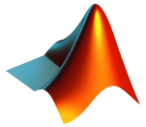
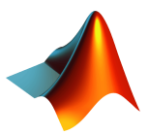
MATLAB 프로그래밍 및 실습

5강. 반복문



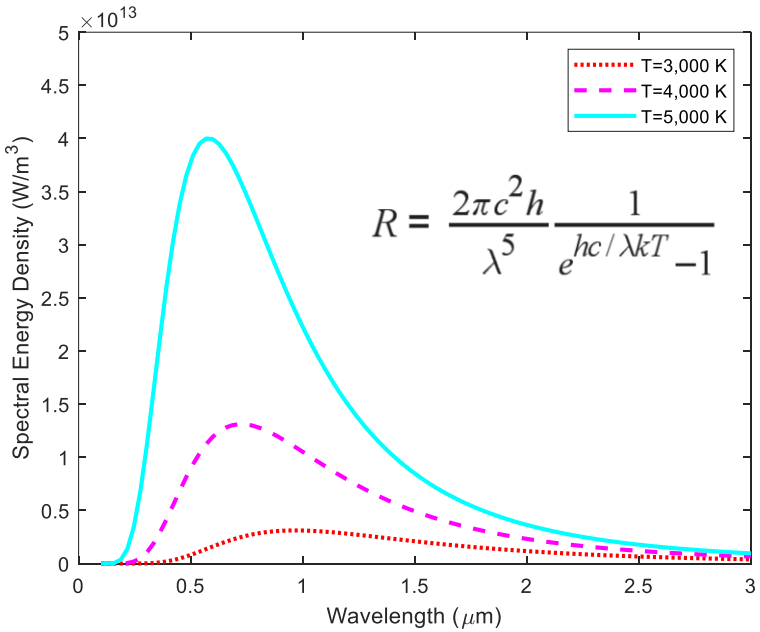
반복문



반복문이 필요한 이유

```
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(['Max score: ' num2str(max(scores))])
disp(['Avg score: ' num2str(mean(scores))])
```

- 점수를 100개 넣고 싶다면?
- 점수 개수가 매번 바뀐다면?



- 그래프를 3개 그리려면 똑같은 문장을 3번 써야할까?

반복작업을 스마트하게 바꿀 방법이 있지 않을까?

```
% 주어진 자연수가 3의 배수인지 맞히는 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
```

- 매번 실행하자니 귀찮다.

$$image = \begin{bmatrix} 36 & 0 & 18 & 9 \\ 27 & 54 & 9 & 0 \\ 81 & 63 & 72 & 45 \end{bmatrix}$$

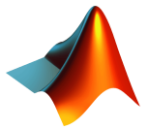
$$blurred = [40 \quad 30]$$

- 이미지 크기에 상관없이 잘 돌아가게 짜고 싶다.

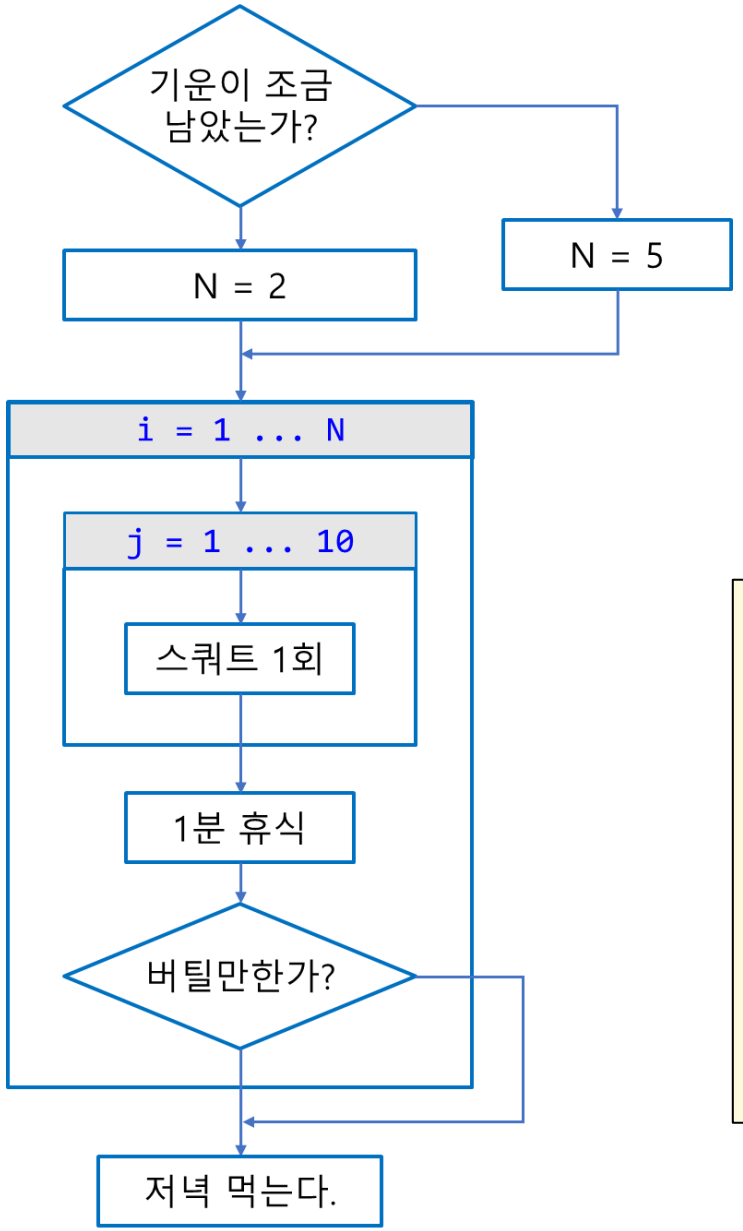
$$\left\lfloor \frac{36+0+18+27+54+9+81+63+72}{9} \right\rfloor = 40$$

$$\left\lfloor \frac{0+18+9+54+9+0+63+72+45}{9} \right\rfloor = 30$$

for문



C 학생의 퇴근 프로세스 – 의사코드



if-else

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

for
for

if-else, break

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

for문 – 간단한 예제

```
% 스쿼트 N회 실시 (N은 사용자 입력으로 받음)
N = input('How many times?: ');
for i = 1:N
    disp(['squat: ' num2str(i, '%02d') '-th'])
end
```

```
% 100까지 모든 3의 배수 출력
for i = 3:3:100
    fprintf('%3d\n', i)
end
```

```
% 기본문법
for i = first:incr:last
    명령문1;
    명령문2;
    ...
end
```

first:incr:last의 **원소 개수**만큼 반복

for i =(여기에)
- 음수 간격 가능
- 실수 가능
- 빈 행렬 가능 (실행 안함)
- 2차원 행렬 가능 (잘 안 씀)

- first, incr, last에 따라 **i가 변하며** 반복 수행
- incr는 생략하면 1
- first, incr, last에는 실수도 들어갈 수 있다.
- 써놓고 보니 콜론 연산자와 비슷하다? → 그 콜론 연산자가 맞는다!
- 콜론 연산자가 된다면? linspace도 된다!
- 그럼 아무 벡터나 될까? 된다!

하지만 first:incr:last 패턴을 가장 많이 사용한다.

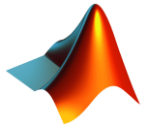
for문 – 간단한 예제

```
% disp를 활용한 예제
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(['Applicant: ' name ' (' age ')'])
disp(['Max score: ' num2str(max(scores))])
disp(['Avg score: ' num2str(mean(scores))])
```

```
% 이 예제를 for를 활용하여 수정
name = input('Type your name: ','s');
age = input('Type your age: ','s');
scores = zeros(5,1);
for i=1:5
    msg = [num2str(i) '-th score: '];
    scores(i) = input(msg);
end
disp(['Applicant: ' name ' (' age ')'])
disp(['Max score: ' num2str(max(scores))])
disp(['Avg score: ' num2str(mean(scores))])
```

※ zeros를 쓴 이유?

- 이제 몇 번이든 반복해서 입력할 수 있다.



for문 – 간단한 예제

```
for k=1:3:10
     x = k^2
end
```

```
>> x =
     1
x =
    16
x =
    49
x =
   100
```

```
k = 1:3:10
x = k.^2;
```

$$n! = \begin{cases} 1 & n = 0 \\ n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1 & n > 0 \end{cases}$$

```
% factorial using for-loop
fact = 1;
N = input('input an integer (>=0): ');
for i=1:N
    fact = fact*i;
end
```

factorial(N)

```
% sum of integers: 1-100

mysum = 0;
for i = 1:100
    mysum = mysum+i;
end
fprintf('sum from 1 to 100 = %d.\n', mysum);
```

$$\sum_{i=1}^{100} i$$

sum(1:100)

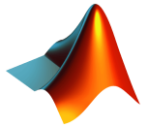
```
% sum of integers of range M-N

M = input('Type starting integer: ');
N = input('Type final integer: ');
mysum = 0;
for i=M:N
    mysum = mysum+i;
end
```

$$\sum_{i=M}^{N} i$$

sum(M:N)

Basel problem

$$\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$$



for문 – 또 다른 예제

```
% Fibonacci numbers

N = input('Which number of fibonacci?: ');
f = zeros(N,1);
f(1) = 1;
f(2) = 1;
for i=3:N
    f(i) = f(i-2)+f(i-1);
end
fprintf('%d-th fibonacci number = %d.\n', N, f(end))
```

fibonacci(N)

* N<3일 때도 잘 될까?

```
%% Tribonacci numbers

N = input('Which number of tribonacci?: ');
t = zeros(N,1);
t(1) = 1;
t(2) = 1;
t(3) = 2;
for i=4:N
    t(i) = t(i-3) + t(i-2) + t(i-1);
end
fprintf('%d-th tribonacci number = %d.\n', N, t(end))
```

* N<4일 때도 잘 될까?

https://mathworld.wolfram.com/TribonacciNumber.html

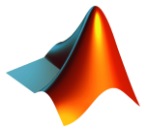
for문 – 좀 있어보이는 예제

```
% calculation of e as series

N = 4;
e = 0;
for i = 0:N
    e = e + 1/factorial(i);
end
fprintf('N = %d -> e = %5.3f, error = %6.3f%%\n', N, e, (e-exp(1))/exp(1)*100);
```

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{1} + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \cdots$$

```
% Leibniz formula for pi

N = 10;
mypi = 0;
for i = 1:N
    mypi = mypi + (-1)^(i-1)*4/(2*i-1);
end
fprintf('N = %d -> pi = %5.3f, error = %6.3f%%\n', N, mypi, (mypi-pi)/pi*100);
```

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4},$$



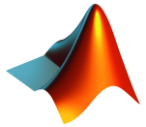
https://en.wikipedia.org/wiki/Leibniz_formula_for_%CF%80
https://en.wikipedia.org/wiki/E_(mathematical_constant)

for문 내에서
i의 적절한 활용이 킬포

같은 작업, 다양한 방법

```
%% two methods for Leibniz formula for pi

N = 10;

% for-loop
mypi1 = 0;
for i = 1:N
    mypi1 = mypi1 + (-1)^(i-1)*4/(2*i-1);
end
disp(['mypi1 = ' num2str(mypi1, '%5.3f')])

% vectorization
sgn = (-1).^(0:N-1);
den = 1:2:(2*N-1);
mypi2 = 4*sum(sgn./den);
disp(['mypi2 = ' num2str(mypi2, '%5.3f')])
```

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4},$$

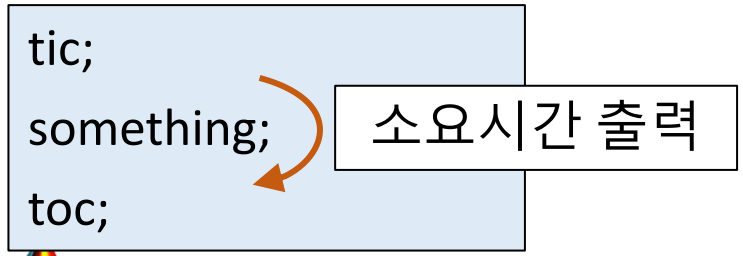


같은 작업, 다양한 방법

```
% 표준편차를 구하는 세 가지 방법
N = 1e4;
x = randn(N,1);
m = mean(x);

% method 1: for-loop
disp('using for-loop')
tic;
mysum = 0;
for i=1:length(x)
    mysum = mysum + (x(i)-m)^2;
end
mystd1 = sqrt(mysum/(N-1));
toc;
```

```
% method 2: vectorization
disp('using vectorization')
tic;
mystd2 = sqrt(sum((x-m).^2)/(N-1));
toc;

% method 3: built-in function
disp('using built-in std')
tic;
mystd3 = std(x);
toc;
```

- 다양한 패턴을 알아두면 좋은 이유?
  - 상황에 따라 for문이 편할 때도 있고, 벡터화가 편할 때가 있다.



원뿔의 부피 계산 – 누가 더 빠를까?

- 지름과 높이가 다른 원뿔 1,000,000개의 부피를 구하시오.

```
% for문 사용
N = 1e6;
D = rand(N,1);
H = rand(N,1);
V = zeros(N,1);

tic;
for i = 1:N
   V(i) = 1/12*pi*(D(i)^2)*H(i);
end
t1 = toc;
```

```
% 원소별 연산(벡터화; vectorization)
N = 1e6;
D = rand(N,1);
H = rand(N,1);
tic;
V = 1/12*pi*(D.^2).*H;
t2 = toc;
```

오호… 벡터화가 훨씬 빠르네!
앞으로 무조건 벡터화만 쓰면 되겠지?

https://www.mathworks.com/help/matlab/matlab_prog/vectorization.html

e 계산 – 누가 더 빠를까?

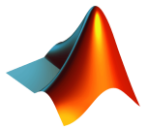
```
N = 1e6;

% using for-loop
e = 0;
tic;
for i = 0:N
    e = e + 1/factorial(i);
end
toc;

% utilizing factorial as vectorization
tic;
e = sum(1./factorial(0:N));
toc;

% smart re-design of for-loop
e = 1;
to_add = 1;
tic;
for i = 1:N
    to_add = to_add/i;
    e = e + to_add;
end
toc;
```

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{1} + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \cdots$$

- 벡터화가 만능은 아니다!
  - 상황에 따라 더 나은 있을 수 있다.
  - 여러 방법을 두루두루 알아두자.

for문 vs 벡터화 (원소별 연산)

- for문을 쓰면
  - **일반적으로** 코드 작성은 쉽지만 실행 시간이 오래 걸린다. (=비싸다.)
- 벡터화 연산은
  - 과정은 쓰지만 (코드 난이도 ↗) **일반적으로** 열매는 달다. (실행시간 ↘)
  - 가독성이 올라간다...?
    - 식이 복잡해지면 벡터화 연산이 오히려 읽기 어려울 수도 있다.
    - 데이터가 적다면 for문이 나을지도 모른다.
  - 코드가 간결해진다.
    - 하지만 "짧아서 멋있는" 코드가 반드시 좋은 코드가 아니다.
    - **짧은 코드보다 가독성이 (거의) 항상 더 중요하다.**
  - 반드시 빠른 것은 아니므로, 더 나은 방법이 있는지 확인하고 작성하자.

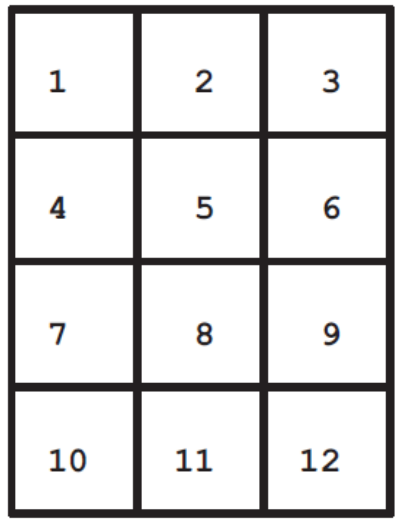


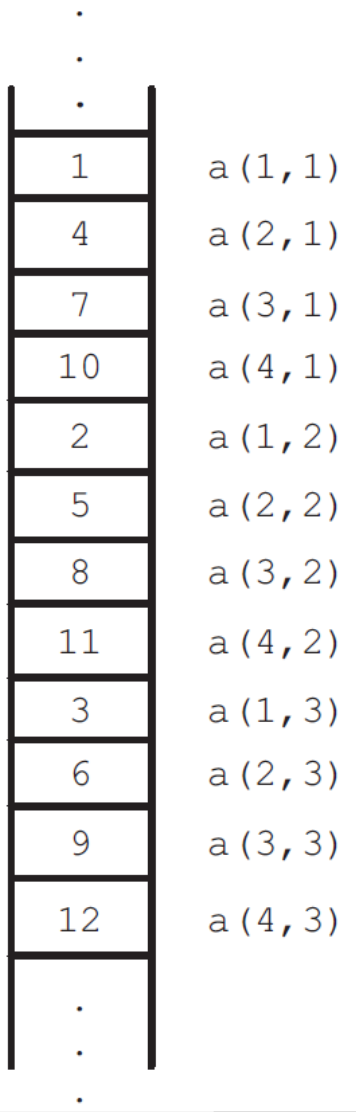
행렬을 미리 만들어두는 이유?

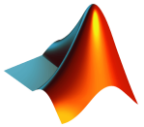
```
N = 1e7;

% no pre-allocation
tic;
for i = 1:N
    a(i) = i;
end
toc;

% pre-allocation (zeros)
tic;
b = zeros(N,1);
for i = 1:N
    b(i) = i;
end
toc;
```

```
% pre-allocation
% (last index first)
tic;
c(N) = 0;
for i = 1:N
    c(i) = i;
end
toc;
```

- 행렬을 미리 만드는 방법
  - zeros
  - 역 인덱싱

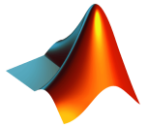






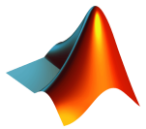
2중 for문 – reshape, repmat을 구현해보자.

- 로직을 먼저 생각 → 로직을 구현



2중 for문 – 구구단

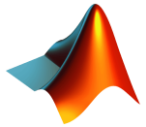
- 로직을 먼저 생각 → 로직을 구현



2중 for문 – boxBlur

- 로직을 먼저 생각 → 로직을 구현

$$\begin{bmatrix} 14 & 0 & 29 & 29 & 32 \\ 59 & 48 & 11 & 0 & 60 \\ 36 & 0 & 33 & 88 & 70 \\ 3 & 0 & 21 & 0 & 0 \\ 54 & 87 & 0 & 62 & 92 \end{bmatrix} \Rightarrow \begin{bmatrix} 25 & 26 & 39 \\ 23 & 22 & 31 \\ 26 & 32 & 40 \end{bmatrix}$$



2중 for문 – boxBlur2

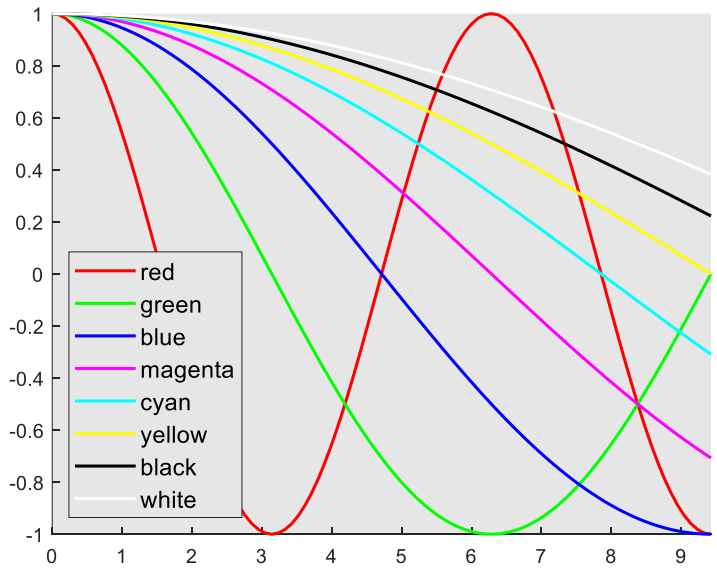
- boxBlur 중, 값이 0인 픽셀은 제외하도록 수정

※ box는 내장함수

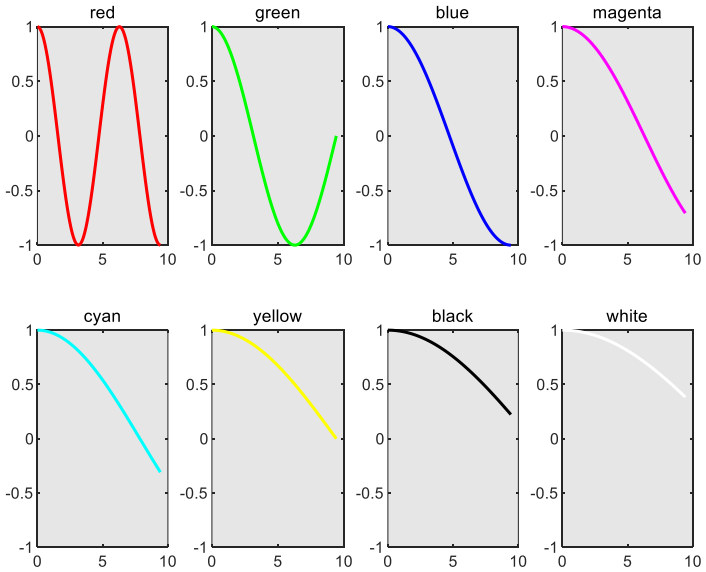
```
% method 1
blurred1 = zeros(size(img)-[2,2]);
for i=1:size(img,1)-2
    for j=1:size(img,1)-2
        ibox = img(i:i+2,j:j+2);
        blurred1(i,j) = floor(sum(ibox,'all')/sum(ibox>0,'all')); % sum(ibox(:))
    end
end

% method 2
blurred2 = zeros(size(img)-[2,2]);
for i=1:size(img,1)-2
    for j=1:size(img,1)-2
        ibox = img(i:i+2,j:j+2);
        map = find(ibox);
        blurred2(i,j) = floor(sum(ibox(:))/length(map));
    end
end
```

plot, subplot에도 for문을 쓸 수 있다.

```
figure, hold on
colors = 'rgbmcykw';
x = linspace(0,3*pi);
for n = 1:length(colors)
    y = cos(x/n);
    plot(x, y, colors(n), 'linewidth',1.5);
end
set(gca, 'color', [.9 .9 .9])
axis tight

h = legend('red', 'green', 'blue', 'magenta', ...
    'cyan', 'yellow', 'black', 'white', ...
    'fontsize', 12, 'location', 'sw');
```



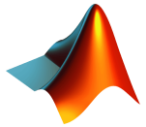

```
figure,
colors = 'rgbmcykw';
titles = {'red', 'green', 'blue', 'magenta', ...
    'cyan', 'yellow', 'black', 'white'};
x = linspace(0,3*pi);
for n = 1:length(colors)
    subplot(2,4,n)
    y = cos(x/n);
    plot(x, y, colors(n), 'linewidth',1.5);
    title(titles{n})
    set(gca, 'color', [.9 .9 .9])
    ylim([-1 1])
end
```



loop index를 쓰는 두 가지 방법

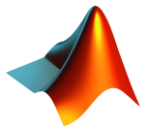
```
%% two ways to use loop index

x = -1:0.1:1;
y = zeros(size(x));
for i = 1:length(x)
    y(i) = x(i)^2;
end

y2 = [];
for ix = x
    y2 = [y2 ix^2];
end
```

- 1:N을 쓰는 것 vs 값을 직접 쓰는 것
- 주로 전자를 많이 사용함

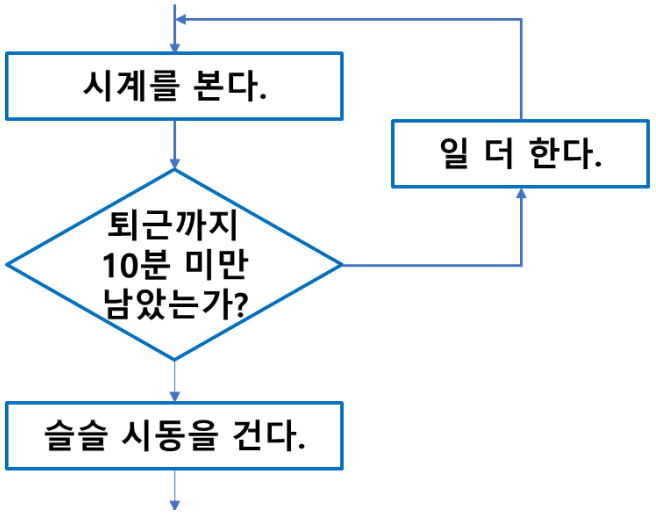


https://lazymatlab.tistory.com/73

while문

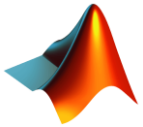


C학생의 퇴근 프로세스 중...




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

- for문은 반복 회수가 정해져 있었다.
- 퇴근 프로세스엔 반복 회수가 정해져 있지 않다!
- loop의 탈출조건이 필요하다!

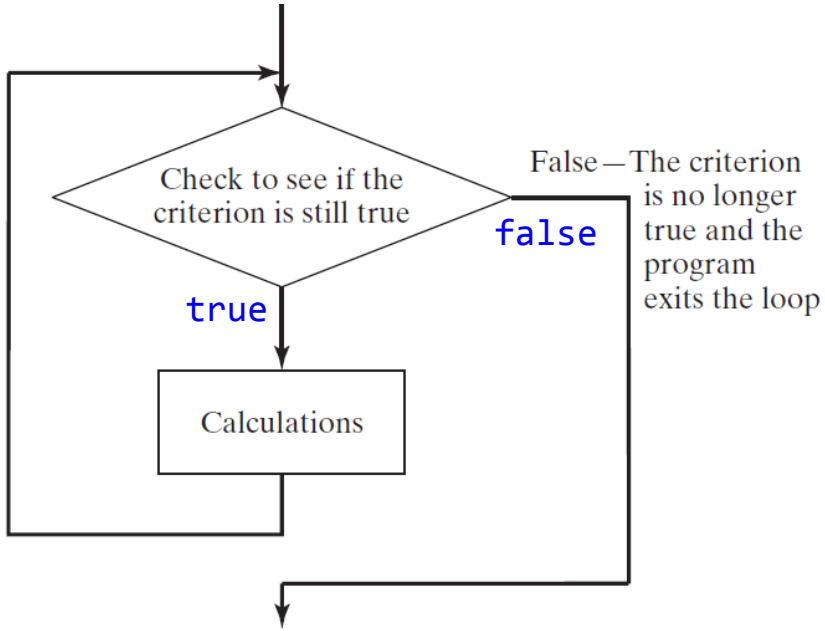


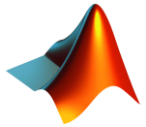
while문 – 간단한 예제

```
% 체력이 고갈되기 직전까지 스쿼트 반복
i = 0;
stamina = 100;
while stamina > 0
    i = i + 1;
    disp(['squat: ' num2str(i, '%03d') '-th'])
    stamina = stamina - 3;
end
```

```
% 기본문법
while 조건문
    명령문1;
    명령문2;
    ...
end
```

- while 뒤의 조건문은 **"탈출조건"**이 아니다!
  - **"계속조건"**이다.
  - 조건문이 true이면 실행-반복, false이면 빠져나감
- while을 빠져나간 직후의 stamina는 얼마일까?



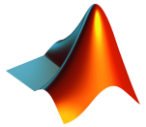


while문 – 간단한 예제

```
% 입력받은 수까지 모든 3의 배수 출력

num = input('input max. number: ');
i = 3;
while i<=num
    fprintf('%3d\n', i)
    i = i + 3;
end
```

※ while을 빠져나온 이후 i는 얼마일까?

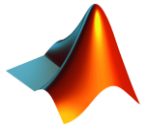
```
% 값을 입력받아 log 값을 출력하되, 음수이면 다시 입력받음

x = input('Enter a positive number: ');
while x<=0
    disp('for x<=0, lox(x) is not defined in real number')
    x = input('I said a POSITIVE number, dude...: ');
end
fprintf('log(%5.2f) is approx. %5.2f.\n', x, log(x))
```



Leibniz formula를 while문으로 다시 써보자.

```
%% Leibniz formula for pi using while-loop

tol = 2e-3;
err = 100; % make initial value larger than tolerance
mypi = 0;
n = 1;

while err>tol
    mypi = mypi + (-1)^(n-1)*4/(2*n-1);
    err = abs(mypi-pi);
    fprintf('N=%4d -> mypi=%7.5f, error=%8.5f\n', n, mypi, mypi-pi);
    n = n + 1;
end
```

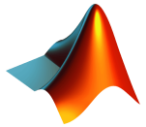
$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4},$$



e 계산을 while문으로 다시 써보자.

```
%% calculation of e using while-loop

tol = 1e-3;
err = 100; % make initial value larger than tolerance
e = 0;
n = 1;

while err>tol
    e = e + 1/factorial(n-1);
    err = abs(e-exp(1));
    fprintf('N=%d -> e=%7.5f, error=%7.5f\n', n, e, e-exp(1));
    n = n + 1;
end
```

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{1} + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \cdots$$



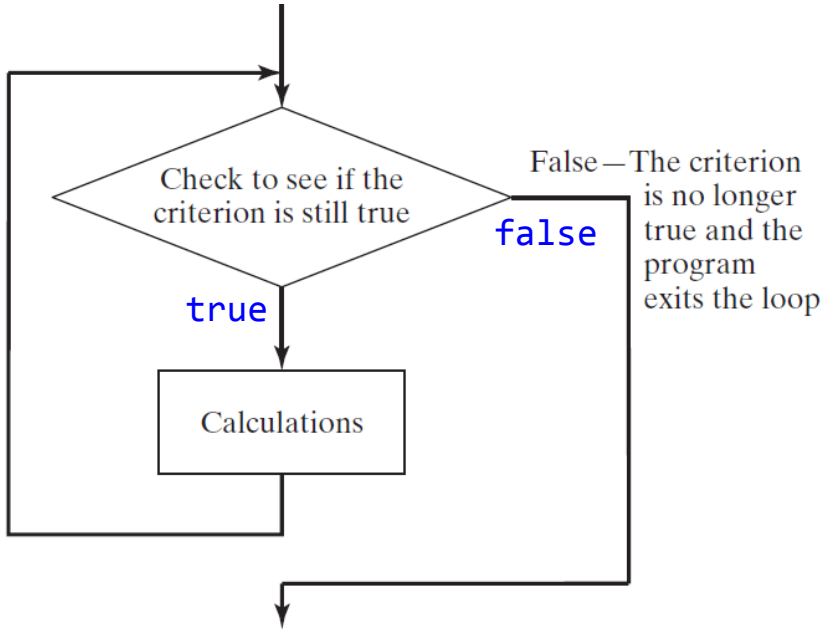
잠깐, 조건문이라고?

- `if, elseif` 뒤에 나오던 바로 그 조건문이 맞는다!
- 빈 행렬, 0, 0.0은 false와 같다.
- 논리연산자를 쓸 수 있다.

```
% 기본문법
while 조건문
    명령문1;
    명령문2;
    ...
end
```

```
%% Leibniz formula for pi using while-loop and calculation limit
tol = 2e-3;
err = 100;
iter_max = 100;
mypi = 0;
n = 1;

while err>tol && n<=iter_max
    mypi = mypi + (-1)^(n-1)*4/(2*n-1);
    err = abs(mypi-pi);
    fprintf('N=%4d -> mypi=%7.5f, error=%10.7f\n', n, mypi, mypi-pi);
    n = n + 1;
end
if n>iter_max
    fprintf('iteration limit exceeded.\n');
else
    fprintf('mypi got close enough.\n')
end
```

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4},$$

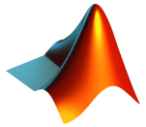


- 조건문에 행렬이 올 때 조심해야 한다.

언제 while을 쓰고 언제 for를 쓸까?

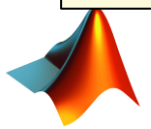
- `for`
  - 반복회수가 명확히 정해져 있을 때
  - 예) 30번째 피보나치 수
- `while`
  - 반복회수가 명확히 정해져 있지 않을 때 (탈출조건이 반복회수가 아닐 때)
  - 예) 1,000,000보다 큰 첫 피보나치 수
  - 예) 측정값을 매 iteration마다 읽어서, 임계값을 넘으면 정지
  - 예) 측정값을 매 iteration마다 읽어서, 임계값을 넘은 것이 N회 이상이 되면 정지

- 아래처럼 생각할 수도 있다.
  - `for`문의 탈출조건: 반복회수가 `for` 뒤의 배열 길이를 넘어갔을 때
  - `while`문의 탈출조건: `while` 뒤의 조건문이 false가 될 때



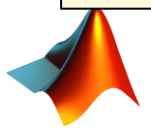
while 내에서 if문의 사용

```
%% 주사위 두 개를 굴려 합이 3이 되어야 빠져나옴
%% 굴리는 도중 같은 눈이 나온 경우의 회수를 셈

mysum = 0;
target = 3;
n_double = 0;
while mysum ~= target
    d1 = randi(6,1);
    d2 = randi(6,1);
    mysum = d1 + d2;
    fprintf('%d, %d', d1, d2)
    if d1==d2
        fprintf(' -> double!')
        n_double = n_double + 1;
    end
    fprintf(newline);
end
fprintf('%d double case occurred.\n', n_double)
```



while 내에서 if문의 사용

```
%% 점수를 grade로 보여줌 (-1이면 빠져나옴)

score = input('input your score (-1 to quit): ');
while score>=0
    if score>100
        disp('wrong score. type again')
    elseif 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
    score = input('input your score: ');
end
```



while 내에서 if문의 사용

```
%% 행렬에서 0을 없애는 방법
A = [1 0 0 2 3 0 4 5 0 0];

% for-loop
B = [];
for i=1:length(A)
    if A(i)~=0
        B = [B A(i)];
    end
end

% while-loop
i = 1;
while i<=length(A)
    if A(i)==0
        A(i) = [];
    else
        i = i + 1;
    end
end
```

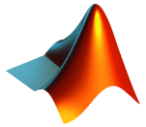
for문 안에서도 if를 쓸 수 있다.

```
V=[5, 17, -3, 8, 0, -7, 12, 15, 20 -6, 6, 4, -2, 16];
n=length(V);
for k=1:n
    if V(k)>0 & (rem(V(k),3)==0 | rem(V(k),5)==0)
        V(k)=2*V(k);
    elseif V(k) < 0 & V(k) > -5
        V(k)=V(k)^3;
    end
end
V
```

Setting n to be equal to the number of elements in V.

for-end loop.

if-elseif-end statement.



for문 안에서도 if를 쓸 수 있다.

```
%% 3D surface with 4 different eqs. for each quadrant

x = -1:0.1:1;
y = -1:0.1:1;

[xx,yy] = meshgrid(x,y);
zz = zeros(size(xx));
for i=1:size(zz,1)
    for j=1:size(zz,2)
        ix = xx(i,j);
        iy = yy(i,j);
        if ix>=0 && iy>=0
            zz(i,j) = ix + iy;
        elseif ix>=0 && iy<=0
            zz(i,j) = ix + iy^2;
        elseif ix<0 && iy>=0
            zz(i,j) = ix^2 + iy;
        else
            zz(i,j) = ix^2 + iy^2;
        end
    end
end
```

$$f(x,y) = \begin{cases} x+y & x \geq 0, y \geq 0 \\ x+y^2 & x \geq 0, y < 0 \\ x^2+y & x < 0, y \geq 0 \\ x^2+y^2 & x < 0, y < 0 \end{cases}$$

break - while을 탈출하는 또 다른 방법

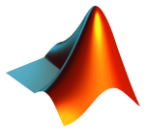
```
%% Leibniz formula for pi using while-loop and break

tol = 2e-3;
err = 100;
iter_max = 100;
mypi = 0;
n = 1;

while err>tol
    mypi = mypi + (-1)^(n-1)*4/(2*n-1);
    err = abs(mypi-pi);
    fprintf('N=%4d -> mypi=%7.5f, error=%10.7f\n', n, mypi, mypi-pi);
    n = n + 1;
    if n>iter_max
        break
    end
end

if n>iter_max
    fprintf('iteration limit exceeded.\n');
else
    fprintf('mypi got close enough.\n')
end
```

```
while 조건문
    ...
    ...
    if 조건문
        break
    end
    ...
    ...
end
```

※ break는 while을 빠져나가는 키워드
(if를 빠져나가는 것이 아님)
※ break를 if 없이 쓰면 어떻게 될까?

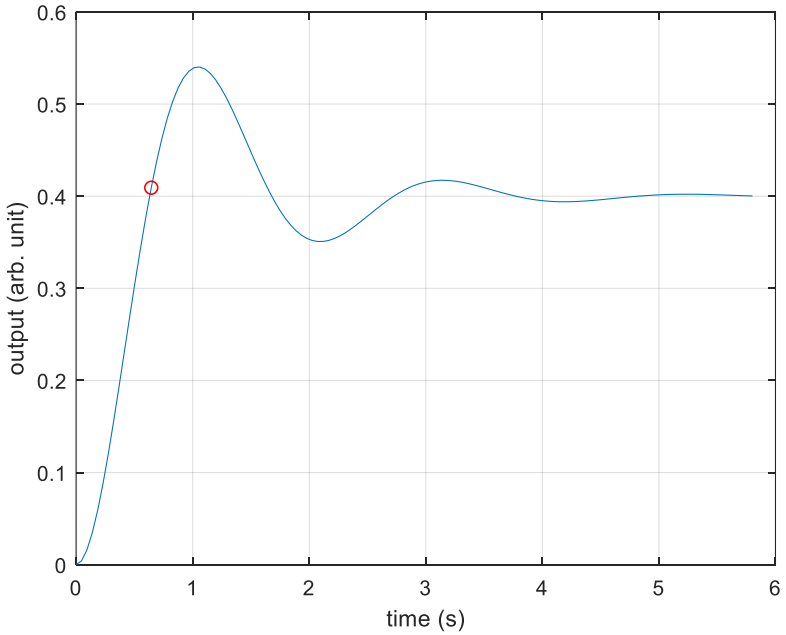
break의 활용

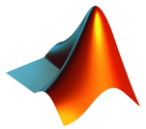
```
%% roll two dices until sum is 3, break if double

mysum = 0;
target = 3;
n_double = 0;
while mysum ~= target
    d1 = randi(6,1);
    d2 = randi(6,1);
    mysum = d1 + d2;
    fprintf('%d, %d', d1, d2)
    if d1==d2
        fprintf(' -> double!\n')
        break
    end
    fprintf(newline);
end
fprintf('%d double case occurred.\n', n_double)
```



break의 활용 – for문에도 쓸 일이 (가끔) 있다.

```
%% step response of a 2nd order system

sys = tf(4,[1 2 10]);
[y, tOut] = step(sys);

figure, hold on, grid on, box on
plot(tOut, y),

for i=1:length(tOut)
    if y(i)>0.4 % found the first point over y=0.4
        plot(tOut(i), y(i), 'ro')
        break
    end
end
xlabel('time (s)')
ylabel('output (arb. unit)')
```

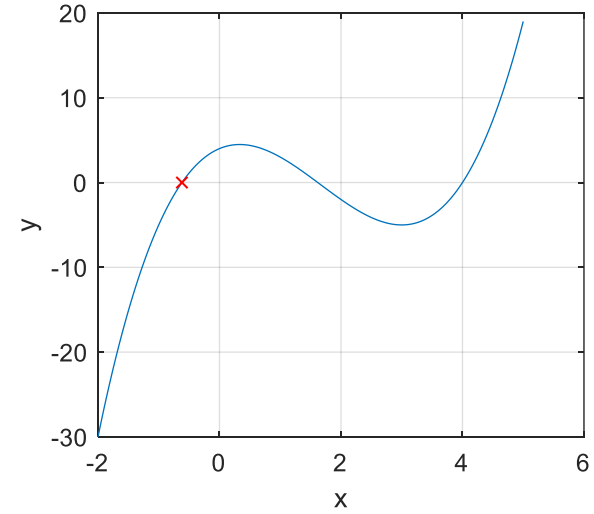


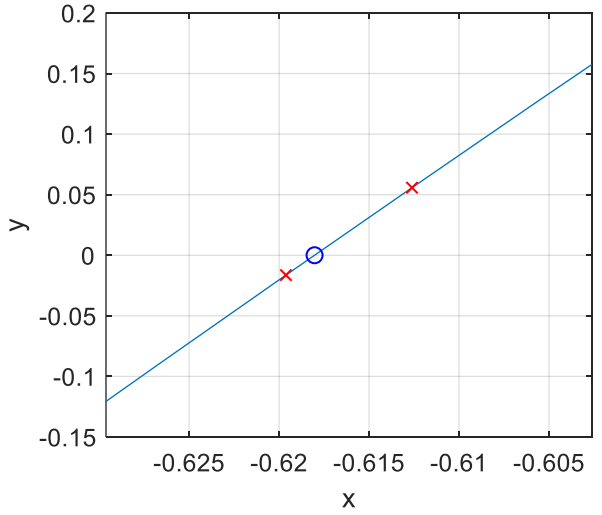


https://www.mathworks.com/help/control/ref/lti.step.html

break의 활용 – for문에도 쓸 일이 (가끔) 있다.

```
a = 1; b = -5; c = 3; d = 4;

x = linspace(-2,5,1000);
y = a*x.^3 + b*x.^2 + c*x + d;

figure,
subplot(2,1,1), plot(x,y), hold on, grid on, box on, xlabel('x'), ylabel('y')
subplot(2,1,2), plot(x,y), hold on, grid on, box on, xlabel('x'), ylabel('y')

for i=1:length(x)
    if y(i)>0
        x1 = x(i-1);
        x2 = x(i);
        y1 = y(i-1);
        y2 = y(i);
        xx = interp1([y1, y2], [x1, x2], 0);
        subplot(2,1,1), plot(xx,0,'rx')
        subplot(2,1,2),
        plot(x(i-1),y(i-1),'rx')
        plot(x(i),y(i),'rx')
        plot(xx,0,'bo')
        xlim([x(i-1)-0.01 x(i)+0.01])
        break
    end
end
```





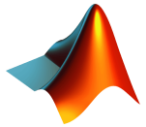
무한루프 - break의 이상한(?) 활용법

- while 뒤 조건문이 항상 참이라면?

```
while 조건문
    ...
    ...
    ...
    ...
    ...
end
```

```
while 1 % 또는 true
    ...
    if 조건문
        break
    end
    ...
end
```

- 무한루프 작성 시 탈출조건(if-break)이 반드시 있어야 한다.

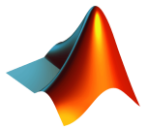


무한루프 활용 예제 – 학점 출력 코드

```
%% 매번 실행해야 해서 귀찮았던 코드

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
```

```
%% 코드 검증을 위한 임시 코드
% loop 빠져나가기: ctrl+C

while true
    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
end
```

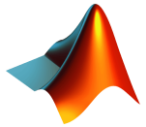
※ 검증을 위한 임시 코드일 뿐,
검증 후에는 제대로 작성해야 한다.



예제 – FizzBuzz

```
%% FizzBuzz

while true
    num = input('input an integer (0 to quit): ');
    if num==0
        break
    end
    if rem(num,15)==0 % 15의 배수를 먼저 확인해야 함
        disp('FizzBuzz!')
    elseif rem(num,3)==0
        disp('Fizz!')
    elseif rem(num,5)==0
        disp('Buzz!')
    end
end
```

tip. 무한루프를 쓰지 않는 경우와 비교해보자.

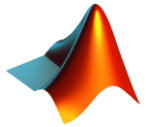


break가 있다면 continue도 있다.

```
%% print only odd numbers

i = 0;
while i<30
    i = i + 1; % this should be before 'if'
    if rem(i,2)==0 % if i is even number
        continue % skip
    end
    disp(i)
end
```

```
%% print only multiples of 7

for n = 1:50
    if rem(n,7) % nonzero == true
        continue
    end
    disp(['Divisible by 7: ' num2str(n)])
end
```

```
while 조건문
    ...
    ...
    if 조건문
        break
    end
    ...
    if 조건문
        continue
    end
    ...
    ...
end
```



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

break가 있다면 continue도 있다.

```
% 값을 입력받아 log 값을 출력하되, 음수이면 다시 입력받음

x = input('Enter a positive number: ');
while x<=0
    disp('for x<=0, lox(x) is not defined in real number')
    x = input('I said a POSITIVE number, dude...: ');
end
fprintf('log(%5.2f) is approx. %5.2f.\n', x, log(x))
```

```
%% 값을 입력받아 log 값을 출력하되, 음수이면 다시 입력받음

while true
    x = input('input a positive number (0 to quit): ');
    if x<0
        disp('lox(x) is not defined in real number for x<0')
        continue
    elseif x==0
        break
    end
    fprintf('log(%5.2f) is approx. %5.2f.\n', x, log(x))
end
```

```
while 조건문
    ...
    ...
    if 조건문
        break
    end
    ...
    if 조건문
        continue
    end
    ...
    ...
end
```

break가 있다면 continue도 있다.

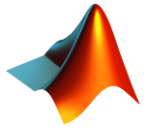
```
%% input scores until 'q' entered

name = input('Type your name: ','s');
age = input('Type your age: ','s');
scores = [];
n = 1;

while true
    msg = [num2str(n) '-th score: '];
    num = input(msg, 's');
    if strcmpi(num, 'q')
        break
    elseif isnan(str2double(num)) % if num contains character
        disp('input must be numeric.')
        continue
    end
    scores = [scores; str2double(num)]; % scores(i) = ...
    n = n + 1;
end
disp(['Applicant: ' name ' (' age ')'])
disp(['Max score: ' num2str(max(scores))])
disp(['Avg score: ' num2str(mean(scores))])
```

tip. n을 쓰지 않는 방법도 가능하다.

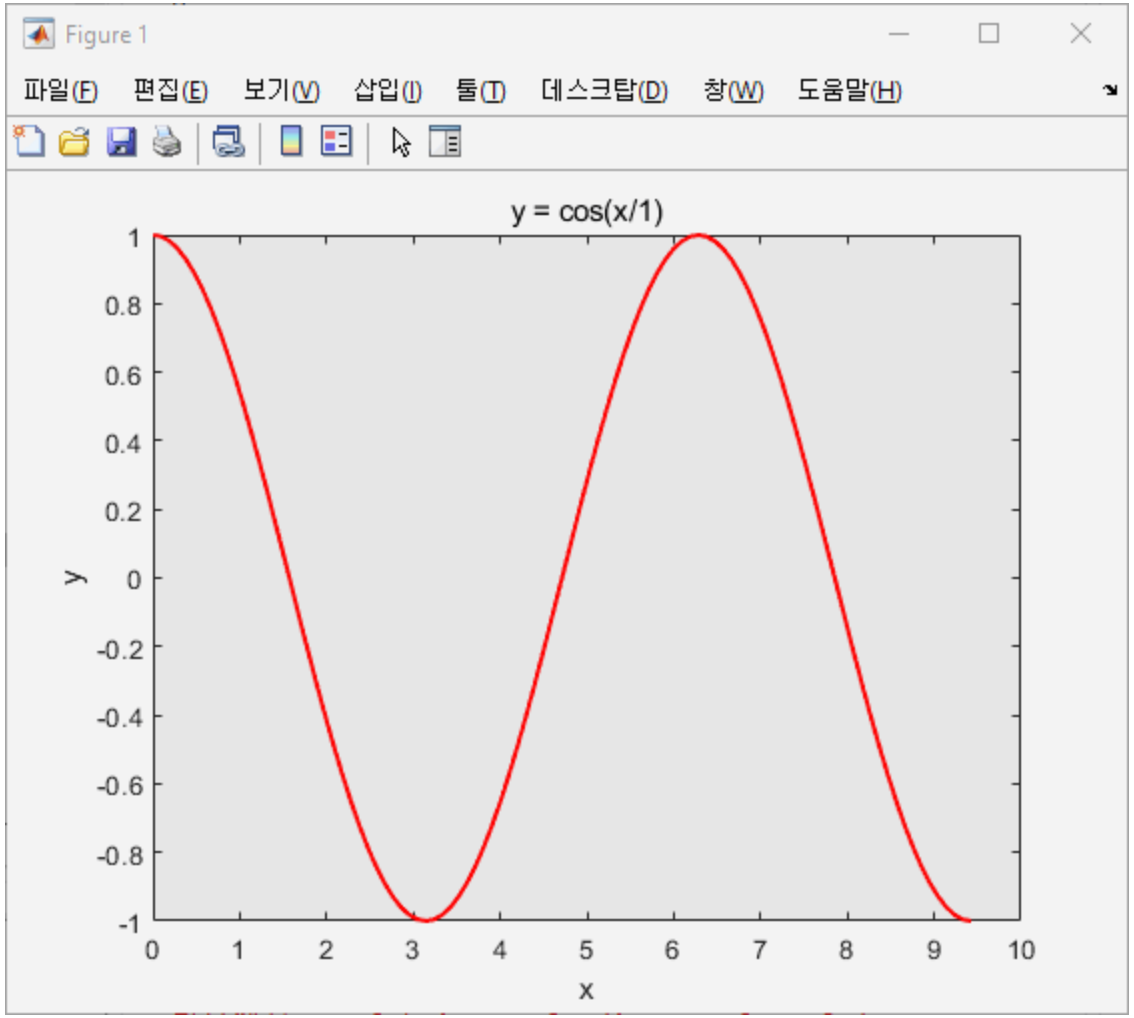
예제 – 숫자를 입력받아, 50보다 큰지 출력

- 숫자 하나를 입력받아, 50보다 큰지 작은지를 출력
- 문자가 들어오면 경고문 출력하고 다시 받음
- 0-100 범위를 벗어나면 경고문 출력하고 다시 받음
- 'q'가 입력되면 종료
- 무엇이 입력되든 에러는 발생하지 않아야 함

- 또 다른 예제
  - Fibonacci 수열의 비율은 golden ratio에 수렴한다?



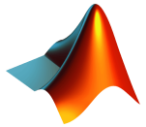
infinite cosine



tip. loop 내에서 pause() 활용

while문을 쓸 때 조심할 점

- 시작 값이 제대로 설정되었는지 확인
- 탈출조건(계속조건)이 제대로 설정되었는지 확인
- 무한루프에 빠질 가능성 없는지 확인



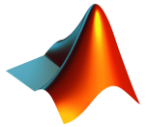
강의요약

- 조건문
  - if, if-else (양자택일), if-elseif-else (다지선다)
  - 조건문에 비교연산자(>, <, ==, ~= 등) , 논리연산자(&, |, ~)를 사용할 수 있다.
  - logical 자료형 (true, false)
  - 문자의 대소비교 (아스키 코드)
- for문 – 반복회수가 정해져 있을 때 (for 뒤의 배열 길이만큼 반복)
  - 2중 for문 가능 (다중 for문도 가능)
  - 벡터화, 내장함수 등과의 속도 비교 → 그때그때 좋은 방법이 다를 수 있다.
  - plot에도 사용할 수 있다.
- while문 – 반복회수가 안 정해져 있을 때 (while 뒤의 조건문이 false일 때 빠져나옴)
  - 탈출조건(계속조건)의 적절한 설계가 핵심이다.
  - break로 탈출가능 (무한루프 작성 시 break 반드시 필요)
  - continue를 만나면 while문 처음으로 돌아간다.

지금까지 배운 것 총정리

- 행렬 다루기 (인덱싱, 원소별 연산, repmat, reshape)
- 사용자 입력, 출력 (input, disp, fprintf)
- 시각화 (plot, hold, plot3, surf, mesh, meshgrid, subplot)
- 조건문 (if, if-else, if-elseif-else)
- 반복문 (for, while, break, continue)

- 이제 배울것
  - 함수
  - 파일입출력 (이미지, 텍스트, 엑셀)



Q&A