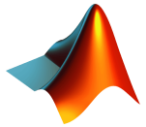
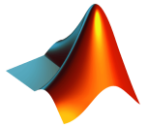
MATLAB 프로그래밍 및 실습

7강. 프로그래밍 스킬의 이해 1



searchpath 개념의 이해

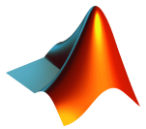


꼭 모든 게 현재 폴더에만 있어야 할까?

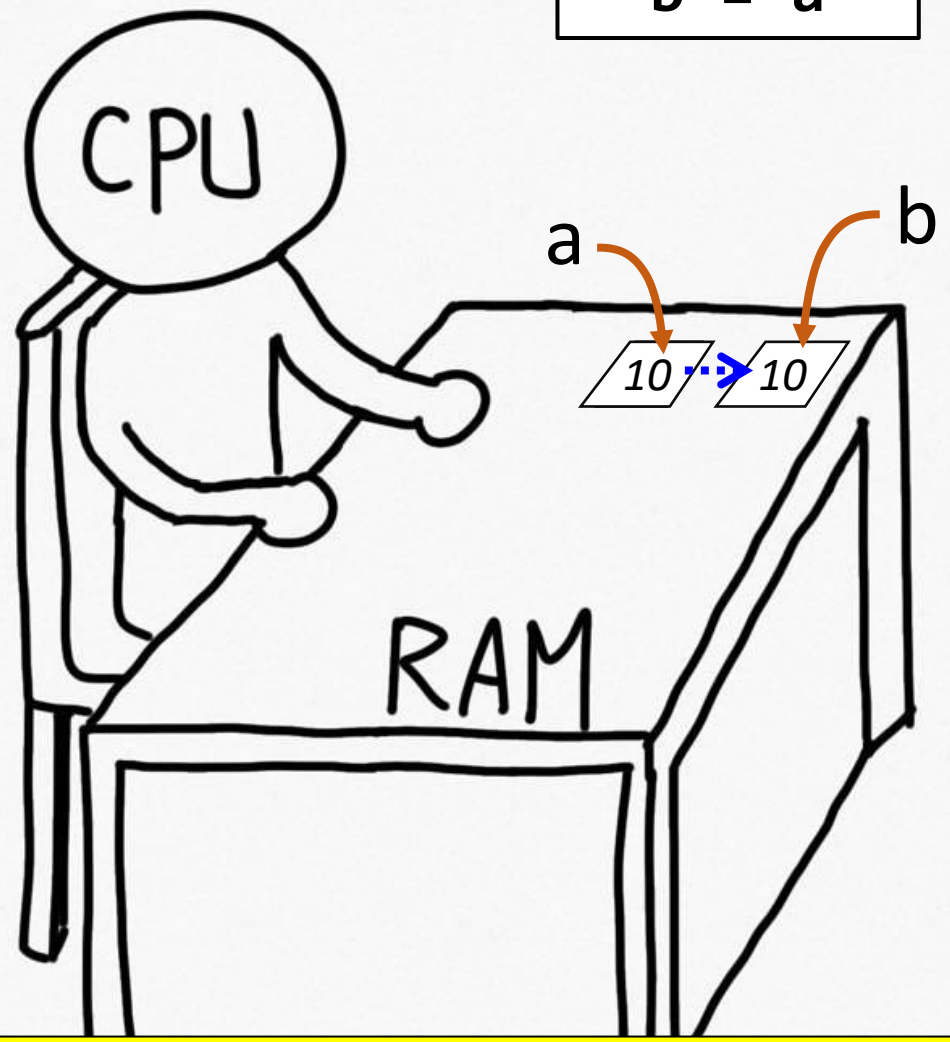
- cameraman.tif는 현재 폴더에 없는데 왜 읽혀질까?
- max, sort, find 같은 내장 함수들은 어디에 있는 걸까?
- 자주 쓰는 함수들을 한 폴더에 모아놓고 쓸 수는 없을까?
- 명령 창에 아래와 같이 쳐보자. (또는 open 함수명)

```
>> which cameraman.tif
C:\Program Files\MATLAB\R2019a\toolbox\images\imdata\cameraman.tif
>>
>> which find
built-in (C:\Program Files\MATLAB\R2019a\toolbox\matlab\elmat\find)
>> which max
built-in (C:\Program Files\MATLAB\R2019a\toolbox\matlab\datafun\max)
>>
```

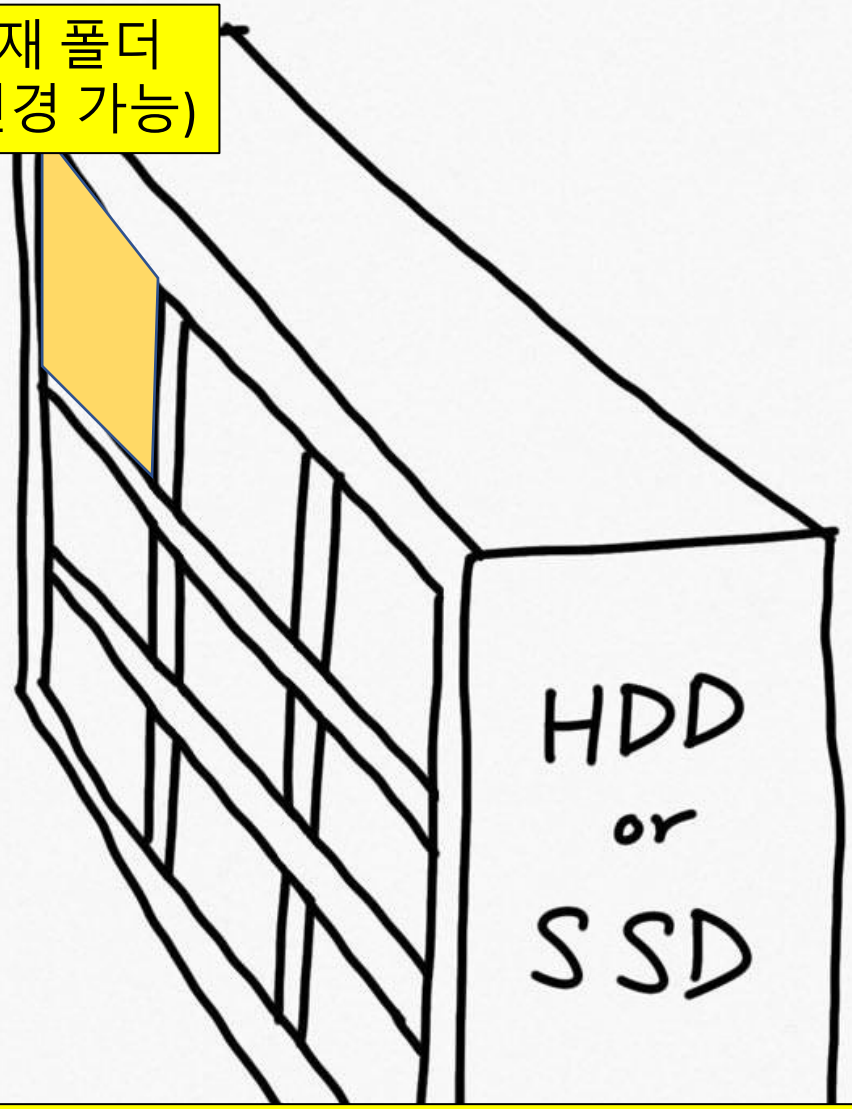
- 여기에 있는 줄 어떻게 알고 가져올까?



```
a = 10
b = a
```



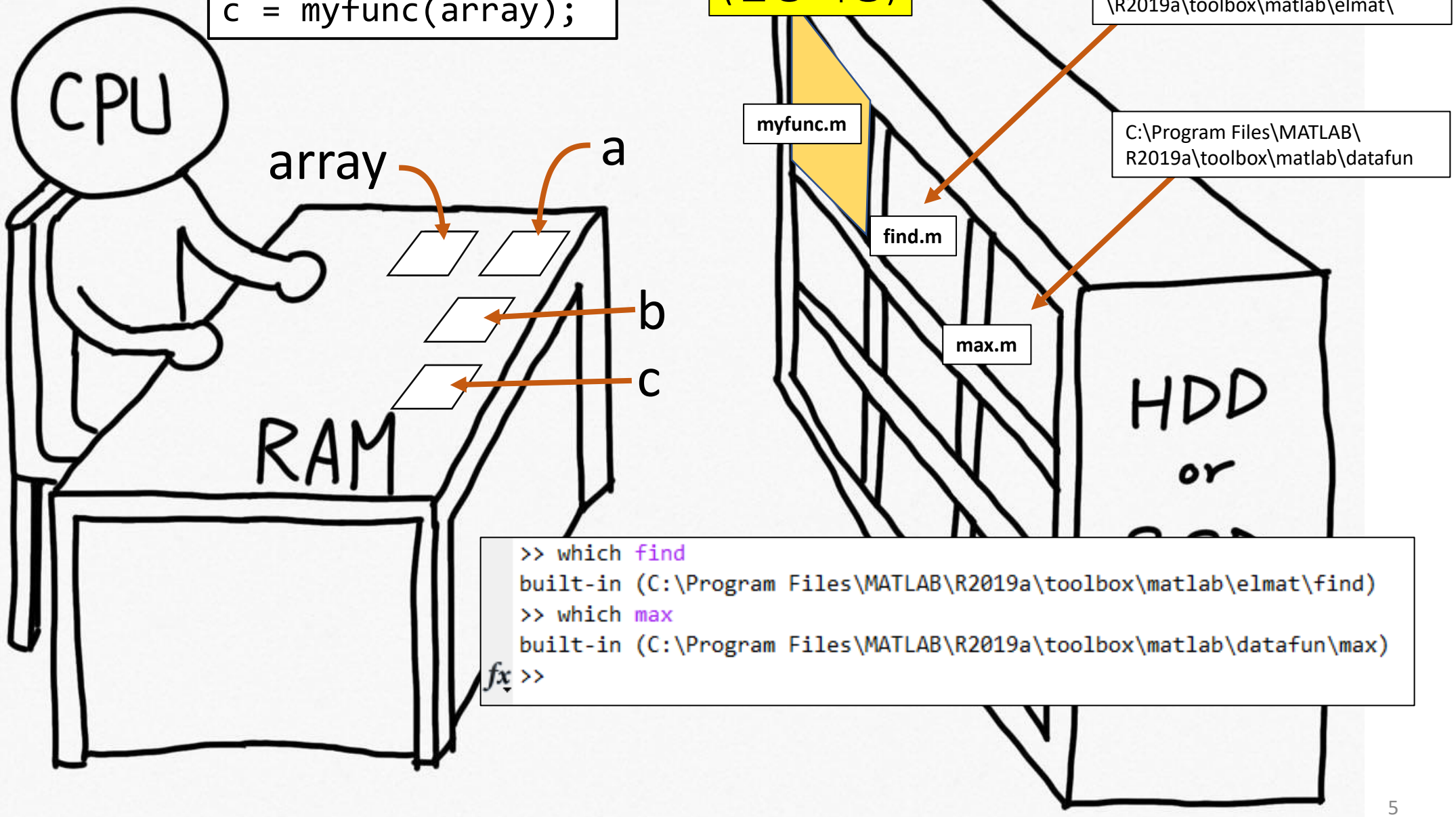
현재 폴더
(변경 가능)



RAM (메모리)
- CPU가 현재 작업 중인 것들을 올려두는 공간
- 컴퓨터의 **RAM** = 매트랩의 **"작업 공간"**

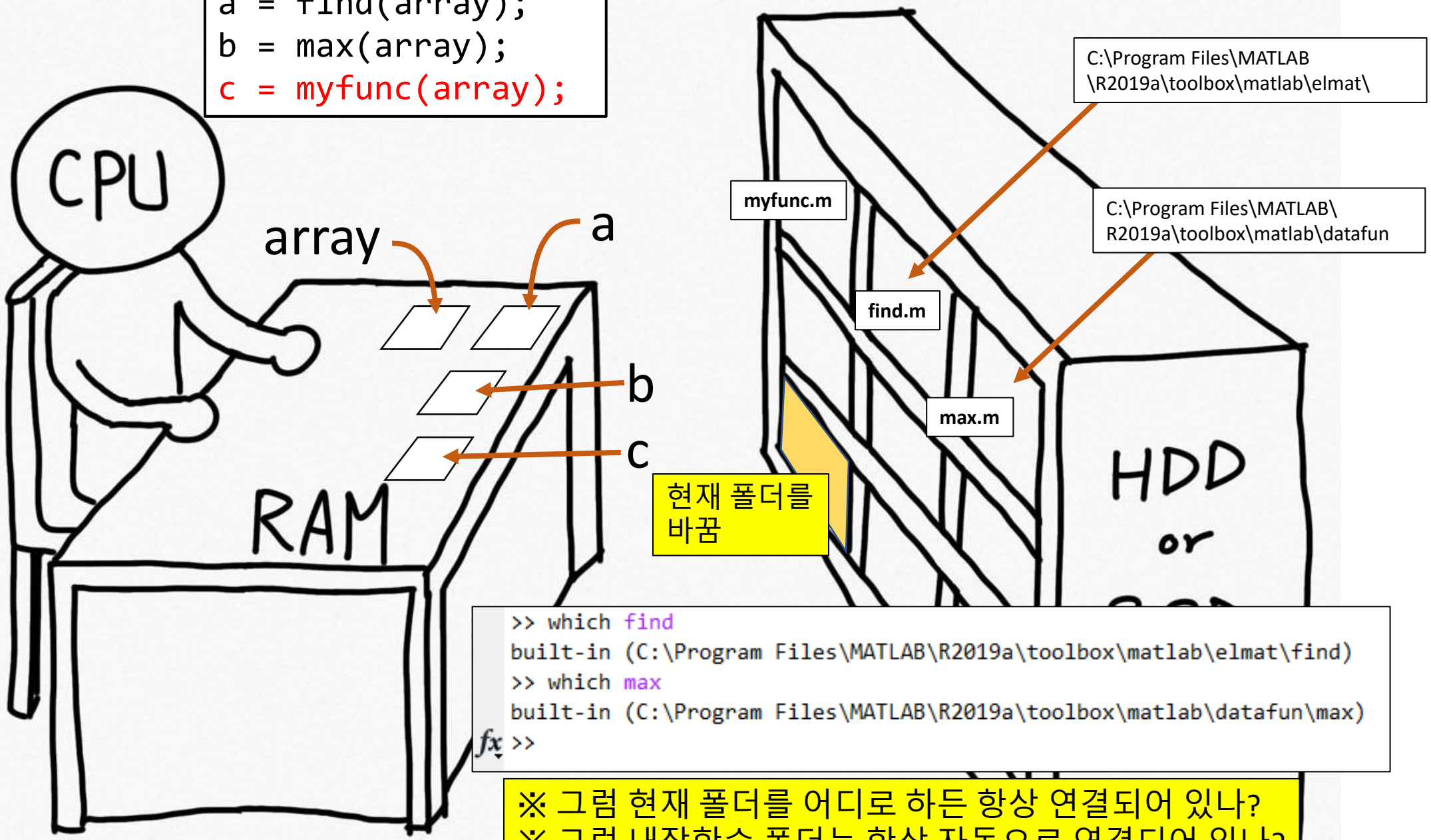
HDD/SSD (저장장치)
- 데이터의 장기간 저장을 위한 공간
- RAM에 비해 느리지만 넓음
- 저장장치의 특정 폴더 = 매트랩의 **"현재 폴더"**

```
a = find(array);
b = max(array);
c = myfunc(array);
```




```
>> which find
built-in (C:\Program Files\MATLAB\R2019a\toolbox\matlab\elmat\find)
>> which max
built-in (C:\Program Files\MATLAB\R2019a\toolbox\matlab\datafun\max)
>>
```

```
a = find(array);
b = max(array);
c = myfunc(array);
```



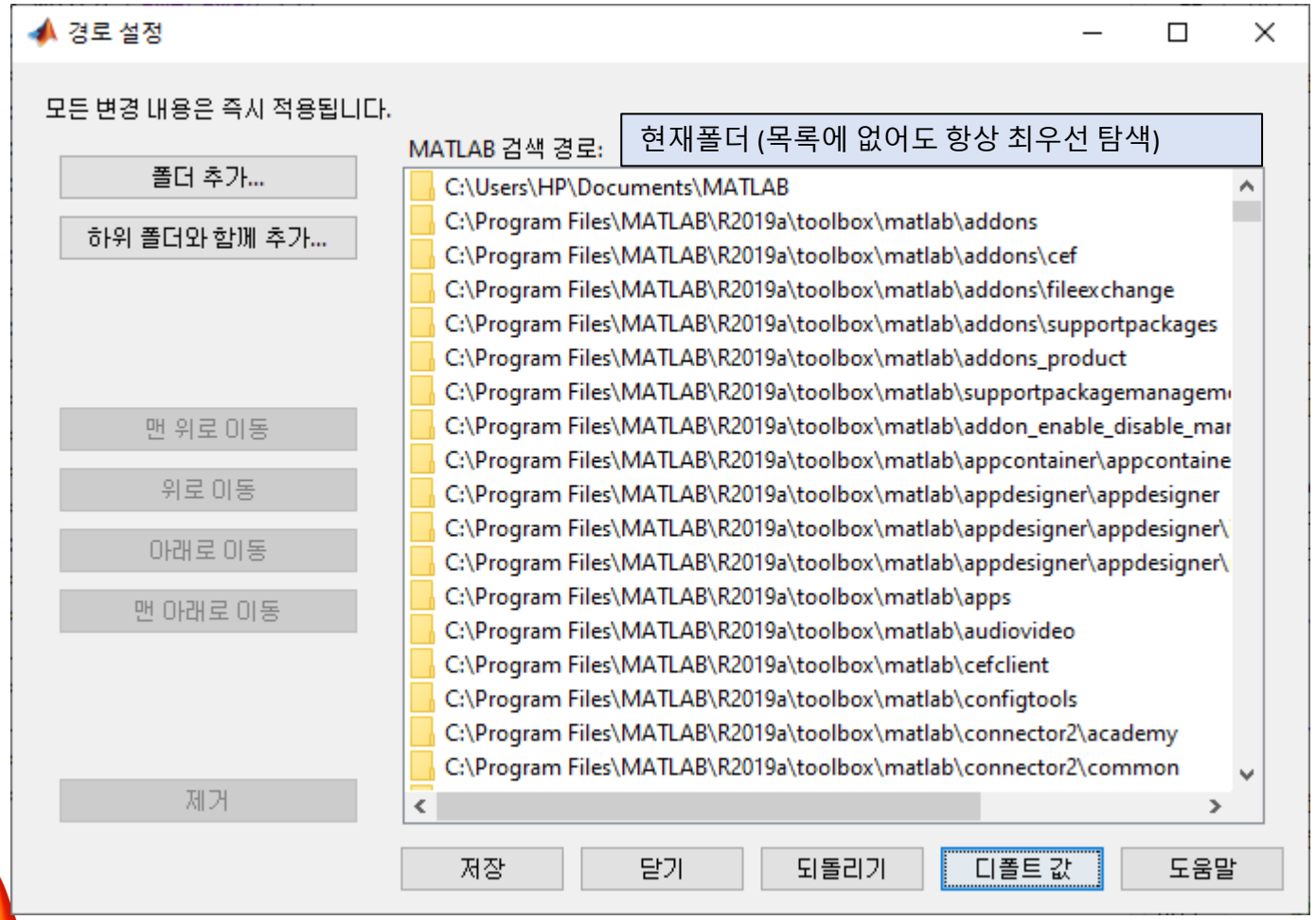
현재 폴더를 바꿈

```
>> which find
built-in (C:\Program Files\MATLAB\R2019a\toolbox\matlab\elmat\find)
>> which max
built-in (C:\Program Files\MATLAB\R2019a\toolbox\matlab\datafun\max)
>>
```

※ 그럼 현재 폴더를 어디로 하든 항상 연결되어 있나?
※ 그럼 내장함수 폴더는 항상 자동으로 연결되어 있나?
※ 그럼 접근 가능한 폴더의 목록은 도대체 어디에?

searchpath

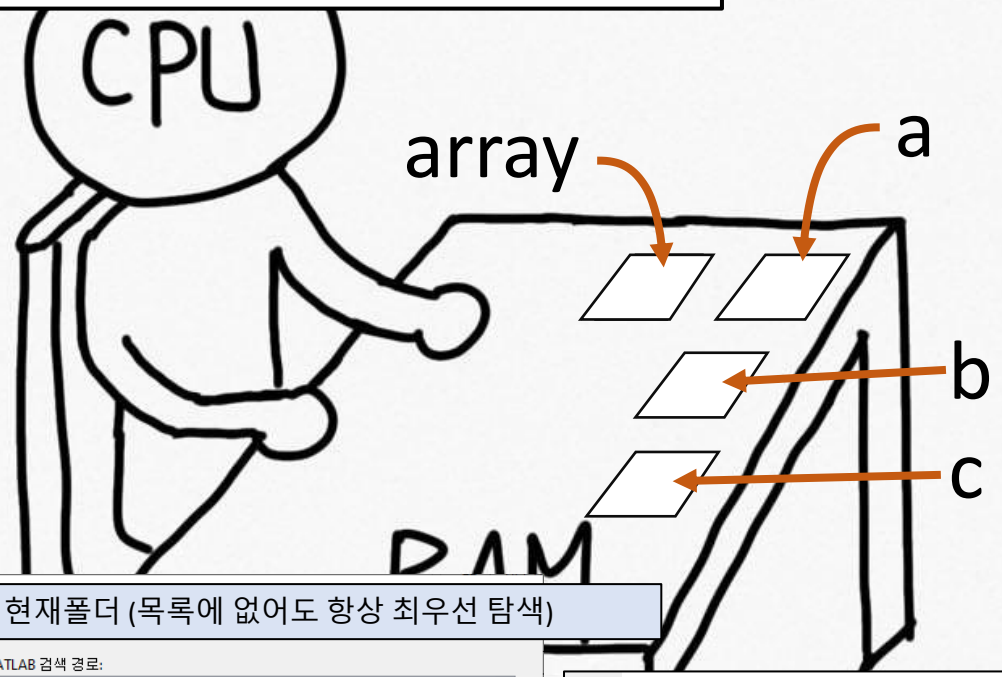
- 툴스트립의 "경로 설정" 클릭, 또는 명령창에 pathtool 입력

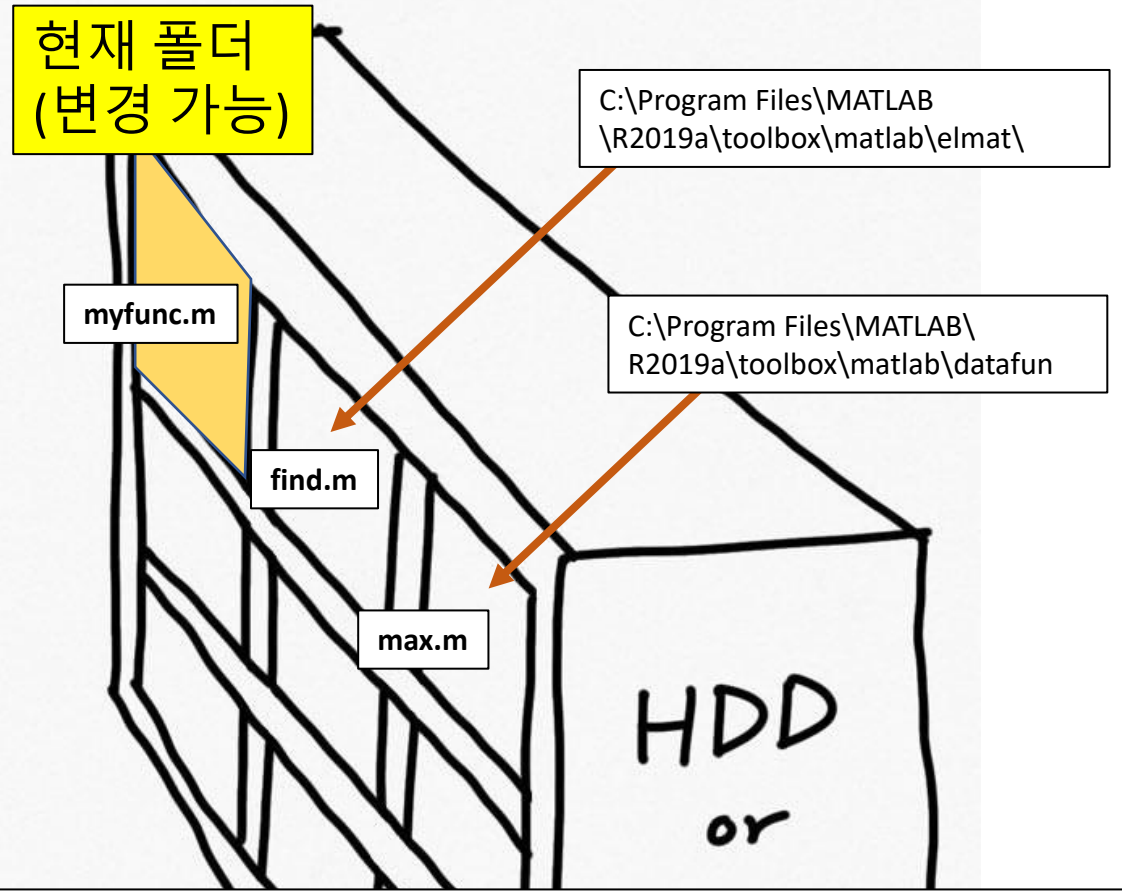


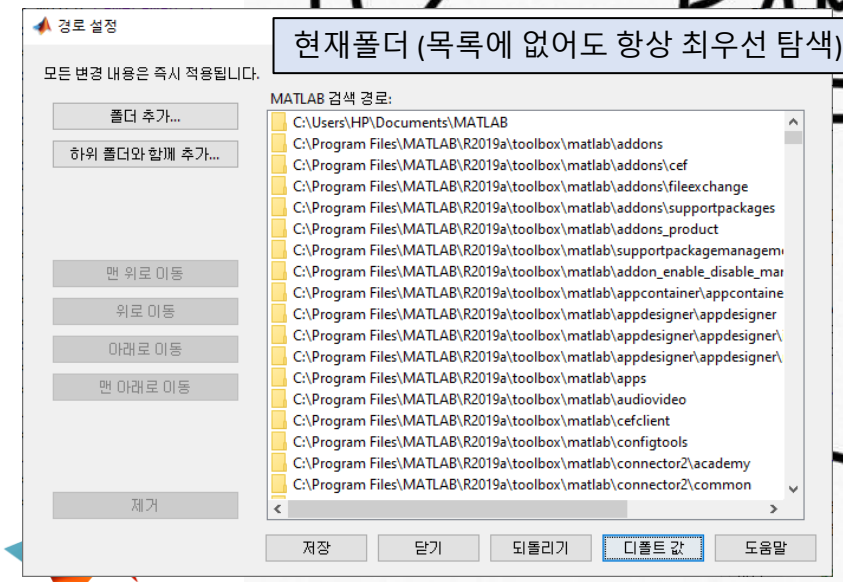
파일(m파일, 이미지 등)을 찾는 순서

1) 현재 폴더
2) MATLAB 검색 경로
   상단부터 아래로 검색

```
a = find(array);
b = max(array);
c = myfunc(array);
imshow(imread('cameraman.tif'));
```





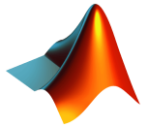


현재폴더 (목록에 없어도 항상 최우선 탐색)

```
>> which cameraman.tif
C:\Program Files\MATLAB\R2019a\toolbox\images\imdata\cameraman.tif
>>
>> which find
built-in (C:\Program Files\MATLAB\R2019a\toolbox\matlab\elmat\find)
>> which max
built-in (C:\Program Files\MATLAB\R2019a\toolbox\matlab\datafun\max)
fx >>
```

searchpath에 내가 만든 폴더를 추가해보자.

1) 찾아가기 편한 곳에 폴더를 하나 만든다.
2) 에디터에서 아무 파일이나 하나 만든다.
   - 스크립트, 함수 아무거나 OK
3) m파일을 위 1)에서 만든 폴더에 저장한다.
4) pathtool 창에서 위 1) 폴더를 추가한다. (최상단에 생김)
5) 저장-닫기를 순서대로 누른다.
   - 저장을 안하면 matlab 재시작 시 해당 폴더가 searchpath에 없음
6) 명령창이나 에디터에서 위 2)에서 만든 파일을 호출해본다.

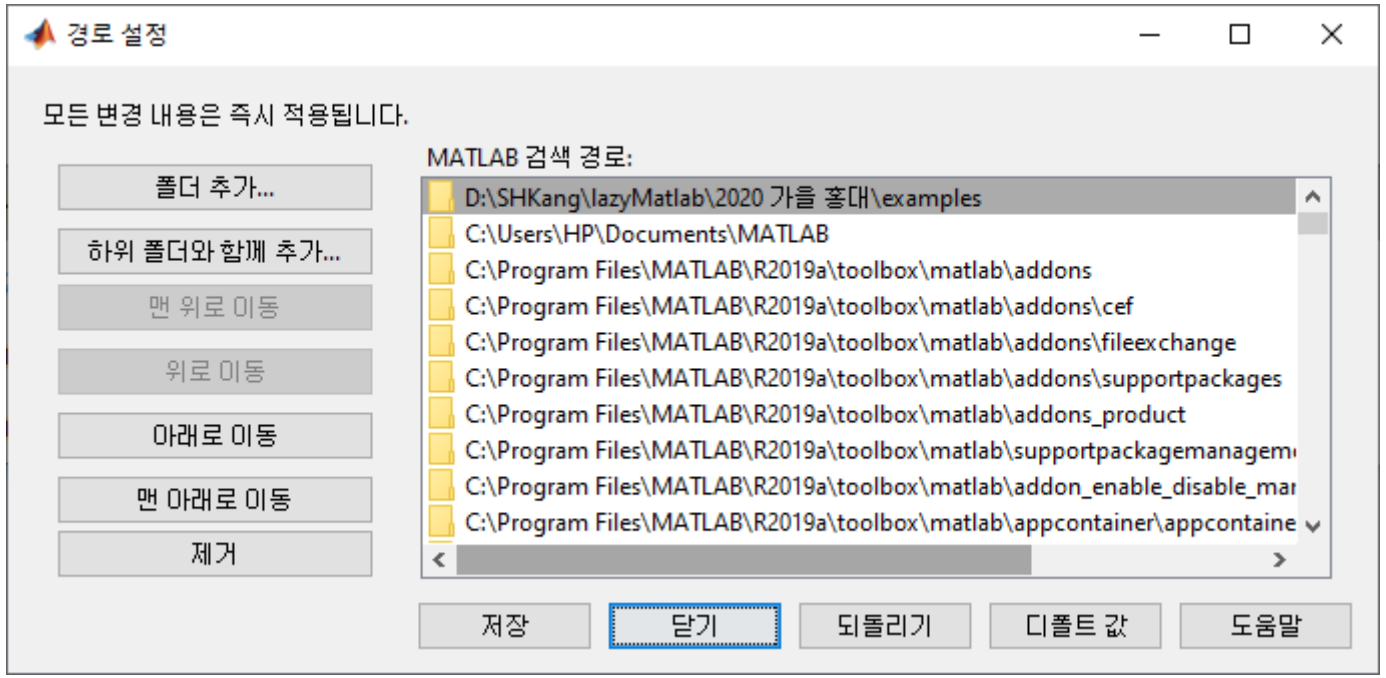


편집기 - D:\SHKang\lazyMatlab\2020 가을 홍대\examples\myadd.m

myadd.m

```
1   function zz = myadd(xx,yy)
2
3 -     zz = xx+yy;
4
5 -   end
```

```
>> cd d:\
>> disp(myadd(10,20))
    30
>>
```

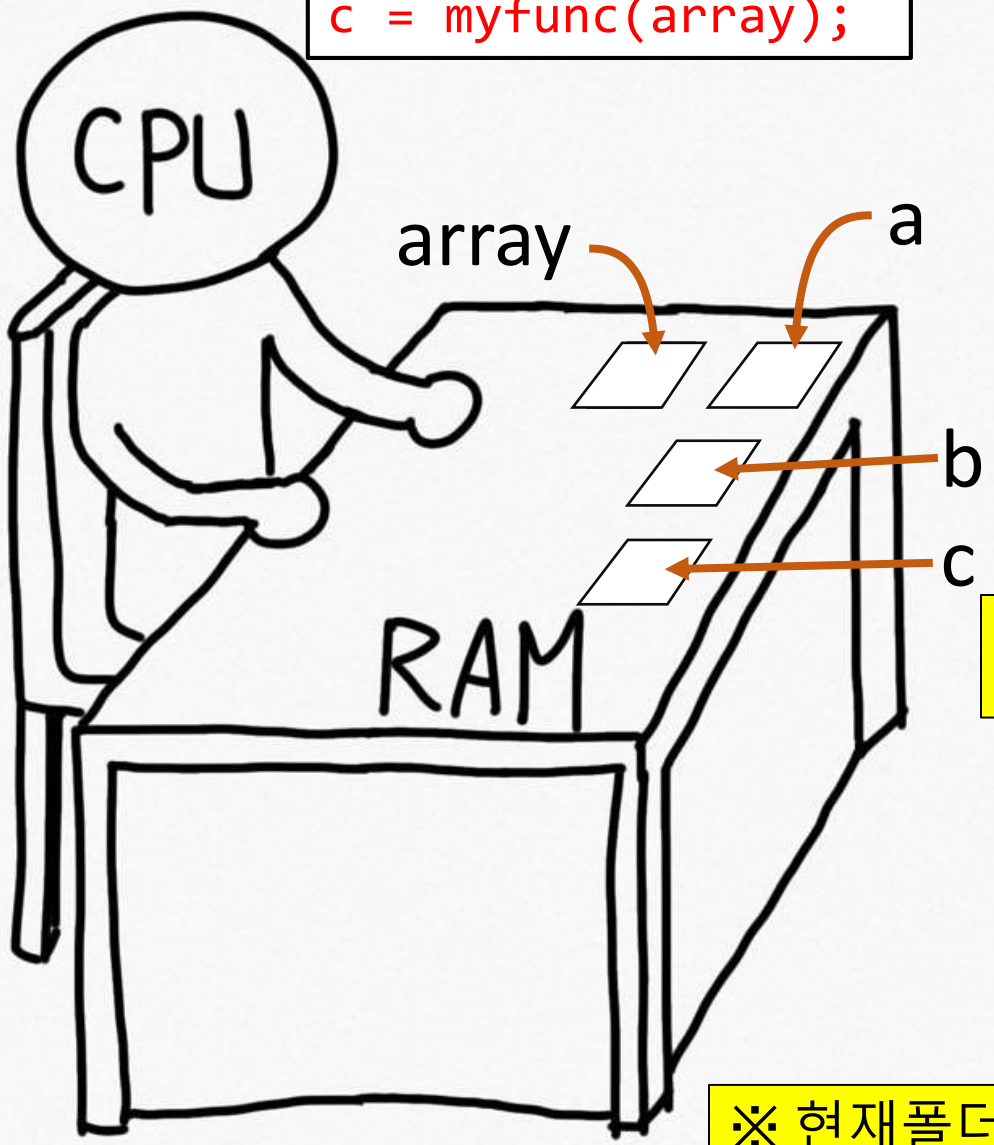


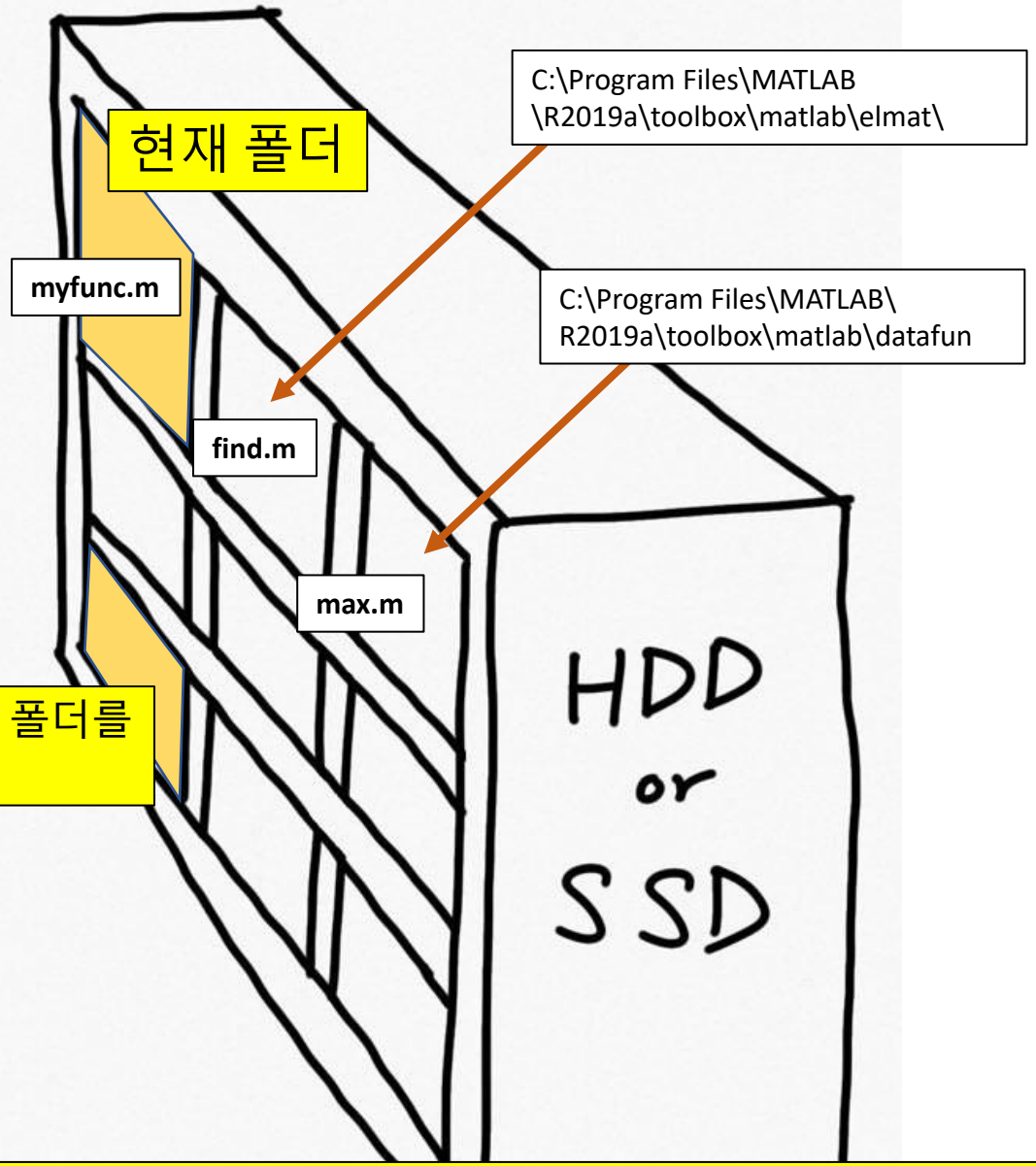
파일(m파일, 이미지 등)을 찾는 순서

1) 현재 폴더
2) MATLAB 검색 경로

상단부터 아래로 검색

```
a = find(array);
b = max(array);
c = myfunc(array);
```



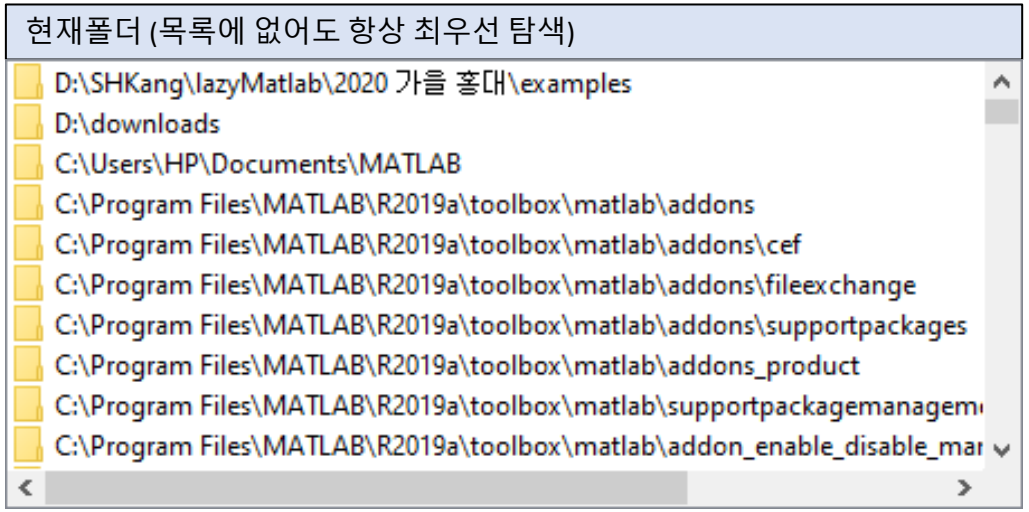


※ 현재폴더는 searchpath에 있든 없든 항상 최우선 검색을 한다.
※ 모든 것은 정의에서 시작한다. 변수든, 함수든, 파일이든.

같은 파일이 여러 폴더에 있으면 어떻게 될까?

- 폴더를 하나 더 만들어서 searchpath에 추가
- 같은 파일을 두 폴더에 모두 복사 (ex. myadd.m)
- myadd.m을 호출하면 어느 것을 호출해올까?

파일 탐색 순서

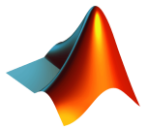


which('myadd','-all')도 가능
단, 함수명은 따옴표로 감싸주어야 함

```
>> which myadd
D:\SHKang\lazyMatlab\2020 가을 홍대\examples\myadd.m
>> which myadd -all
D:\SHKang\lazyMatlab\2020 가을 홍대\examples\myadd.m
D:\downloads\myadd.m                                % Shadowed
>>
```

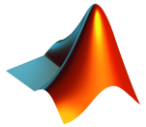
which에 -all 옵션
- searchpath 상에 있는 모든 파일 목록을 탐색
- 탐색결과 중 두 번째부터는 Shadowed

- 현재폴더를 바꿔가며 테스트해보자. (중요)



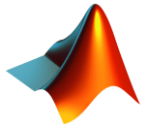
searchpath 관련 몇가지 함수들

- addpath(foldername)
  - 폴더를 searchpath 최상단에 **임시**로 추가
  - 바뀐 searchpath를 저장하지 않으므로
    matlab 재시작 시 searchpath는 원래대로 돌아감
- savepath
  - 현재의 searchpath 목록을 **반영구** 저장 (open pathdef로 확인 가능)
    - pathtool 창에서 default 값을 누르면 searchpath 완전 초기화
  - matlab 재시작 시 searchpath가 적용된 상태로 시작함
- cd = change directory (현재폴더 변경)
- mkdir = make directory (폴더 만들기)
- rmdir = remove directory (폴더 삭제)

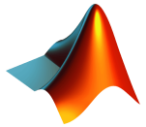


정리

- matlab에서 파일을 탐색하는 순서는 아래와 같다.
  - 현재폴더 (어디든 상관없이 항상 최우선 탐색)
  - searchpath 상단부터 순서대로 탐색
- addpath로 특정 폴더를 searchpath 상단에 임시 저장 가능
- savepath로 searchpath의 반영구 저장 가능
- 자주 쓰는 함수는 폴더 하나에 몰아서 넣어두면 사용하기 편리함
  - 함수를 폴더로 나눠야 할 경우, "하위 폴더와 함께 추가"
- 사용자 정의 함수와 일반 함수의 차이점?
  - searchpath default 설정 시 폴더 연결이 없어지냐 아니냐의 차이 뿐



Code helper

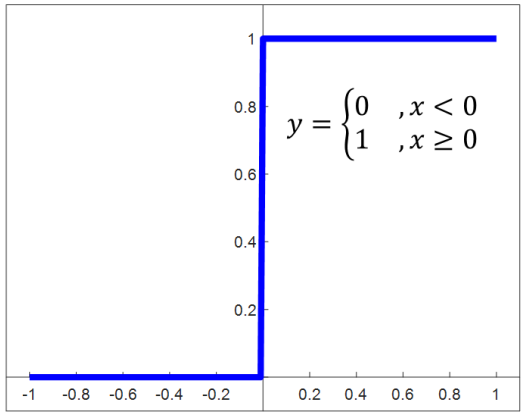


주황색 = 고칠 거리가 있는 곳

* 빨간색 = 문법 오류

```
276 -     N = 1e7;
277
278       % no pre-allocation
279 -     disp('no pre-allocation')
280 -     tic;
281 -     for i = 1:N
282 -         array(i) = i;
283 -     end
284 -     toc;
```

```
% wrong case
x = -1:0.01:1;
if x>=0
    y=1;
else
    y=0;
end
figure, plot(x, y)
```




```
778       %% score -> grade (ABCDF)
779 -     clc
780
781 -     while true
782 -         score = input('input your score: ');
783 -         if score>=90
784 -             disp('You got A.')
785 -         elseif score>=80
786 -             disp('You got B.')
787 -         elseif score>=70
788 -             disp('You got C.')
789 -         elseif score>=60
790 -             disp('You got D.')
791 -         else
792 -             disp('You got F.')
793 -         end
794 -     end
795
796       %% FizzBuzz
797
798 -     clear
799 -     clc
```

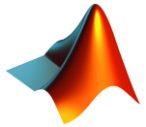
* break가 있으면?

주황색 = 고칠 거리가 있는 곳

```
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) = [];
```

```
function [m, s] = get_mean_std(x)

 m = mean(x);
 s = std(x);

 end
```

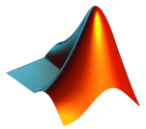
```
x = rand(10000,1);

[m, s] = get_mean_std(x)

get_mean_std(x)

s = get_mean_std(x)

[~,s] = get_mean_std(x)
```



주황색 = 고칠 거리가 있는 곳

```
while true
    name = input('type your name: ', 's');
    if length(name)==0
        disp('Name should have at least 1 character.')
    else
        fprintf('Hello, %s!\n', name)
        break
    end
end
```

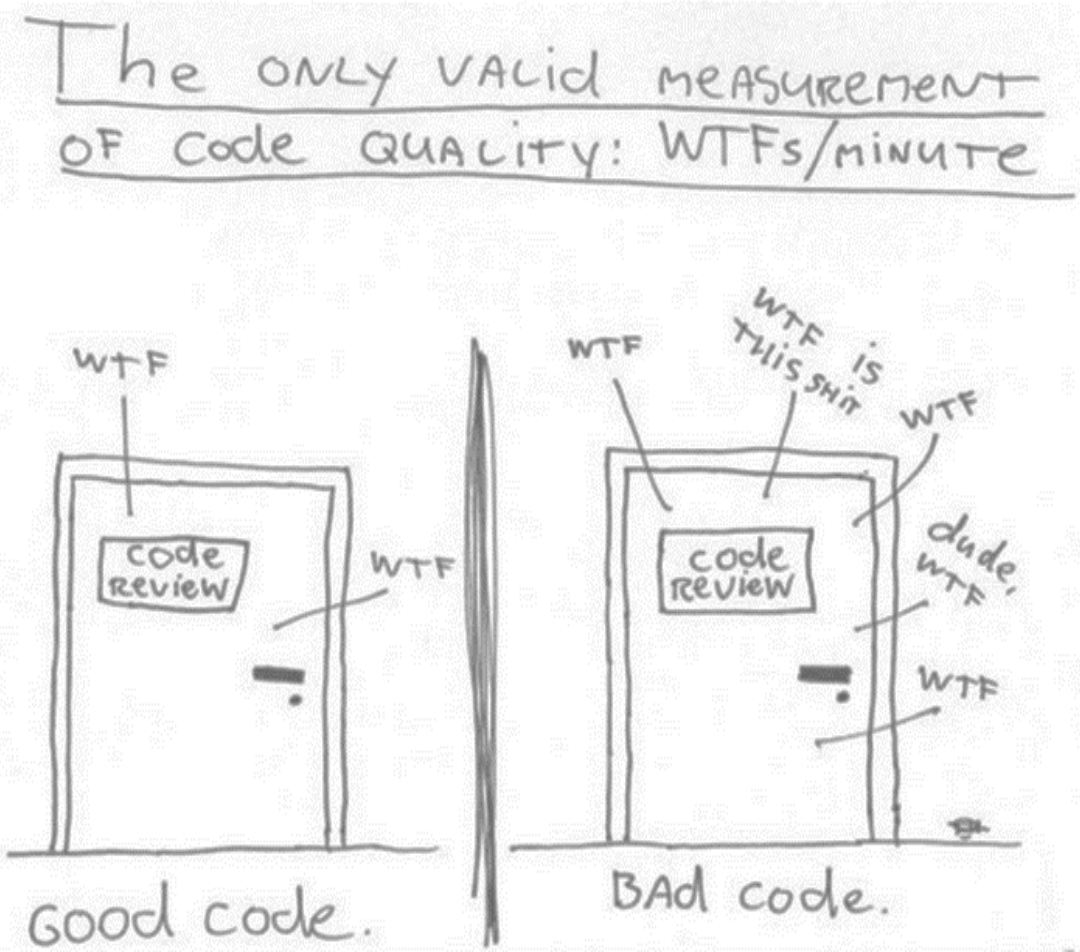
```
n1 = input('type a number: ');
n2 = input('type a number: ');

if n1>0 & n2>0
    disp('both positive')
else
    disp('at least one of inputs is nonpositive')
end
```



Good Programming Practice

좋은 코드 vs 나쁜 코드

코드 품질을 측정하는
유효한 단 한가지 방법:
1분 당 욕 나오는 회수



작명의 고통

- 명확한 이름을 써라.
  - 이름 자체로 설명이 가능하도록

```
a = 1e6;
b = rand(N,1);
c = rand(N,1);
d = zeros(N,1);

for i = 1:a
   d(i) = 1/12*pi*(b(i)^2)*c(i);
end
```

```
N = 1e6;
D = rand(N,1); % diameter?
H = rand(N,1); % height?
V = zeros(N,1); % volume?

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

  - 검색 가능한 이름을 써라.

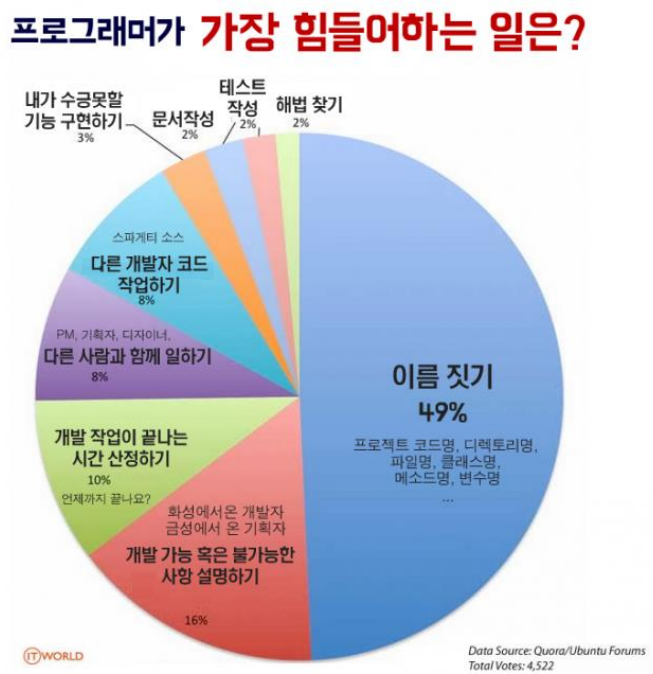
```
hour = 3600;
```

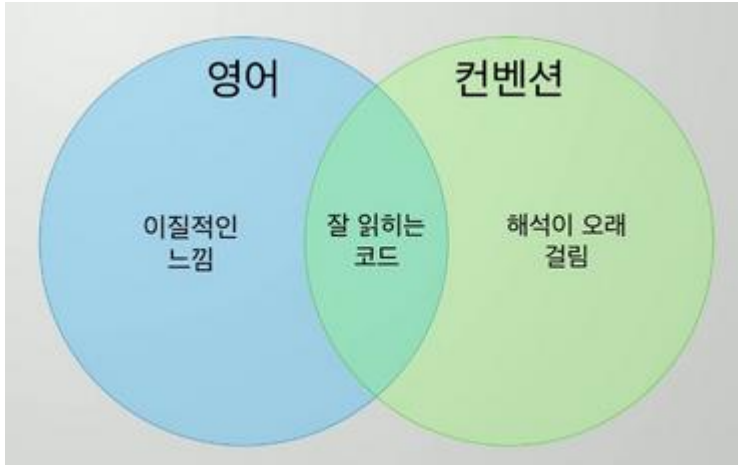
```
seconds_in_an_hour = 3600;
```

  - 너무 길지도, 너무 짧지도 않은 이름

```
get_top_plate_vertex_position(top_plate);
```

```
get_tp_vpos(tp);
```





https://dzone.com/articles/best-practices-variable-and
https://www.youtube.com/watch?v=rbSnkiqPnJI

작명의 고통

- 연관된 변수들은 일관된 이름을 갖도록

```
max_iteration_number = 100;
min_repetition_num = 10;
```

- 함수명은 이름만 봐도 동작을 알 수 있도록

```
function c = calc_numbers(a, b)
```

```
function multi = multiply_numbers(num1, num2)
```

```
function test(in)
```

```
function test_true(in)
```

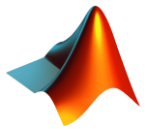
- 함수 이름에는 (가급적) 동사를 써라.

```
props_nickel = material_props('nickel');
```

```
props_nickel = get_material_props('nickel');
```

- 함수 이름은 유일해야 한다.

https://www.slideshare.net/devcatpublications/ndc2018
https://www.youtube.com/watch?v=Jz8Sx1XYb04&feature=youtu.be
https://lazymatlab.tistory.com/109

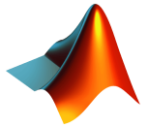


하드코딩

- 바뀔 가능성이 있는 변수는 코드 앞부분에 몰아두자.
  - 제대로 짜면 중간은 신경 끌 수 있다.

```
err = 100;
mypi = 0;
n = 1;

while err>2e-3 && n<=1000
    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>1000
    fprintf('iteration limit exceeded.\n');
else
    fprintf('mypi got close enough.\n')
end
```

```
tol = 2e-3;
err = 100;
iter_max = 1000;
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
```

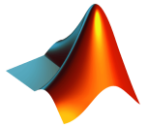


하드코딩

- 바뀔 가능성이 있는 변수는 코드 앞부분에 몰아두자.
  - 제대로 짜면 중간은 신경 끌 수 있다.

```
img = imread('cameraman.tif');

blurred = zeros(254,254);
for i=1:254
    for j=1:254
        blurred(i,j) = floor(mean2(img(i:i+2,j:j+2)));
    end
end
```

```
img = imread('cameraman.tif');
box_size = [3 3];

blurred = zeros(size(img,1)-2,size(img,2)-2);
for i=1:size(blurred,1)
    for j=1:size(blurred,2)
        blurred(i,j) = floor(mean2(img(i:i+box_size(1)-1,j:j+box_size(2)-1)));
    end
end
```

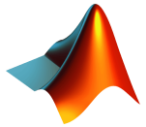


하드코딩

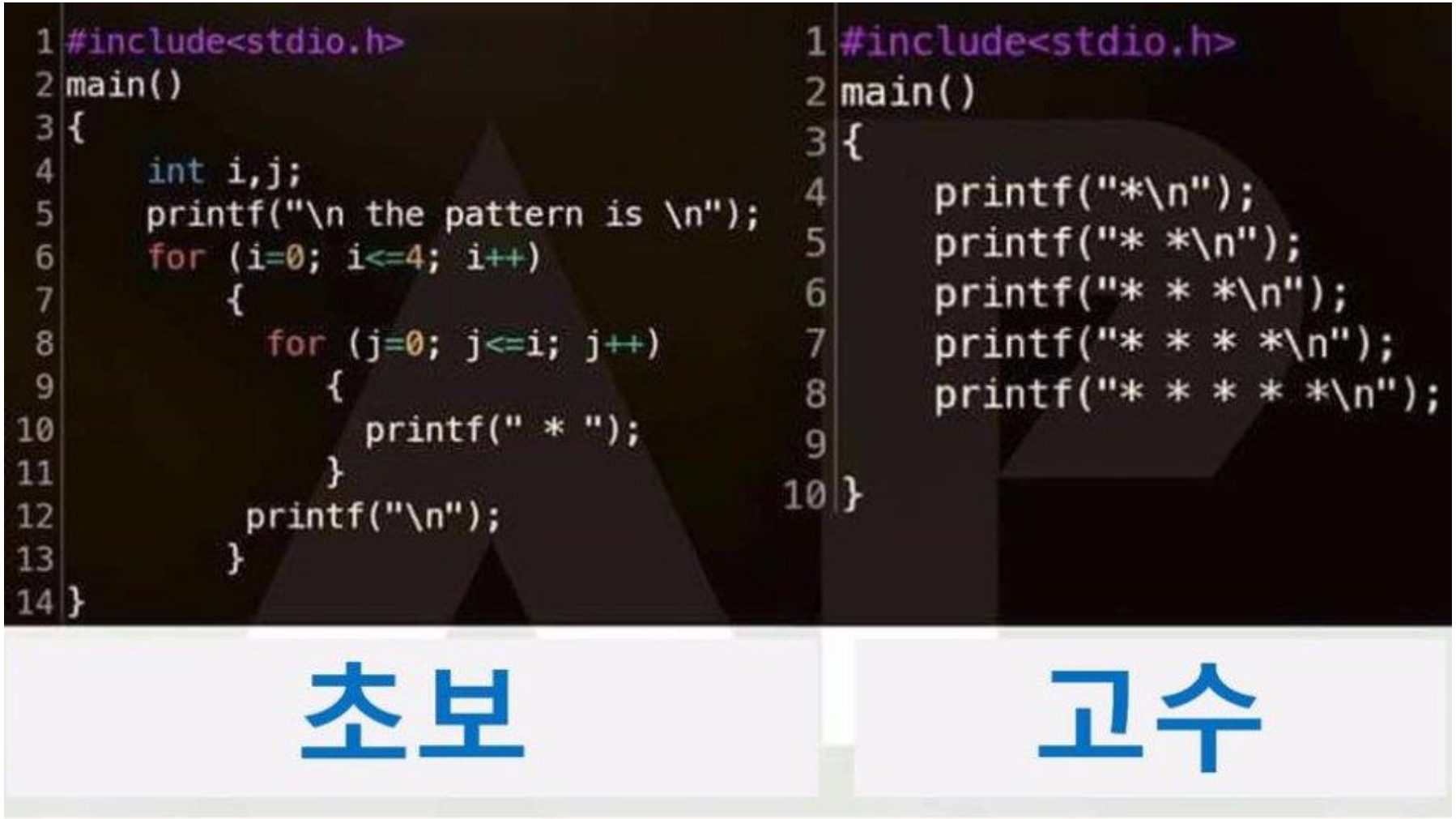
- 함수는 그 자체로 완벽한 모듈이어야 한다.
  - 어떤 입력이 들어와도 동작해야 한다.
  - 하드코딩은 (웬만하면) 절대 하지 말자.

```
function blurred = boxBlur(img)

blurred = zeros(254,254);

for i=1:254
    for j=1:254
        ibox = img(i:i+2,j:j+2);
        blurred(i,j) = floor(mean2(ibox));
    end
end

end
```

```
function blurred = boxBlur(img, box_size)

blurred = zeros(size(img,1)-box_size(1),size(img,2)-box_size(2));

for i=1:size(blurred,1)
    for j=1:size(blurred,2)
        ibox = img(i:i+box_size(1)-1,j:j+box_size(2)-1);
        blurred(i,j) = floor(mean2(ibox));
    end
end

end
```



그럼 하드코딩은 항상 나쁜가...?



```c
#include<stdio.h>
main()
{
    int i,j;
    printf("\n the pattern is \n");
    for (i=0; i<=4; i++)
        {
          for (j=0; j<=i; j++)
              {
                printf(" * ");
              }
          printf("\n");
        }
}
```

초보

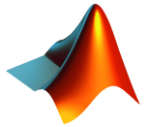
```c
#include<stdio.h>
main()
{
    printf("*\n");
    printf("* *\n");
    printf("* * *\n");
    printf("* * * *\n");
    printf("* * * * *\n");

}
```

고수

주석은 얼마나 열심히 달아야 할까?

```
% get Planck constant
h_kg_m2_per_s = get_planck_h();

h = get_planck_h(); % kg m2 / s

% get blurred_image
blurred = boxBlur(img);

% take username by input
name = input('Type your name: ');

x = 0:0.1:1;
y = x.^2;
orange = [255, 128, 0]/255;
plot(x, y, ':^', 'color', orange)
plot(x, y, ':^', 'color', '#900606') % dark red
```

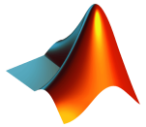
- 나쁜 코드
  - 주석이 없어서 이해하기 어려운 코드
  - 주석이 너무 많아서 가독성이 떨어지는 코드
- 좋은 코드
  - 딱 필요한 부분에만 주석이 있는 코드
  - 주석이 없어도 잘 읽히는 코드



https://brunch.co.kr/@lkj28/92

주석은 얼마나 열심히 달아야 할까?

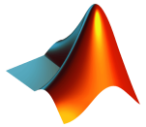
```
%% input scores until 'q' entered

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') % if 'q' is typed
        break
    elseif isnan(str2double(num)) % if num contains character
        disp('input must be numeric.')
        continue
    end
    scores = [scores; str2double(num)];
    n = n + 1;
end
```



짧은 코드 = 좋은 코드?

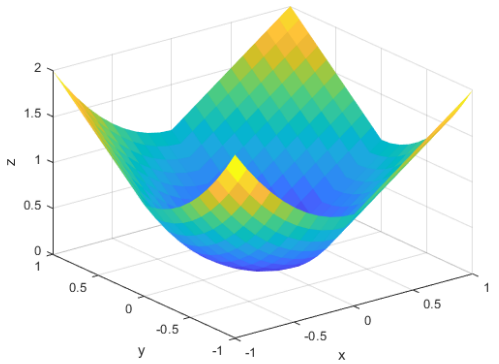
```
% boxblur2 - ignore zero-valued pixels
blurred = zeros(size(img)-[2,2]);
for i=1:size(img,1)-2
    for j=1:size(img,1)-2
        ibox = sum(img(i:i+2,j:j+2);
        blurred(i,j) = floor(sum(ibox,'all')/sum(ibox>0,'all'));
    end
end
```

```
% boxblur2 - ignore zero-valued pixels
blurred = zeros(size(img)-[2,2]);
for i=1:size(img,1)-2
    for j=1:size(img,1)-2
        blurred(i,j) = floor(sum(img(i:i+2,j:j+2),'all')/ ...
                             sum(img(i:i+2,j:j+2)>0,'all'));
    end
end
```



짧은 코드 = 좋은 코드?

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




```
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 % 1st quadrant
            zz(i,j) = ix + iy;
        elseif ix<0 && iy>=0 % 2nd quadrant
            zz(i,j) = ix^2 + iy;
        elseif ix<0 && iy<0  % 3rd quadrant
            zz(i,j) = ix^2 + iy^2;
        else % 4th quadrant
            zz(i,j) = ix + iy^2;
        end
    end
end
```

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

[xx,yy] = meshgrid(x,y);

quad1 = (xx>=0) & (yy>=0);
quad2 = (xx <0) & (yy>=0);
quad3 = (xx <0) & (yy <0);
quad4 = (xx>=0) & (yy <0);

zz = quad1.*( xx    + yy    ) + ...
     quad2.*( xx.^2 + yy    ) + ...
     quad3.*( xx.^2 + yy.^2 ) + ...
     quad4.*( xx    + yy.^2 );
```

짧은 코드 = 좋은 코드?

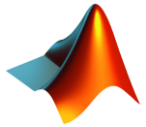
가독성 = 성능 >>>>> 코드 길이

Quaternion multiplication

×	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

출처: https://en.wikipedia.org/wiki/Quaternion

$$
\begin{aligned}
\mathbf{q}_1 &= w_1 + x_1\mathbf{i} + y_1\mathbf{j} + z_1\mathbf{k} \\
\mathbf{q}_2 &= w_2 + x_2\mathbf{i} + y_2\mathbf{j} + z_2\mathbf{k} \\
\mathbf{q}_1 * \mathbf{q}_2 &= (w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2) \\
&+ (w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2)\mathbf{i} \\
&+ (w_1y_2 - x_1z_2 + y_1w_2 + z_1x_2)\mathbf{j} \\
&+ (w_1z_2 + x_1y_2 - y_1x_2 + z_1w_2)\mathbf{k}
\end{aligned}
$$

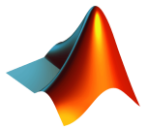


https://personal.utdallas.edu/~sxb027100/dock/quaternion.html

그 외에도...

- 중요한 것들
  - 가독성 (readability)
  - 확장성 (scalability)
  - 이식성 (portability)
  - 리팩토링 (refactoring)
- 버려야 할 습관
  - 개발도구 (IDE) 무시하기
  - 매번 바퀴를 다시 발명하기
  - 중복된 부분이 많은 코딩하기
  - 동작원리를 모르고 복붙하기
  - 너무 이른 최적화

https://brunch.co.kr/@lkj28/92
https://mingrammer.com/translation-13-simple-rules-for-good-coding/
https://lazymatlab.tistory.com/33
http://www.ciokorea.com/news/27205



Q&A