시오.

```
print("합계 : {}".format(?))
print("평균 : {}".format(?))
```

```
python 점수 입력 >> 100
머신러닝 점수 입력 >> 80 <= 입력 값
딥러닝 점수 입력 >> 60
합계 : 240
평균 : 80.0
```

변수 `number`을 입력 받아 백의 자리 미만을 버리는 코드를 완성하시오

```
number = int(input("정수 입력 >"))
print(??)
```

```
정수 입력 >> 456
400
```

38000원은 만원짜리 3장, 오천원짜리 1장, 천원짜리 3장으로 표현 가능하다.  
이처럼 입력 받은 금액을 최소 지폐 개수로 표현하는 코드를 작성하시오.

```
money = int(input("금액 입력 >>"))
money_10000 = ???
money_5000 = ???
money_1000 = ???
print("만원짜리", money_10000, "장")
print("오천원짜리", money_5000, "장")
print("천원짜리", money_1000, "장")
```

```
금액 입력 >> 38000
만원짜리 3 장
오천원짜리 1 장
천원짜리 3 장
```

## 문자열 포매팅(Formatting)

- 문자열 안의 특정한 값을 삽입해야 할 때 사용

```
str3 = "오늘은 2월 5일입니다."
```

```
str3 = "오늘은 2월 6일입니다."
```

- “%d”를 이용해서 정수 대입

```
day = 5
str4 = "오늘은 2월 %d일입니다."%day
print(str4)
```

오늘은 2월 5일입니다.

- 2개 이상 값을 포매팅 할 때

```
day = 5
month = 2
str5 = "오늘은 %d월 %d일입니다."%(month, day)
print(str5)
```

오늘은 2월 5일입니다.

## 문자열 포맷 코드

- 문자열 내 값 삽입

코드	설명
%s	문자열(string)
%c	문자 1개
%d	정수(Integer)
%f	실수(float-point)
%%	Literal % (문자 '%' 자체)

## format 함수를 사용한 포매팅

```
day = 5
month = 2
str6 = "오늘은 {}월 {}일입니다.".format(month, day)
print(str6)
```

오늘은 2월 5일입니다.

## f-string을 사용한 포매팅

```
day = 5
month = 2
str7 = f"오늘은 {month}월 {day}일입니다."
print(str7)
```

변수 `x`에는 100을 대입, 변수 `y`에는 200을 대입 후 변수 `add`에는 두 변수의 합을 대입하고 포매팅을 이용하여 아래와 같이 출력하시오.

```
x = 100
y = 200
add = x + y
print(?)
```

100와 200의 합은 300입니다

함수	설명
<code>count('문자')</code>	문자열에 포함된 문자 개수 세기
<code>find('문자')</code>	문자 위치 알려주기
<code>index('문자')</code>	문자 위치 알려주기
<code>join('문자')</code>	각각의 문자 사이에 '문자' 삽입하기
<code>upper()</code>	소문자를 대문자로 바꾸기
<code>lower()</code>	대문자를 소문자로 바꾸기
<code>lstrip()</code>	왼쪽 공백 지우기
<code>rstrip()</code>	오른쪽 공백 지우기
<code>strip()</code>	양쪽 공백 지우기
<code>replace('문자1', '문자2')</code>	문자열1을 문자열 2로 바꾸기
<code>split()</code>	문자열 나누기

초를 입력 받아 “00시간 00분 00초” 형태로 출력하시오.

```
time = int(input("시간 입력 >> "))
hour = ?
minute = ?
second = ?
print("{}시간 {}분 {}초".format(hour, minute, second))
```

시간 입력 >> 7533 2시간 5분 33초	시간 입력 >> 1123 0시간 18분 43초	시간 입력 >> 3723 1시간 2분 3초
-----------------------------	------------------------------	----------------------------

## 문자열 곱하기

```
s = "x"  
print(s*10)
```

xxxxxxxxxx

```
s = "안녕하세요"  
print(s*2)
```

안녕하세요안녕하세요

## 지수 연산자 (\*\*)

```
num = int(input("정수 입력 >>"))
power = int(input("지수 입력 >>"))
print("{}의 {}승은 {}입니다.".format(num, power, num**power))
```

```
정수 입력 >>2
지수 입력 >>3
2의 3승은 8입니다.
```

= (대입 연산자)

$a = 3$      $b = a + 1$      $b = b + 3$

$+ =$ ,  $- =$ ,  $* =$ ,  $/ =$ ,  $\% =$   
(복합 대입 연산자)

$a += b \rightarrow a = a + b$

$a -= 3 \rightarrow a = a - 3$

## 대입(복합) 연산자 실습

```
num = 27
```

```
num += 3
```

```
num
```

```
num = 27
```

```
num = num + 3  
      30
```

```
30
```



## 대입(복합) 연산자 실습

```
num = 27
```

```
num += 3
```

```
num += 3
```

```
num += 3
```

```
num
```

```
num = 27
```

```
num = num + 3 30
```

```
num = num + 3 33
```

```
num = num + 3 36
```

```
num
```

```
36
```

> >= < <=

== !=

같다

같지 않다

a > b a >= b

a == b a != b



결과값  
(True, False)

## 비교 연산자 실습

```
a = 3  
b = 7
```

print(a > b)	→	False
print(a <= b)	→	True
print(a == b)	→	False
print(a != b)	→	True

## 논리연산자

- 논리형(True, False)을 연산해주는 연산자

not

not 논리

and or

논리 and 논리 논리 or 논리

## 논리 연산자 not

- 논리값을 뒤집는 역할
- True → False
- False → True

a	not a
True	False
False	True

```
a = 3
b = 7
not a < b
      True
```

```
a = 3
b = 7
not a == b
      False
```

## 논리 연산자 and

- 두 값이 모두 True일 경우만 True

a	b	a and b
False	False	False
False	True	False
True	False	False
True	True	True

$3 > 5 \text{ and } 10 == 20$   
\_\_\_\_\_   
**False**      **False**

$3 > 5 \text{ and } 10 < 20$   
\_\_\_\_\_   
**False**      **True**

$3 < 5 \text{ and } 10 < 20$   
\_\_\_\_\_   
**True**      **True**



## 논리 연산자 or

- 두 값이 하나라도 True이면 True

a	b	a or b
False	False	False
False	True	True
True	False	True
True	True	True

$3 > 5 \text{ or } 10 == 20$

False

False

$3 > 5 \text{ or } 10 < 20$

False

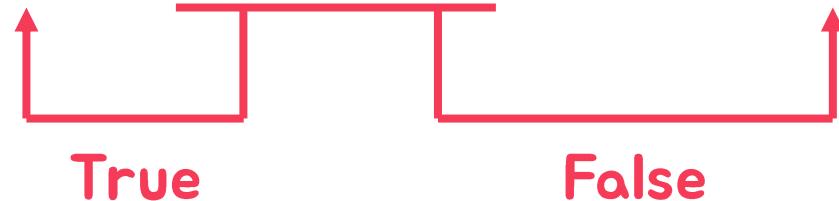
True

$3 < 5 \text{ or } 10 < 20$

True

True

a if 조건식 else b



```
score = 80
"합격" if score >= 60 else "불합격"
```

True

```
score = 50
"합격" if score >= 60 else "불합격"
```

False

키보드로 정수를 입력 받아 홀수인지 짝수인지 판별하시오

?

정수 입력 >> 33  
33는(은) 홀수입니다.

?

정수 입력 >> 22  
22는(은) 짝수입니다.

두 개의 정수를 입력 받아 큰 수에서 작은 수를 뺀 결과값을 출력하  
시오.

?

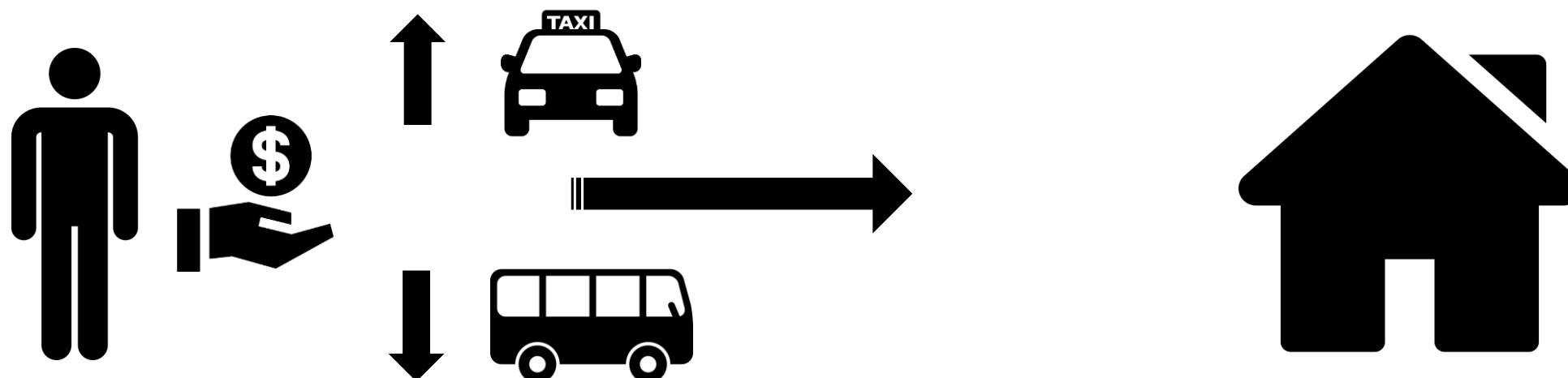
```
첫 번째 정수 입력 >> 5
두 번째 정수 입력 >> 10
두 수의 차 : 5
```

?

```
첫 번째 정수 입력 >> 33
두 번째 정수 입력 >> 5
두 수의 차 : 28
```

## 조건문

- 조건에 따라 실행 흐름을 다르게 하는 문법

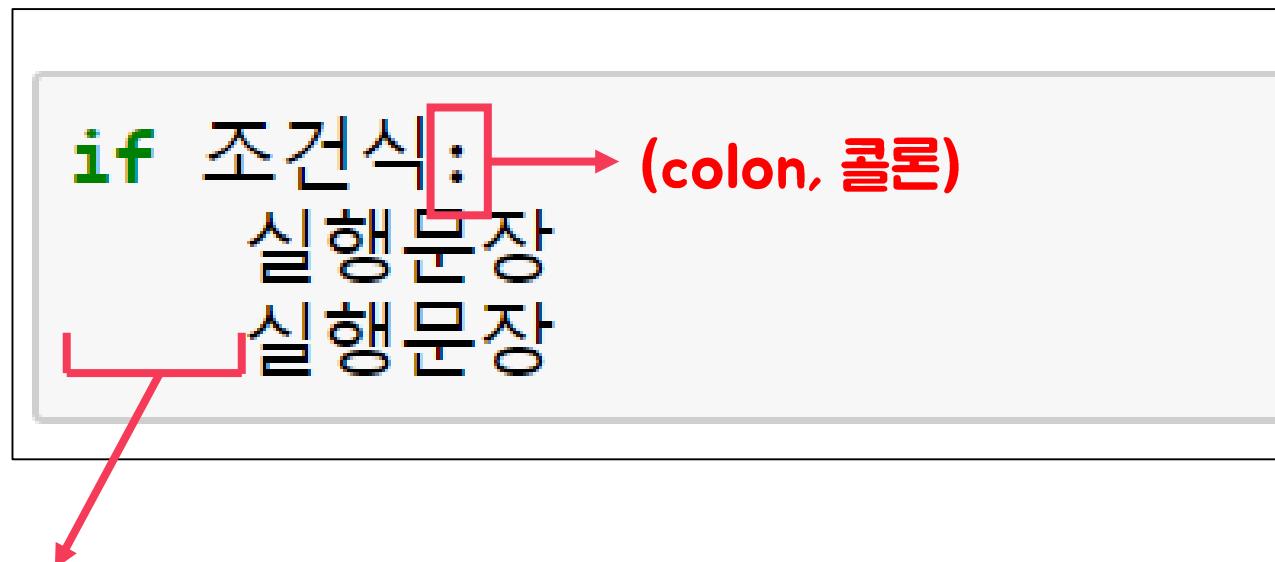


if

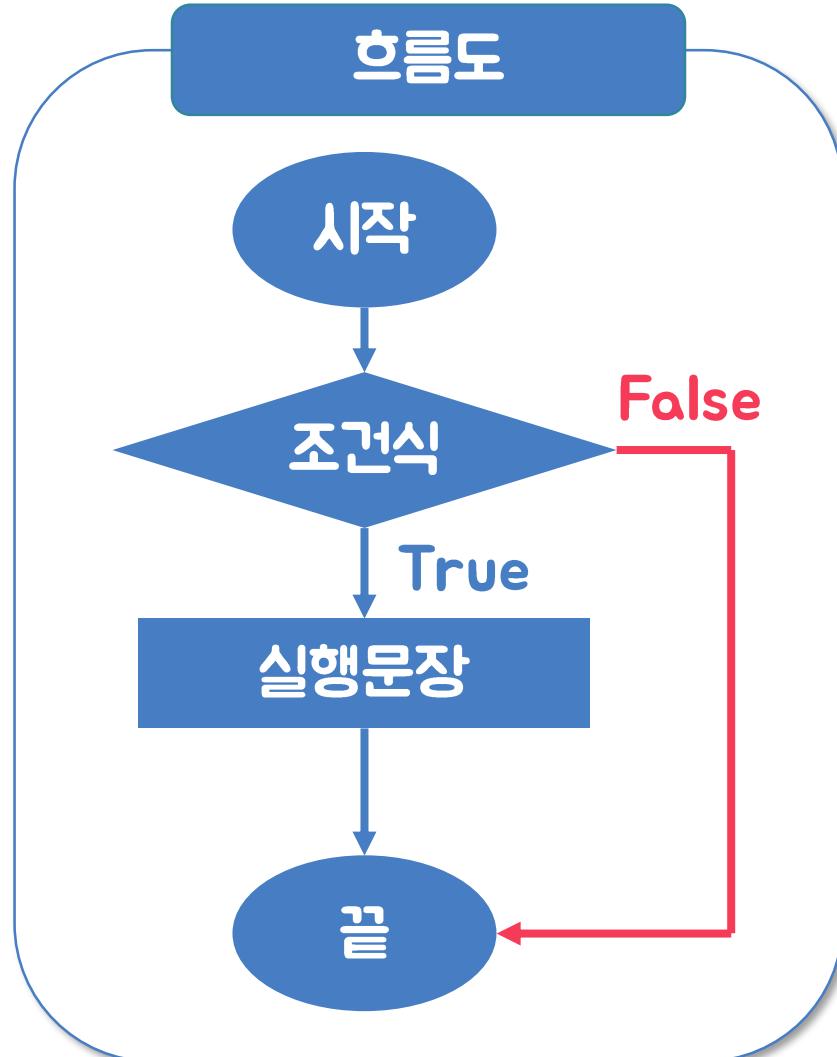
elif

else

## 조건식이 True일 경우 실행문장 실행



들여쓰기 (Tab, Space\*4)



```
if True:  
    print("실행문장 실행")
```

실행문장 실행

```
if False:  
    print("실행문장 실행")
```

```
if True:  
    print("실행문장 실행")  
print("if문 밖에 있는 실행문장")
```

실행문장 실행  
if문 밖에 있는 실행문장

```
if False:  
    print("실행문장 실행")  
print("if문 밖에 있는 실행문장")
```

if문 밖에 있는 실행문장

조건문을 사용하여 변수 money가 100000이상이면 “택시를 탄다”를 출력하시오. (비교연산자 이용)

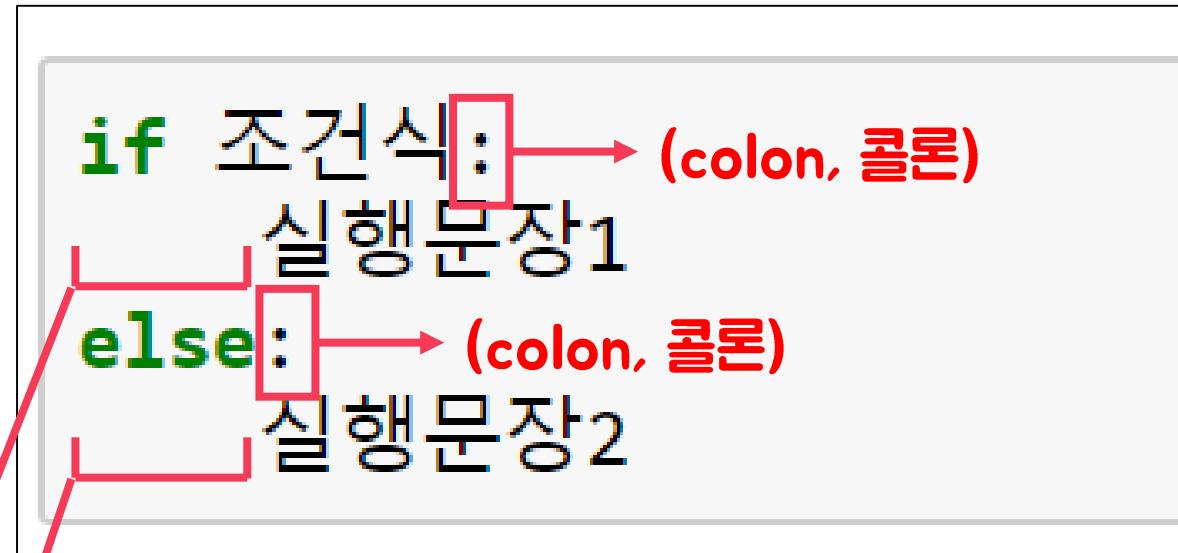
```
money = 11000
?
```

택시를 탄다.

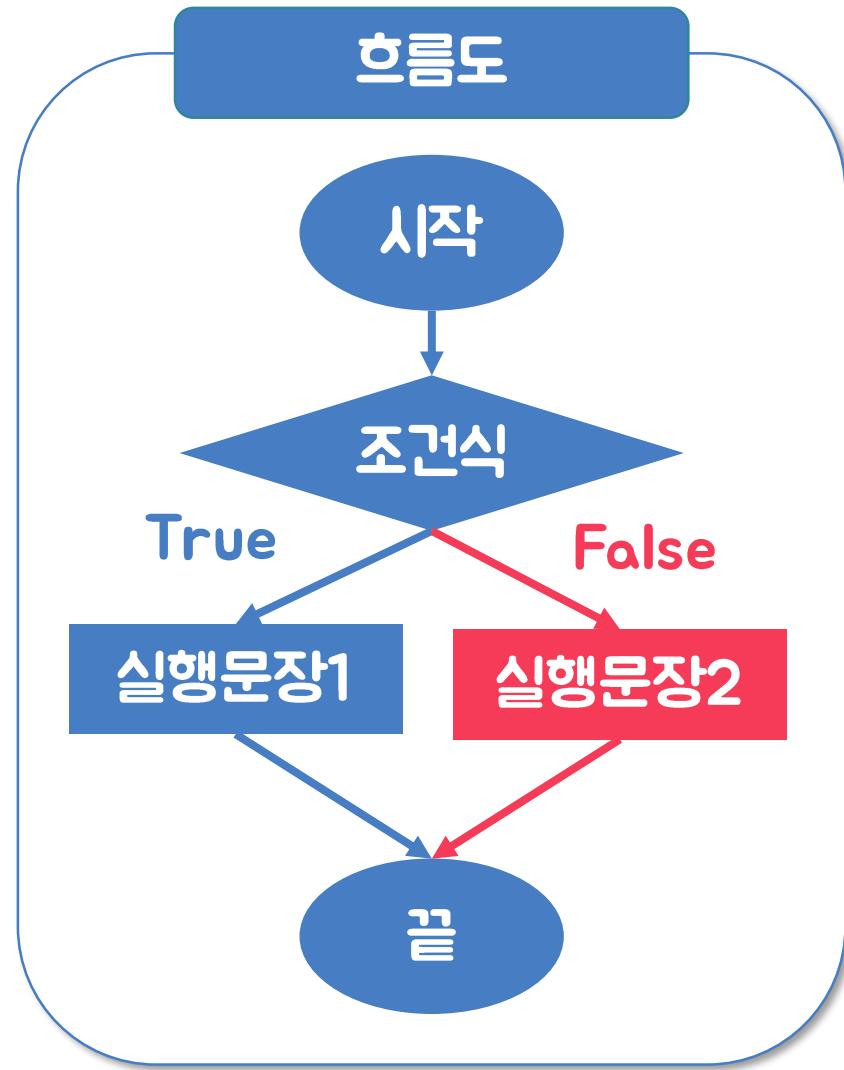
```
money = 9000
?
```

조건식이 **True**일 경우 실행문장1 실행

조건식이 **False**일 경우 실행문장2 실행



들여쓰기 (Tab, Space\*4)



```
if True:  
    print("실행문장1")  
else:  
    print("실행문장2")
```

실행문장1

```
if False:  
    print("실행문장1")  
else:  
    print("실행문장2")
```

실행문장2

조건문을 사용하여 변수 money가 100000이상이면 “택시를 탄다”를 출력하고 10000미만이면 “버스를 탄다“를 출력하시오.

```
money = 11000
```

```
?
```

택시를 탄다.

```
money = 9000
```

```
?
```

버스를 탄다.

키보드로 변수 num을 입력 받고 num이 3의 배수이면서 5의 배수이면 “3과 5의 배수입니다”를 출력하고 아니라면 “3과 5의 배수가 아닙니다“를 출력하시오

```
num = ?  
?
```

정수 입력 >> 30  
3과 5의 배수입니다.

```
num = ?  
?
```

정수 입력 >> 7  
3과 5의 배수가 아닙니다.

마트 계산대 프로그램입니다.  
10000원짜리 추석선물세트를 구입했을 때 지불해야하는 금액을  
계산해보세요. 단 11개 이상 구매시에는 10% 할인이 됩니다.

?

사려는 상품의 갯수를 입력하세요 >> 9  
가격은 90000원 입니다

?

사려는 상품의 갯수를 입력하세요 >> 12  
가격은 108000원 입니다

조건식이 **True**일 경우 실행문장1 실행

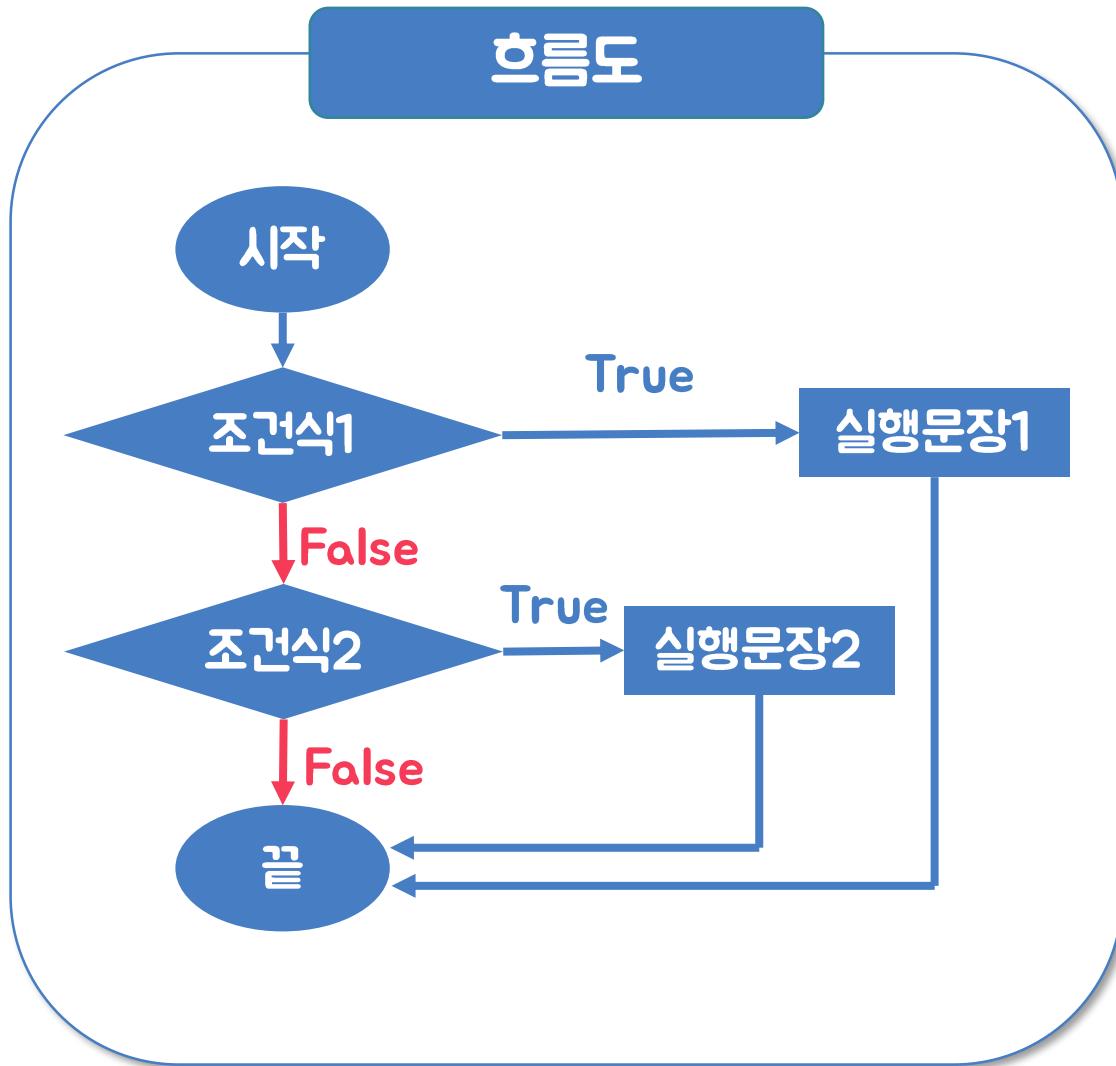
조건식이 **False**일 경우 다음 조건식 확인

```
if 조건식1:  
    실행문장1  
elif 조건식2:  
    실행문장2
```

```
if 조건식1:  
    실행문장1  
elif 조건식2:  
    실행문장2  
else:  
    실행문장3
```

```
if 조건식1:  
    실행문장1  
elif 조건식2:  
    실행문장2  
elif 조건식3:  
    실행문장3  
else:  
    실행문장4
```

# python 조건문 elif



```
if True:  
    print("실행문장1")  
elif True:  
    ...  
  
if False:  
    print("실행문장1")  
elif False:  
    print("실행문장2")  
elif True:  
    print("실행문장2")  
  
실행문장2
```

변수 num1과 num2에 숫자를 입력 받아 크기를 비교하시오.

```
num1 = ?  
num2 = ?  
?
```

```
첫 번째 정수 입력 >> 5  
두 번째 정수 입력 >> 3  
첫 번째 정수가 더 큽니다.
```

```
num1 = ?  
num2 = ?  
?
```

```
첫 번째 정수 입력 >> 7  
두 번째 정수 입력 >> 13  
두 번째 정수가 더 큽니다.
```

```
num1 = ?  
num2 = ?  
?
```

```
첫 번째 정수 입력 >> 7  
두 번째 정수 입력 >> 7  
두 수가 똑같습니다.
```

변수 score에 점수를 입력 받아서 다음과 같이 학점을 부여하시오.

100이하, 90이상 → A

90미만, 80이상 → B

80미만, 70이상 → C

70미만, 60이상 → D

60미만 → F

```
score = ?  
?
```

점수 입력 >> 98  
98점은 A학점 입니다.

점수 입력 >> 70  
70점은 C학점 입니다.

점수 입력 >> 36  
36점은 F학점 입니다.

## 자판기 프로그램을 만들어보자.

- 메뉴 출력
- 돈 입력
- 메뉴 선택
- 결과 확인
  - 잔액부족 출력
  - 잔액 환전 (5000원, 1000원, 500원, 100원)

금액을 입력하세요 : 8500

[1]음료1(1000원) [2]음료2(700원) [3]음료3(500원)

메뉴를 고르세요 1

잔돈 : 7500원

오천원 : 1장

천원 : 2장

오백원 : 1개

## 리스트(list) 란?

- 파이썬의 자료구조 형태중 하나
- 순서가 있는 수정 가능한 객체의 집합
- 대괄호( [ ] )로 작성되어지며, 리스트 내부의 값은 콤마( , )으로 구분
- 추가, 수정, 삭제 가능

## 리스트명 =

```
a = [ ]
```

```
b = [1, 2, 3]
```

```
c = ['My', 'name', 'is', 'YH']
```

```
d = [1, 2, 'My', 'name']
```

```
e = [1, 2, ['My', 'name']]
```

## 인덱싱(indexing)

- 무엇인가를 '가리킨다'는 의미

## 슬라이싱(Slicing)

- 무엇인가를 '잘라낸다'는 의미

## 리스트 [인덱스]

- 인덱스에 위치한 값 반환

```
list1 = [2, 5, 7, 9, 10]
print(list1[0])
print(list1[3])
print(list1[2]+list1[-1])
```

```
list2 = [1, 2, 3, ['a', 'b', 'c']]
```

- 변수 temp에 ['a', 'b', 'c']를 저장하고 출력하시오.
- list2에서 문자 'b'만 뽑아서 출력하시오.

## 리스트[**start** 인덱스 : **end** 인덱스]

- **start** 인덱스부터 **end** 인덱스 바로 전까지 값 반환 (**start**  $\leq$  **x** < **end**)

```
list3 = [0, 1, 2, 3, 4]  
list3[1:3]
```

```
[1, 2]
```

```
list3[:2]
```

```
[0, 1]
```

```
list3[3:]
```

```
[3, 4]
```

```
list3[3:4]
```

```
[3]
```

```
list4 = [1, 2, 3]
list5 = [3, 4, 5, 6]
```

```
[1, 2, 3, 3, 4, 5, 6]
```

## 리스트.append(값)

- 맨 뒤에 값 추가

```
list5 = [0, 1, 2, 3, 4]
list5.append(5)
list5
```

```
[0, 1, 2, 3, 4, 5]
```

```
list5.append(6)
list5
```

```
[0, 1, 2, 3, 4, 5, 6]
```

## 리스트.insert(인덱스, 값)

- 인덱스 위치에 값 추가

```
list5 = [0, 1, 2, 3, 4]
list5.insert(1, 5)
list5
```

```
[0, 5, 1, 2, 3, 4]
```

```
list5.insert(5, 6)
list5
```

```
[0, 5, 1, 2, 3, 6, 4]
```

# python 리스트(list) 값 수정



```
list6 = [0, 1, 2, 3, 4]  
list6
```

```
[0, 1, 2, 3, 4]
```

```
print("수정 전 :",list6[1])  
list6[1] = 7  
print("수정 후 :",list6[1])
```

```
수정 전 : 1  
수정 후 : 7
```

```
list6
```

```
[0, 7, 2, 3, 4]
```

```
print(list6[2:4], list6[2:3])  
list6[2:4] = 7
```

```
[7, 4] [7]
```

```
-----  
TypeError
```

```
t)
```

```
<ipython-input-17-c640f2364ee6>  
    1 print(list6[2:4], list6[  
----> 2 list6[2:4] = 7
```

```
TypeError: can only assign an it
```

```
print(list6[2:4])  
list6[2:4] = [7]  
list6
```

```
[2, 3]
```

```
[0, 7, 7, 4]
```

## del 키워드 이용

```
list7 = [0, 1, 2, 3, 4, 5]
del list7[1]
list7
```

```
[0, 2, 3, 4, 5]
```

```
list7 = [0, 1, 2, 3, 4, 5]
del list7[1:5]
list7
```

```
[0, 5]
```

## 리스트.remove(값)

```
list7 = ['a','b','c','d','e']
list7.remove('b')
list7
```

```
['a', 'c', 'd', 'e']
```

```
list7.remove('b')
```

```
-----
ValueError
t)
```

```
<ipython-input-42-ac74ba81fef3
----> 1 list7.remove('b')
```

```
ValueError: list.remove(x): x
```

## 리스트.sort()

- 리스트에 있는 값을 오름차순으로 정렬

```
list8 = [9, 77, 13, 51, 100, 3]
list8
[9, 77, 13, 51, 100, 3]

list8.sort()
list8
[3, 9, 13, 51, 77, 100]
```

## 리스트.reverse()

- 리스트에 있는 값을 역순으로 뒤집음

```
list9 = [9, 77, 13, 51, 100, 3]
```

```
list9
```

```
[9, 77, 13, 51, 100, 3]
```

```
list9.reverse()
```

```
list9
```

```
[3, 100, 51, 13, 77, 9]
```

## 리스트.sort() 와 리스트.reverse() 이용

- 리스트에 있는 값을 내림차순으로 정렬

```
list10 = [9, 77, 13, 51, 100, 3]  
list10
```

```
[9, 77, 13, 51, 100, 3]
```

```
list10.sort()  
list10
```

```
[3, 9, 13, 51, 77, 100]
```

```
list10.reverse()  
list10
```

```
[100, 77, 51, 13, 9, 3]
```

## 리스트.index()

- 찾고자 하는 값의 위치 반환

```
list11 = ['a', 'b', 'c', 'd', 'e', 'f']
list11.index('c')
```

2

## 리스트.pop()

- 마지막 값을 반환 후 리스트에서 제거

```
list12 = ['a', 'b', 'c', 'd', 'e', 'f']
list12.pop()
```

'f'

```
list12
```

```
['a', 'b', 'c', 'd', 'e']
```



## len(리스트)

- 리스트의 값 개수 반환

```
list13 = [0, 1, 2]
len(list13)
```

3

```
list14 = ['a','b','c','d','e','f']
len(list14)
```

6

## 튜플(tuple) 이란?

- 파이썬의 자료구조 형태중 하나
- 순서가 있는 집합
- 소괄호(())로 작성되어지며, 튜플의 내부 값은 콤마(,)으로 구분
- 추가, 수정, 삭제 불가능

튜플명 = (요소1, 요소2, 요소3, ...)

```
a = ()  
b = (1, 2, 3)  
c = ('My', 'name', 'is', 'YH')  
d = (1, 2, 'My', 'name')  
e = (1, 2, ('My', 'name'))
```

## 튜플[인덱스]

- 인덱스에 위치한 값 반환

```
tuple1 = (0, 1, 2, 3, ('a','b','c'), 5)
```

```
tuple1[2]
```

```
2
```

```
tuple1[4]
```

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

## 튜플[start 인덱스 : end 인덱스]

- start 인덱스부터 end 인덱스 바로 전까지 값 반환 (start  $\leq$  x < end)

```
tuple1 = (0, 1, 2, 3, ('a','b','c'), 5)
```

```
tuple1[1:3]
```

```
(1, 2)
```

```
tuple1[3:]
```

```
(3, ('a', 'b', 'c'), 5)
```

## len(튜플)

- 튜플의 값 개수 반환

```
tuple1 = (0, 1, 2, 3, ('a', 'b', 'c'), 5)
len(tuple1)
```

6

```
tuple2 = ('a', 'b', 'c', 'd', 'e', 'f')
len(tuple2)
```

6

## 튜플은 추가, 수정, 삭제 불가능

```
tuple1 = (0, 1, 2, 3, ('a','b','c'), 5)
```

```
tuple1[0] = 3
```

```
---
```

```
TypeError
```

```
t)
```

```
<ipython-input-53-5e0f22de5ab3> in <module>
```

```
----> 1 tuple1[0] = 3
```

```
Traceback (most r
```

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

## 공통점

- 타입과 상관 없이 일련의 **요소(Element)**를 갖을 수 있다.
- 요소의 순서를 관리한다.

## 차이점

- 리스트는 **가변적(mutable)**이며, 튜플은 **불변적(immutable)**
- 리스트는 요소가 몇 개 들어갈지 명확하지 않은 경우에 사용
- 튜플은 요소 개수를 사전에 정확히 알고 있을 경우에 사용

in : 찾고자 하는 값(x)이 포함되어 있으면 True

not in : 찾고자 하는 값(x)이 포함되어 있지 않으면 True

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

```
str1 = "파이썬 최고"  
  
"파이썬" in str1  
  
True  
  
"파이썬" not in str1  
  
False
```

```
list1 = [77, 38, 10]  
  
33 in list1  
  
False  
  
33 not in list1  
  
True
```

- 프로그램 내에서 똑같은 명령을 일정 횟수만큼 **반복하여 수행**하도록 제어하는 명령문
- 반복문 종류는 **while**문, **for**문이 있다.



**while**

: 반복 횟수가 명확하지 않을 때

**for**

: 반복 횟수가 명확할 때

1부터 1000까지 출력하시오.

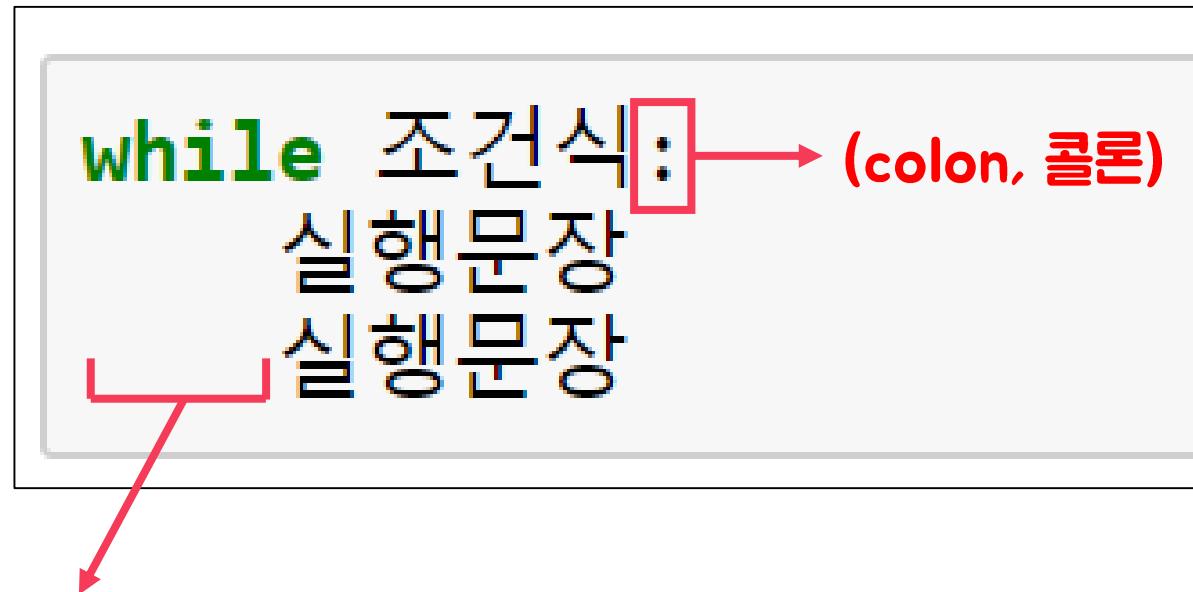
```
print(1)
print(2)
print(3)
print(4)
print(5)
print(6)
...
...
```

1  
2  
3  
4  
5  
6

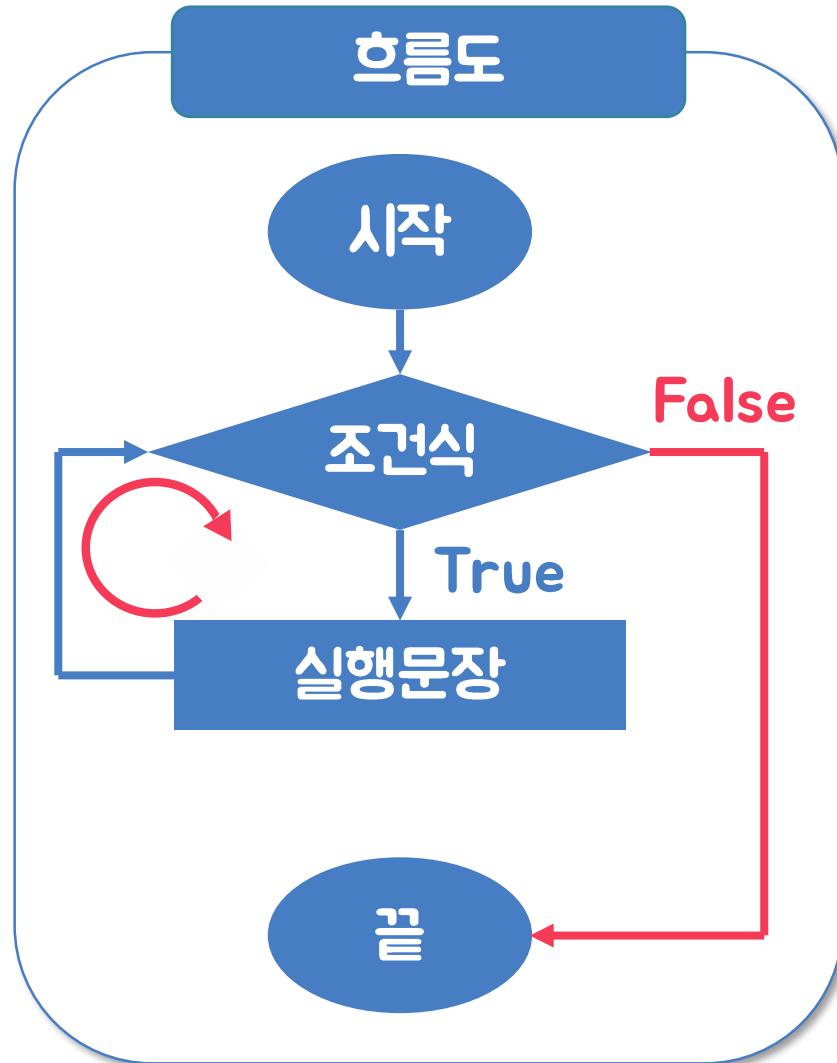
```
for i in range(1,1001):
    print(i)
```

1  
2  
3  
4  
996  
997  
998  
999  
1000

## 조건식이 True일 경우 실행문장 반복



들여쓰기 (Tab, Space\*4)



1부터 3까지 출력하시오.

```
number = 1 ①
while number <= 3: ② ⑤ ⑧ ⑪
    print(number) ③ ⑥ ⑨
    number += 1 ④ ⑦ ⑩
```



**while**문을 사용해서 “파이썬 최고!!“를 13번 출력하시오.

## while 조건식: 실행문장

```
num = 0
while num < 10:
    print("파이썬 최고!!")
    num += 1
```

## break

- 반복문을 나가는 기능

```
while True:  
    print("무한루프")
```

무한루프  
무한루프

```
while True:  
    print("무한루프")  
    break
```

무한루프

```
number = 1  
while True:  
    print(number)  
    number += 1  
    if number > 3:  
        break
```

1  
2  
3

두개의 정수를 입력 받아서 더하는 코드를 작성하시오.  
(단, 두개의 정수가 0이 들어올 때 까지 반복한다.)

첫 번째 정수 입력 >> 1
두 번째 정수 입력 >> 2
두 정수의 합 : 3
첫 번째 정수 입력 >> 7
두 번째 정수 입력 >> 3
두 정수의 합 : 10
첫 번째 정수 입력 >> 13
두 번째 정수 입력 >> 77
두 정수의 합 : 90
첫 번째 정수 입력 >> 0
두 번째 정수 입력 >> 0
프로그램이 종료되었습니다.

## 다이어트 관리 프로그램

1. 현재 몸무게와 목표몸무게를 입력 받고 주차 별 감량 몸무게를 입력 받으세요.
2. 목표몸무게를 달성하면 축하한다는 문구를 출력하고 입력을 멈추세요!

```
현재 몸무게 : 80
목표 몸무게 : 70
1주차 감량 몸무게 : 2
2주차 감량 몸무게 : 3
3주차 감량 몸무게 : 4
4주차 감량 몸무게 : 5
66 kg 달성!! 축하합니다!
```

랜덤으로 1부터 50사이의 숫자를 뽑으면 뽑은 숫자를 맞추는 Up,  
Down 게임 예제

```
숫자를 입력하세요 >> 25
25보다 작은 수입니다.
숫자를 입력하세요 >> 13
13보다 작은 수입니다.
숫자를 입력하세요 >> 5
5보다 큰 수입니다.
숫자를 입력하세요 >> 8
8보다 작은 수입니다.
숫자를 입력하세요 >> 6
6보다 큰 수입니다.
숫자를 입력하세요 >> 7
정답을 맞추셨습니다.
```

## 라이브러리 import

```
import random
```

## random.randint() 사용

- 1~10 사이의 숫자 랜덤 추출

```
random.randint(1, 10)
```

5

랜덤으로 1부터 50사이의 숫자를 뽑으면 뽑은 숫자를 맞추는 Up,  
Down 게임 예제

```
숫자를 입력하세요 >> 25
25보다 작은 수입니다.
숫자를 입력하세요 >> 13
13보다 작은 수입니다.
숫자를 입력하세요 >> 5
5보다 큰 수입니다.
숫자를 입력하세요 >> 8
8보다 작은 수입니다.
숫자를 입력하세요 >> 6
6보다 큰 수입니다.
숫자를 입력하세요 >> 7
정답을 맞추셨습니다.
```

문자열 또는 리스트 또는 튜플이 들어갔을 때  
안에 있는 요소를 **하나씩 반복**

```
for 변수 in 문자열(or 리스트 or 튜플):
```

```
    print(변수)
```

들여쓰기 (Tab, Space\*4)

(colon, 콜론)

## for문 예시

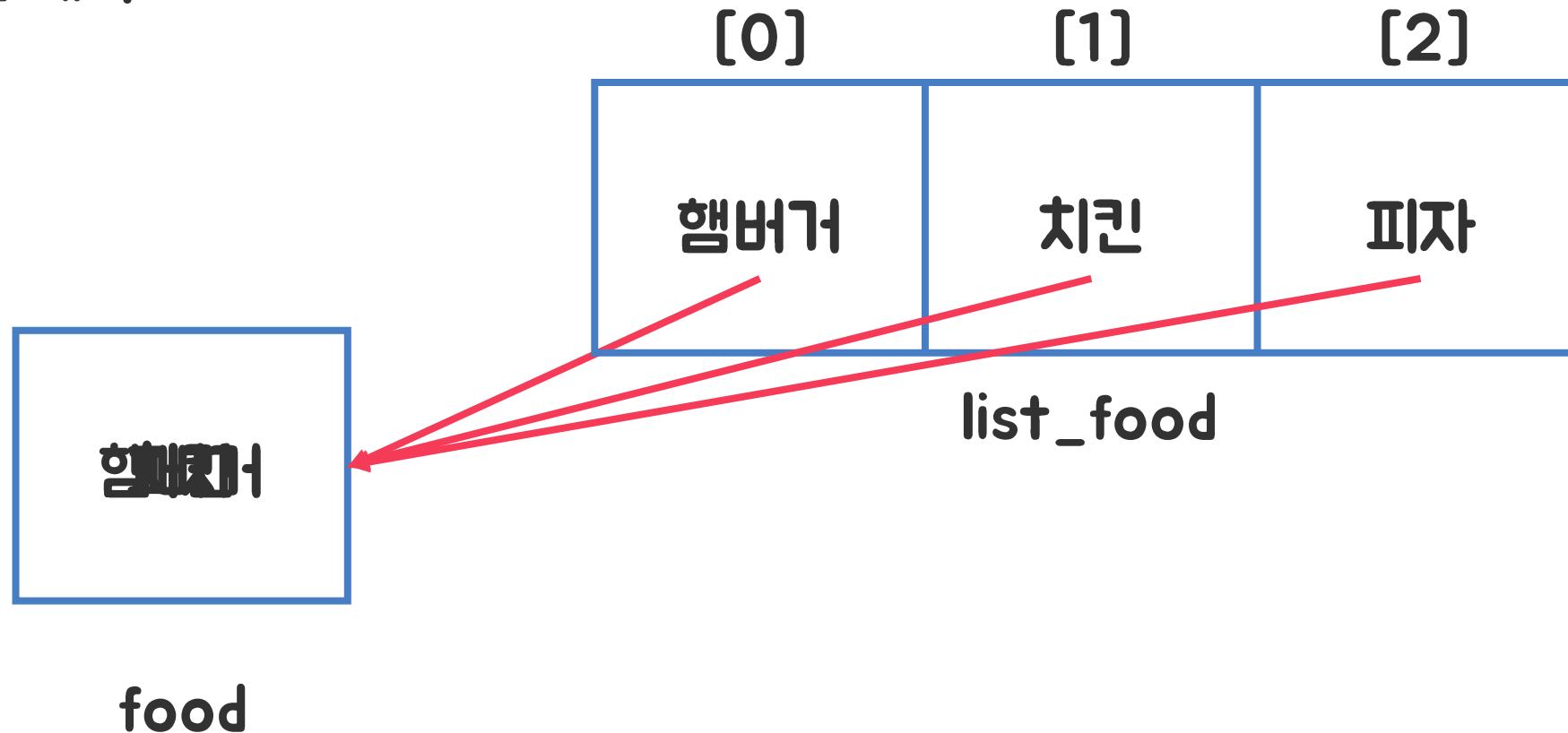
```
list_food = ["햄버거", "치킨", "피자"]
for food in list_food:
    print(food)
```

①

② ④ ⑥

③ ⑤ ⑦

## for문 예시



## for문 예시

```
hi = "안녕하세요"  
for s in hi:  
    print(s)
```

안  
녕  
하  
세  
요

```
tuple_food = ("햄버거", "치킨", "피자")  
for food in tuple_food:  
    print(food)
```

햄버거  
치킨  
피자

1. 1부터 100사이의 숫자 중 3의 배수인 값들의 합을 출력하세요.

정답 : 1683

2. for문을 이용하여 구구단 2단을 출력하시오.

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18

For문을 이용하여 다음 list에 들어있는 요소 중  
**가장 큰 수**를 찾아 출력하세요.

```
list2 = [4,5,2,1,99,15,2,7,27]
```

```
?
```

```
99
```

For문을 이용하여 다음 list에 들어있는 요소 중  
**가장 작은 수를 찾아 출력하세요.**

```
list2 = [4,5,2,1,99,15,2,7,27]  
?
```

1

5명에 대한 정보처리기사 자격증 시험 점수가 리스트에 담겨있습니다. 이때 각 점수가 합격 점수인지 불합격 점수인지 판별하여 출력하시오.

(60점 이상 합격)

```
score_list = [90, 45, 70, 60, 55]  
?
```

- 1번 학생은 합격입니다.
- 2번 학생은 불합격입니다.
- 3번 학생은 합격입니다.
- 4번 학생은 합격입니다.
- 5번 학생은 불합격입니다.

## range() 함수 사용

- 필요한 만큼의 숫자를 만들어내는 유용한 기능
- range(**시작할 숫자, 종료할 숫자, 증가량**)
- range(1, 10, 1) → 1부터 9까지 1씩 증가
- range(1, 100, 3) → 1부터 99까지 3씩 증가
- range(10, 1, -1) → 10부터 2까지 1씩 감소(-1씩 증가)

## range() 함수 사용

```
for i in range(1, 10, 1):  
    print(i)
```

1

2

3

4

5

6

7

8

9

[1, 2, 3, 4, 5, 6, 7, 8, 9]

## print() 함수

- end 속성

```
for i in range(1, 10, 1):
    print(i, end=" ")
```

1 2 3 4 5 6 7 8 9

```
for i in range(1, 10, 1):
    print(i, end="\n")
```

1  
2  
3  
4  
5  
6  
7  
8  
9

## range() 함수 사용

- 필요한 만큼의 숫자를 만들어내는 유용한 기능
- range(**기본값 0**, 종료할 숫자, **기본값 1**)
- range(3, 10) → 3부터 9까지 1씩 증가
- range(10) → 0부터 9까지 1씩 증가

```
for i in range(3, 10):  
    print(i, end=" ")
```

3 4 5 6 7 8 9

```
for i in range(10):  
    print(i, end=" ")
```

0 1 2 3 4 5 6 7 8 9

1. for문을 이용하여 97부터 77까지 출력하시오.

?

97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77

2. for문을 이용하여 23부터 40까지 출력하시오

?

23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

```
list1 = [[1,2],[3,4],[5,6]]  
for i,j in list1:  
    print(i, j)
```

1 2  
3 4  
5 6

```
a, b = 1, 7  
print(a)  
print(b)
```

1  
7

두개의 정수를 키보드로 입력 받아 첫 번째 정수부터 두 번째 정수까지 출력되는 소스코드를 작성하시오.

```
start = ?  
end = ?  
?
```

```
첫 번째 정수 입력 >> 10  
두 번째 정수 입력 >> 30  
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

숫자를 입력 받고 입력 받은 숫자의 **약수**를 구하시오.  
(약수란 어떤 수를 나누어 떨어지게 하는 수)

```
num = int(input("정수 입력 > "))  
?
```

```
정수 입력 >> 32  
32의 약수 : 1 2 4 8 16 32
```

## 별을 찍어보자

```
*  
**  
***  
****  
*****
```

```
*****  
****  
***  
**  
*
```

```
*  
**  
***  
****  
*****
```

```
*****  
****  
***  
**  
*
```



# 별을 찍어보자

\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*

\*\*  
\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*



# 별을 찍어보자

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*