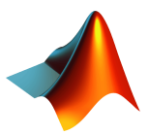
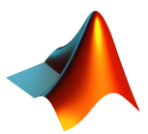


MATLAB 프로그래밍 및 실습

6강. 함수 기초



함수가 필요한 이유



표준편차를 구하는 코드를 짜보았다.

- 분산 (variance): 편차의 제곱의 평균
- 표준편차 (standard deviation): $\sqrt{\text{분산}}$

$$s = \sqrt{\frac{\sum (x_i - m)^2}{N}}$$

% mycode.m

...
...
...
...

```
m = mean(x);  
N = length(x);  
s = sqrt(sum((x-m).^2)/N);
```

...
...

```
m = mean(x);  
N = length(x);  
s = sqrt(sum((x-m).^2)/N);
```

...
...

```
m = mean(x);  
N = length(x);  
s = sqrt(sum((x-m).^2)/N);
```

```
m = mean(x);  
N = length(x);  
s = sqrt(sum((x-m).^2)/N);
```

그런데 내가 짠 코드가 글썽... [충격...경악...]

- 하나하나 찾아서 다 고쳐야 한다! $\pi\pi\pi$
- 그런데 파일이 한두개가 아니라면...?!

$$s = \sqrt{\frac{\sum (x_i - m)^2}{N - 1}}$$

% mycode.m

...
...
...
...

```
m = mean(x);  
N = length(x);  
s = sqrt(sum((x-m).^2)/N);
```

...
...

```
m = mean(x);  
N = length(x);  
s = sqrt(sum((x-m).^2)/N);
```

...
...

```
m = mean(x);  
N = length(x);  
s = sqrt(sum((x-m).^2)/N);
```

```
m = mean(x);  
N = length(x);  
s = sqrt(sum((x-m).^2)/N);
```



그것만 문제가 아니다.

- 10개의 다른 입력에 대해 10번 계산
- 10개의 다른 변수명이 필요하다.

$$s = \sqrt{\frac{\sum (x_i - m)^2}{N - 1}}$$

% mycode.m

...
...
x1 = ...
...

```
m1 = mean(x1);  
N1 = length(x1);  
s1 = sqrt(sum((x1-m1).^2)/N1);
```

...
x2 = ...

```
m2 = mean(x2);  
N2 = length(x2);  
s2 = sqrt(sum((x2-m2).^2)/N2);
```

...
x3 = ...

```
m3 = mean(x3);  
N3 = length(x3);  
s3 = sqrt(sum((x3-m3).^2)/N3);
```

```
m = mean(x);  
N = length(x);  
s = sqrt(sum((x-m).^2)/N);
```



그럼 이걸 어떻게 해결하면 좋을까?

- 같은 코드를 여기저기 복붙해서 사용
 - 문제점1: 코드 수정이 필요할 때 사용된 곳을 모두 찾아서 고쳐야 한다.
 - 문제점2: 코드의 범위가 불분명할 경우 배포가 어렵다.
 - 문제점3: 변수들이 뒤죽박죽이 된다.
- 해결책: 별도의 파일을 만들어서 필요할 때만 불러서 쓰면 어떨까?

% mycode.m

x1 = ...
s1 = calcSTD(x1);

x2 = ...
s2 = calcSTD(x2);

x3 = ...
s3 = calcSTD(x3);

calcSTD.m

```
function s = calcSTD(x)
m = mean(x);
N = length(x);
s = sqrt(sum((x-m).^2)/(N-1));
```

별도의 파일로 작성 = "모듈화"

- 모듈화 → 코드의 관리, 확장, 배포, 재사용이 쉬워진다.
- 예) 표준편차 계산 코드를 별도의 파일로 작성 (calcSTD.m)
 - 필요할 때 해당 파일을 불러서 쓰면 된다.
 - 수정(디버깅, 기능 추가)이 필요하면 해당 파일만 수정하면 된다.
 - 배포가 필요하면 해당 파일만 배포하면 된다.
 - 코드 내부 동작을 알 필요가 없다. 정확히 정의된 입출력만 있으면 된다.
 - 가독성이 향상된다. 인터페이스만 있으면 된다.

% mycode.m

x1 = ...
s1 = calcSTD(x1);

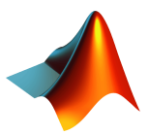
x2 = ...
s2 = calcSTD(x2);

x3 = ...
s3 = calcSTD(x3);

calcSTD.m

```
function s = calcSTD(x)
m = mean(x);
N = length(x);
s = sqrt(sum((x-m).^2)/(N-1));
```

프로그램 코드
= 모듈의 시퀀스



지금까지 했던 것들을 다시 살펴보자.

`% 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))
```

fibonacci(N)처럼 쓸 수 있는
tribonacci(N)을 만들고 싶다면?

`% 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
```

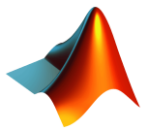
get_grade(85)만 쓰면
'B'가 출력되게 하고 싶다면?
Grade를 나누는 점수 기준이 바뀐다면?

`% boxBlur`

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

boxBlur(img)만 쓰면
알아서 blurred가 나왔으면 좋겠다면?

➔ 모듈화가 해답이다!



간단한 함수를 만들어보자.

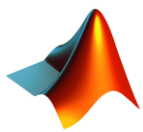
```
myadd.m x
1 function z = myadd(x,y)
2
3     z = x+y;
4
5     end
```

```
+33
getConeVol.m* x
1 function V = getConeVol(diameter, height)
2
3     V = 1/3*pi*(diameter/2).^2.*height;
4
5     end
```

- 새 파일 만들기: ctrl+N
- 왼쪽과 같이 작성 후 m파일 저장
 - 저장은 "현재 폴더"에
- 명령창에서 함수를 호출해보자.

```
>> myadd(10,100)
ans =
    110
>> getConeVol(4,5)
ans =
    20.9440
```

- 호출하는 방법은 내장함수와 다를 게 없다.
 - factorial(10), sin(pi), size(x,1), ...
 - 일단 만들어주면 내장함수처럼 쓸 수 있다.



함수는 이렇게 생겼다.

```
myadd.m
1 function z = myadd(x,y)
2
3     z = x+y;
4
5 end
```

```
+33
getConeVol.m*
1 function V = getConeVol(diameter, height)
2
3     V = 1/3*pi*(diameter/2).^2.*height;
4
5 end
```

```
funcname.m
function output = funcname(input1, input2, ...)
...
...
output = ...

end
```

1) 첫 줄은 아래처럼 생겼다.

`function` 출력 = 함수이름(입력1, 입력2, ...)

2) `end`로 끝난다.

필수는 아니지만 웬만하면 붙이자.

3) 함수명과 파일명이 같다.

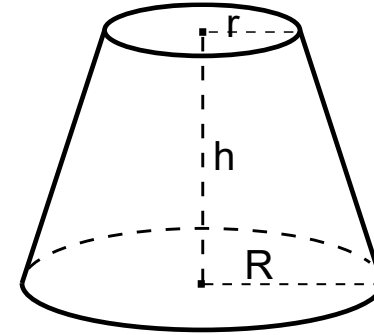
4) 출력인자(반환값)가 함수 내부에서 정의된다.

※ 첫 줄이 `function`으로 시작하지 않는 파일 = "스크립트(script)"

tip. 인자(argument)? 매개변수(parameter)? 호출? 반환?

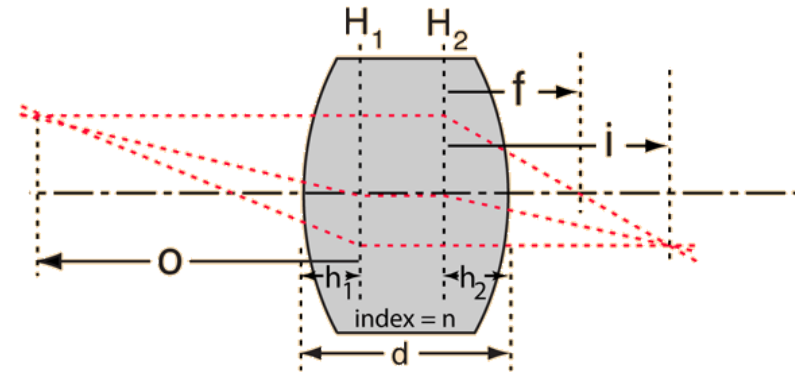
간단한 함수 예제

```
getVolTruncCone.m
1 function V = getVolTruncCone(H,r1,r2)
2
3 V = pi/3*H.*(r1.^2 + r1.*r2 + r2.^2);
4
5 end
```



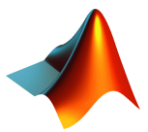
$$V = \frac{\pi H}{3} (R_1^2 + R_1 R_2 + R_2^2)$$

```
getFofThickLens.m
1 function f = getFofThickLens(n,R1,R2,d)
2
3 f = 1./((n-1)*(1./R1 - 1./R2 ...
4         + (n-1)*d./(n*R1.*R2)));
5
6 end
```



<http://hyperphysics.phy-astr.gsu.edu/hbase/geoopt/priplan.html>

$$\frac{1}{f} = (n - 1) \left[\frac{1}{R_1} - \frac{1}{R_2} + \frac{(n - 1)d}{nR_1R_2} \right]$$



간단한 함수 예제

```

+27 get_quadeq_disc.m
1 function disc = get_quadeq_disc(a,b,c)
2
3     disc = b^2-4*a*c;
4
5 end

```

$$ax^2 + bx + c = 0$$

$$D = b^2 - 4ac$$

```

+28 tribonacci.m
1 function tt = tribonacci(N)
2
3     t = zeros(N,1);
4     t(1) = 1;
5     t(2) = 1;
6     t(3) = 2;
7     for i=4:N
8         t(i) = t(i-3) + t(i-2) + t(i-1);
9     end
10
11     tt = t(end);

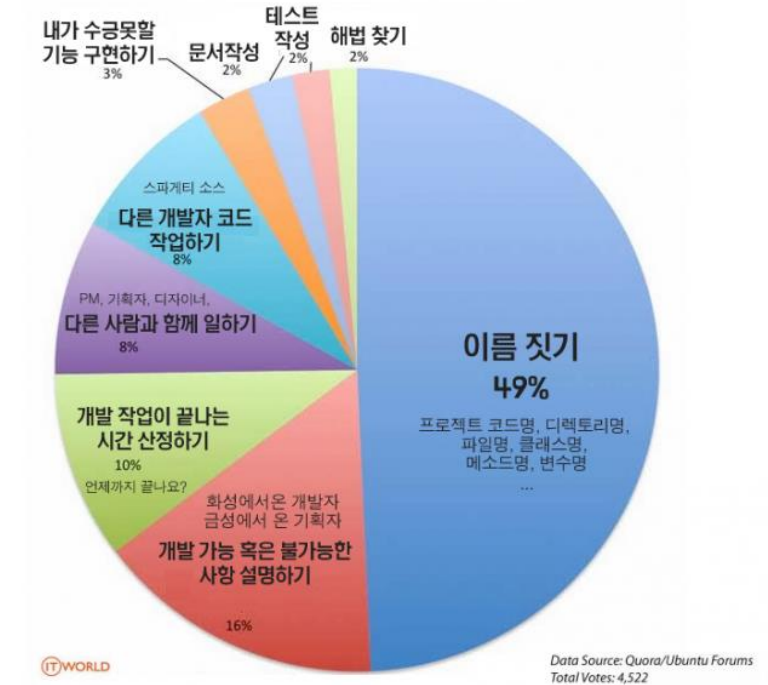
```

```

+29 boxBlur.m
1 function blurred = boxBlur(img)
2
3     blurred = zeros(size(img,1)-2,size(img,2)-2);
4     for i=1:size(blurred,1)
5         for j=1:size(blurred,2)
6             blurred(i,j) = floor(mean2(img(i:i+2,j:j+2)));
7         end
8     end

```

프로그래머가 가장 힘들어하는 일은?



간단한(?) 함수 예제

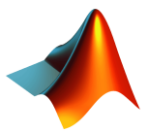
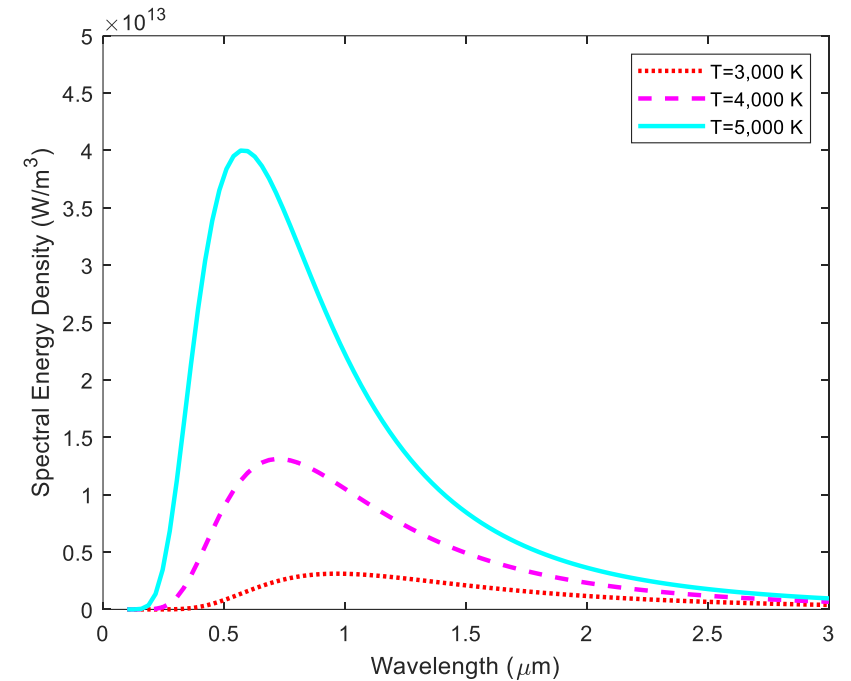
```
+24 getSED.m x
1 function R = getSED(wavelength, T)
2 %getSED Return spectral energy density of black body radiation
3 % wavelength = wavelength of radiation, in meter
4 % T = temperature of black body, in Kelvin
5
6 h = 6.626e-34; % Planck Constant, J.s
7 c = 3e8; % speed of light, m/s
8 k = 1.38e-23; % Boltzmann constant, J/K
9
10 R = 2*pi*c^2*h./(wavelength.^5)./(exp(h*c./wavelength/k/T)-1);
```

$$R = \frac{2\pi c^2 h}{\lambda^5} \frac{1}{e^{hc/\lambda kT} - 1}$$

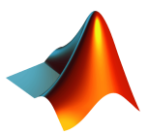
```
wl = linspace(0.1,3)/1e6; % wavelength, meter
```

```
colors = 'rmc';
styles = [":", "--", "-"];
temps = [3000, 4000, 5000];
```

```
figure, hold on, box on
for i = 1:length(temps)
    plot(wl*1e6, getSED(wl, temps(i)), colors(i)+styles(i))
end
```



함수의 동작 이해



함수의 동작을 이해해보자.

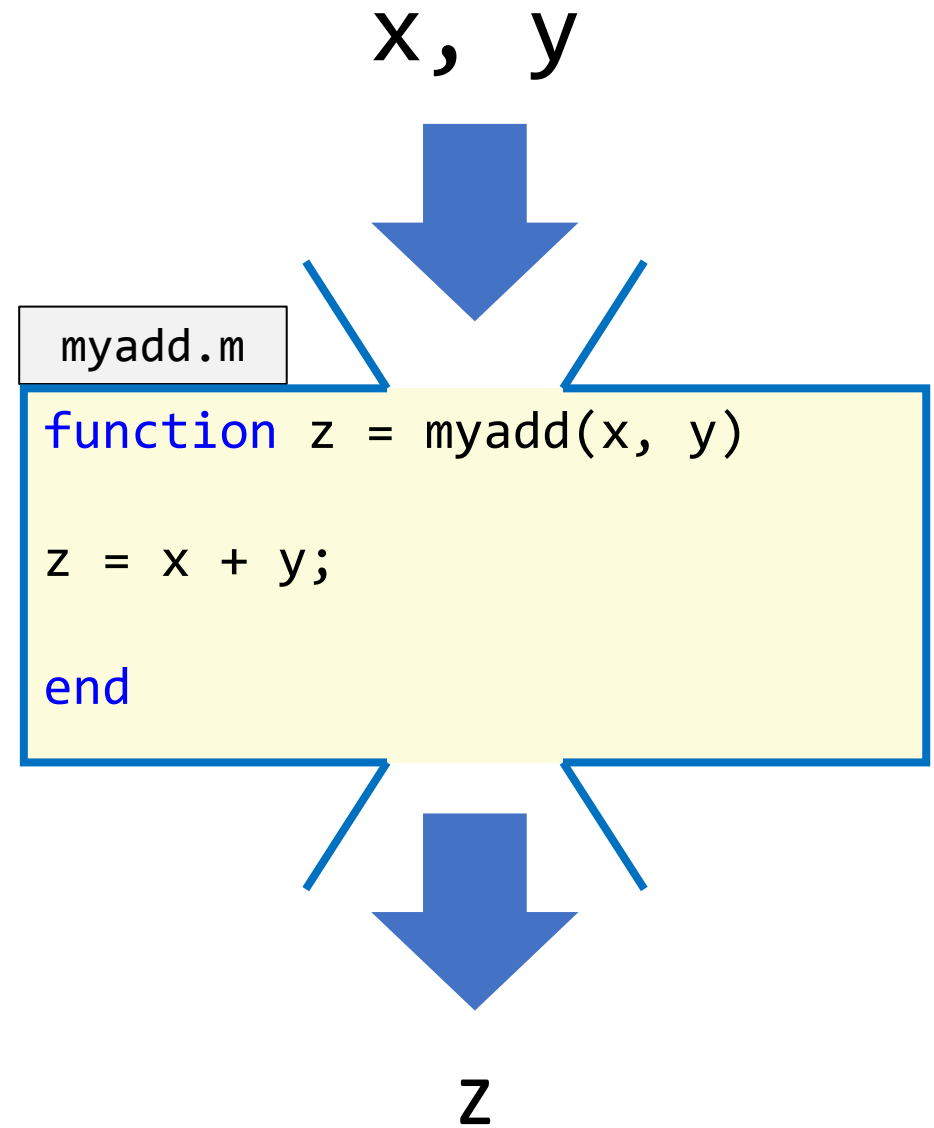
```
a = myadd(10, 100);  
x = myadd(3, 4);  
myadd(10, 20);
```

myadd(10) → ?

```
disp(myadd(11, 22));  
disp(myadd(exp(1), -exp(1)));
```

```
z = myadd(10, 40) - myadd(20, 30);
```

```
xx = pi;  
yy = 2*pi;  
zz = myadd(xx, yy);
```

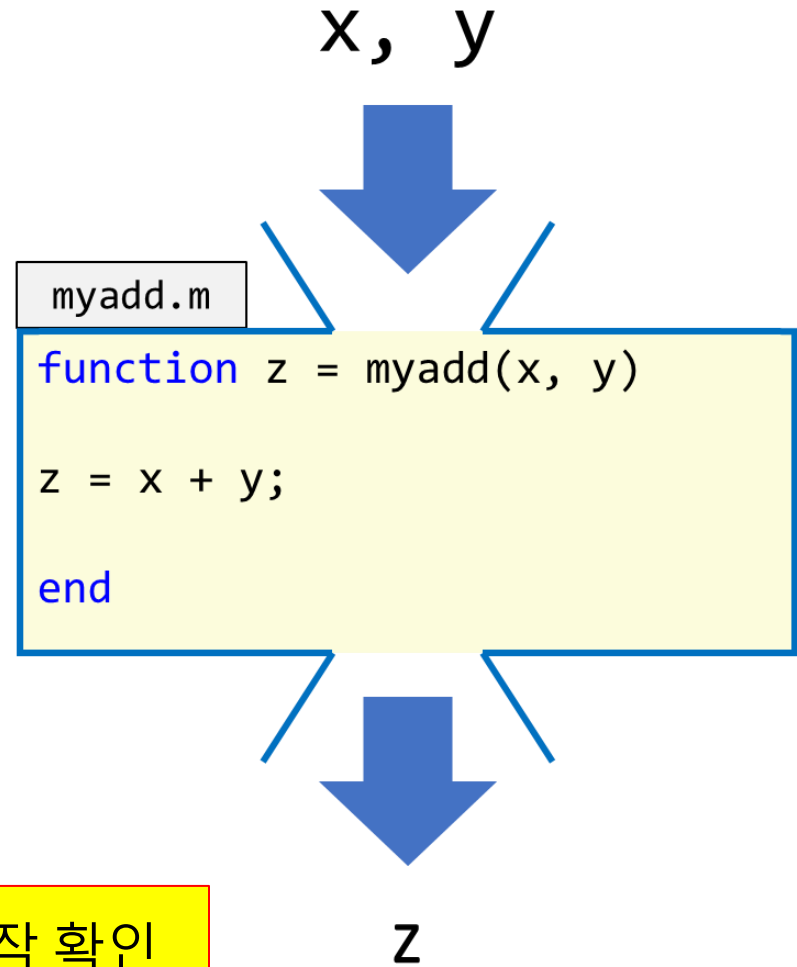


※ 호출한 곳의 변수(workspace)와 함수 내부 변수(workspace)는 완전히 별개이다.

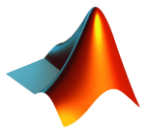
함수는 workspace를 공유하지 않는다.

- 공유하면 무슨 일이 생길까?

```
x = 10;  
y = 20;  
myadd(); % z stored in workspace  
  
z = 100; % z overwritten!  
  
...  
...  
...  
  
myadd(); % z overwritten again!  
...      % which x and y are being used?  
...  
  
myadd();  
...
```

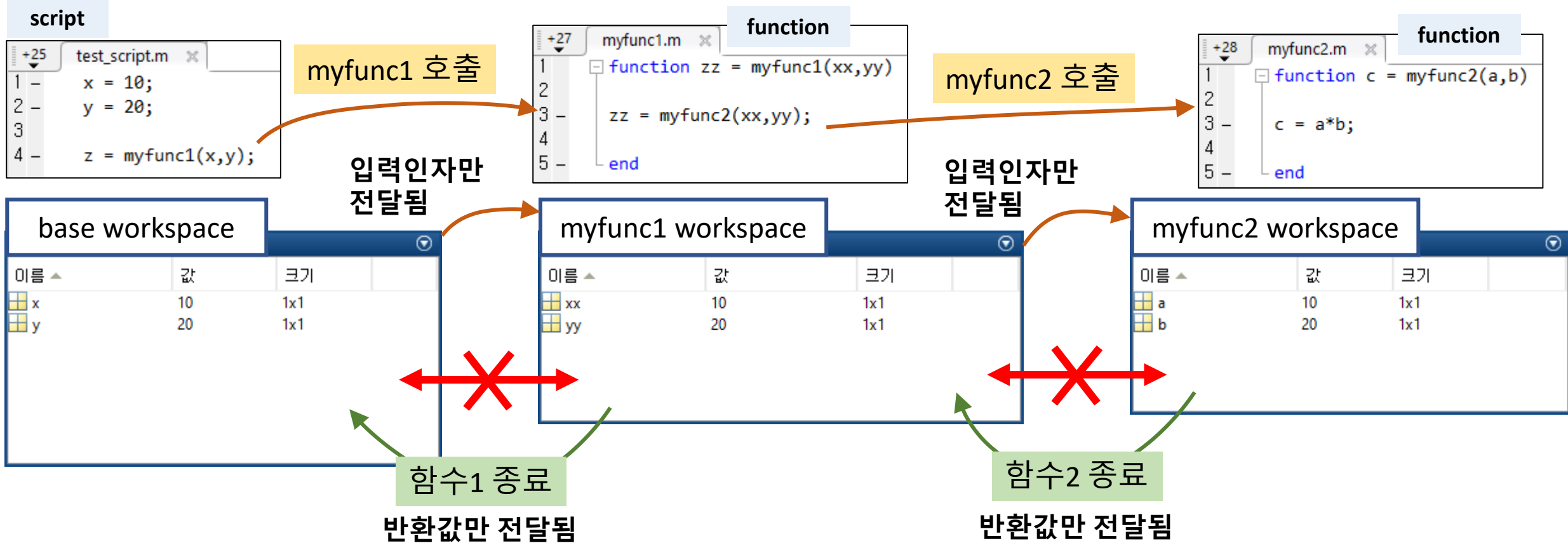


※ breakpoint로 동작 확인



function workspace vs base workspace

※ breakpoint로
동작 확인



- matlab 시작 직후 생성
- 비우는 방법
 - clear
 - matlab 종료

- 함수1이 호출될 때 생성
- 입력인자가 정의된 상태로 시작
- 함수1이 종료되면 완전히 사라짐
- 함수1이 다시 호출되면 새로운 workspace가 생성됨

- 함수2가 호출될 때 생성
- 입력인자가 정의된 상태로 시작
- 함수2가 종료되면 완전히 사라짐
- 함수2가 다시 호출되면 새로운 workspace가 생성됨

script vs function

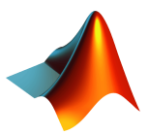
※ F5로 한번에 쭉 실행하는 코드 작성에 익숙해지자.

```
+32 triarea_script.m x
1 - clear ★
2 -
3 - b = 5;
4 - h = 3;
5 - a = 0.5*(b.*h);
```

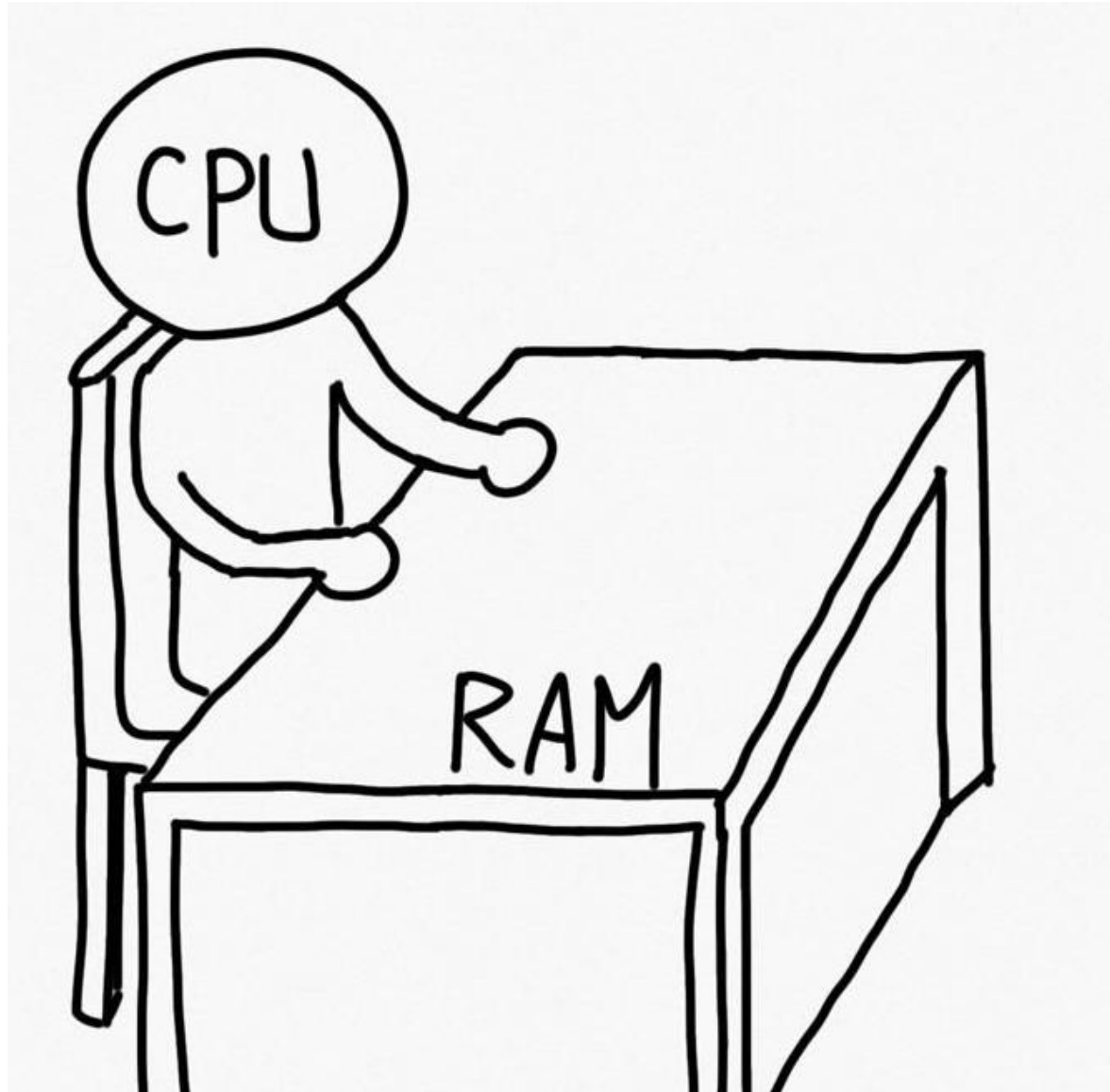
※ clear의 역할은?

```
+31 triarea_function.m x
1 - function a = triarea_function(b,h)
2 -
3 - a = 0.5*(b.*h);
4 -
5 - end
```

- 함수나 스크립트나 한 줄씩 실행하는 건 똑같아 보인다. 하지만...
 - 스크립트에서 생성된 변수는 base workspace에 생성된다.
 - base workspace 변수는 clear 실행 또는 matlab 종료 전까지 없어지지 않는다.
 - clear가 실행되지 않는 한, 스크립트에서 정의하는 변수는 계속 덮어쓴다.
 - 즉, 스크립트 코드끼리는 base workspace를 공유한다.
 - function 내에서 생성되는 변수는 function workspace에만 존재한다.
 - 입력인자도 마찬가지이다.



script는 base workspace를 공유한다.



mycode1.m

```
x = 1;  
y = 2;  
z = 3;
```

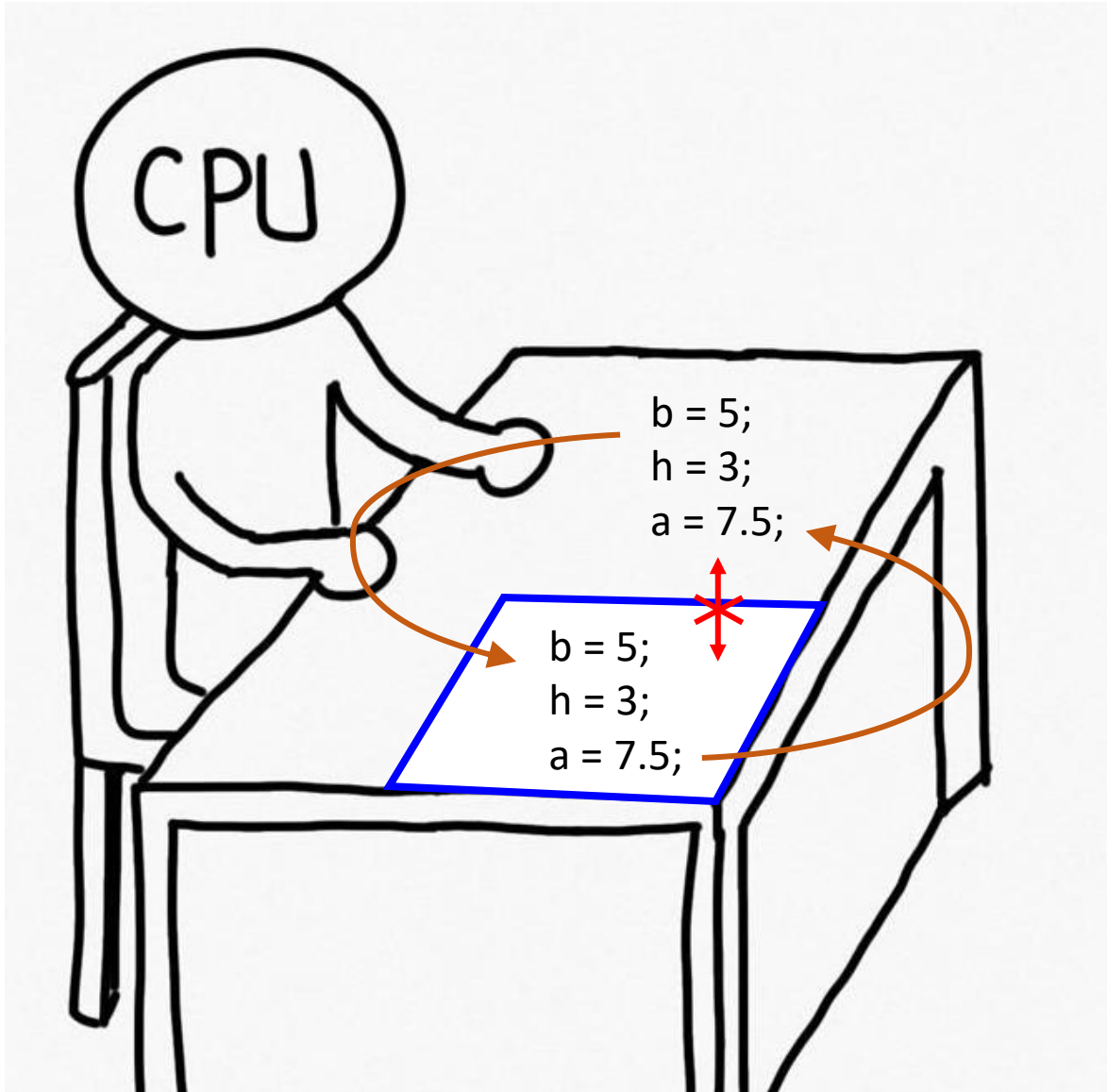
mycode2.m

```
a = 10;  
b = 20;  
x = 30;  
y = 40;
```

※ clear 하기 전까지는 base workspace의 변수는 지워지지 않는다.

(지금까지 코드 실행 전 열심히 clear 했던 이유)

function workspace는 따로 관리된다.



```
+32 triarea_script.m x
1 - clear
2
3 - b = 5;
4 - h = 3;
5 - a = triarea_function(b,h);
```

```
+31 triarea_function.m x
1 - function a = triarea_function(b,h)
2
3 - a = 0.5*(b.*h);
4
5 - end
```

script

vs

function

첫 줄이 **function**으로
시작하지 않음

구분법

첫 줄이 **function**으로
시작함

base workspace

변수 생성 위치

function workspace

base workspace에서
clear가 실행되거나
matlab이 종료될 때

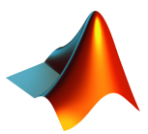
생성된 변수의 소멸

함수 종료 시

파일명 [엔터]

실행 방법

파일명(입력인자) [엔터]

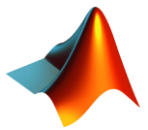
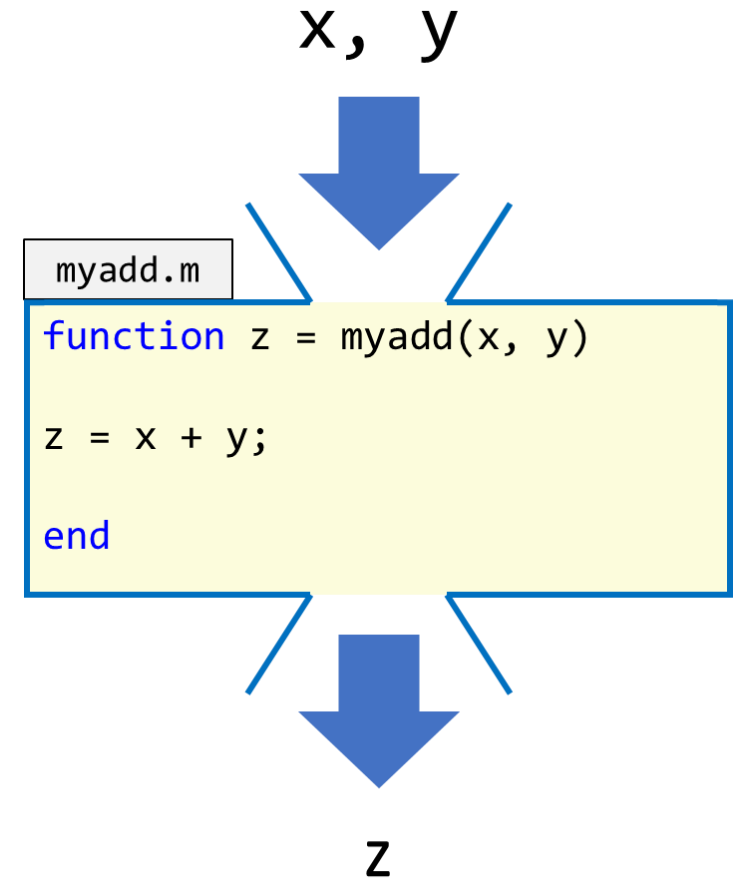


https://www.mathworks.com/help/matlab/matlab_prog/scripts-and-functions.html

https://www.mathworks.com/help/matlab/matlab_prog/base-and-function-workspaces.html

함수가 갖는 + 가져야 할 특징

- 입력인자, 반환값을 통해서만 외부와 소통한다.
 - workspace는 완전히 분리되어 동작한다. (data hiding)
 - base 또는 다른 함수의 workspace에 접근 불가
- 내부 동작원리를 몰라도 쓸 수 있어야 한다.
 - 인터페이스가 정확히 정의되어 있어야 한다.
- 한 가지 동작만을 하는 것이 좋다.
 - 동작이 여러 개라면 여러 함수로 나누는 것이 좋다.
- 하나의 독립된 동작모듈이어야 한다.
 - 그 자체로 완전한 것이어야 한다.
 - 동작하려면 다른 무언가가 필요한 함수는 나쁜 함수이다.



함수 사용법? 우리는 이미 써오고 있었다!

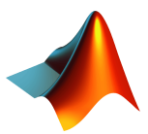
```
>> triarea_function(2,2)
ans =
     2
>> myadd(10,100)
ans =
    110
>> getConeVol(2,3/pi)
ans =
    1.0000
```

- 이름만 봐선 내장함수와 구분조차 할 수 없다.
- 기존 매트랩 함수를 열어보자.
 - factorial
 - std

```
triarea_function.m
1 function a = triarea_function(b,h)
2
3     a = 0.5*(b.*h);
4
5 end

myadd.m
1 function z = myadd(x,y)
2
3     z = x+y;
4
5 end

getConeVol.m*
1 function V = getConeVol(diameter, height)
2
3     V = 1/3*pi*(diameter/2).^2.*height;
4
5 end
```



함수는 언제 종료되는가?

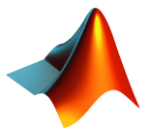
1) 함수 코드를 끝까지 실행하고 나서

```
+33  getConeVol.m*  
1  function V = getConeVol(diameter, height)  
2  
3  V = 1/3*pi*(diameter/2).^2.*height;  
4  
5  end
```

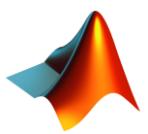
2) `return`을 만났을 때

```
+31  getConeVol2.m*  
1  function V = getConeVol2(diameter, height)  
2  
3  if ~all(size(diameter)==size(height))  
4      warning('input size mismatch')  
5      V = [];  
6      return  
7  end  
8  
9  V = 1/3*pi*(diameter/2).^2.*height;  
10  
11 end
```

※ `size(diameter)==size(height)`라고 쓰면?



함수의 여러 가지 형태



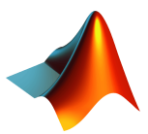
반환값은 꼭 하나여야 할까?

- 반환값이 여러 개인 함수

- 원뿔의 밑면 반지름, 높이 → 부피, 겉넓이
- 이차 방정식의 계수 3개 → 두 근
- 벡터 → 평균, 표준편차
- 복소수 → 크기, 위상각
- 재료 이름 → 재료의 물성

- 이미 반환값이 여러 개인 내장함수들

- min, max, find, ...



반환값이 2개 이상인 함수

```
+34 get_abs_angle.m x
1 function [mag, ang] = get_abs_angle(x)
2
3     mag = abs(x);
4     ang = angle(x);
5
6 end
```

```
+33 get_mean_std.m x
1 function [m, s] = get_mean_std(x)
2
3     m = mean(x);
4     s = std(x);
5
6 end
7
```

```
+25 get_material.m x
1 function [rho,E,G] = get_material(name)
2
3     mu = 0.33;
4     if strcmpi(name, 'Aluminium')
5         rho = 2700;
6         E = 68.9e9;
7         G = E/(2*(1+mu));
8     elseif strcmpi(name, 'Ti')
```

funcname.m

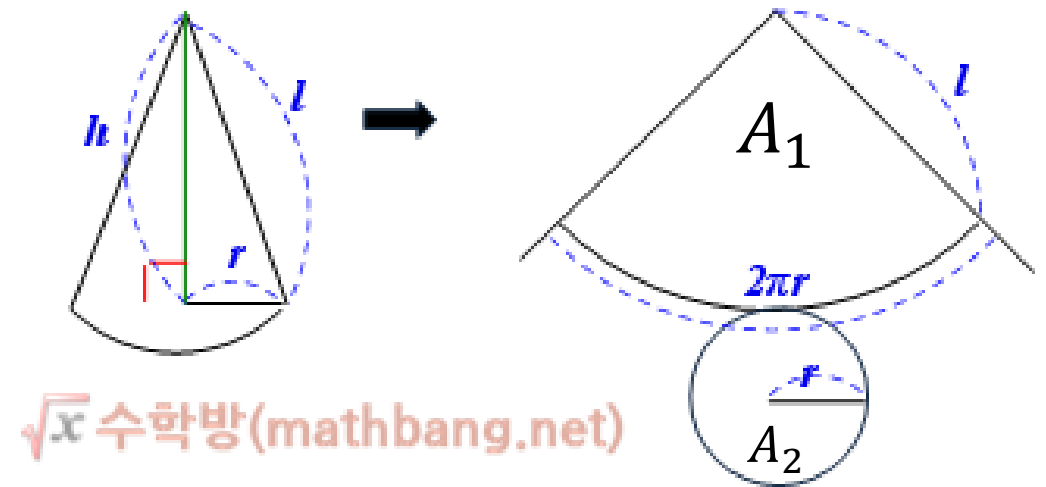
```
function [out1, out2, ...] = funcname(input1, input2, ...)
...
...
out1 = ...
out2 = ...

end
```

```
>> x = randn(10000,1);
>> [m,s] = get_mean_std(x)
m =
    -0.0014
s =
    0.9894
>> get_mean_std(x)
ans =
    -0.0014
★ >> s = get_mean_std(x)
s =
    -0.0014
★ >> [~,s] = get_mean_std(x)
s =
    0.9894
```

반환값이 2개 이상인 함수

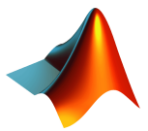
```
+32  getConeVol3.m  x
1  function [V, A] = getConeVol3(diameter, height)
2  %getConeVol Calculate volumes of cones
3  % input: diameter, height
4  %     - Must be matrices with same size
5  %     - If size mismatches -> returns empty array
6  % output: volumes of cones, surface areas of cones
7
8  if ~all(size(diameter)==size(height))
9      warning('input size mismatch')
10     V = [];
11     return
12 end
13
14 V = 1/3*pi*(diameter/2).^2.*height;
15
16 L = sqrt((diameter/2).^2 + height.^2);
17 A1 = pi*L.*(diameter/2);
18 A2 = pi*(diameter/2).^2;
19 A = A1+A2;
20
21 end
```



$$V = \frac{1}{3}\pi r^2 h$$

$$A_1 = \pi l r$$

$$A_2 = \pi r^2$$



반환값은 꼭 있어야 할까?

- 반환값이 없는 함수

- 숫자나 문자 입력 → 특정 포맷으로 출력
- 온도와 파장 입력 → 흑체복사 그래프만 그림
- i, N → for문이 몇% 진행됐는지 출력

funcname.m

```
function funcname(input1, input2, ...  
...  
...  
end
```

- 이미 반환값이 없는 내장함수

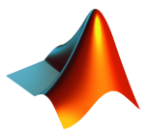
- disp
- fprintf는?
- plot은?

```
N = 10000;  
x = rand(N,1);  
  
for i=1:N  
    if floor(i/N*10)~=floor((i-1)/N*10)  
        dispperc(i,length(x),'perc')  
    end  
    ...  
    ...  
end
```

+35

dispperc.m

```
function dispperc(i,N,mode)  
    switch mode  
        case 'perc'  
            fprintf('%d%% progressing...\n',round(i/N*100));  
        case 'quot'  
            fprintf('%d/%d progressing...\n',i,N);  
    end
```



입력은 꼭 있어야 할까?

- 입력이 없는 함수
 - 각종 물리 상수를 반환하는 함수
 - `get_G()` → 중력 상수를 반환 (6.67408×10^{-11})
 - `get_h()` → 플랑크 상수를 반환 ($6.62607004 \times 10^{-34}$)
 - 함수 호출 시 ()를 붙일까? 말까?
 - 안 붙여도 되지만, 함수임을 드러내기 위해 붙이자.

funcname.m

```
function [out1, out2, ...] = funcname
...
...
end
```

```
+36 get_G.m x
1 function G = get_G
2
3 G = 6.67408e-11;
4
5 end
```

```
>> G = get_G()
G =
    6.6741e-11
>> h = get_h()
h =
    6.6261e-34
```

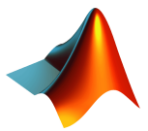
```
+35 get_h.m x
1 function h = get_h
2
3 h = 6.62607004e-34;
4
5 end
```

입출력이 모두 없는 함수도 가능할까?

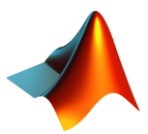
```
+36 hello.m x
1 function hello
2
3     disp('Hello, world!')
4
5 end
```

```
+37 setfig.m x
1 function setfig
2
3     disp('xlabel(''x'')')
4     disp('ylabel(''y'')')
5     disp('zlabel(''z'')')
6     disp('title(''mytitle'')')
7     disp('set(gca, 'fontSize', 12)')
```

```
>> hello
Hello, world!
>>
>> setfig
xlabel('x')
ylabel('y')
zlabel('z')
title('mytitle')
set(gca, 'fontSize', 12)
>>
```



함수 활용



잘 만든 함수 + 도움말을 활용해보자.

- 내가 만든 함수를 가장 먼저 쓰는 사람은 누구일까?
- 내가 지난 달에 만든 함수를 내가 정확히 기억하고 있을까?
 - "입력이 diameter가 먼저였나? height가 먼저였나?"
 - "diameter를 넣는 거였나? radius를 넣는 거였나?"


```
>> help getConeVol
getConeVol Calculate volumes of cones

V = getConeVol(diameter, height)


input: diamter, height
      - must be matrices with same size
output: volumes of cones
```

```
fx >> getConeVol(
    getConeVol(diameter,height)
    자세한 도움말...
```

```
+31 myfunction.m x
1 function c = myfunction(a, b)
2
3 c = 1/3*pi*(a/2).^2.*b;
4
5 end
```

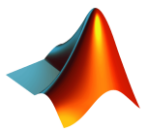


```
+30 getConeVol.m x
1 function V = getConeVol(diameter, height)
2 %getConeVol Calculate volumes of cones
3 %
4 % V = getConeVol(diameter, height)
5 %
6 % input: diamter, height
7 %       - must be matrices with same size
8 % output: volumes of cones
9
10 V = 1/3*pi*(diameter/2).^2.*height;
11
12 end
```



잘 만든 함수란?

- 함수 이름만 보아도 동작을 유추할 수 있는 함수
 - function1 → getConeVol
 - 함수명은 가급적 동사로 시작하자. (coneVol → getConeVol)
- 입출력 변수명에서 의미를 유추할 수 있는 함수
 - getConeVol(a,b) → getConeVol(diameter, height)
- 도움말이 충실히 작성된 함수
 - 도움말에서 인터페이스를 정확히 정의
 - 더 좋은 함수는? → 주석이 없어도 잘 읽히는 함수!
- 한 가지 일만 하는 함수
 - 두 가지 이상의 동작을 하는 함수? → 가독성 저하, 디버깅 어려워짐
 - 한 가지 동작만을 하는 함수들로 나뉘라.



몇 가지 함수 예제

```
+27 temp_C2F.m x
1 function F = temp_C2F(C)
2 %temp_C2F Convert temperature Celcius to Fahrenheit
3 %
4 % F = temp_C2F(C)
5
6 F = C*1.8 + 32;
7
8 end
```

화씨 -> 섭씨

```
+27 temp_F2C.m x
1 function C = temp_F2C(F)
2 %temp_F2C Convert temperature Fahrenheit to Celcius
3 %
4 % C = temp_F2C(F)
5
6 C = (F-32)/1.8;
7
8 end
```

섭씨 -> 화씨

```
+30 calc_ball_max_height.m x
1 function h_max = calc_ball_max_height(v0)
2 %calc_ball_max_height Calculate max height of ball thrown vertically
3 %
4 % h_max = calc_ball_max_height(v0)
5 %
6 % input: v0 = initial velocity (m/s)
7 % output: h_max = max height (m)
8
9 h_max = v0^2/9.81^2;
10
11 end
```

수직으로 던진 공의
최대 높이

```
+29 count_numel_over_n.m x
1 function [num, U, L] = count_numel_over_n(A, n)
2 %count_numel_over_n Count number of elements of A>n
3 %
4 % [num, U, L] = count_numel_over_n(A, n)
5 %
6 % input: A = input matrix
7 %         n = a number over which elements of A will be counted
8 %
9 % output: num = number of elements of A over n
10 %           U = a logical array; 1 if A>n, else 0
11 %           L = 1-U
12
13 num = sum(A(:)>n);
14 U = (A>n);
15 L = logical(1-U);
16
17 end
```

A 원소 중 n보다
큰 것의 개수 반환

```
+28 get_grade.m x
1 function grade = get_grade(score, grades, scorecuts)
2 %get_grade Return grade char array for given score, grades and scorecuts
3 %
4 % grade = get_grade(score, grades, scorecuts)
5 %
6 % input: score = a number to get grade
7 %         grades = a char arrays containing grades
8 %               ex.) grades = 'ABCD';
9 %         scorecuts = scores to cutoff
10 %
11 % output: grade char array
12
13 position = sum(score>=scorecuts);
14 N = length(grades);
15 grade = grades(N-position);
16
17 end
```

점수 -> grade

함수를 script로? script를 함수로?

```
+32  getConeVol3.m x
1  function [V, A] = getConeVol3(diameter, height)
2  %getConeVol Calculate volumes of cones
3  % input: diameter, height
4  %     - Must be matrices with same size
5  %     - If size mismatches -> returns empty array
6  % output: volumes of cones, surface areas of cones
7
8  if ~all(size(diameter)==size(height))
9      warning('input size mismatch')
10     V = [];
11     return
12 end
13
14 V = 1/3*pi*(diameter/2).^2.*height;
15
16 L = sqrt((diameter/2).^2 + height.^2);
17 A1 = pi*L.*(diameter/2);
18 A2 = pi*(diameter/2).^2;
19 A = A1+A2;
20
21 end
```

```
+33  getConeVol3_script.m x
1  %getConeVol Calculate volumes of cones
2  % input: diameter, height
3  %     - Must be matrices with same size
4  %     - If size mismatches -> returns empty array
5  % output: volumes of cones, surface areas of cones
6
7  diameter = 10;
8  height = 20;
9
10 if ~all(size(diameter)==size(height))
11     warning('input size mismatch')
12     V = [];
13     return
14 end
15
16 V = 1/3*pi*(diameter/2).^2.*height;
17
18 L = sqrt((diameter/2).^2 + height.^2);
19 A1 = pi*L.*(diameter/2);
20 A2 = pi*(diameter/2).^2;
21 A = A1+A2;
```

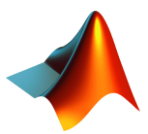
※ 스크립트에서 사용자가 바꿀만한 부분은 맨 앞부분에 몰아놓자.

자주 하는 실수들

- 함수명과 파일명이 다른 경우

```
+26
1  function V = getConeVolume(diameter, height)
2  %getConeVol Calculate volumes of cones
3  % input: diamter, height
4  %      - must be matrices with same size
5  % output: volumes of cones
6
7  V = 1/3*pi*(diameter/2).^2.*height;
8
9  end
```

- 함수명과 같은 변수가 workspace에 있는 경우
 - 변수를 먼저 읽어온다.



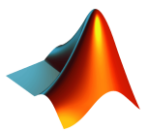
자주 하는 실수들

- workspace가 공유되는 것으로 착각하는 경우
 - 함수 입력인자를 넣지 않은 경우 → "입력 인수가 부족합니다."
 - 반환값을 변수에 저장하지 않은 경우 → "정의되지 않은 변수입니다."

```
+30  getConeVol.m x
1  function V = getConeVol(diameter, height)
2  %getConeVol Calculate volumes of cones
3  %
4  % V = getConeVol(diameter, height)
5  %
6  % input: diameter, height
7  %       - must be matrices with same size
8  % output: volumes of cones
9
10  V = 1/3*pi*(diameter/2).^2.*height;
11
12  end
```

```
>> diameter = 10;
>> height = 20;
>> V = getConeVol;
입력 인수가 부족합니다.
오류 발생: getConeVol (line 10)
V = 1/3*pi*(diameter/2).^2.*height;
```

```
>> diameter = 10;
>> height = 20;
>> getConeVol(diameter, height);
>> disp(V)
'V'은(는) 정의되지 않은 함수 또는 변수입니다.
```



자주 하는 실수들

- 함수의 반환값을 제대로 쓰지 않은 경우

```
+30
getConeVol.m
1 function V = getConeVol(diameter, height)
2 %getConeVol Calculate volumes of cones
3 %
4 % V = getConeVol(diameter, height)
5 %
6 % input: diamter, height
7 %       - must be matrices with same size
8 % output: volumes of cones
9
10 V = 1/3*pi*(diameter/2).^2.*height;
11
12 end
```

```
function getConeVol(diameter, height)

V = 1/3*pi*(diameter/2).^2.*height;

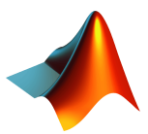
end
```

```
function [V, A] = getConeVol3(diameter, height)

V = 1/3*pi*(diameter/2).^2.*height;

L = sqrt((diameter/2).^2 + height.^2);
A1 = pi*L.*(diameter/2);
A2 = pi*(diameter/2).^2;
disp(A1+A2)

end
```



자주 하는 실수들

- 함수의 입력인자를 내부에서 새로 정의하는 경우

```
+30  getConeVol.m x
1  function V = getConeVol(diameter, height)
2  %getConeVol Calculate volumes of cones
3  %
4  % V = getConeVol(diameter, height)
5  %
6  % input: diameter, height
7  %      - must be matrices with same size
8  % output: volumes of cones
9
10 - V = 1/3*pi*(diameter/2).^2.*height;
11
12 - end
```

```
>> diameter = 10;
>> height = 20;
>> getConeVol(diameter, height)
ans =
    523.5988
```

```
function V = getConeVol(diameter, height)

diameter = 10;
height = 20;

V = 1/3*pi*(diameter/2).^2.*height;

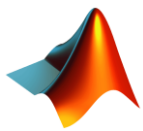
end
```

```
function V = getConeVol(diameter, height)

diameter = input('type diameter: ');
height = input('type height: ');

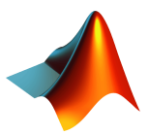
V = 1/3*pi*(diameter/2).^2.*height;

end
```



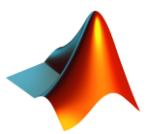
자주 하는 실수들

- 함수명 규칙에 맞지 않은 경우
 - 함수명 규칙은 변수명 규칙과 같다.
 - 알파벳, 숫자, 밑줄만 허용 (첫 글자는 알파벳만 허용)
 - 한글, 콜론, 세미콜론, 콤마, 빈칸, 키워드 안됨
 - 호출 시 대소문자 구분함 (윈도우 파일명은 대소문자 구분하지 않음)
 - 내장함수명 지양
- command window에서 함수를 정의하려 한 경우
 - 함수는 editor에서만 정의 가능
- 함수 파일을 F5로 실행한 경우
 - 입력이 없는 함수인 경우에만 오류없이 실행됨



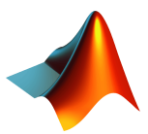
자주 하는 실수들

- 첫 줄을 `function`으로 시작하지 않은 경우
 - 첫 줄을 `function`으로 시작하지 않은 파일은 함수가 아니라 스크립트
- 함수를 `end`로 닫아주지 않은 경우
 - 메인 함수 1개만 있는 경우는 없어도 됨
 - 로컬 함수가 있는 경우는 모든 함수는 `end`로 닫아주어야 함



정리

- 함수를 만드는 이유 = 코드의 모듈화
 - 코드의 관리, 확장, 배포, 재사용이 쉬워진다. 일단 만들면 내장함수처럼 쓸 수 있다.
- 만드는 법: 새 m파일 만들기, function 키워드로 시작
 - `function [out1, out2, ...] = funcname(in1, in2, ...)`
- script vs function
 - script는 base workspace를 사용, script 파일 간에는 base workspace 공유
 - function은 자기만의 function workspace만을 사용
- 함수의 입출력
 - 입력이 없을 수도 있고 2개 이상일 수도 있다.
 - 출력이 없을 수도 있고 2개 이상일 수도 있다.
- 핵심포인트
 - 함수는 workspace를 공유하지 않는다.
 - base workspace 및 다른 함수의 workspace와는 완전히 독립된 workspace를 생성 및 관리한다.
 - 함수명과 파일명은 같아야 한다.
 - 그 자체로 완전한 것이어야 한다.



강의 로드맵

기초 문법

행렬 다루기

- 행렬 만들기
- 인덱싱, 행렬 연산

시각화

- plot, plot3
- mesh, surf

조건문과 반복문

- if, elseif, else
- for, while

함수

- 사용자 정의 함수 기초

기타

- 파일입출력 (이미지, 텍스트, 엑셀 파일 다루기)
- 프로그래밍 스킬
- cell, struct 자료형

고급 문법

시각화 고급

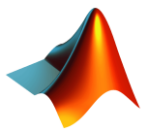
- 그래픽 객체
- gca, gcf, axes, set, get
- 애니메이션

함수 고급

- 로컬함수, 익명함수, 함수핸들
- varargin, nargin

수치해석 활용

- 선형보간
- 회귀분석
- 수치미분
- 수치적분
- 방정식의 해
- 미분방정식의 해
- 테일러 전개
- ...
- ...



Q&A

