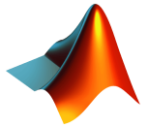
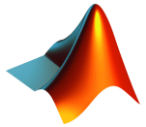
MATLAB 프로그래밍 및 실습

9강. 함수 고급 - part1

(로컬함수, 익명함수)



로컬 함수
local function



https://www.youtube.com/watch?v=oOKRmzrtkrY

함수에서 다른 함수를 호출하는 경우

```
% test_script.m

x = input('x: ');
y = input('y: ');
a = myfunc1(x, y);
```

```
% myfunc1.m
function z = myfunc1(x, y)

z1 = sum_pow2(x, y);
z2 = sum_pow3(x, y);
z = z1 * z2;

end
```

```
% sum_pow2.m
function z = sum_pow2(x, y)

z = 2^x + 2^y;

end
```

```
% sum_pow3.m
function z = sum_pow3(x, y)

z = 3^x + 3^y;

end
```

- myfunc1에서 반복적으로 사용되는 코드
  -> sum_pow2, sum_pow3을 함수화
- sum_pow2와 sum_pow3를 쓸 일이 myfunc1 계산 외에는 없다면?
  -> 파일 3개는 항상 같이 다녀야 한다.
  -> 파일관리(searchpath 관리)를 해야 한다. 불편하다.

원뿔의 밑면 지름과 높이 -> 부피와 겉넓이

```
diameter = [1,2,3,4,5];
height = [5,4,3,2,1];

[vol, area] = getConeProp(diameter, height);
```

```
function [vol, area] = getConeProp(diameter, height)

vol = get_cone_volume(diameter, height);
area = get_cone_area(diameter, height);

end
```

```
function vol = get_cone_volume(diameter, height)

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

end
```

```
function area = get_cone_area(diameter, height)

L = sqrt((diameter/2).^2 + height.^2);
area_side = pi*L.*(diameter/2);
area_bottom = pi*(diameter/2).^2;
area = area_side + area_bottom;

end
```

- 활용범위가 제한적인 함수 -> 파일관리가 귀찮다.
  - 통계자료 중 특별한 조건을 만족하는 회수를 구하는 함수
  - 특정 실험데이터에만 항상 적용해야 하는 데이터처리 함수

로컬 함수가 이걸 해결해준다.

```
% test_myfunc1_script_only

a = myfunc1(2, 3);
```

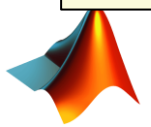
스크립트

```
% myfunc1.m

function z = myfunc1(x, y)
z1 = sum_pow2(x, y);
z2 = sum_pow3(x, y);
z = z1 * z2;
end

function z = sum_pow2(x, y)
z = 2^x + 2^y;
end

function z = sum_pow3(x, y)
z = 3^x + 3^y;
end
```

함수

```
% test_myfunc1_with_local_funcs

a = myfunc1(2, 3);

function z = myfunc1(x, y)
z1 = sum_pow2(x, y);
z2 = sum_pow3(x, y);
z = z1 * z2;
end

function z = sum_pow2(x, y)
z = 2^x + 2^y;
end

function z = sum_pow3(x, y)
z = 3^x + 3^y;
end
```

스크립트



메인 함수와 로컬 함수, 함수의 흐름

```
% test_myfunc1_script_only

a = myfunc1(2, 3);
```

스크립트

```
% myfunc1.m

function z = myfunc1(x, y)
z1 = sum_pow2(x, y);
z2 = sum_pow3(x, y);
z = z1 * z2;
end

function z = sum_pow2(x, y)
z = 2^x + 2^y;
end

function z = sum_pow3(x, y)
z = 3^x + 3^y;
end
```

함수

myfunc1: 메인 함수

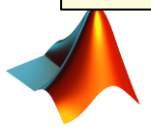
* 외부에서 호출 가능
* 외부에서는 myfunc1만 보임
* 메인 함수는 가장 위에 있어야 함

sum_pow2: 로컬 함수

* 외부에서 호출 불가능 (외부에서 보이지 않음)
* myfunc1 내에서만 호출 가능
* 로컬 함수끼리는 순서 변경 가능

sum_pow3: 로컬 함수

* 외부에서 호출 불가능 (외부에서 보이지 않음)
* myfunc1 내에서만 호출 가능
* 로컬 함수끼리는 순서 변경 가능



메인 함수와 로컬 함수, 함수의 흐름

```
% test_myfunc1_with_local_funcs

a = myfunc1(2, 3);

function z = myfunc1(x, y)
z1 = sum_pow2(x, y);
z2 = sum_pow3(x, y);
z = z1 * z2;
end

function z = sum_pow2(x, y)
z = 2^x + 2^y;
end

function z = sum_pow3(x, y)
z = 3^x + 3^y;
end
```

스크립트

myfunc1: 로컬 함수

* 외부에서 호출 불가능 (외부에서 보이지 않음)
* 작성된 m 파일에서만 호출 가능
* 로컬 함수끼리는 순서 변경 가능

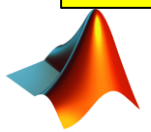
sum_pow2: 로컬 함수

* 외부에서 호출 불가능 (외부에서 보이지 않음)
* 작성된 m 파일에서만 호출 가능
* 로컬 함수끼리는 순서 변경 가능

sum_pow3: 로컬 함수

* 외부에서 호출 불가능 (외부에서 보이지 않음)
* 작성된 m 파일에서만 호출 가능
* 로컬 함수끼리는 순서 변경 가능

※ 이 경우 메인 함수는 없음



로컬 함수의 특징

```
% myfunc1.m

function z = myfunc1(x, y)
z1 = sum_pow2(x, y);
z2 = sum_pow3(x, y);
z = z1 * z2;
end

function z = sum_pow2(x, y)
z = 2^x + 2^y;
end

function z = sum_pow3(x, y)
z = 3^x + 3^y;
end
```

"함수 이름은 유일해야 한다."

- m파일 하나에 로컬 함수 여러 개 작성 가능
  - 메인 함수는 반드시 최상단에 위치하며, 파일명과 같은 이름
  - 메인 함수가 없는 스크립트는 로컬함수명과 파일명이 달라야 함
  - 로컬 함수 간의 순서는 상관없음
- m파일에 함수가 2개 이상이면 (=로컬 함수가 있다면)
  - 메인 함수를 포함한 모든 함수는 end로 닫아주어야 함
- 로컬 함수는 자신이 포함된 m파일에서만 호출 가능
- 같은 이름의 m파일이 있어도 로컬 함수가 우선됨
- 로컬 함수명은 파일명(메인 함수명)과 달라야 함
  - 같으면 어떤 일이 일어날까?
  - 스크립트에 정의된 함수들은 모두 로컬 함수임에 주의

로컬 함수의 특징

```
function z = myfunc2(x, y)
z1 = sum_pow2(x, y);
z2 = sum_pow3(x, y);
z = z1 * z2;
end

function z = sum_pow2(x, y)
z = power2(x) + power2(y);
end

function z = sum_pow3(x, y)
z = power3(x) + power3(y);
end

function p = power2(n)
p = 2^n;
end

function p = power3(n)
p = 3^n;
end
```

- 로컬 함수가 다른 로컬 함수 호출 가능
  - 로컬 함수 간의 순서는 무관
  - 함수가 정의되어 있기만 하면 OK

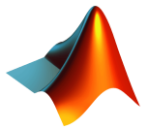
메인 함수	vs	로컬 함수
function: m 파일의 첫 번째 함수 script: 메인 함수 없음	함수 작성 위치	function: m 파일의 두 번째 이후의 함수들 script: 모든 함수
외부에서 호출할 수 있음 (명령창 또는 다른 파일)	호출 가능 위치	자신이 포함된 파일에서만 호출 가능
함수명은 파일명과 **같아야** 한다.	함수명	함수명은 파일명과 **달라야** 한다.

공통점

`function` [출력1, 출력2, …] = 함수이름(입력1, 입력2, …)

workspace를 공유하지 않는다.

함수명 규칙은 변수명 규칙과 같다.



로컬 함수의 활용

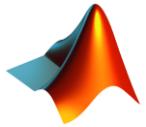
```
function [V, A] = getConeProp2(diameter, height)

if ~sizechk(diameter, height)
    warning('input size mismatch')
    V = [];
    A = [];
    return
end

V = get_cone_volume(diameter, height);
A = get_cone_area(diameter, height);

end

function TF = sizechk(diameter, height)
TF = all(size(diameter)==size(height));
end

function vol = get_cone_volume(diameter, height)
vol = 1/3*pi*(diameter/2).^2.*height;
end

function area = get_cone_area(diameter, height)
L = sqrt((diameter/2).^2 + height.^2);
area_side = pi*L.*(diameter/2);
area_bottom = pi*(diameter/2).^2;
area = area_side + area_bottom;
end
```

"함수는 한 가지 동작만을 해야 한다."

로컬 함수의 활용

```
function names = get_random_names(N)

lastnames = randsample(get_lastnames(),N);
firstnames = randsample(get_firstnames(),N);

names = firstnames + " " + lastnames;

end

function names = get_lastnames

names = [    "Smith" ;    "Johnson";    "Scott";    "Torres";
             "Nguyen";    "Hill";    "Flores";    "Green"];
end

function names = get_firstnames

names = [    "James";    "David";    "Christopher";    "George";
             "Ronald";    "William";    "Thomas";    "Donald"];
end
```

"가독성은 중요하다."



언제 유용한가?

- 활용범위가 제한적인 함수를 작성할 때
- 하나의 파일에서 함수들을 관리하는 것이 더 유리할 때
  - 파일 여러 개 관리하기 귀찮을 때

prob3.m

```
function prob3(N)

% project Euler, problem #3
%
% The prime factors of 13195 are 5, 7, 13 and 29.
% What is the largest prime factor of the number 600851475143 ?

if ~nargin
    N = 600851475143;
end

w_factors(N);
wo_factors(N);
wo_factors_modified(N);
wo_factors_matmatt_python(N);
wo_factors_spinachp_python(N);
wo_factors_matmatt_modified(N);
end

function w_factors(N) ...

function wo_factors(N) ...

function wo_factors_matmatt_python(N) ...

function wo_factors_spinachp_python(N) ...

function wo_factors_matmatt_modified(N) ...

function wo_factors_modified(N) ...
```

```
function assignment1

D = randn(100,1) + 10;
H = randn(100,1) + 5;
VCone = getConeVol(D, H);
fprintf('std of volumes of cones: %5.2f\n',calcSTD(VCone));

D1 = randn(100,1) + 20;
D2 = randn(100,1) + 10;
H = randn(100,1) + 5;
VTruncCone = getVolTruncCone(H, D1/2, D2/2);
fprintf('std of volumes of truncated cones: %5.2f\n',calcSTD(VTruncCone));

end

function V = getConeVol(D, H) ...

function V = getVolTruncCone(H, r1, r2) ...
```

```
function maxOfTwoFuncs
x = pi/2:0.01:4*pi;
y1 = fn1(x);
y2 = fn2(x);
figure, plot(max(y1,y2))

end

function y = fn1(x)
y = exp(-x/10).*sin(x);
end

function y = fn2(x)
y = log(x).*cos(x);
end
```

언제 유용한가?

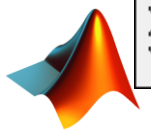
- 함수 코드의 가독성을 높이고 싶을 때

getConeProp2.m

```
1   function [V, A] = getConeProp2(diameter, height)
2   %getConeVol Calculate volumes of cones
3    % input: diamter, 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 ~sizechk(diameter, height)
9        warning('input size mismatch')
10       V = [];
11       A = [];
12       return
13   end
14
15   V = get_cone_volume(diameter, height);
16
17   A = get_cone_area(diameter, height);
18
19   end
20
21  function TF = sizechk(diameter, height)...
26
27  function vol = get_cone_volume(diameter, height)...
32
33  function area = get_cone_area(diameter, height)...
```

get_random_names.m

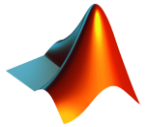
```
1   function names = get_random_names(N)
2   % ref: https://www.al.com/news/2019/10/50-most-common-last
3    % ref: https://rong-chang.com/namesdict/popular_names.htm
4
5    lastnames = randsample(get_lastnames(),N);
6    firstnames = randsample(get_firstnames(),N);
7
8    names = firstnames + " " + lastnames;
9
10   end
11
12  function names = get_lastnames...
68
69  function names = get_firstnames...
```



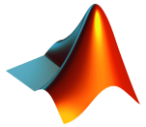
로컬함수 작성 시 주의할 점

- 함수 사이에 스크립트가 올 수 없다.
- function 작성 시
  - 모든 코드는 함수에 속해있어야 한다.
- script 작성 시
  - 모든 스크립트는
    첫 번째 로컬 함수 이전에 작성한다.

```
a = myfunc1(2, 3);

function z = myfunc1(x, y)
z1 = sum_pow2(x, y);
z2 = sum_pow3(x, y);
z = z1 * z2;
end

b = myfunc1(3,4);

function z = sum_pow2(x, y)
z = 2^x + 2^y;
end

function z = sum_pow3(x, y)
z = 3^x + 3^y;
end
```



익명함수와 함수 핸들
anonymous function and function handle



https://www.youtube.com/watch?v=CIDo9LgVXq8

함수 = m 파일?

- 함수는 항상 파일로 작성해야 할까?
  - 파일을 만든다.

    = 파일을 **관리**해야 한다.

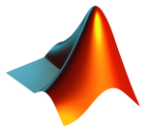
    = 파일의 **경로를 관리**해야 한다. (searchpath)

    = **버전 관리**를 해야 한다.

    = 명령 창에서 정의할 수 **없다.**

- 그런 거 필요 없는 아주 간단한 한줄짜리 임시 함수를 만들 수는 없을까?

```
x          ->    y = x^3 -5*x^2 + 3*x + 4
u, v       ->    w = exp(u^2+v^2)*u/v^2
```



있다! - 익명함수

1) 이렇게 생겼다.

함수명 = @(입력인자) 표현식

2) 한줄이다.

3) 명령창에서 정의할 수 있다.
: 파일을 만들지 않아도 된다.
: 에디터(m파일)에서도 할 수 있다.

4) function_handle이라는 게 생긴다.
: whos로 볼 수 있다. -> 함수? 변수?!

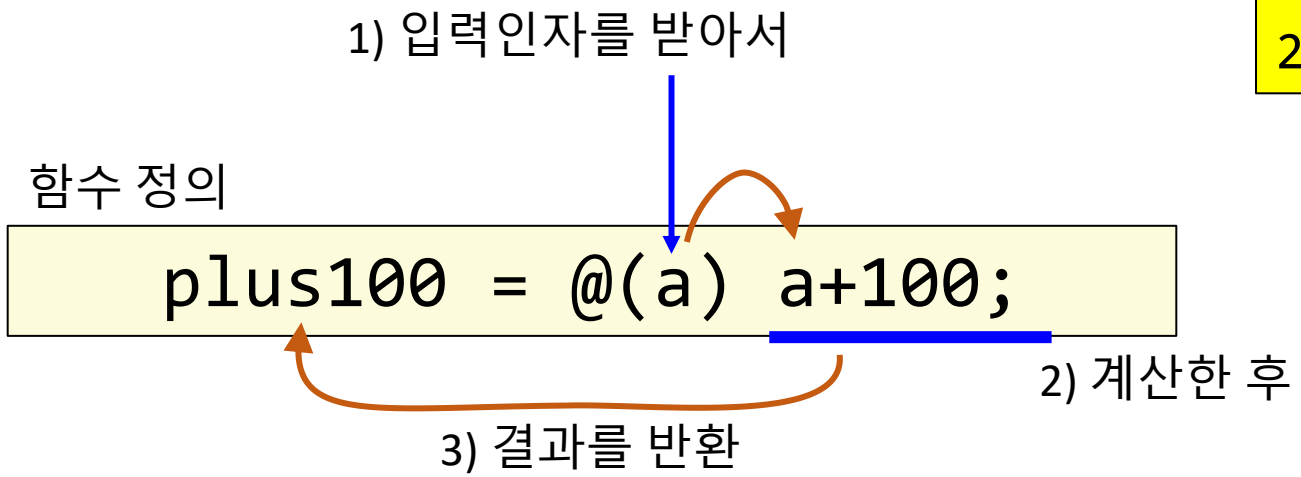
5) 함수 호출은 일반 함수와 같다.
: 호출 형태로는 사용자 정의 함수, 로컬함수, 내장함수와 구별할 수 없다.

6) 반환값을 별도로 정의하지 않는다.

```
>> clear
>> plus100 = @(a) a+100;
>> doubleit = @(x) x*2;
>> squareit = @(x) x.^2;
>> fun1 = @(x) x^3 -5*x^2 + 3*x + 4;
>> plus100
plus100 =
  function_handle with value:
    @(a)a+100
>> whos
  Name          Size              Bytes  Class

  doubleit      1x1                  32  function_handle
  fun1          1x1                  32  function_handle
  plus100       1x1                  32  function_handle
  squareit      1x1                  32  function_handle

>> plus100(23)
ans =
   123
>> doubleit(100)
ans =
   200
>> squareit(5)
ans =
    25
>> fun1(2)
ans =
    -2
```

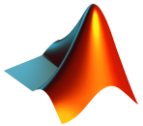
익명함수의 동작 원리 - 함수 정의

1) 이렇게 생겼다.
   함수명 = @(입력인자) 표현식
2) 한 줄이다.



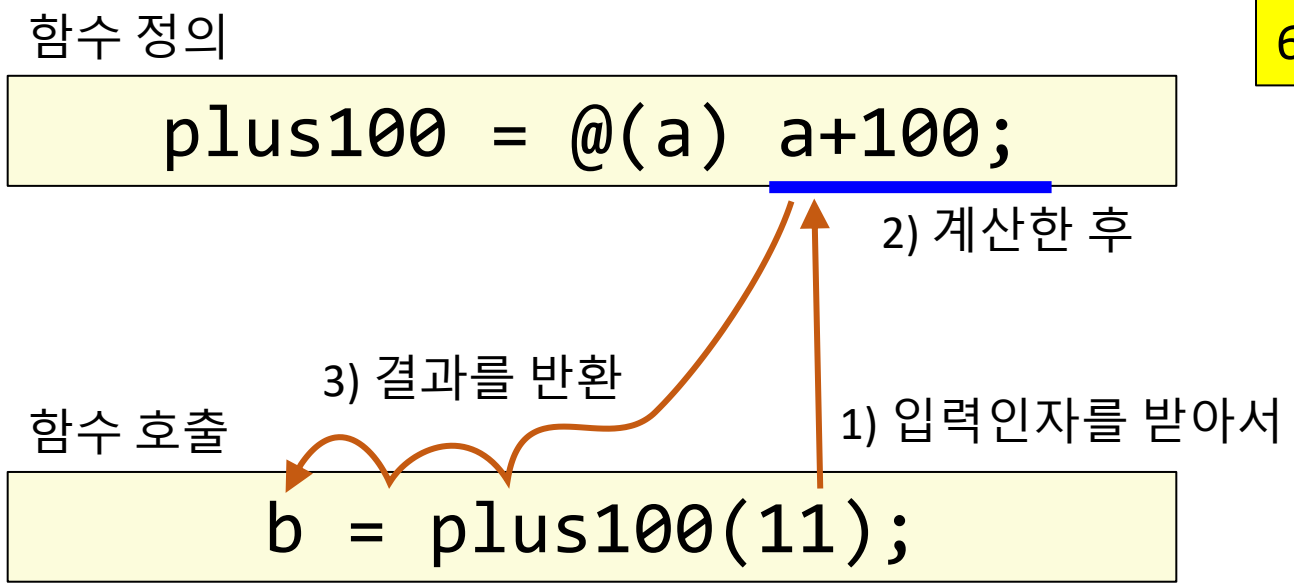
함수 호출

```
b = plus100(11);
```



익명함수의 동작 원리 - 함수 호출

1) 이렇게 생겼다.
   함수명 = @(입력인자) 표현식
2) 한 줄이다.
5) 함수 호출은 일반 함수와 같다.
6) 반환값을 별도로 정의하지 않는다.



* 같은 동작의 사용자 정의 함수

```
function b = plus100(a)
b = a+100;
end
```

간단한 익명함수 예제

익명함수

```
doubleit = @(x) x*2;
```

```
squareit = @(x) x.^2;
```

```
fun1 = @(x) x^3 -5*x^2 + 3*x + 4;
```

```
get_circle_area = @(r) pi*r.^2;
```

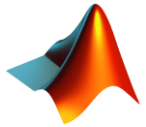
동일한 기능의 사용자 정의 함수

```
function xx = doubleit(x)
xx = x*2;
end
```

```
function xx = squareit(x)
xx = x.^2;
end
```

```
function y = fun1(x)
y = x^3 -5*x^2 + 3*x + 4;
end
```

```
function A = get_circle_area(r)
A = pi*r.^2;
end
```

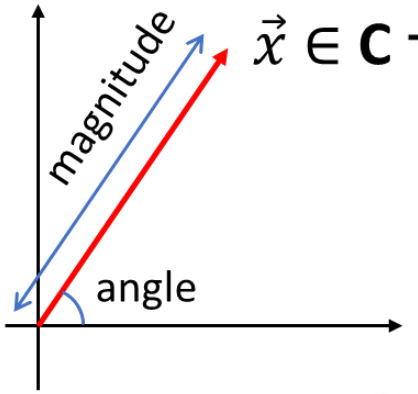


행렬을 반환하는 익명함수

```
get_mean_std = @(x) [mean(x), std(x)];
```

$$\left[\bar{x}, \sqrt{\frac{\sum|x - \bar{x}|^2}{n-1}} \right]$$

```
get_abs_angle = @(z) [abs(z), angle(z)];
```



$\vec{x} \in \mathbf{C} \rightarrow$ [magnitude, angle]

```
roll_dices = @(N) randi(6,[1, N]);
```

x N

```
assorted = @(n) {zeros(n), ones(n), magic(n), rand(n)};
```

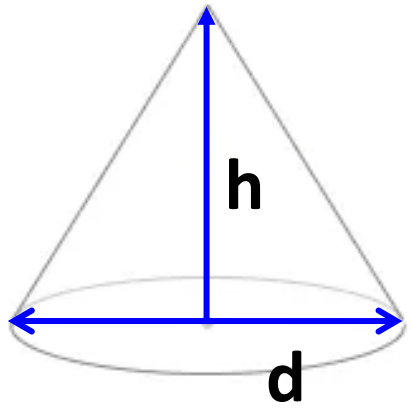
입력이 여러 개인 익명함수

함수명 = @(입력1, 입력2, ...) 표현식

```
getConeVol = @(d, h) 1/3*pi*(d/2).^2.*h;
```

getConeVol.m*

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

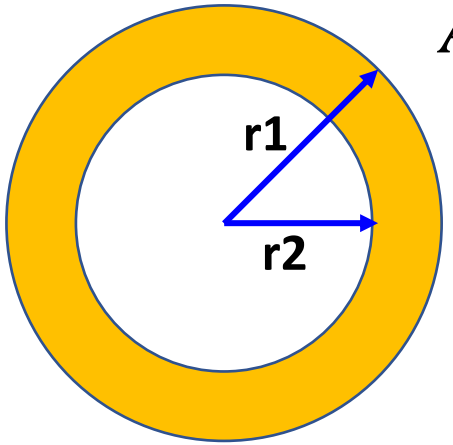


$$V = \frac{\pi}{3}\left(\frac{d}{2}\right)^2 h$$

```
getRingArea = @(r1, r2) pi*abs(r1.^2-r2.^2);
```

getRingArea.m

```
1  function A = getRingArea(r1, r2)
2
3  A = pi*abs(r1.^2-r2.^2);
4
5  end
```



$$A = \pi\left|r_1^2 - r_2^2\right|$$

입력이 여러 개인 익명함수

함수명 = @(입력1, 입력2, ...) 표현식

```
get_quadeq_disc = @(a, b, c) b^2-4*a*c;
```

get_quadeq_disc.m

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

$$ax^2 + bx + c = 0$$
$$D = b^2 - 4ac$$

```
roll_3dices = @(n1, n2, n3) [randi(n1), randi(n2), randi(n3)];
```

roll_3dices.m

```
1   function nums = roll_3dices(n1, n2, n3)
2
3 -     nums = [randi(n1), randi(n2), randi(n3)];
4
5 -   end
```

x 3

```
magics = @(n1, n2, n3) {magic(n1), magic(n2), magic(n3)};
```

출력이 없는 익명함수

함수명 = @(입력인자) 표현식

표현식의 반환값이 없는 경우

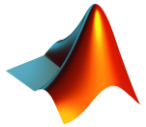
```
greeting = @(name) disp("Hello, " + name + "!");
```

```
function greeting(name)

disp("Hello, " + name + "!");

end
```

```
>> greeting = @(name) disp("Hello, " + name + "!");
>> greeting("Hong")
Hello, Hong!
>> s = greeting("Hong")
Error using disp
Too many output arguments.
Error in @(name)disp("Hello, "+name+"!")
```

```
>> greetingf = @(name) fprintf('Hello, %s!\n', name);
>> greetingf('Hong')
Hello, Hong!
>> s = greetingf('Hong');
Hello, Hong!
>> s
s =
    13
```

??



입력이 없는 익명함수

함수명 = @() 표현식

괄호는 있어야 함

```
get_G = @() 6.67408e-11;
```

```
>> get_G = @() 6.67408e-11;
>> get_G
get_G =
  function_handle with value:
    @()6.67408e-11
>> get_G()
ans =
   6.6741e-11
```

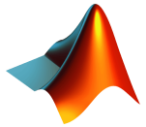
※ 이때는 호출 시 ()를 붙여야 한다.

get_G.m

```
1   function G = get_G
2
3 -     G = 6.67408e-11;
4
5 -   end
```

```
>> get_G
ans =
   6.6741e-11
>> get_G()
ans =
   6.6741e-11
```

※ 이때는 ()를 붙여도 되고 안 붙여도 되지만 함수임을 드러내기 위해 붙이는 것이 좋다.



입출력이 모두 없는 익명함수

함수명 = @() 표현식

괄호는 있어야 함

표현식의 반환값이 없는 경우

```
greeting = @() disp('Hello, world!');
```

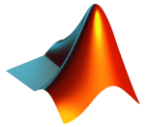
```
>> greeting = @() disp('Hello, world!');
>> greeting
greeting =
  function_handle with value:
    @()disp('Hello, world!')
>> greeting()
Hello, world!
```

※ 이때는 호출 시 ()를 붙여야 한다.

```
function greeting

disp('Hello, world!');

end
```

```
>> greeting
Hello, world!
>> greeting()
Hello, world!
```

※ 이때는 ()를 붙여도 되고 안 붙여도 되지만 함수임을 드러내기 위해 붙이는 것이 좋다.



언제 유용한가?

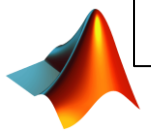
3) 명령창에서 정의할 수 있다.
: 파일로 만들지 않아도 된다.
: 에디터(m파일)에서도 할 수 있다.

- 파일로 만들 필요가 없는 간단한 함수 작성 시

```
>> roll_dices = @(N) randi(6,[1, N]);
>> roll_dices(5)
ans =
     5     1     4     4     1
>> roll_dices(3)
ans =
     6     4     5
>> roll_dices(10)
ans =
     3     5     3     3     6     3     1     6     3     2
```

* 사용자 정의 함수는 명령창에서 정의 불가능

```
roll_dices = @(N) randi(6,[1, N]);

Ndices = input('How many dices?: ');
Nrounds = input('How many rounds?: ');

results = zeros(Nrounds, Ndices);
for i=1:Nrounds
    results(i,:) = roll_dices(Ndices);
end
```



익명함수의 한계

- 한줄짜리 함수만 가능 (표현식이 1개인 경우만)

tribonacci.m

```
function tt = tribonacci(N)

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

tt = t(end);
```

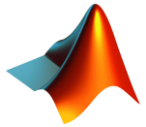
```
>> fn = @() disp('Hello,'), disp('world');
fn =
  function_handle with value:
    @()disp('Hello,')
world
```

왜 안될까?

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

boxBlur.m

```
function blurred = boxBlur(img)

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+2,j:j+2)));
    end
end
```



익명함수를 쓸 때 주의할 점

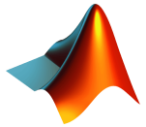
```
>> a = 1; b = 2; c = 3;
>> fn = @(x) a*x.^2 + b*x + c;
>> fn(1)
ans =
     6
>> a = 0; b = 0; c = 0;
>> fn(1)
ans =
     6
>> clear a b c
>> fn(1)
ans =
     6
```

※ a, b, c는 익명함수를 만들 때의 값으로 고정된다.

```
>> clear a b c
>> fn = @(x) a*x.^2 + b*x + c;
>> fn
fn =
  function_handle with value:
    @(x)a*x.^2+b*x+c
>> fn(1)
Undefined function or variable 'a'.
Error in @(x)a*x.^2+b*x+c
```

※ a, b, c가 없어도
- 익명함수 정의 시 오류 발생하지 않음
- 함수 호출 시 오류 발생함

입력인자가 아닌 변수는
- 익명함수 정의 전에 정의되어 있어야 하며
- 익명함수를 정의할 때의 값으로 고정된다.



익명함수를 쓸 때 주의할 점

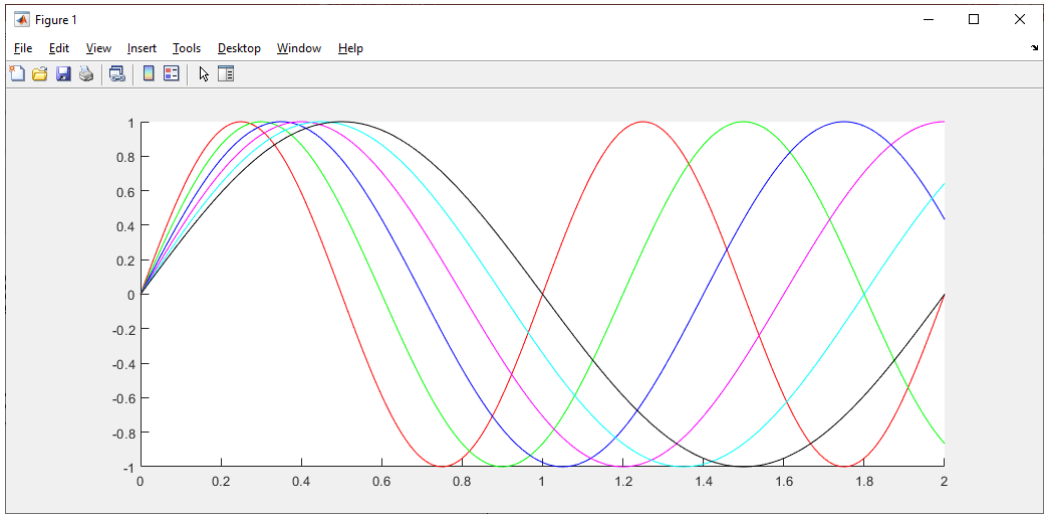
주기를 바꿔가며 sin 파형 plot

```
1     mysin = @(x) sin(2*pi*x/p);
2     x = 0:0.01:2;
3     ps = 1:0.2:2;
4
5     figure, hold on,
6     colors = 'rgbmck';
7     for i=1:length(ps)
8         p = ps(i);
9         plot(x, mysin(x), colors(i))
10    end
```

익명함수를 만들 때는 error 발생하지 않음

```
Undefined function or variable 'p'.
```

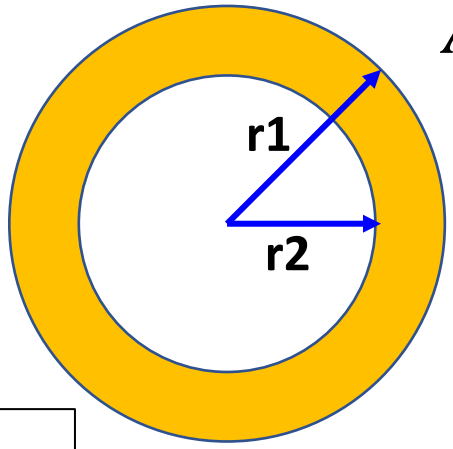
```
1     mysin = @(p,x) sin(2*pi*x/p);
2     x = 0:0.01:2;
3     ps = 1:0.2:2;
4
5     figure, hold on,
6     colors = 'rgbmck';
7     for i=1:length(ps)
8         plot(x,mysin(ps(i),x),colors(i))
9     end
```



익명함수를 쓸 때 주의할 점

- 벡터화를 해야 할까?

```
getRingArea = @(r1, r2) pi*abs(r1.^2-r2.^2);
```



$$A = \pi\left|r_1^2 - r_2^2\right|$$

```
>> getRingArea = @(r1, r2) pi*abs(r1.^2-r2.^2);
>> getRingArea([4,5,6],[1,2,3])
ans =
   47.1239   65.9734   84.8230
```

```
>> getRingArea = @(r1, r2) pi*abs(r1^2-r2^2);
>> getRingArea([4,5,6],[1,2,3])
Error using  ^  (line 51)
Incorrect dimensions for raising a matrix to a power. Check that the
matrix is square and the power is a scalar. To perform elementwise matrix
powers, use '.^'.
Error in @(r1,r2)pi*abs(r1^2-r2^2)
```

※ 행렬곱 등을 할 게 아니라면
벡터화를 하는 것이 안전하다.

함수 핸들?

1) 이렇게 생겼다.

함수명 = @(입력인자) 표현식

2) 한줄이다.

3) 명령창에서 정의할 수 있다.
: 파일로 만들지 않아도 된다.
: 에디터(m파일)에서도 할 수 있다.

4) function_handle이라는 게 생긴다.
: whos로 볼 수 있다. -> 함수? 변수?!

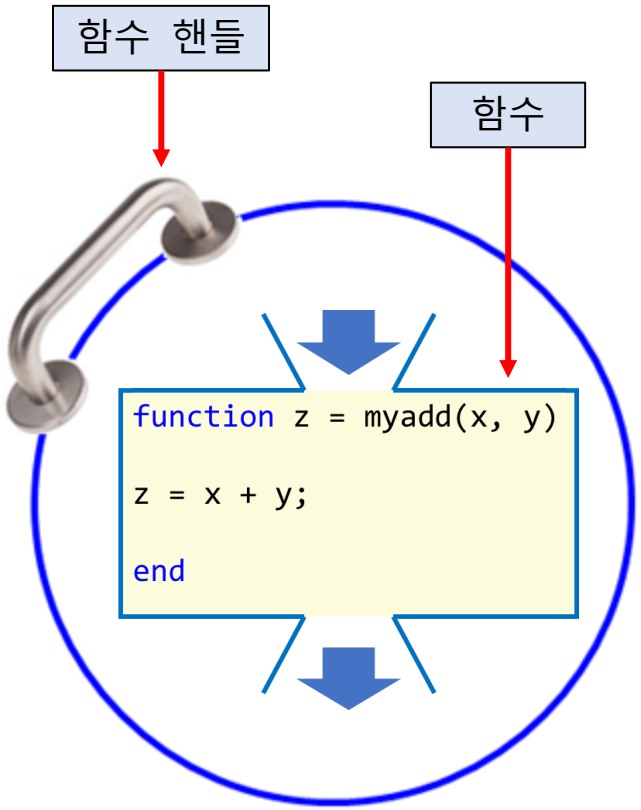
5) 함수 호출은 일반 함수와 같다.
: 호출 형태로는 사용자 정의 함수, 로컬함수, 내장함수와 구별할 수 없다.

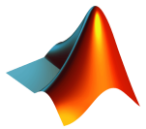
6) 반환값을 별도로 정의하지 않는다.

```
>> clear
>> plus100 = @(a) a+100;
>> doubleit = @(x) x*2;
>> squareit = @(x) x.^2;
>> fun1 = @(x) x^3 -5*x^2 + 3*x + 4;
>> plus100
plus100 =
  function_handle with value:
    @(a)a+100
>> whos
  Name          Size              Bytes  Class

  doubleit      1x1                  32  function_handle
  fun1          1x1                  32  function_handle
  plus100       1x1                  32  function_handle
  squareit      1x1                  32  function_handle

>> plus100(23)
ans =
   123
>> doubleit(100)
ans =
   200
>> squareit(5)
ans =
    25
>> fun1(2)
ans =
    -2
```

함수 핸들이란?

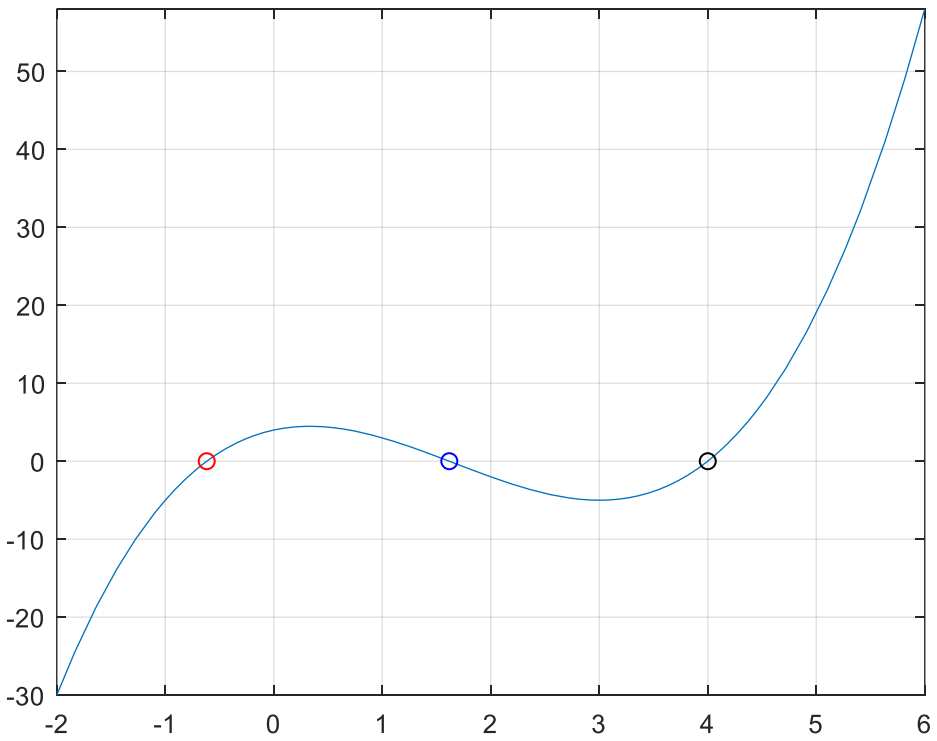
- 함수 = 모듈화된 코드
- 함수 핸들 = 함수에 손잡이(핸들)을 달아 **"자료형"**의 형태로 만든 것
  - "자료형"이므로 int8형, double형 등과 **동등한 지위**를 가짐
  - "자료형"이므로 **"변수"**로 취급할 수 있음
  - "변수"이므로 **workspace**에 생성됨
  - "변수"이므로 함수에 **인자**로 전달하거나, 전달받을 수 있음
  - "파일"이 아니므로 searchpath와는 무관함



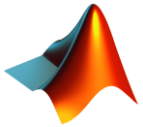


함수 핸들이 왜 필요할까?

```
myfun = @(x) x.^3 -5*x.^2 + 3*x + 4;
fplot(myfun, [-2, 6])

x1 = fzero(myfun,-1);
hold on,
plot(x1, 0, 'ro')

x2 = fzero(myfun, [1,2]);
plot(x2, 0, 'bo')

x3 = fzero(myfun, [3,5]);
plot(x3, 0, 'ko')
```



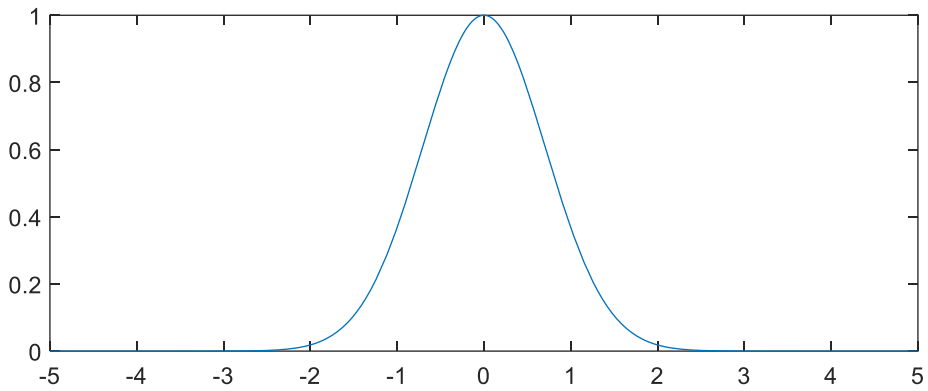
* 함수 함수 (function functions)
  - 함수 핸들을 입력인자로 받는 함수



함수 핸들이 왜 필요할까?

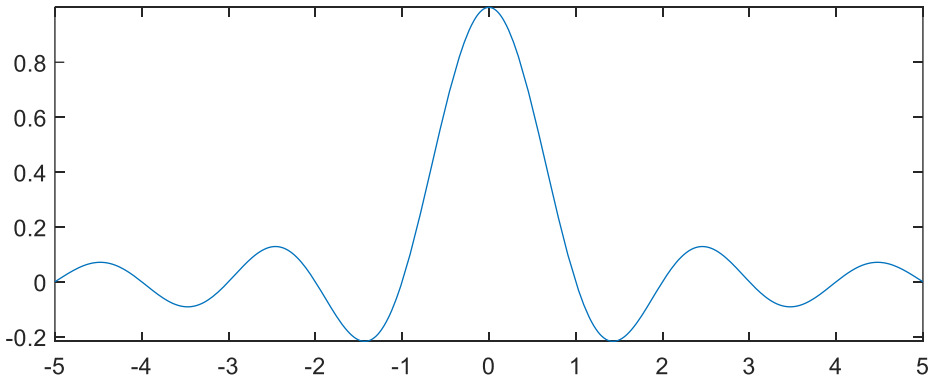
```
myfun = @(x) 1./exp(x.^2);
figure, fplot(myfun);
integral(myfun, -1, 1)
```

$$\int_{-1}^{1} \frac{1}{e^{x^2}} \, dx$$




```
figure, fplot(@sinc);
integral(@sinc, 0, 1)
```

$$\int_{0}^{1} \frac{\sin(\pi x)}{\pi x} \, dx$$



https://www.integral-calculator.com/

함수 핸들이 왜 필요할까?

```
C = ["Michael"; "Kate"; "Dylan"];
greeting = @(name) "Hello, " + name + "!";
arrayfun(greeting, C)
```

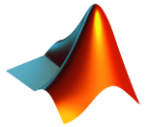
```
ans =
  3x1 string array
    "Hello, Michael!"
    "Hello, Kate!"
    "Hello, Dylan!"
```

```
A = {magic(3), magic(4), magic(5)};
cellfun(@(A) mean(A,'all'), A)
```

```
ans =
    5.0000    8.5000   13.0000
```

```
doGrading = @get_grade; % function alias
doGrading(85, 'ABCDF', [60 70 80 90])
% same as get_grade(...)
```

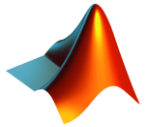
```
function grade = get_grade(score, grades, scorecuts)

position = sum(score>=scorecuts);
N = length(grades);
grade = grades(N-position);

end
```



셀 배열엔 무엇이든 들어갈 수 있다!

```
roll_cubeDices = @(N) randi(6,[1, N]);
roll_3dices = @(n1, n2, n3) [randi(n1), randi(n2), randi(n3)];
choose_dices = @(n) randsample([4,6,8,12,20], n);

dice_funcs = {choose_dices, roll_cubeDices, roll_3dices};

% choose random 3 dices
Nfaces = dice_funcs{1}(3);
% roll chosen 3 dices
nums3 = dice_funcs{3}(Nfaces(1), Nfaces(2), Nfaces(3));
```

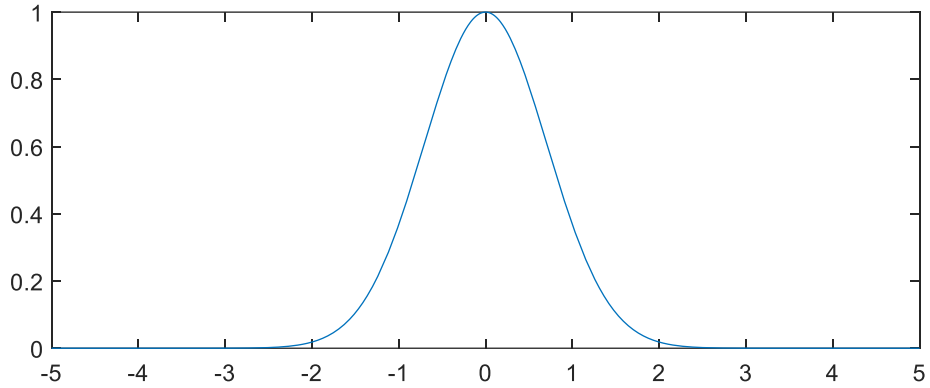
- 함수 핸들을 원소로 갖는 셀 배열
- 셀 배열도 변수이므로 함수의 인자가 될 수 있다.



함수 핸들을 만드는 방법

- 익명함수 – 그 자체가 이미 함수 핸들

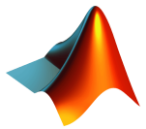
```
myfun = @(x) 1./exp(x.^2);
figure, fplot(myfun);
integral(myfun, -1, 1)
```



- 사용자 정의 함수 - @함수명
  - 로컬함수, 내장함수 모두 가능

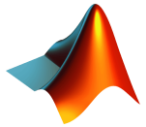
```
+33  getConeVol.m*
1    function V = getConeVol(diameter, height)
2
3 –      V = 1/3*pi*(diameter/2).^2.*height;
4
5    end
```

```
figure,
fsurf(@getConeVol, [1, 3, 1, 5])
```



일반함수/로컬함수 vs 함수핸들/익명함수

`function` [출력인자] = 함수명(입력인자)	정의 방법	(함수명) = @(입력인자) 표현식;
파일	형태	변수
디스크 드라이브	위치	workspace
메인: searchpath에 파일이 있어야 함 로컬: m파일 내에 정의되어 있어야 함	호출하려면	workspace에 함수핸들이 있어야 함
없음	명령창에서 정의할 수	있음

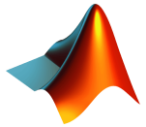


익명함수 사용 시 주의할 점

- 모든 함수는 정의에서 시작한다.
  - 사용자 정의함수의 정의
    - 메인 함수: 파일의 존재 유무 (searchpath 상에서)
    - 로컬 함수: 파일 내에서의 함수 정의 유무
  - 익명함수의 정의
    - workspace 상에서의 함수 핸들의 존재 유무
    - 익명함수 = 함수 핸들 = 변수의 한 종류

- 익명함수와 사용자 정의함수(m파일) 이름이 겹치면?
  - 익명함수 먼저 (=변수 먼저)

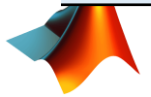
- 익명함수와 로컬 함수의 이름이 겹치면?
  - 익명함수를 코드에서 정의하는 경우 -> 에러
  - 익명함수가 workspace에 있는 경우 -> 로컬함수 먼저



```
greeting = @() disp('Hello! (anony)');

greeting();

function greeting

disp('Hello! (local)');

end
```

함수 총정리

	사용자 정의 함수 (메인함수)	로컬함수	익명함수 (함수 핸들)
함수 작성 위치	function 파일의 맨 위	- function 파일의 메인함수 아래 - 스크립트에 정의된 모든 함수	아무데서나
형태	파일		변수
정의 방법	`function` [출력인자] = 함수명(입력인자)		(함수명) = @(입력인자) 표현식;
함수 존재 위치	디스크 드라이브		workspace
명령창에 정의할 수	없음		있음
호출하려면	searchpath에 파일이 있어야 함	m파일 내에 정의되어 있어야 함	workspace에 함수핸들이 있어야 함



Q&A