



대한상공회의소  
서울기술교육센터

나 예 호 교수

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 대입 하고 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
\\	문자 "\"
'\''	단일 인용부호( ' )
'\"'	이중 인용부호( " )

## 인덱싱(indexing)

- 무엇인가를 '가리킨다'는 의미

## 슬라이싱(Slicing)

- 무엇인가를 '잘라낸다'는 의미

```
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 가져오  
기

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** = "20250205Sunny"  
                    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 : 7.6666666666666667

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  
딥러닝 점수 입력 >> 60  
합계 : 240  
평균 : 80.0
```

<= 입력 값

변수 number을 입력 받아 백의 자리 미만을 버리는 코드를 완성하십시오

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

```
정수 입력 >> 456  
400
```

38000원은 만원짜리 3장, 오천원짜리 1장, 천원짜리 3장으로 표현 가능하다.

이처럼 입력  
성하시오.

```
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

하는 코드를 작

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

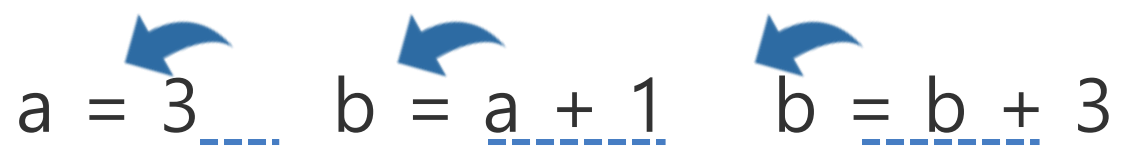
## 지수 연산자 (\*\*)

```
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)	 $a = 3 \quad b = a + 1 \quad b = b + 3$
+=, -=, *=, /=, %= (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
```

```
num
```

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

>   >=   <   <=	$a > b$ $a >= b$
==   !=	$a == b$ $a != b$

결과값  
(True, False)

같다

같지  
않  
다

## 비교 연산자 실습

```
a = 3
```

```
b = 7
```

```
print(a > b) → False
```

```
print(a <= b) → True
```

```
print(a == b) → False
```

```
print(a != b) → True
```

## 논리 연산자

- 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 and 10 == 20  
False False

3 > 5 and 10 < 20  
False True

3 < 5 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 or 10 == 20  
False False

3 > 5 or 10 < 20  
False True

3 < 5 or 10 < 20  
True True

a if 조건식 else b



True False

```
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 출력하시오.

?

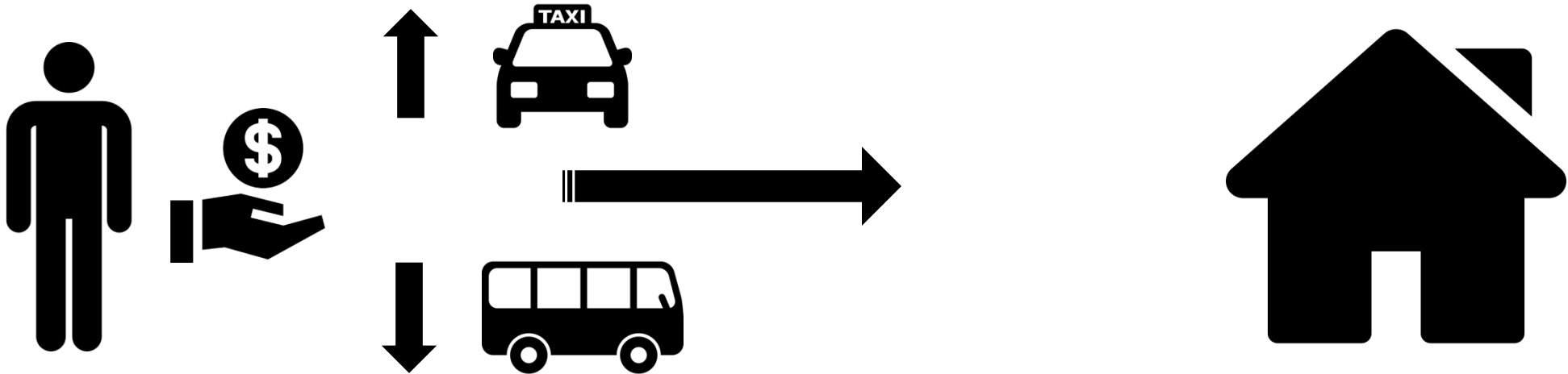
```
첫 번째 정수 입력 >> 5
두 번째 정수 입력 >> 10
두 수의 차 : 5
```

?

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

## 조건문

- 조건에 따라 실행 흐름을 다르게 하는 문법



if

elif

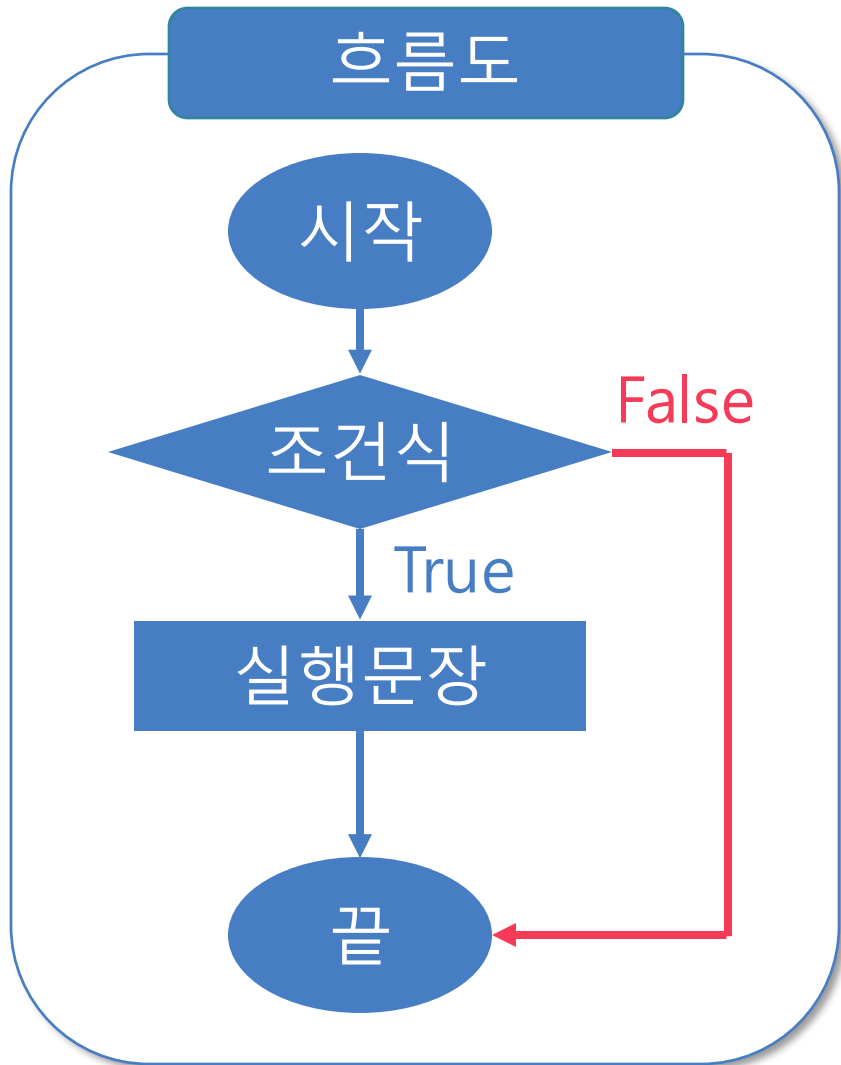
else

조건식이 True일 경우 실행문장 실행

```
if 조건식:
    실행문장
    실행문장
```

(colon, 콜론)

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 10000이상이면 "택시를 탄다"를 출력하시오. (비교연산자 이용)

```
money = 11000  
?
```

택시를 탄다.

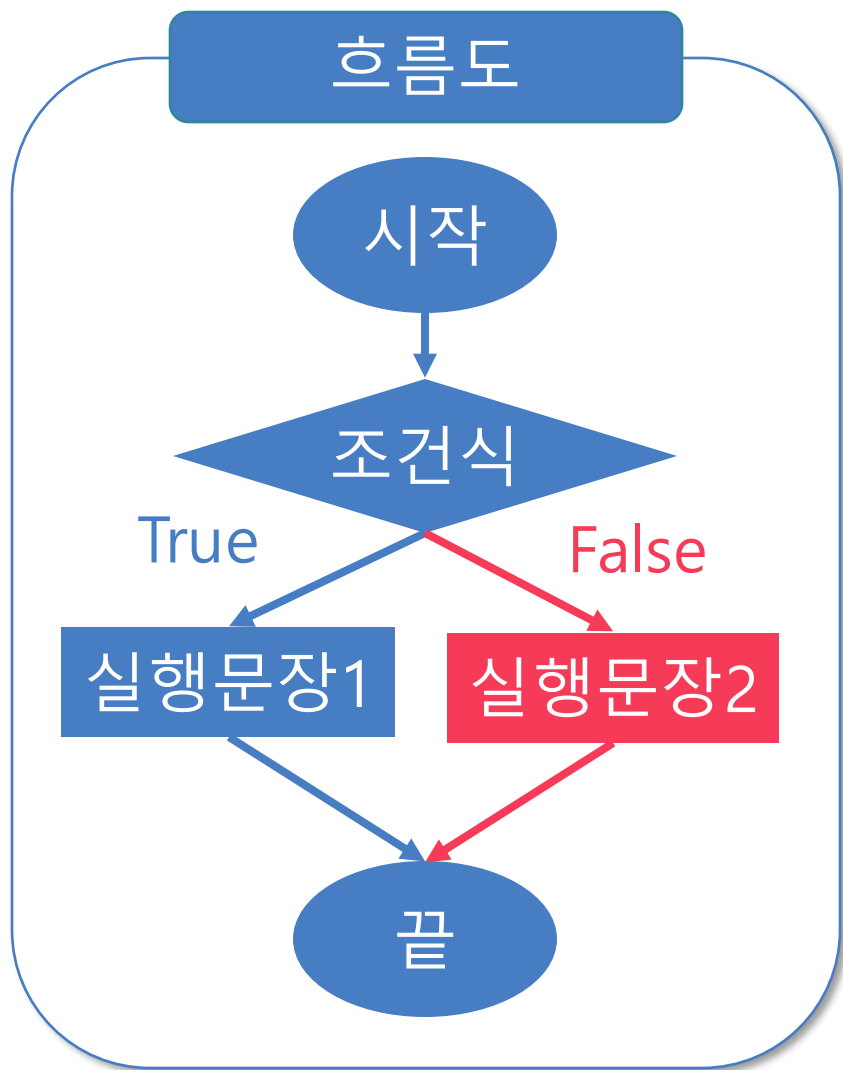
```
money = 9000  
?
```



조건식이 **True**일 경우 **실행문장1** 실행  
조건식이 **False**일 경우 **실행문장2** 실행

```
if 조건식:      (colon, 콜론)  
    실행문장1  
else:          (colon, 콜론)  
    실행문장2
```

들여쓰기 (Tab, Space\*4)



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

실행문장1

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

실행문장2

조건문을 사용하여 변수 money가 10000이상이면 "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

?

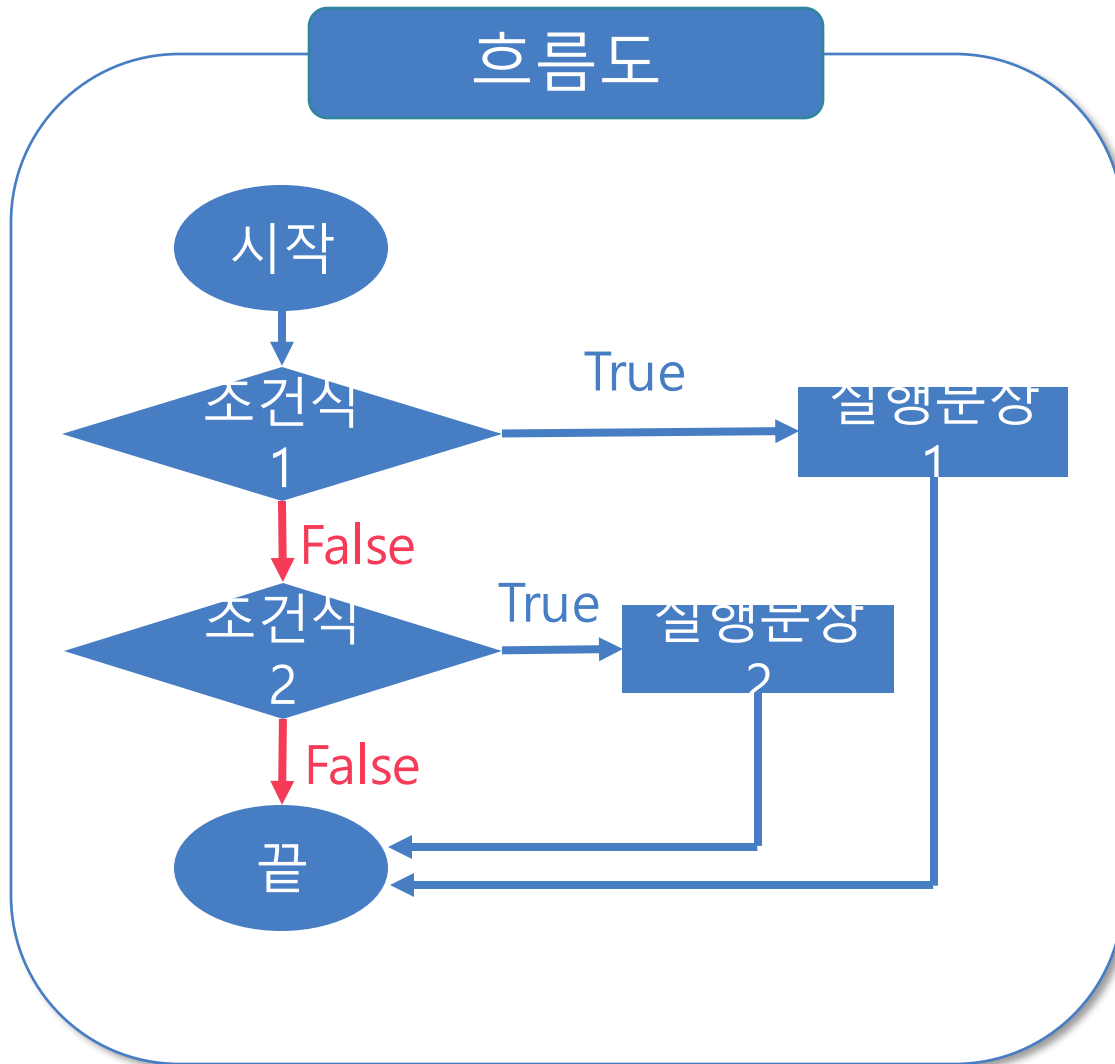
사려는 상품의 갯수를 입력하세요 >> 12  
가격은 108000원 입니다

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
```



```
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 큼니다.

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

첫 번째 정수 입력 >> 7  
두 번째 정수 입력 >> 13  
두 번째 정수가 더 큼니다.

```
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 <= 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]
```

```
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 **오름차순**으로 **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 **역순**으로 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 **내림차순**으로 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 <= 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                                Traceback (most recent call  
t)  
<ipython-input-53-5e0f22de5ab3> in <module>  
----> 1 tuple1[0] = 3  
  
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 포함 되어 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