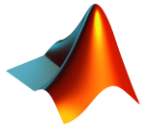
MATLAB 프로그래밍 및 실습

4강. 조건문



우리가 지금까지 해온 것들

```
name = input('Type your name: ','s');
age = input('Type your age: ','s');
scores = zeros(5,1);
scores(1) = input('first score:  ');
scores(2) = input('second score: ');
scores(3) = input('third score:  ');
scores(4) = input('fourth score: ');
scores(5) = input('fifth score:  ');
disp(' ')
disp(['Applicant: ' name ' (' age ')'])
disp(' ')
disp(['Ma
disp(['Av
```

- 점수를 100개 넣고 싶다면?
- 점수 개수가 매번 바뀐다면?

```
x = 10:0.1:22;
y = 95000./(x.^2);

xd = 10:2:22;
yd = [950 640 460 340 250 180 140];

plot(x,y,'-','linewidth',1.0)
xlabel('DISTANCE (cm)')
ylabel('INTENSITY (lux)')
title('Light Intensity as a Function of Distance');
axis([8 24 0 1200])
text(14,700, 'Comparison between theory and experiment')
hold on
plot(xd,yd,'ro--','linewidth',1.0,'markersize',4)
legend('Theory','Experiment')
```

- 선을 몇 개 그릴지 사용자 입력으로 조절하고 싶다면?

- 지금까지 해온 것들
  - 한줄한줄 순서대로 실행
- 상황에 따른 유연한 제어가 어렵다.
  - 조건에 따른 분기
  - 원하는 회수만큼 반복
  - 특정 조건이 만족되는 동안 계속 반복

- 특정 조건에 따라 반복적으로 값을 수정하고 싶다면?

```
>> a(2,3) = 100
a =
     1     5     9    13    17
     2     6   100    14    18
     3     7    11    15    19
     4     8    12    16    20
```

노트북을 바꾸고 싶다.

https://lazymatlab.tistory.com/75
『프로그래밍 언어 공부는
프로그래밍 **언어** 공부가 아니다.
프로그래밍 언어 공부이다.』

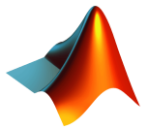
방법1

- 추천 노트북을 무작정 검색해서 하나씩 살핀다.
- 추천 노트북들은 너무 비싼걸 깨닫는다.
- 생각해보니 예산을 안 정했다.
- 예산을 정한다.
- 다시 검색한다.
- 생각해보니 노트북의 목적이 불분명하다.
- 노트북을 새로 살 목적을 정한다.
- 다시 검색한다.
- 그런데 아무리 검색해도 데탑이 훨씬 싸다.
- 타겟을 데탑으로 바꿔서 다시 검색한다.
- …
- …

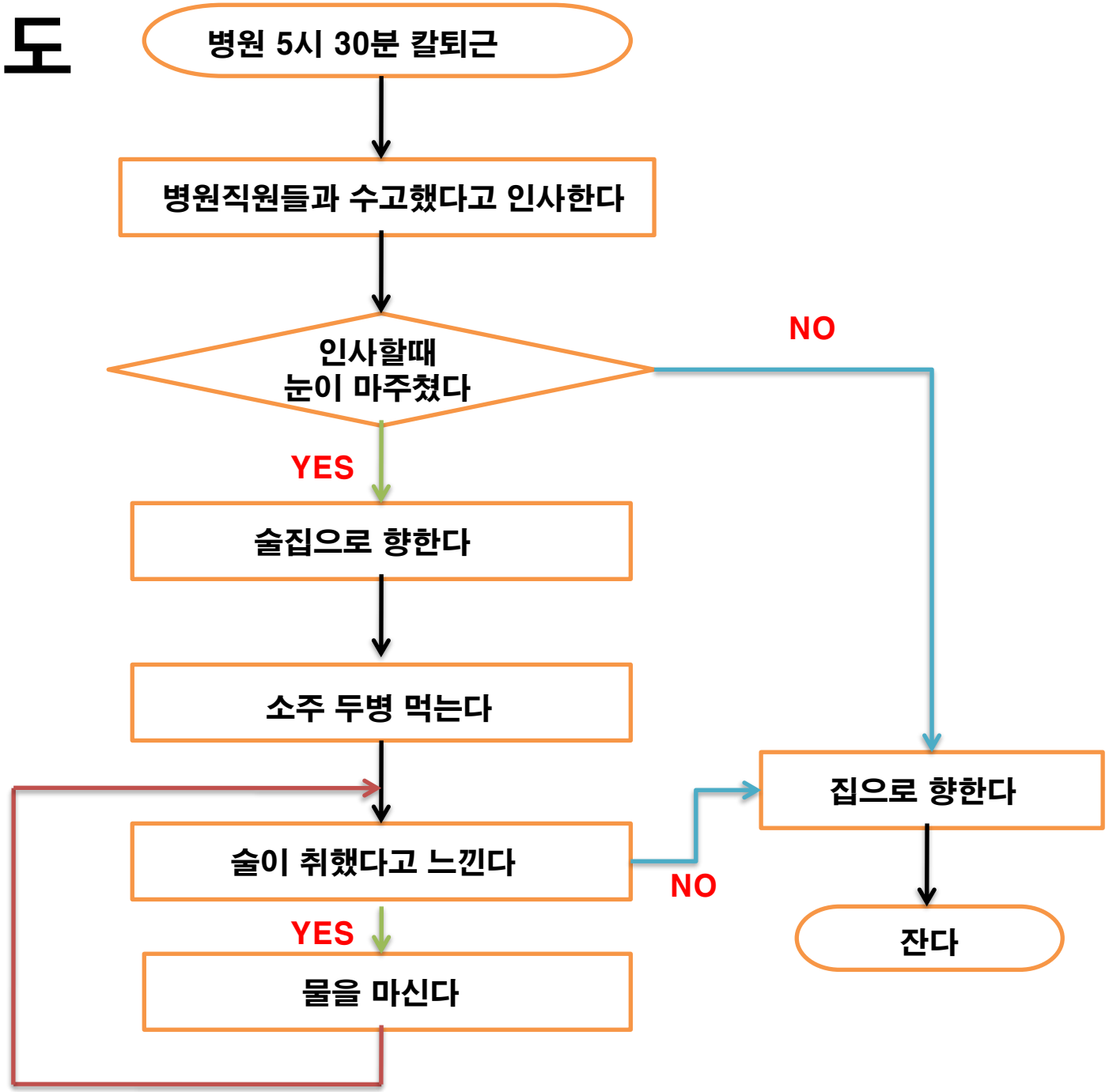
방법2

- 예산, 구매목적, 데탑/노트북을 먼저 정한다.
  - 노트북으로 정했다.
- 목적에 맞는 추천 노트북을 검색한다.
- 목록을 모아서 다음을 반복한다.
  - i번째 노트북 상세페이지를 연다.
  - 예산을 넘는가? -> 제외
  - 2kg을 넘는가? -> 제외
  - 13.3인치 미만인가? -> 제외
  - 상품평이 별로인가? -> 제외
  - 브랜드가 별로인가? -> 제외
  - i를 1 증가시키고 반복
- 남는 것들을 종합평가하여 하나를 선정한다.

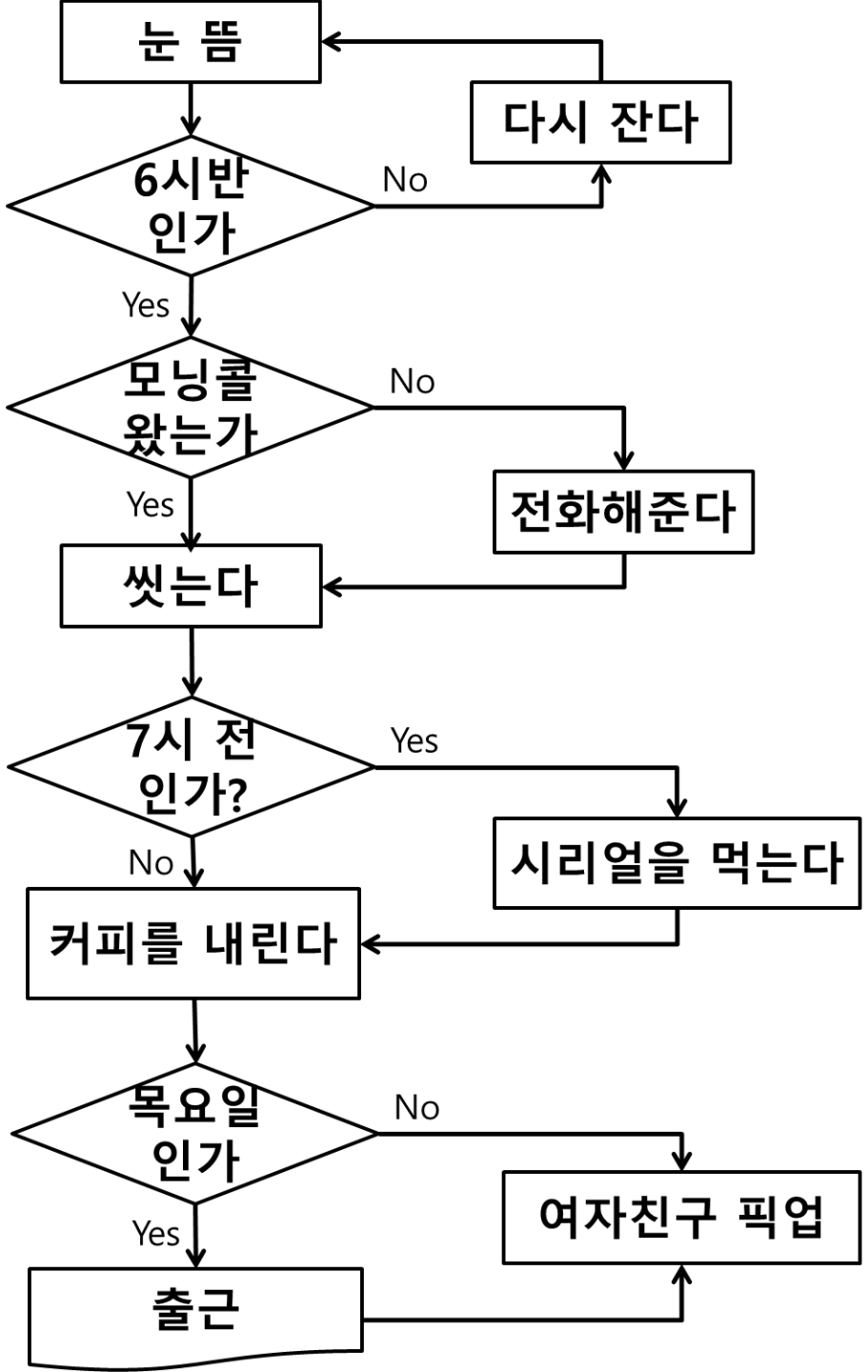
- 작업의 분리 및 구조화 → “알고리즘”, “로직”



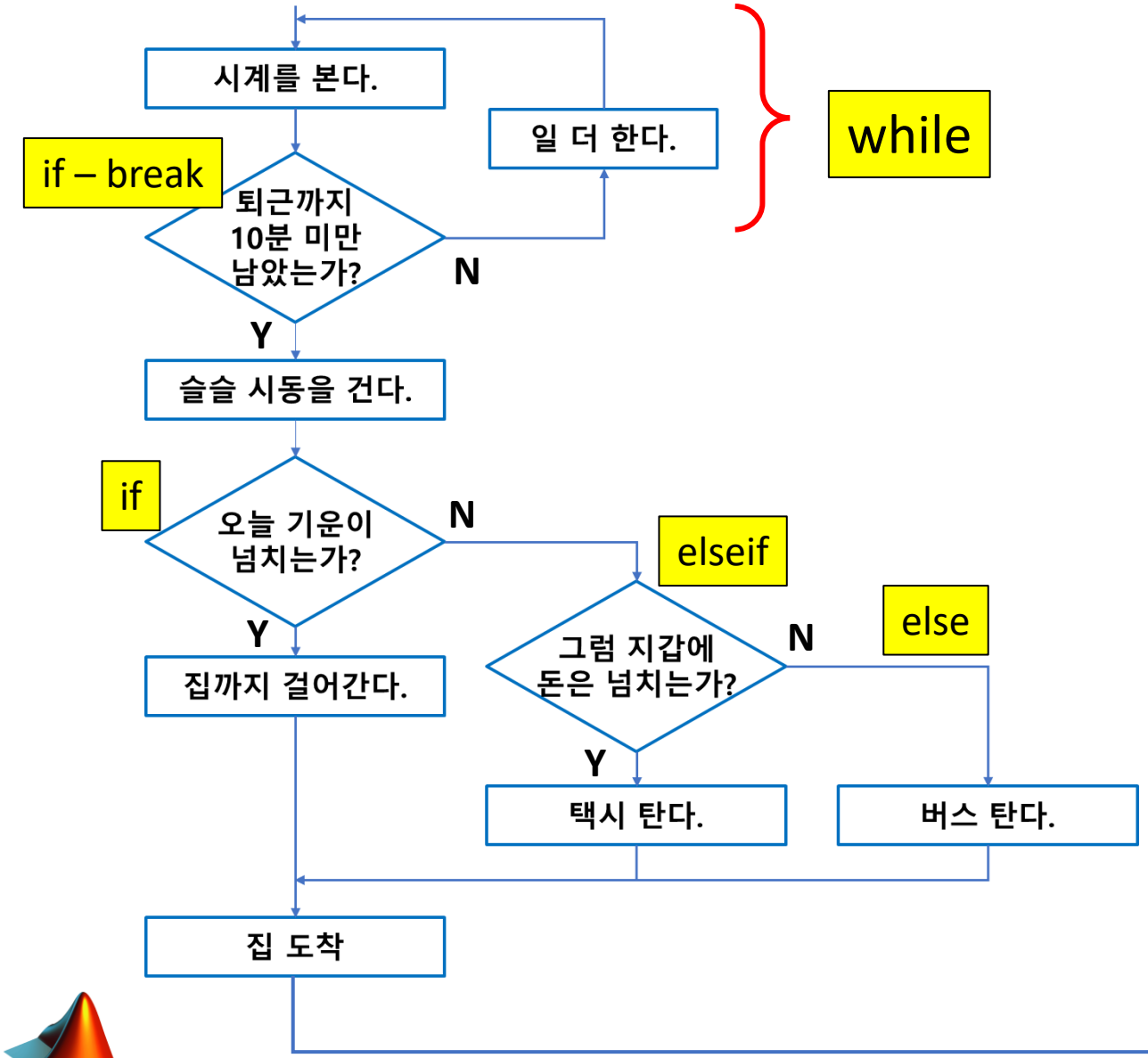
A 학생의 퇴근 순서도

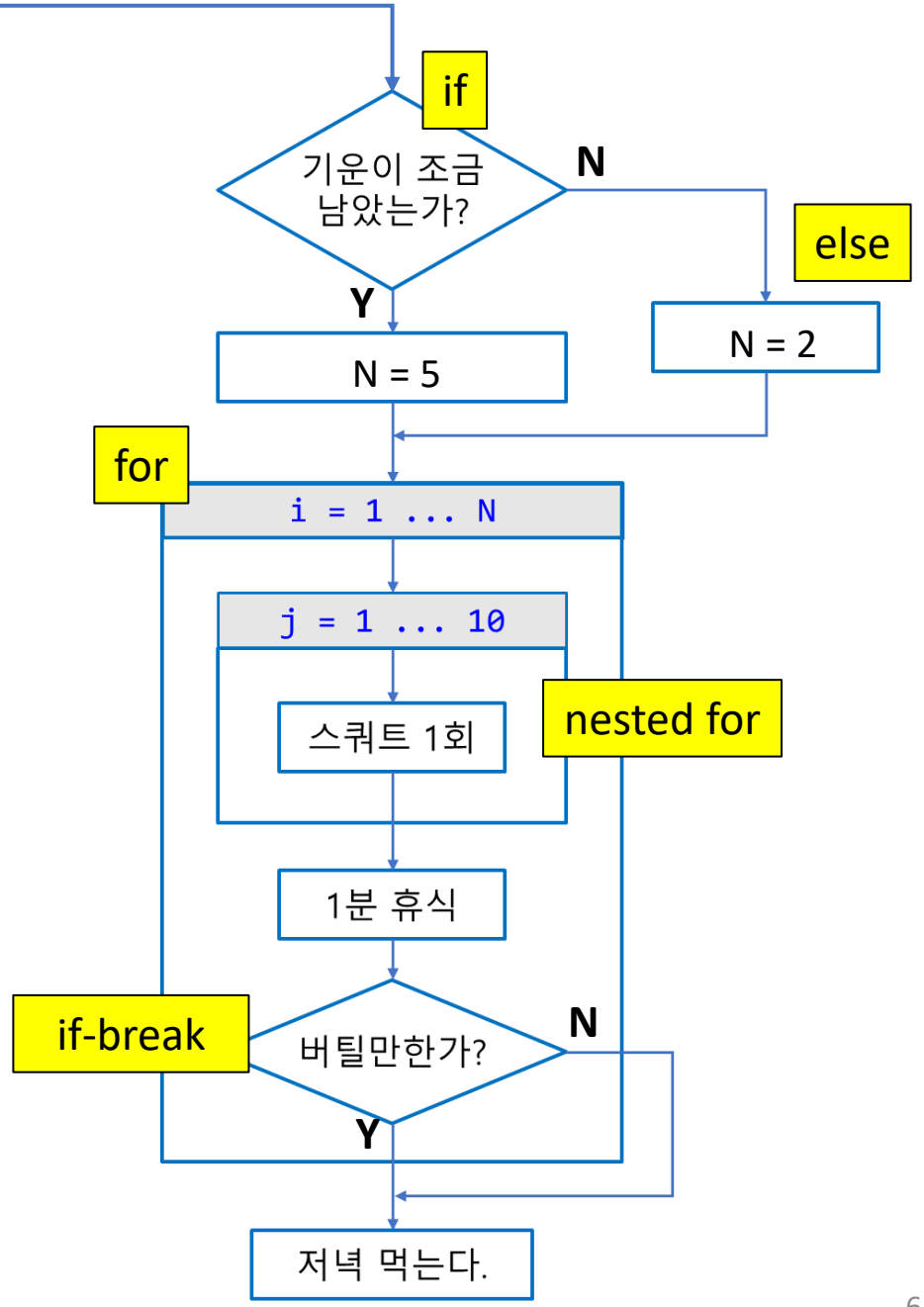

병원 5시 30분 칼퇴근
병원직원들과 수고했다고 인사한다
인사할때
눈이 마주쳤다
NO
YES
술집으로 향한다
소주 두병 먹는다
술이 취했다고 느낀다
NO
YES
물을 마신다
집으로 향한다
잔다

B 학생의 출근 순서도

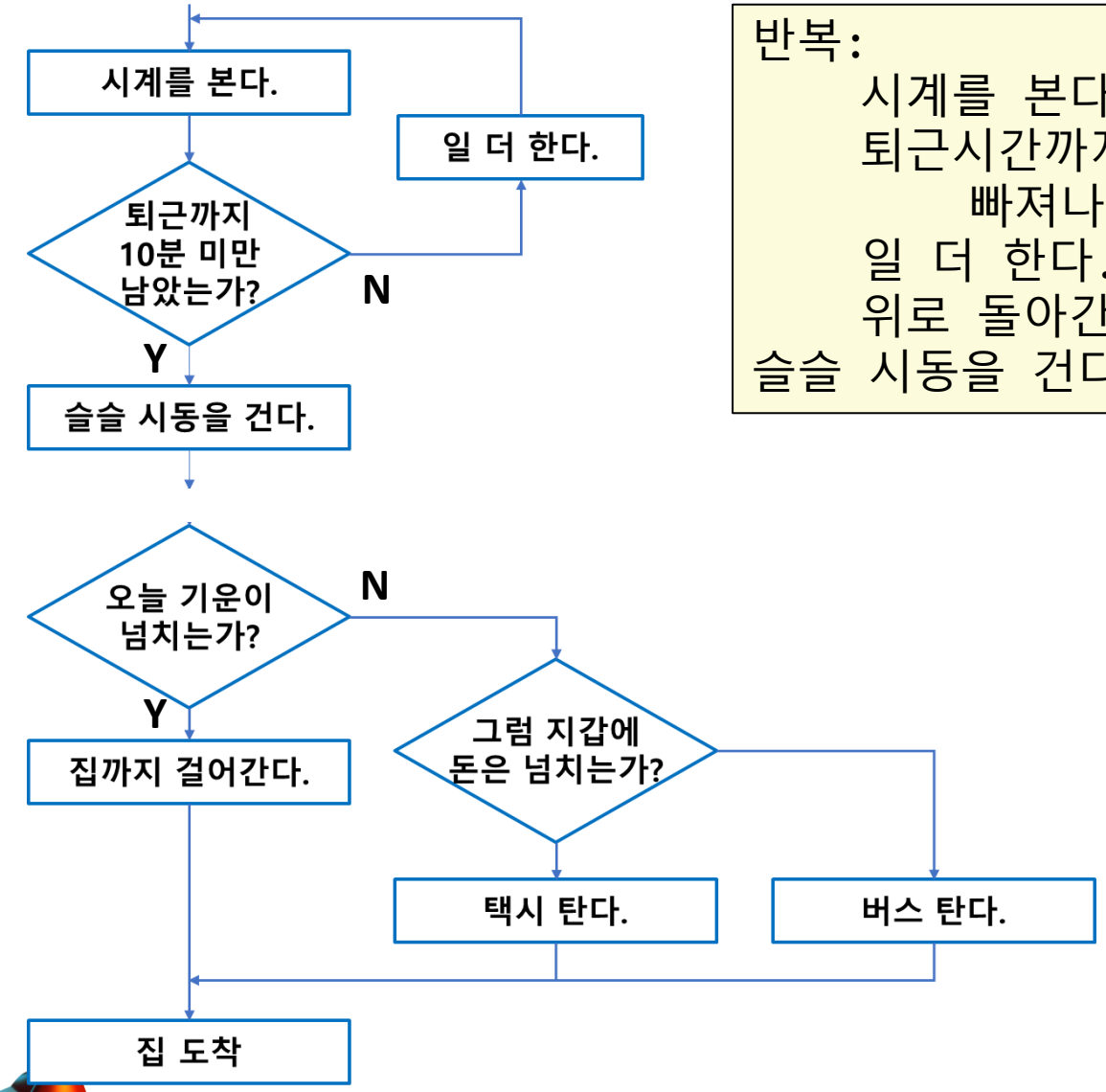

눈 뜸
다시 잔다
6시반 인가
No
Yes
모닝콜 왔는가
No
전화해준다
Yes
씻는다
7시 전 인가?
Yes
시리얼을 먹는다
No
커피를 내린다
목요일 인가
No
여자친구 픽업
Yes
출근

C 학생의 퇴근 순서도


시계를 본다.
일 더 한다.
while
if – break
퇴근까지
10분 미만
남았는가?
N
Y
슬슬 시동을 건다.
if
오늘 기운이
넘치는가?
N
elseif
Y
그럼 지갑에
돈은 넘치는가?
N
else
Y
집까지 걸어간다.
택시 탄다.
버스 탄다.
집 도착


if
기운이 조금
남았는가?
N
else
Y
N = 5
N = 2
for
i = 1 ... N
j = 1 ... 10
스쿼트 1회
nested for
1분 휴식
if-break
버틸만한가?
N
Y
저녁 먹는다.

의사코드 (pseudo code)



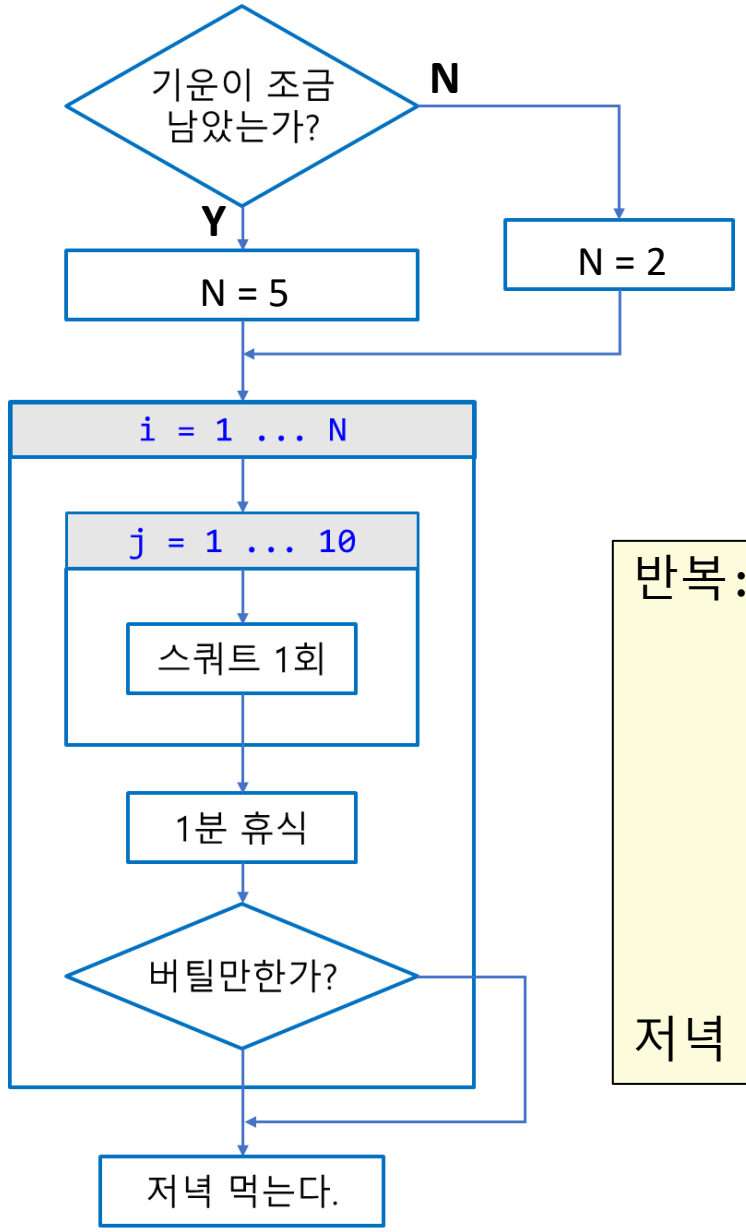
while, if-break

```
반복:
    시계를 본다.
    퇴근시간까지 10분 미만이면:
        빠져나간다.
    일 더 한다.
    위로 돌아간다.
슬슬 시동을 건다.
```

if-elseif-else

```
오늘 기운이 넘치면:
    집까지 걸어간다.
그건 아니지만 지갑에 돈이 많다면:
    택시를 탄다.
그것도 아니면
    버스를 탄다.
집 도착
```

의사코드 (pseudo code)



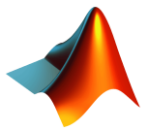
if-else

```
기운이 남았다면:
    N = 5
그렇지 않다면:
    N = 2
```

for
for
if-break

```
반복: N회
    반복: 10회
        스쿼트 1회 실시
    1분 휴식
    버틸만하다면:
        위로 돌아간다.
    그게 아니라면:
        빠져나간다.
저녁 먹는다.
```

조건문

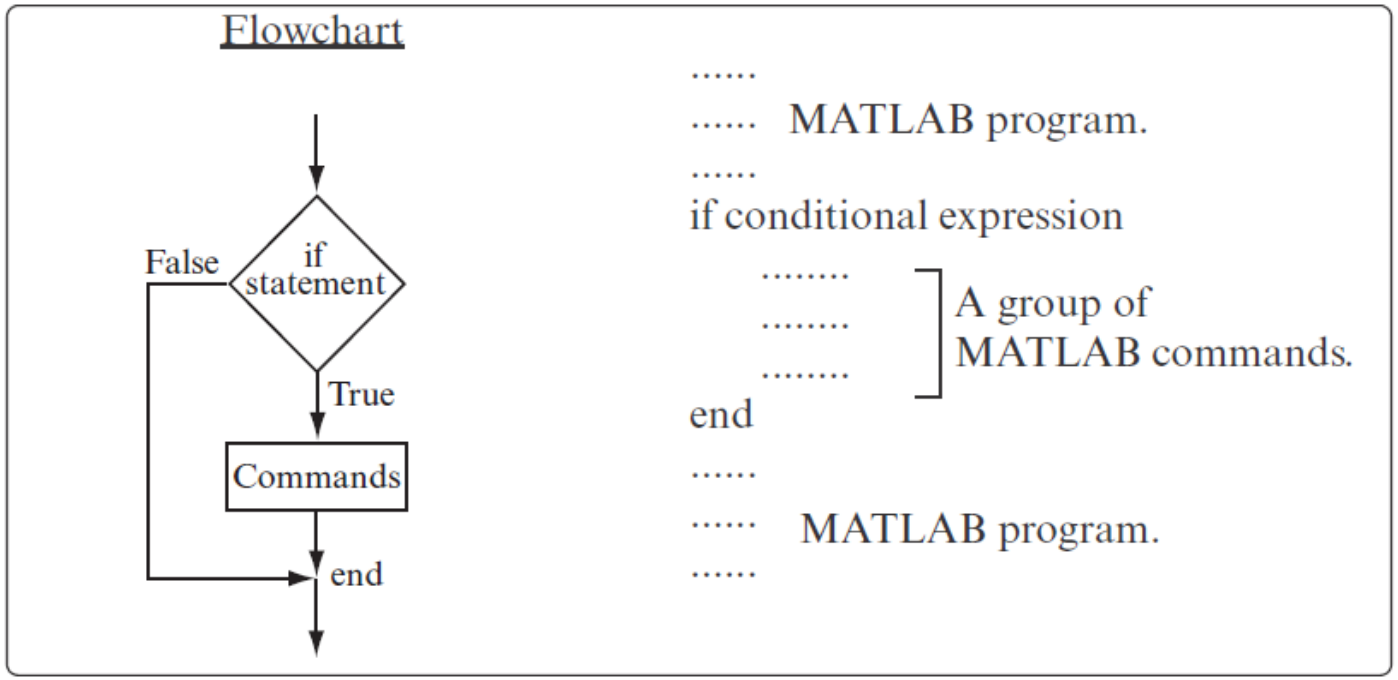


간단한 예제부터 시작해보자.

```
% 난수를 생성하고, 0.5보다 큰지 판단
n = rand;
if n>0.5
    disp('BIG!')
end
if n<=0.5
    disp('Small...')
end
```

```
% 수를 입력받아, 50보다 큰지 판단
n = input('put a number (0~100)');
if n>50
    disp('BIG!')
end
if n<=50
    disp('Small...')
end
```

```
% 기본문법

if 조건문
    명령문;
    명령문;
    ...
    ...
end
```

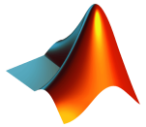


if문 예제

```
% 2차 방정식이 실근을 갖는지 출력
a = input('coeff. of x^2');
b = input('coeff. of x^1');
c = input('coeff. of x^0');
disc = b^2-4*a*c;
if disc>=0
    disp('Eq. has real solutions.')
end
if disc<0
    disp('Eq. has complex solutions.')
end
```

```
% 아래처럼 짜면 안된다.
if disc>=0
    disp('Eq. has real solutions.')
end
disp('Eq. has complex solutions.')
```

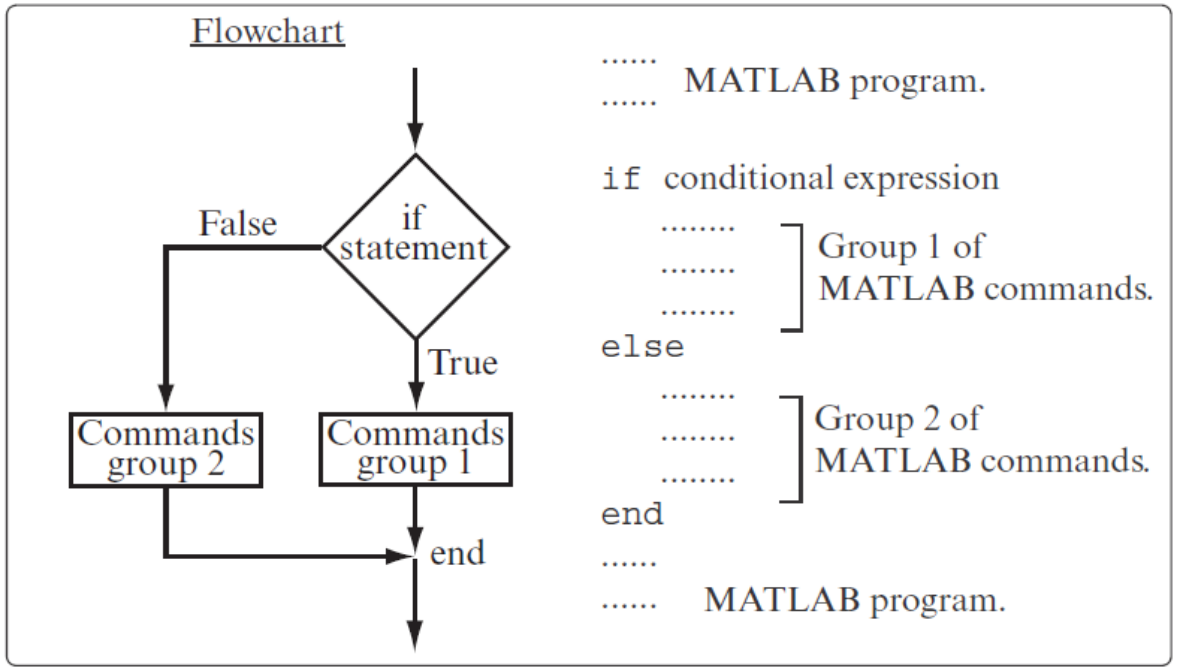
- if문이 두 번 나오니 불편하다.
- 하나로 합칠 수 없을까?

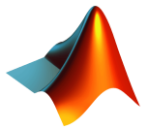


if-else (feat. 양자택일)

```
% 2차 방정식이 실근을 갖는지 출력
a = input('coeff. of x^2: ');
b = input('coeff. of x^1: ');
c = input('coeff. of x^0: ');
disc = b^2-4*a*c;
if disc>=0
    disp('Eq. has real solutions.')
else
    disp('Eq. has complex solutions.')
end
```

```
% 기본문법

if 조건문
    명령문;
    ...
else
    명령문;
    ...
end
```

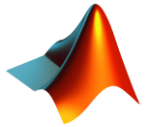




if-else 예제

```
% 60점 넘으면 pass, 아니면 fail
score = input('input your score: ');
if score>=60
    disp('Passed the exam. Congrats!')
else
    disp('Failed... better luck next time!')
end
```

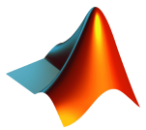
- 그런데 말입니다.
- 점수별로 A부터 F까지 출력하고 싶다면?
- `if-end, if-end, if-end, if-end`...라고 쓰고 싶지는 않습니다.



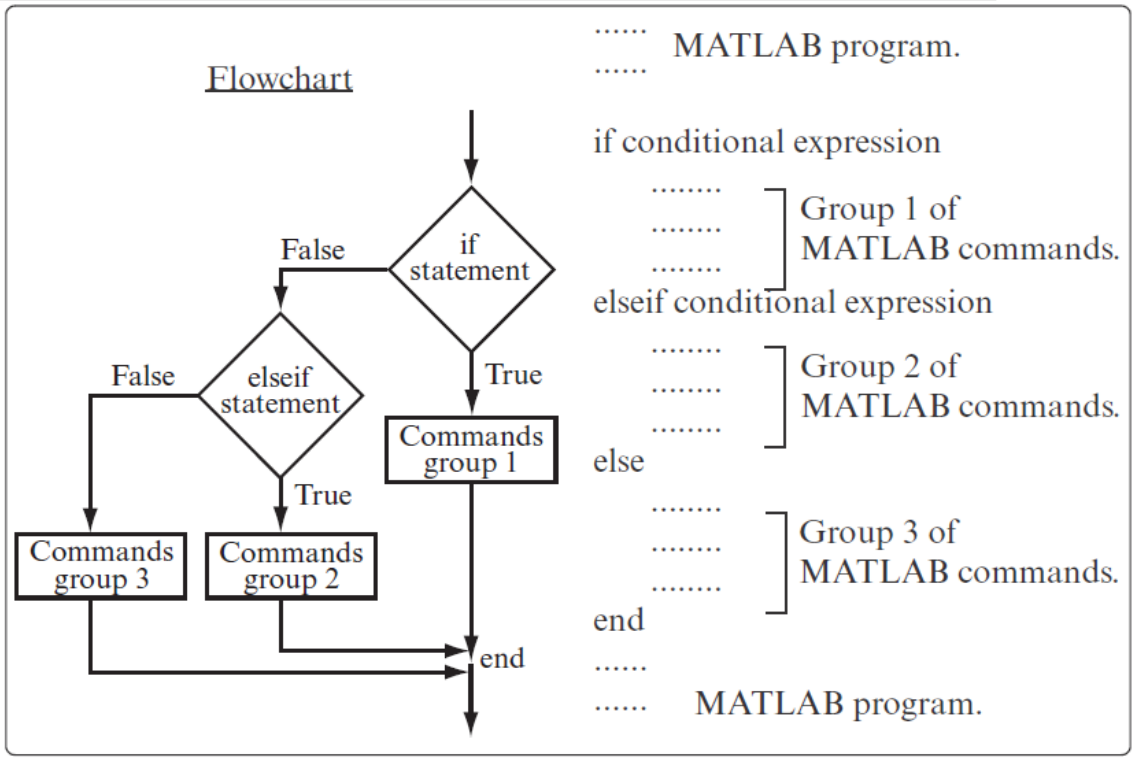
if-elseif-else (feat. 다지 선다)

```
% 점수별로 A부터 F까지 출력
score = input('input your score: ');
if score>=90
    disp('You got A.')
elseif score>=80
    disp('You got B.')
elseif score>=70
    disp('You got C.')
elseif score>=60
    disp('You got D.')
else
    disp('You got F.')
end
```

- 각 조건의 순서에 주의하자.
- fprintf를 쓰면 조금 더 간단히 쓸 수 있다.
- 주의: `elseif`와 `else if`는 다르다.



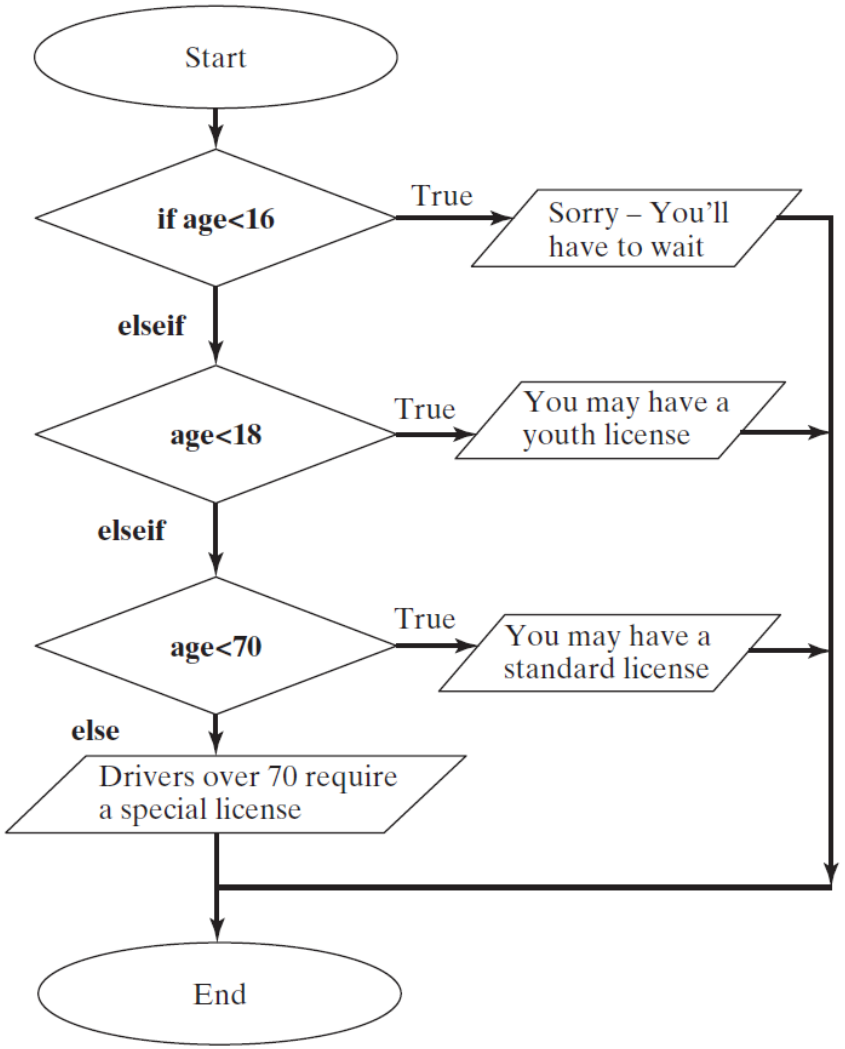
```
% 기본문법
if 조건문
    명령문;
elseif
    명령문;
else
    명령문;
end
```



if-elseif-else (feat. 다지 선다)

```
% Driver's license status
age = input('input your age: ');
if age<16 % be careful of the order
     disp('Sorry - You''ll have to wait.')
elseif age<18
     disp('You may have a youth license.')
elseif age<70
     disp('You may have a standard license.')
else
     disp('Drivers over 70 requires a special license.')
end
```

```
% 2차 방정식이 어떤 근을 갖는지 출력
a = input('coeff. of x^2: ');
b = input('coeff. of x^1: ');
c = input('coeff. of x^0: ');
disc = b^2-4*a*c;
if disc>0
    disp('Eq. has two distinct real solutions.')
elseif disc==0
    disp('Eq. has two identical real solutions.')
else
    disp('Eq. has two distinct complex solutions.')
end
```



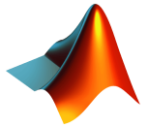
조건문은 어떻게 쓸까? – 1) 비교연산자

```
a = input('input 1st number: ');
b = input('input 2nd number: ');

% check if a and b is equal or not
if a==b
    disp('equal')
elseif a~=b % else part can be omitted.
    disp('not equal')
end

if a>=b
    disp('1st is equal to or larger than 2nd.')
end

if a<=b
    disp('1st is equal to or smaller than 2nd.')
end

if a>b
    disp('1st > 2nd')
elseif a<b
    disp('1st < 2nd')
end
```

Relational operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
~=	Not Equal to

- 실수를 비교할 때는 조심하자.

```
>> 0.1+0.2 == 0.3
ans =
  logical

>> sin(pi) == 0
ans =
  logical
```

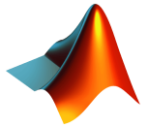


조건문은 어떻게 쓸까? – 2) 논리연산자

```
% check if the eq. has two distinct solutions
a = input('coeff. of x^2: ');
b = input('coeff. of x^1: ');
c = input('coeff. of x^0: ');
disc = b^2 - 4*a*c;
if (a~=0) & (disc~=0)
    disp('Eq. has two distinct solutions')
end
```

INPUT		OUTPUT			
		and	or	xor	not
A	B	A & B	A | B	xor(A, B)	~A
false	false	false	false	false	true
false	true	false	true	true	true
true	false	false	true	true	false
true	true	true	true	false	false

- 쪼개서 생각해보자.
- 2중 if문으로도 작성할 수 있으나…
- and(A, B), or(A, B)도 있으나 잘 안 쓰임
  - and, or 함수는 인자를 2개밖에 받지 못함
  - &, |는 인자를 몇 개든 받을 수 있음



https://www.mathworks.com/help/matlab/logical-operations.html

연산자 우선순위

Precedence	Operation
1 (highest)	Parentheses (if nested parentheses exist, inner ones have precedence)
2	Exponentiation
3	Logical NOT (~)
4	Multiplication, division
5	Addition, subtraction
6	Relational operators (>, <, >=, <=, ==, ~=)
7	Logical AND (&)
8 (lowest)	Logical OR (|)

```
>> 1 | 0 & 0
ans =
  logical
   1
>> (1 | 0) & 0
ans =
  logical
   0
```

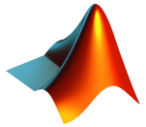
```
>> 3+4<16/2
ans =
     1
>> 3+(4<16)/2
ans =
    3.5000
```

+ and / are executed first.

The answer is 1 since 7 < 8 is true.

4 < 16 is executed first, and is equal to 1, since it is true.

3.5 is obtained from 3 + 1/2.



```
>> x=-2; y=5;
>> -5<x<-1
ans =
     0
>> -5<x & x<-1
ans =
     1
>> ~(y<7)
ans =
     0
>> ~y<7
ans =
     1
>> ~((y>=8)|(x<-1))
ans =
     0
>> ~(y>=8)|(x<-1)
ans =
     1
```

Define variables x and y.

This inequality is correct mathematically. The answer, however, is false since MATLAB executes from left to right. $-5 < x$ is true (=1) and then $1 < -1$ is false (0).

The mathematically correct statement is obtained by using the logical operator &. The inequalities are executed first. Since both are true (1), the answer is 1.

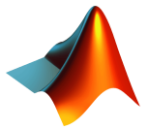
$y < 7$ is executed first, it is true (1), and ~1 is 0.

※ 수식=사람이 이해하는 식
→ 컴퓨터가 이해하는 식으로 수정 필요

~y is executed first, y is true (1) (since y is nonzero), ~1 is 0, and $0 < 7$ is true (1).

$y >= 8$ (false), and $x < -1$ (true) are executed first. OR is executed next (true). ~ is executed last, and gives false (0).

$y >= 8$ (false), and $x < -1$ (true) are executed first. NOT of ($y >= 8$) is executed next (true). OR is executed last, and gives true (1).



and, or는 두 종류가 있다.

INPUT		OUTPUT			
		element-wise		short circuit	※ scalar끼리만 연산 가능
A	B	A & B	A | B	A && B	A || B
false	false	원소별 연산		A만으로 판단 완료	B까지 확인 필요
false	true			A만으로 판단 완료	B까지 확인 필요
true	false			B까지 확인 필요	A만으로 판단 완료
true	true			B까지 확인 필요	A만으로 판단 완료

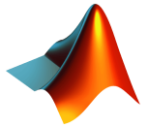
- &&와 ||는 언제 필요한가?
  - A 또는 B의 결과를 얻는데 시간이 오래 걸리는 경우
  - A의 결과에 따라 B를 확인할 필요가 없는 경우
  - A, B 모두 스칼라일 땐 → &&나 ||
  - 둘 중 하나라도 행렬일 땐 → &나 |

```
b = 1;
a = 20;
x = (b ~= 0) && (a/b > 18.5)
```

```
x = logical
   1
```

```
b = 0;
x = (b ~= 0) && (a/b > 18.5)
```

```
x = logical
   0
```



https://www.mathworks.com/help/matlab/logical-operations.html
https://www.mathworks.com/help/matlab/ref/logicaloperatorsshortcircuit.html

logical 자료형이란?

```
% check if the eq. has two distinct solutions
a = input('coeff. of x^2: ');
b = input('coeff. of x^1: ');
c = input('coeff. of x^0: ');
disc = b^2 - 4*a*c;
if (a~=0) & (disc~=0)
    disp('Eq. has two distinct solutions')
end
```

INPUT		OUTPUT			
		and	or	xor	not
A	B	A & B	A | B	xor(A, B)	~A
false	false	false	false	false	true
false	true	false	true	true	true
true	false	false	true	true	false
true	true	true	true	false	false

- 다시 한번 쪼개서 생각해보자.
  - true = `logical 1`
  - false = `logical 0`
- `logical`은 함수이기도 하다.
  - logical(A): A와 같은 크기의 logical array 반환
  - 0, 0.0, false → `logical 0`
  - 그 외 (문자 포함) → `logical 1`

※예제

```
>> (a~=0)
ans =
  logical
   1
>> (disc~=0)
ans =
  logical
   0
```

a = 1, b = 2, c =1

```
>> logical(0.1+0.2-0.3)
ans =
  logical
   1
```

```
>> logical([1 0 0 -1 -0.1])
ans =
  1×5 logical 배열
   1   0   0   1   1
```

https://www.mathworks.com/help/matlab/ref/logical.html

비교, 논리연산자를 행렬에 사용하면?

```
>> a = magic(3)
a =
     8     1     6
     3     5     7
     4     9     2
>> a>3
ans =
  3×3 logical 배열
   1   0   1
   0   1   1
   1   1   0
```

- 행렬 (op) 스칼라
- 스칼라 (op) 행렬
  - 각 원소를 비교

```
>> b = reshape(1:9, 3,3)
b =
     1     4     7
     2     5     8
     3     6     9
>> a>b
ans =
  3×3 logical 배열
   1   0   0
   1   0   0
   1   1   0
```

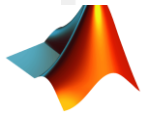
- 행렬 (op) 행렬
  - 각 위치별로 비교
  - 크기가 다르면 에러

```
>> c = [0 0 1 2 3];
>> d = [1 0 4 5 0];
>> c & d
ans =
  1×5 logical 배열
   0   0   1   1   0
>> c && d
|| 및 && 연산자에 대한 피연산자
다.
>> xor(c,d)
ans =
  1×5 logical 배열
   1   0   0   0   1
```

```
>> not(c)
ans =
  1×5 logical 배열
   1   1   0   0   0
>> ~c
ans =
  1×5 logical 배열
   1   1   0   0   0
```

- 행렬 (& 또는 |) 행렬
- xor(행렬, 행렬)
- not(행렬), ~(행렬)
  - 각 행렬을 logical로 변환 후 연산

```
>> a = rand(3)
a =
    0.0430    0.7317    0.5470
    0.1690    0.6477    0.2963
    0.6491    0.4509    0.7447
>> (a>0.5) & (magic(3)>3)
ans =
  3×3 logical 배열
   0   0   1
   0   1   0
   1   0   0
```



```
>> b=[15 6 9 4 11 7 14]; c=[8 20 9 2 19 7 10];
```

Define vectors b and c.

```
>> d=c>=b
```

Checks which c elements are larger than or equal to b elements.

```
d =
     0     1     1     0     1     1     0
```

Assigns 1 where an element of c is larger than or equal to an element of b.

```
>> b == c
```

Checks which b elements are equal to c elements.

```
ans =
     0     0     1     0     0     1     0
>> b~=c
```

Checks which b elements are not equal to c elements.

```
ans =
     1     1     0     1     1     0     1
>> f=b-c>0
```

Subtracts c from b and then checks which elements are larger than zero.

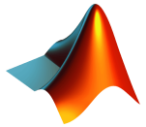
```
f =
     1     0     0     1     0     0     1
>> A=[2 9 4; -3 5 2; 6 7 -1]
```

Define a 3×3 matrix A.

```
A =
     2     9     4
    -3     5     2
     6     7    -1
>> B=A<=2
```

Checks which elements in A are smaller than or equal to 2. Assigns the results to matrix B.

```
B =
     1     0     0
     1     0     1
     0     0     1
```



if 뒤의 조건문에 행렬이 오면?

```
% if 뒤 조건문이 행렬이라면?
a = rand(1,3);
disp(a)
if a>0.3
    disp('every element > 0.3')
else
    disp('1 or more elements <= 0.3')
end
```

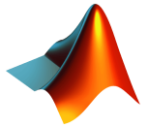
```
>>
    0.2259    0.1707    0.2277
1 or more elements <= 0.3
    0.4357    0.3111    0.9234
every element > 0.3
```

- 행렬의 모든 원소가 true(logical 1)이어야 true로 인식
- all(A)
  - A가 모두 true일 때만 true

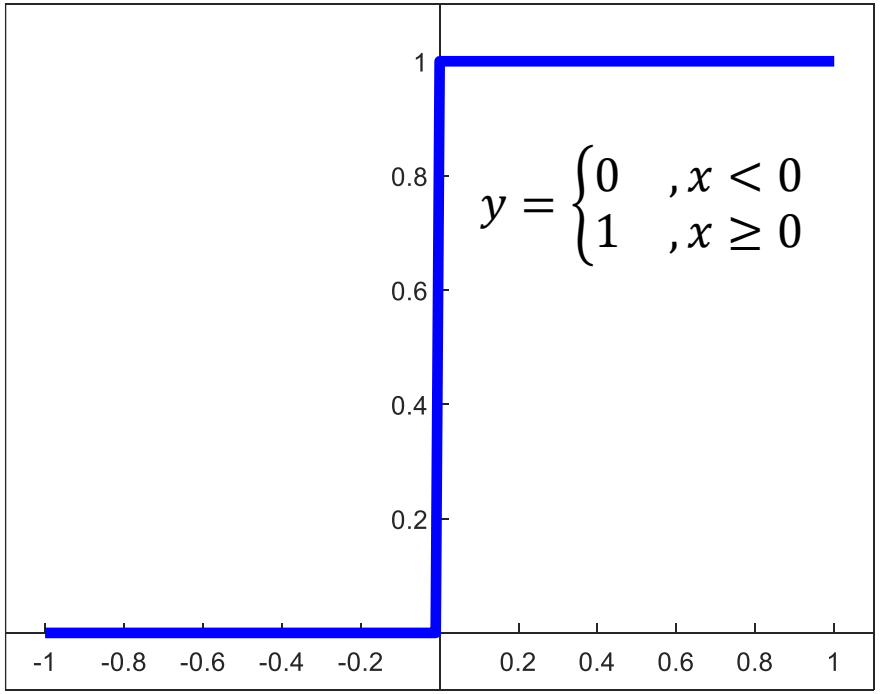
```
>> all([1 0 0 0])
ans =
  logical
   0
```

- any(A)
  - A 원소 중 하나라도 true이면 true

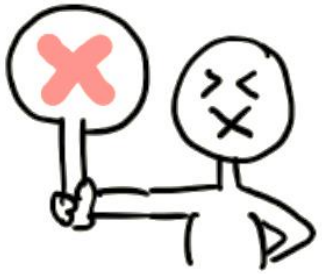
```
>> any([1 0 0 0])
ans =
  logical
   1
```



if 뒤의 조건문에 행렬이 오면?




```
x = -1:0.01:1;
if x>=0
    y=1;
else
    y=0
end

plot(x, y)
```



※※ 알아서 원소별로 계산해주지 않는다!

```
x = -1:0.01:1;
y = (x>=0);
plot(x, y)
```

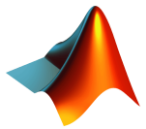
※ 수식=사람이 이해하는 식
→ 컴퓨터가 이해하는 식으로 수정 필요

if 뒤 조건문에 &나 |가 오면?

```
% if 뒤 조건문에서 행렬 (& 또는 |) 행렬

if ones(1,4) | zeros(1,3)
    disp('true')
end

if [1 0 0 0] & zeros(1,3)
    disp('true')
end

if ones(1,4) & zeros(1,3) % error
    disp('true')
end

a = ones(1,4) | zeros(1,3); % error
```

상황	OUTPUT			
	element-wise		short circuit (only scalars)	
	A & B	A | B	A && B	A || B
일반	원소별 연산		A가 거짓이면 거짓	A가 거짓이면 B 확인 필요
			A가 참이면 B 확인 필요	A가 참이면 참
if 뒤 조건문	short-circuit 동작		위와 같음	위와 같음

- if 뒤의 &나 |는 short-circuiting
- if A | B
  - A 원소가 모두 nonzero이면: 참 (B는 스킵)
- if A & B
  - A의 원소 중 하나라도 0이면: 거짓 (B는 스킵)



logical 자료형과 double 자료형의 연산

```
>> 3&7
ans =
     1
```
3 AND 7.

3 and 7 are both true (nonzero), so the outcome is 1.

```
>> a=5|0
a =
     1
```
5 OR 0 (assign to variable a).

1 is assigned to a since at least one number is true (nonzero).

```
>> ~25
ans =
     0
```
NOT 25.

The outcome is 0 since 25 is true (nonzero) and the opposite is false.

```
>> t=25*((12&0)+(~0)+(0|5))
t =
    50
```
Using logical operators in a math expression.

```
>> x=[9 3 0 11 0 15]; y=[2 0 13 -11 0 4];
```
Define two vectors x and y.

```
>> x&y
ans =
     1     0     0     1     0     1
```
The outcome is a vector with 1 in every position where both x and y are true (nonzero elements), and 0s otherwise.

```
>> z=x|y
z =
     1     1     1     1     0     1
```
The outcome is a vector with 1 in every position where either or both x and y are true (nonzero elements), and 0s otherwise.

```
>> ~(x+y)
ans =
     0     0     0     1     1     0
```
The outcome is a vector with 0 in every position where the vector x + y is true (nonzero elements), and 1 in every position where x + y is false (zero elements).

```
>> 5>8
ans =
     0
```
Checks if 5 is larger than 8.

Since the comparison is false (5 is not larger than 8) the answer is 0.

```
>> a=5<10
a =
     1
```
Checks if 5 is smaller than 10, and assigns the answer to a.

Since the comparison is true (5 is smaller than 10) the number 1 is assigned to a.

```
>> y=(6<10)+(7>8)+(5*3==60/4)
y =
     2
```
Using relational operators in math expression.

Equal to 1 since 6 is smaller than 10.

Equal to 0 since 7 is not larger than 8.

Equal to 1 since 5*3 is equal to 60/4.

logical 자료형의 활용

```
% random 행렬에서 0.5보다 큰 값만 남기는 방법
A = rand(3,4);
disp('original A')
disp(A)

% method 1
map = (A>0.5);
B = A.*map; % A = A.*(A>0.5);
disp('elements with <=0.5 to 0')
disp(B)

% method 2
map = (A>0.5);
C = A(map);
disp('elements with <=0.5 removed')
disp(C)

% method 3
map = (A<=0.5);
D = A;
D(map) = []; % D(A<=0.5) = [];
disp('elements with <=0.5 removed')
disp(D)
```

행렬 인덱스에 logical array
→ logical 1인 곳의 값만 선택됨
→ numeric array와 다른 동작

```
% how many are larger than 0.5?
disp('number of elements with >0.5')
disp(sum(sum(A>0.5)))
disp(sum(A>0.5,'all'))
disp(sum(A(:)>0.5))

% make elements with <0.5 to 0.5
E = max(A, 0.5);
disp('elements with <0.5 coerced to 0.5')
disp(E)

% make elements with <0.5 to 0.1
map = (A<0.5);
F = A;
F(map) = 0.1; % F(A<0.5) = 0.1;
disp('elements with <0.5 coerced to 0.1')
disp(F)

% make elements with <0.5 to 0.1
map = (A>=0.5);
G = A.*map + (1-map).*.1;
disp('elements with <0.5 coerced to 0.1')
disp(G)
```

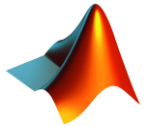
logical 자료형의 활용

```
% 점수별로 A부터 F까지 출력
score = input('input your score: ');
if score>=90
    disp('You got A.')
elseif score>=80
    disp('You got B.')
elseif score>=70
    disp('You got C.')
elseif score>=60
    disp('You got D.')
else
    disp('You got F.')
end
```

```
% 점수별로 A+부터 F까지 출력
% logical array를 사용한 예제

score = input('input your score: ');
grades = ['A+'; 'A0'; 'B+'; 'B0';
          'C+'; 'C0'; 'D+'; 'D0'; 'F '];

scorecuts = [95 90 85 80 75 70 65 60];
idx = length(scorecuts)+1-sum(score>=scorecuts);
yourgrade = grades(idx,:);
to_print = ['You got ' yourgrade '.\n'];
fprintf(to_print)
```

https://www.mathworks.com/help/matlab/matlab_prog/find-array-elements-that-meet-a-condition.html
https://www.mathworks.com/help/matlab/math/array-indexing.html

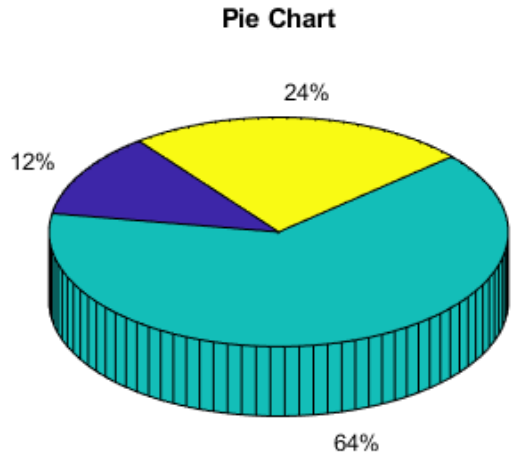
- 주의할 점
  - logical 자료형은 double과 연산이 가능하지만 엄연히 다른 자료형이다.
  - logical과 자연수 array는 행렬 인덱스로 사용될 때 동작이 다르다.



switch-case

```
x = linspace(-pi/3,pi/3,100);
figure,

to_draw = input('what to draw?: ', 's');
switch to_draw
    case 'sin'
        plot(x, sin(x))
    case 'cos'
        plot(x, cos(x))
    case 'tan'
        plot(x, tan(x))
    case 'quad'
        plot(x, x.^2)
    case 'cubic'
        plot(x, x.^3)
    case 'exp'
        plot(x, exp(x))
    otherwise
        disp('invalid option')
end
```

```
x = [12 64 24];
plottype = 'pie3';

switch plottype
    case 'bar'
        bar(x)
        title('Bar Graph')
    case {'pie','pie3'}
        pie3(x)
        title('Pie Chart')
    otherwise
        warning('Unexpected plot type. No plot created.')
end
```



- 주의할 점
  - case에 비교연산자를 쓸 수 없음

    https://www.mathworks.com/help/matlab/ref/switch.html

문자는 어떻게 대소비교를 할까?

- 비교연산자의 동작은 숫자와 동일하다.
  - char (op) char
  - char (op) char array
  - char array (op) char array
- 단, 문자의 **아스키 코드** 값을 비교한다.
  - 'A' = 65, 'Z' = 90
  - 'a' = 97, 'z' = 122
- strcmp
  - 두 문자열이 같은지 비교 (대소문자 구분 O)
- strcmpi
  - 두 문자열이 같은지 비교 (대소문자 구분 X)
  - ex) strcmpi('y', 'Y') → logical 1
- upper(str): 문자열 str을 모두 대문자로 변경
- lower(str): 문자열 str을 모두 소문자로 변경

```
>> double('Hello')
ans =
    72   101   108   108   111
>> char([87 111 114 108 100])
ans =
    'World'
>> char(65)
ans =
    'A'
>> double('z')
ans =
   122
>> 'A'-0
ans =
    65
>> 'Hello'-0
ans =
    72   101   108   108   111
```

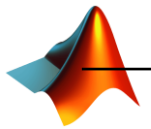
제어 문자 | 공백 문자 | 구두점 | 숫자 | 알파벳

10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진	문자
0	0x00	NUL	32	0x20	SP	64	0x40	@	96	0x60	`
1	0x01	SOH	33	0x21	!	65	0x41	A	97	0x61	a
2	0x02	STX	34	0x22	"	66	0x42	B	98	0x62	b
3	0x03	ETX	35	0x23	#	67	0x43	C	99	0x63	c
4	0x04	EOT	36	0x24	$	68	0x44	D	100	0x64	d
5	0x05	ENQ	37	0x25	%	69	0x45	E	101	0x65	e
6	0x06	ACK	38	0x26	&	70	0x46	F	102	0x66	f
7	0x07	BEL	39	0x27	'	71	0x47	G	103	0x67	g
8	0x08	BS	40	0x28	(	72	0x48	H	104	0x68	h
9	0x09	HT	41	0x29	)	73	0x49	I	105	0x69	i
10	0x0A	LF	42	0x2A	*	74	0x4A	J	106	0x6A	j
11	0x0B	VT	43	0x2B	+	75	0x4B	K	107	0x6B	k
12	0x0C	FF	44	0x2C	,	76	0x4C	L	108	0x6C	l
13	0x0D	CR	45	0x2D	-	77	0x4D	M	109	0x6D	m
14	0x0E	SO	46	0x2E	.	78	0x4E	N	110	0x6E	n
15	0x0F	SI	47	0x2F	/	79	0x4F	O	111	0x6F	o
16	0x10	DLE	48	0x30	0	80	0x50	P	112	0x70	p
17	0x11	DC1	49	0x31	1	81	0x51	Q	113	0x71	q
18	0x12	DC2	50	0x32	2	82	0x52	R	114	0x72	r
19	0x13	DC3	51	0x33	3	83	0x53	S	115	0x73	s
20	0x14	DC4	52	0x34	4	84	0x54	T	116	0x74	t
21	0x15	NAK	53	0x35	5	85	0x55	U	117	0x75	u
22	0x16	SYN	54	0x36	6	86	0x56	V	118	0x76	v
23	0x17	ETB	55	0x37	7	87	0x57	W	119	0x77	w
24	0x18	CAN	56	0x38	8	88	0x58	X	120	0x78	x
25	0x19	EM	57	0x39	9	89	0x59	Y	121	0x79	y
26	0x1A	SUB	58	0x3A	:	90	0x5A	Z	122	0x7A	z
27	0x1B	ESC	59	0x3B	;	91	0x5B	[	123	0x7B	{
28	0x1C	FS	60	0x3C	<	92	0x5C	₩	124	0x7C	|
29	0x1D	GS	61	0x3D	=	93	0x5D	]	125	0x7D	}
30	0x1E	RS	62	0x3E	>	94	0x5E	^	126	0x7E	~
31	0x1F	US	63	0x3F	?	95	0x5F	_	127	0x7F	DEL

몇가지 유용한 비교연산 함수들

https://www.mathworks.com/help/matlab/ref/is.html

함수명	동작	예시	참고
isempty(A)	비어있는 배열인지 확인 숫자, 문자 모두 가능	>> isempty([]) ans = logical 1	>> isempty(' ') ans = logical 0
ismember(A, B)	A의 각 원소가 B에 있는지 확인 숫자, 문자 모두 가능	>> ismember('aeiou','Hello World') ans = 1×5 logical 배열 0 1 0 1 0	
isequal(A, B)	A와 B의 값이 같은지 확인 (자료형은 구분하지 않음)	>> isequal([1 2 3],[1 2 3]') ans = logical 0	isequal('A',65) → true isequal(true,1) → true
ischar	문자열 배열인지 확인	>> ischar('hongik') ans = logical 1	>> ischar(['He', 108, 108, 'o']) ans = logical 1
isnumeric	숫자 배열인지 확인	>> isnumeric(pi) ans = logical 1	>> isnumeric(true) ans = logical 0



find 함수의 활용법

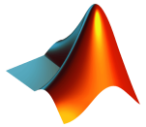
- find(X): X의 **nonzero** element의 위치를 **1차원 인덱스로** 반환
  - nonzero element == 'true' element
  - find(~X)는 zero element의 위치를 반환

```
>> find([1 2 0 0 3 4 0 5])
ans =
     1     2     5     6     8
>> a = [1 0 0
        0 2 3
        4 5 0];
>> find(a)
ans =
     1
     3
     5
     6
     8
```

```
>> a = [1 0 0
        0 2 3
        4 5 0];
>> [r,c,v] = find(a);
>> [r c v]
ans =
     1     1     1
     3     1     4
     2     2     2
     3     2     5
     2     3     3
```

```
>> a = magic(3)
a =
     8     1     6
     3     5     7
     4     9     2
>> a>3
ans =
  3×3 logical 배열
   1   0   1
   0   1   1
   1   1   0
>> find(a>3)
ans =
     1
     3
     5
     6
     7
     8
```

```
>> a
a =
     1     5     9
     2     6    10
     3     7    11
     4     8    12
>> a(:)
ans =
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
    11
    12
```



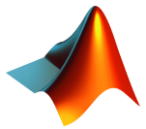
벡터에서 0을 없애는 7가지 방법 (+2: 반복문)

```
A = [1 0 0 2 3 0 4 5 0 0];

% method 1: find zero-value index and remove
idx = find(A==0);
A(idx) = [];

% method 2: same as 1, single line
A(find(A==0))=[];

% method 3: 'find' finds nonzero-value indices
A = A(find(A));

% method 4: 'find' gives nonzero values
[~,~,A] = find(A);

% method 5: using ~=0 instead of 'find'
A = A(A~=0);

% method 6: using logical
A = A(logical(A));

% method 7: same as 2, not using 'find'
A(A==0) = [];
```

예제 – 3의 배수인지 맞추는 미니게임

```
% 주어진 자연수가 3의 배수인지 맞추는 미니게임
n = randi(100);
msg = ['is ' num2str(n) ' divisible by 3? (Y/N): '];
TF = input(msg, 's');
answer = (rem(n,3)==0);

if strcmpi(TF,'Y') % upper() can be used
    TF = 1;
elseif strcmpi(TF,'N') % upper() can be used
    TF = 0;
end

if TF==answer
    disp('Good boy!')
else
    disp('Eeh~')
end
```

https://lazymatlab.tistory.com/75
『프로그래밍 언어 공부는
프로그래밍 **언어** 공부가 아니다.
프로그래밍 언어 공부이다.』



- 작업의 분리 및 구조화 → “알고리즘”, “로직”

예제 – 윤년 판단기

1. Years evenly divisible by 400 are leap years.
2. Years evenly divisible by 100 but *not* by 400 are not leap years.
3. All years divisible by 4 but *not* by 100 are leap years.
4. All other years are not leap years.

1. 서력 기원 연수가 4로 나누어 떨어지는 해는 **윤년**으로 한다. ...
2. 서력 기원 연수가 4, 100으로 나누어 떨어지는 해는 평년으로 한다. ...
3. 서력 기원 연수가 4, 100, 400으로 나누어 떨어지는 해는 **윤년**으로 둔다.

```
yr = input('type the year: ');
fprintf('Year %d is ', yr);
if rem(yr,400)==0
    disp('a leap-year.')
elseif rem(yr,100)==0
    disp('not a leap-year')
elseif rem(yr,4)==0
    disp('a leap-year')
else
    disp('not a leap-year')
end
```

https://lazymatlab.tistory.com/75
『프로그래밍 언어 공부는
프로그래밍 **언어** 공부가 아니다.
프로그래밍 언어 공부이다.』

- 작업의 분리 및 구조화 → "알고리즘", "로직"

Q&A