MATLAB 프로그래밍

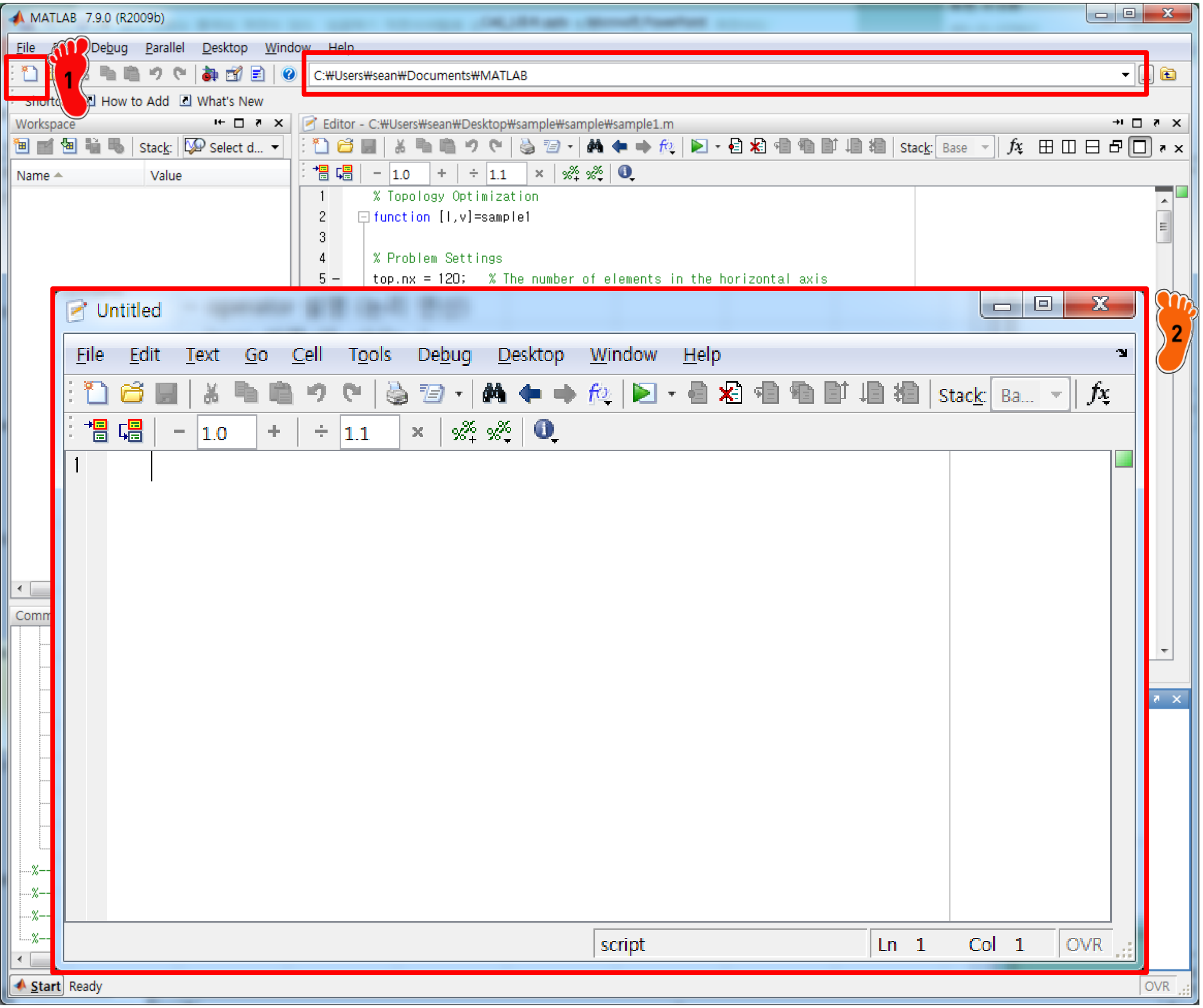
Computational Design Laboratory
Department of Automotive Engineering
Hanyang University, Seoul, Korea

CONTENTS

- M-files
- Input-output
- Structured programming
- Passing functions to m-files
- Case study
- Assignment

- **M-files**
  - ✓ **Script file**
  - ✓ **Function file**
  - ✓ **Subfunction**

SCRIPT FILE



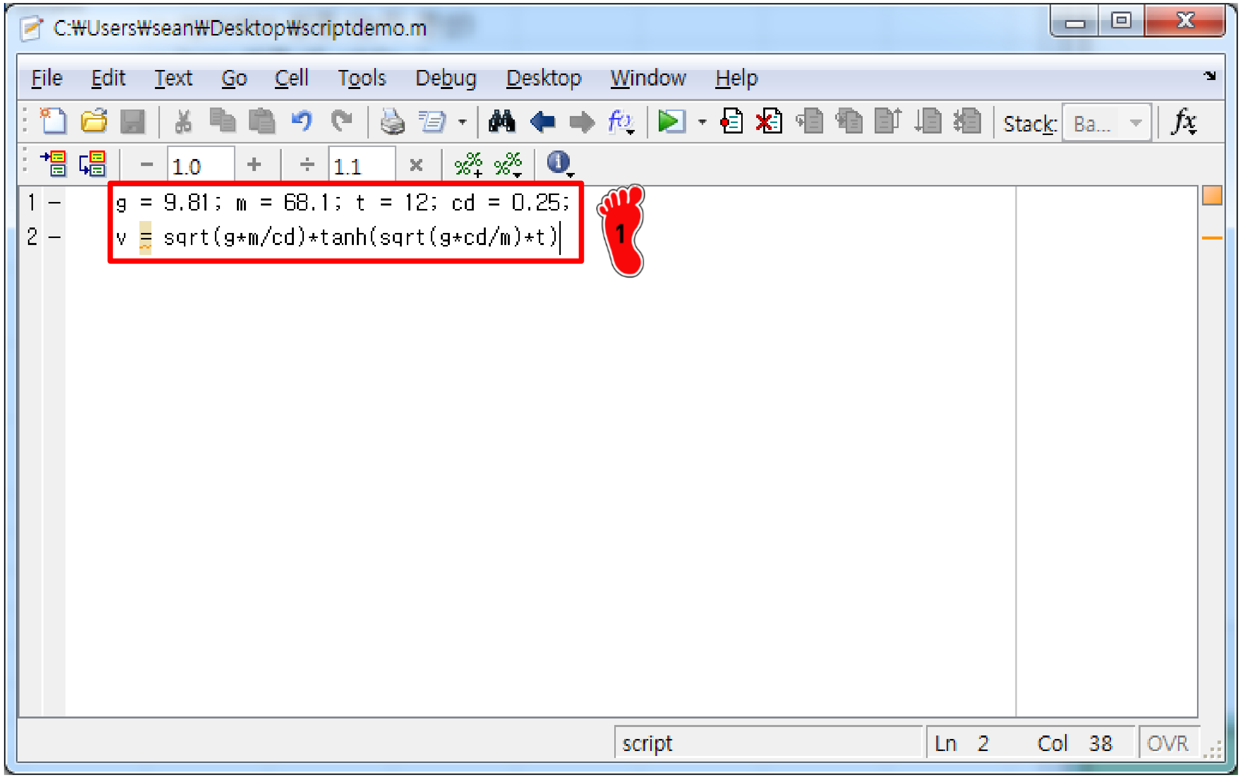
1 New m-file 을 클릭하면 script 를 작성할 수 있는 빈 창이 생성됨

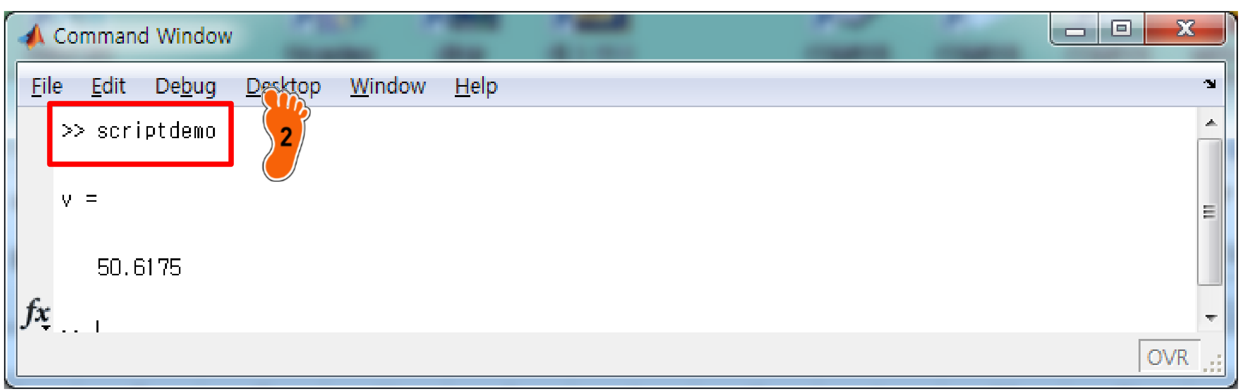
2 이 m-file 을 저장하면 *.m 파일로 저장되고 현재 매틀랩에서 설정되어있는 경로는 상단에 표시되어 있음

따라서 저장된 경로와 매틀랩에서 설정된 경로가 일치해야 script file 을 실행시킬 수 있음

혹은 script file 을 저장 후 F5 키를 이용하여 실행시키면 자동으로 매틀랩 설정 경로가 script file 이 저장된 경로로 변경

EXAMPLE

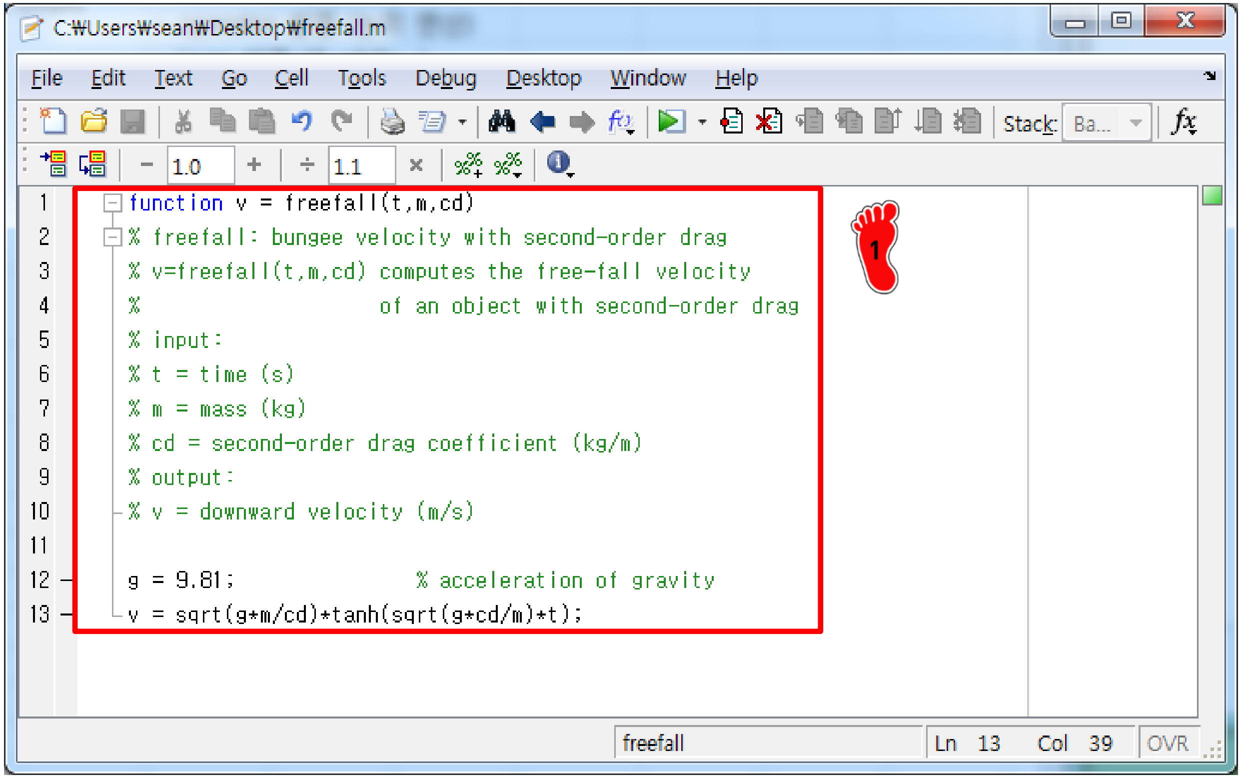


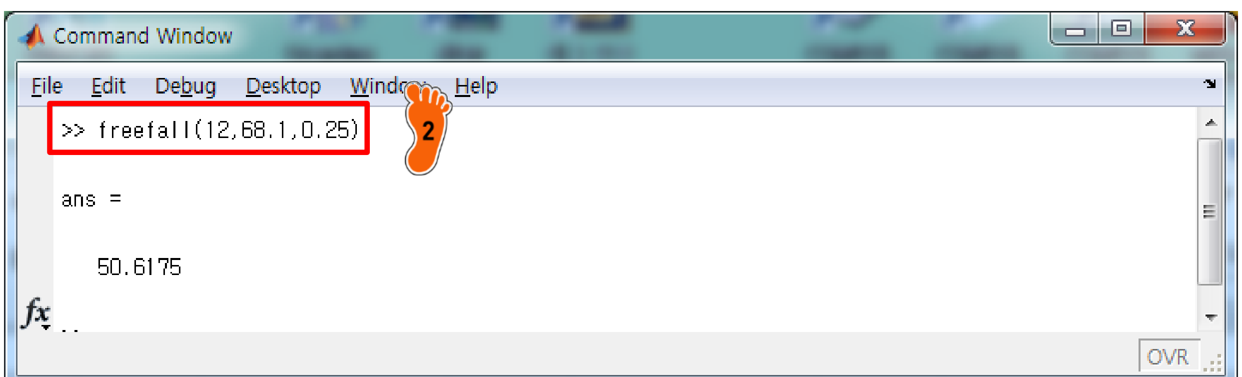


1 전 시간에 배운 커멘트 창에 입력하는 방식으로 script 를 작성 후 scriptdemo 라는 파일 이름으로 저장

2 command window 에 scriptdemo 라고 저장된 파일 이름을 입력할 경우 결과값을 출력

FUNCTION FILE





1 function outvar = funcname(arglist)

outvar = value;

함수의 기본 형태

주의할 점
함수로 정의한outvar 가 함수 내부에 동일한 이름으로 존재해야하고 m-file 을 funcname 으로 저장

2 Command window에서 funcname(arglist) 를 입력하여 결과값 출력

FUNCTION FILE

C:\Users\sean\Desktop\freefall.m

```matlab
function v = freefall(t,m,cd)
% freefall: bungee velocity with second-order drag
% v=freefall(t,m,cd) computes the free-fall velocity
%                     of an object with second-order drag
% input:
% t = time (s)
% m = mass (kg)
% cd = second-order drag coefficient (kg/m)
% output:
% v = downward velocity (m/s)

g = 9.81;               % acceleration of gravity
v = sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t);
```

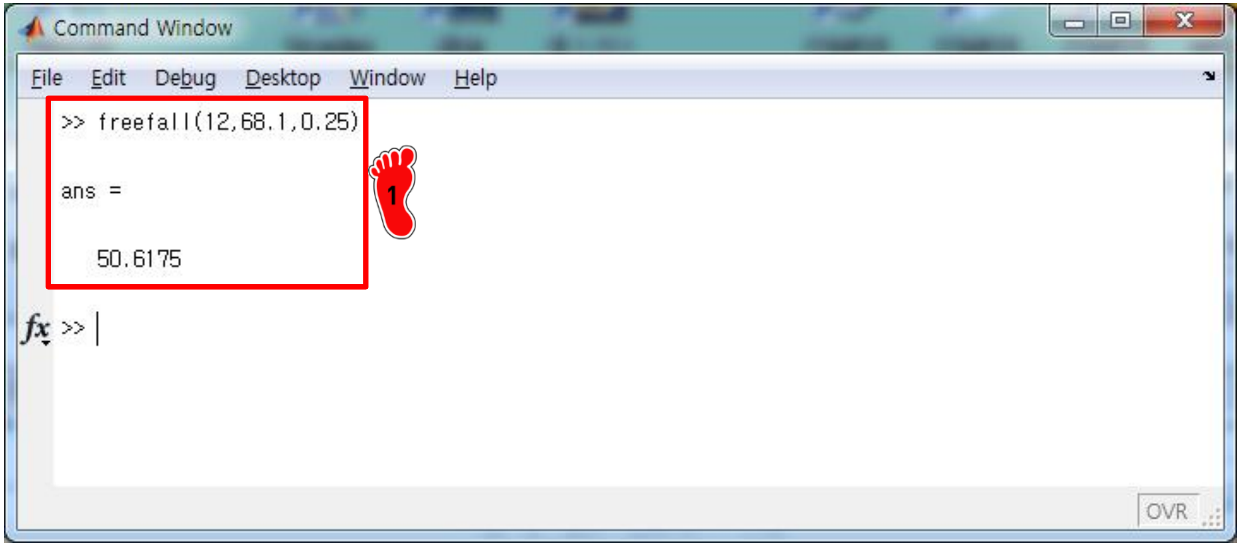
freefall Ln 13 Col 39 OVR

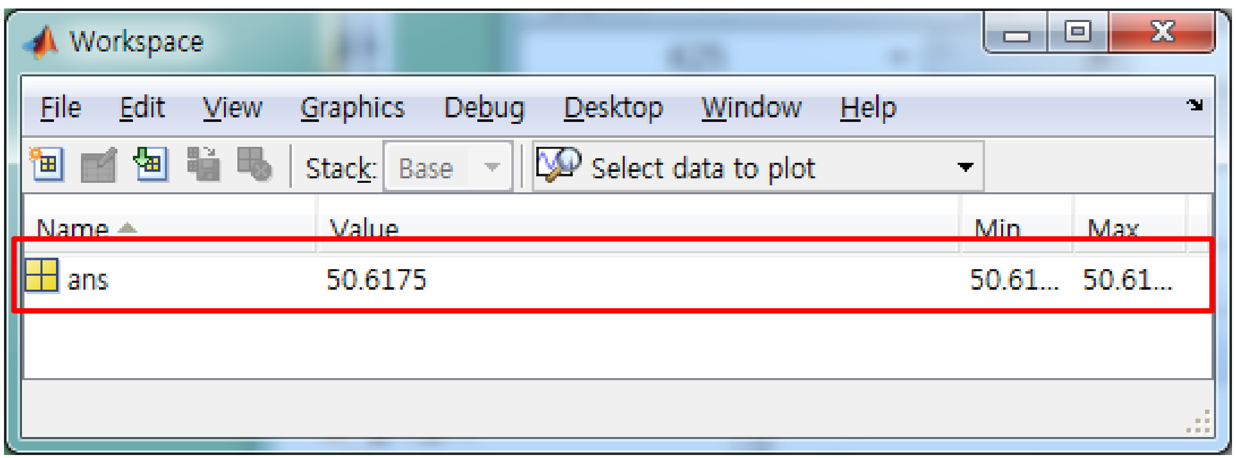
```
>> help freefall
  freefall: bungee velocity with second-order drag
  v=freefall(t,m,cd) computes the free-fall velocity
                     of an object with second-order drag
  input:
  t = time (s)
  m = mass (kg)
  cd = second-order drag coefficient (kg/m)
  output:
  v = downward velocity (m/s)
```

Command window에 help freefall 을 입력할 경우 script 창에서 주석으로 입력했던 내용이 출력

주석 처리는 % 기호로 하며 script 창에서는 초록색으로 표시됨

FUNCTION FILE

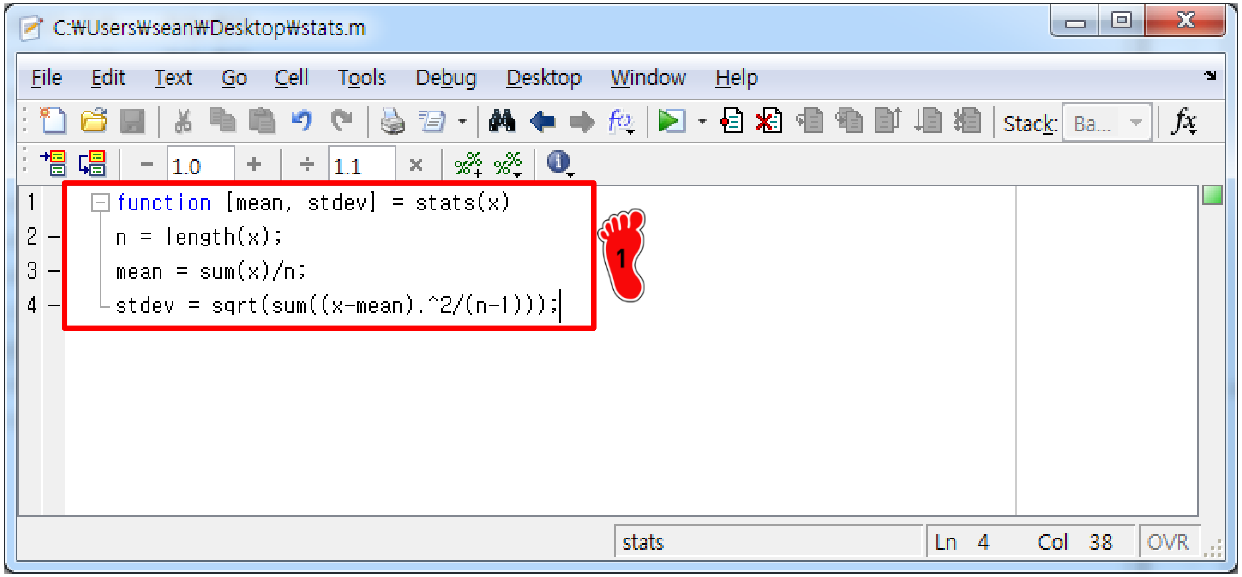




1 함수에 적용된 argument 들은 workspace 에 저장되지 않음

따라서 예제와 같이 t, m, cd 는 v 를 계산하기 위해 잠시 사용될 뿐, 최종적으로 저장되지 않음

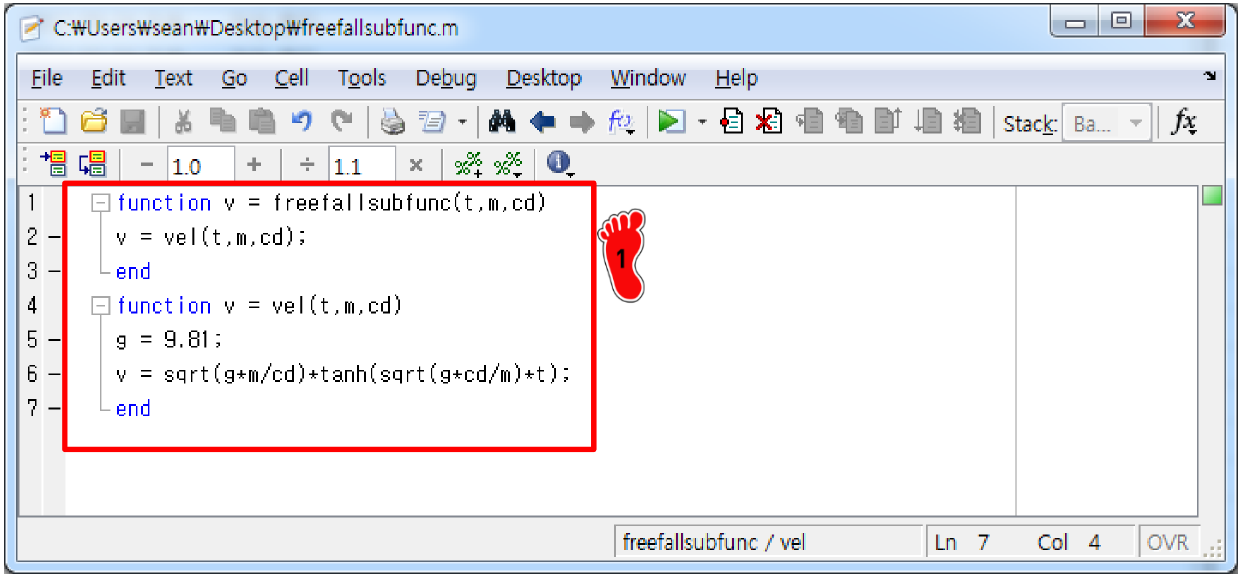
FUNCTION FILE



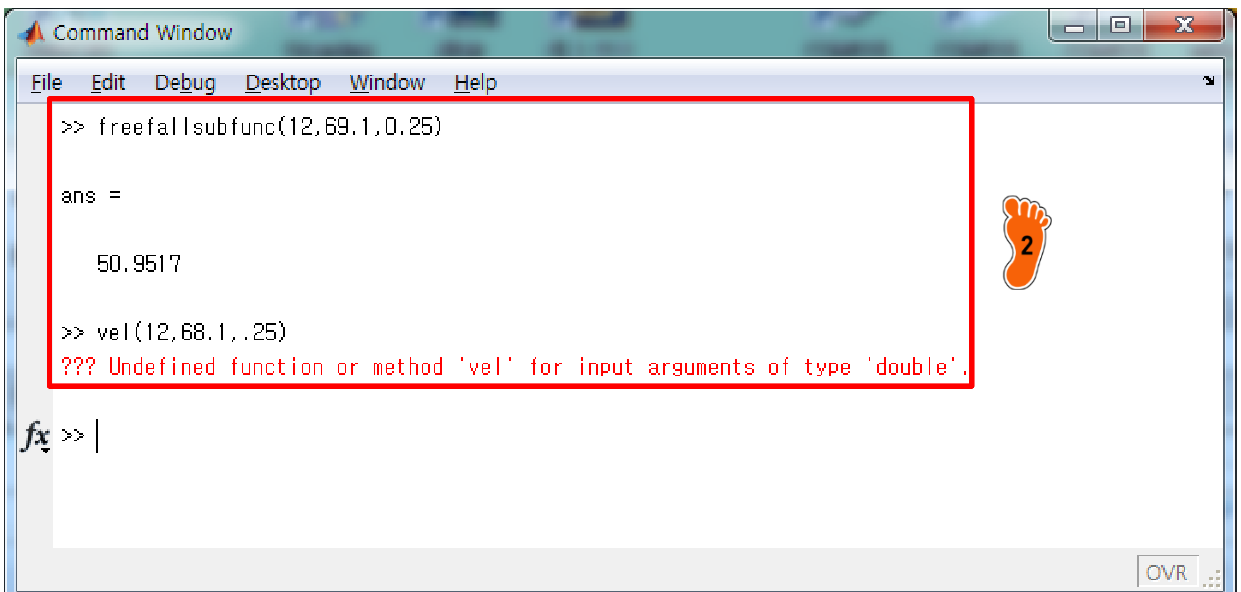
1 평균과 분산을 계산하는 함수 예제

여기서 outvar 이 한 개가 아닌 여러 개의 원소를 갖는 벡터로 지정할 수 있음

SUBFUNCTION



1



2

1 함수 안에 또 다른 함수를 입력하여 적용 가능

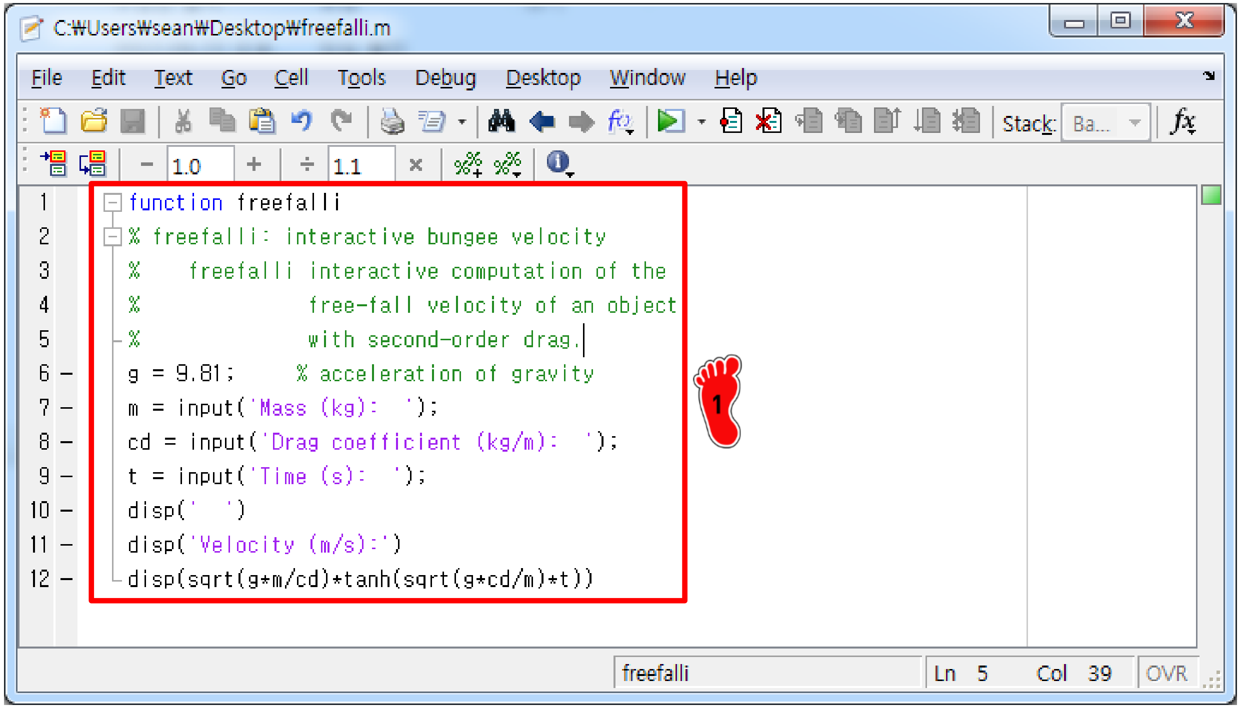
이 때는 end를 이용하여 함수가 끝난 것을 표시

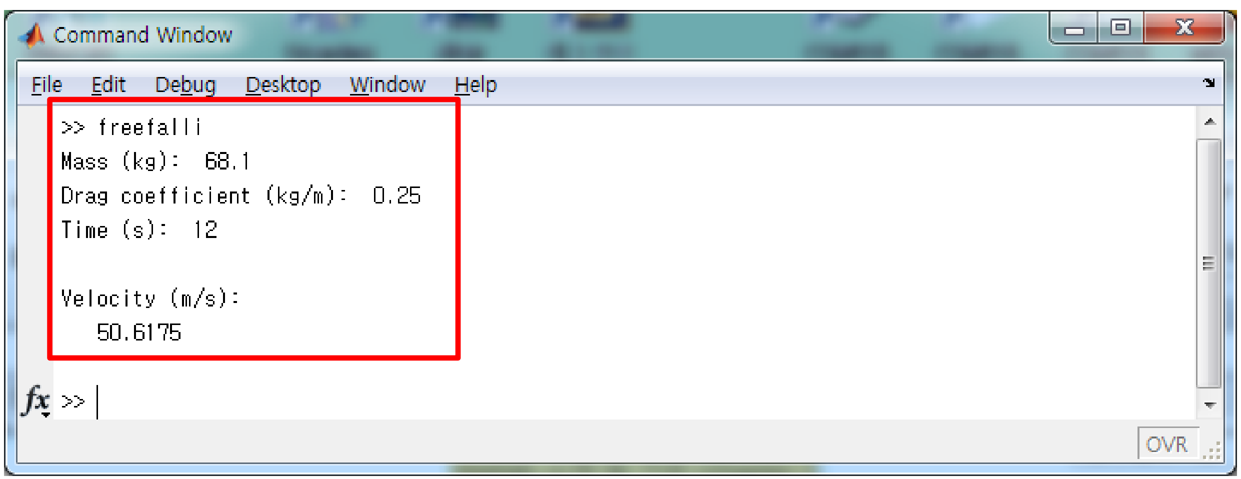
m-file 명은 처음 정의한 함수 이름으로 저장

2 vel 함수는 freefallsubfunc 안에 있는 함수로 인식할 수 없음

- **Input-output**
  - ✓ **Input function**
  - ✓ **Fprint function**

INPUT FUNCTION



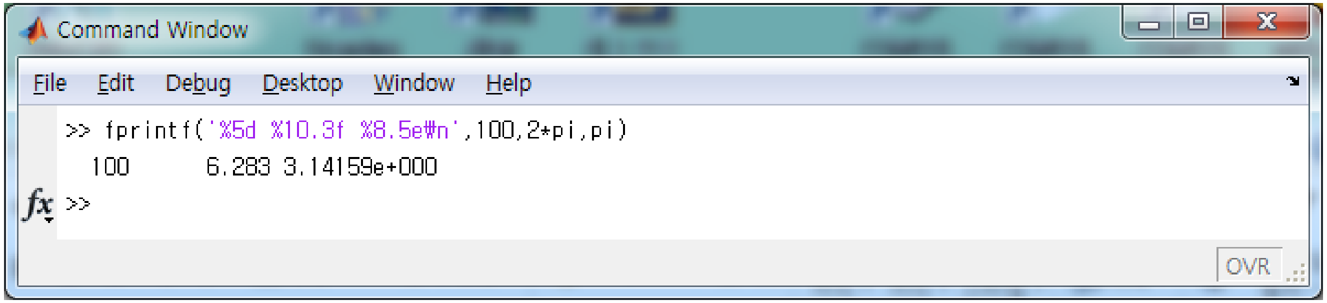


함수에 arguments 가 없어도 다른 방식으로 arguments 를 입력 가능

input 함수를 이용하여 직접 arguments 를 입력 가능

이 경우도 입력받은 변수는 workspace 에 저장되지 않음

FPRINT FUNCTION



1 fprint 함수는 command 창에 값을 출력

다양한 format 을 이용하여 출력하는 형식을 변경 가능

Format Code	Description
%d	Interger format
%e	Scientific format with lowercase e
%E	Scientific format with uppercase E
%f	Decimal format
%g	The more compact of %e or %f
Control Code	Description
\n	Start new line
\t	Tab

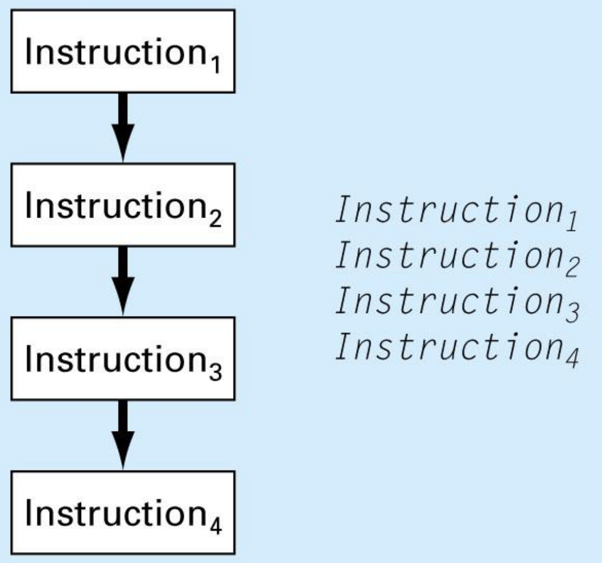
- **Structured programming**
  - ✓ **Algorithm**
  - ✓ **If structure**
  - ✓ **Switch structure**
  - ✓ **For ... end structure**
  - ✓ **While structure**
  - ✓ **Examples**

ALGORITHM

- **Benefits to writing organized, well-structured code**
  - ✓ Shorter time to develop, test, and update
- **Three fundamental control structures**
  - ✓ Sequence, selection and repetition
- **Flowchart**
  - ✓ Visual or graphical representation of an algorithm
- **Pseudocode**
  - ✓ Code-like statements in place of the graphical symbols of the flowchart
  - ✓ Bridges the gap between flowcharts and computer code

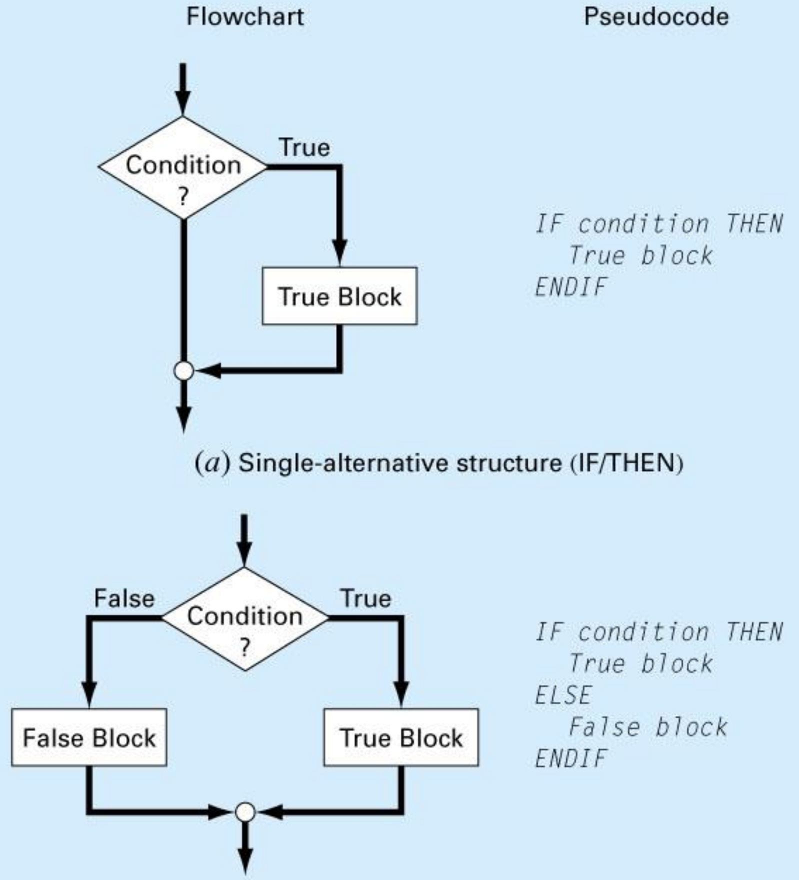
LOGICAL REPRESENTATION (1)

- **Sequence**



(*a*) Flowchart (*b*) Pseudocode

- **Selection (1)**

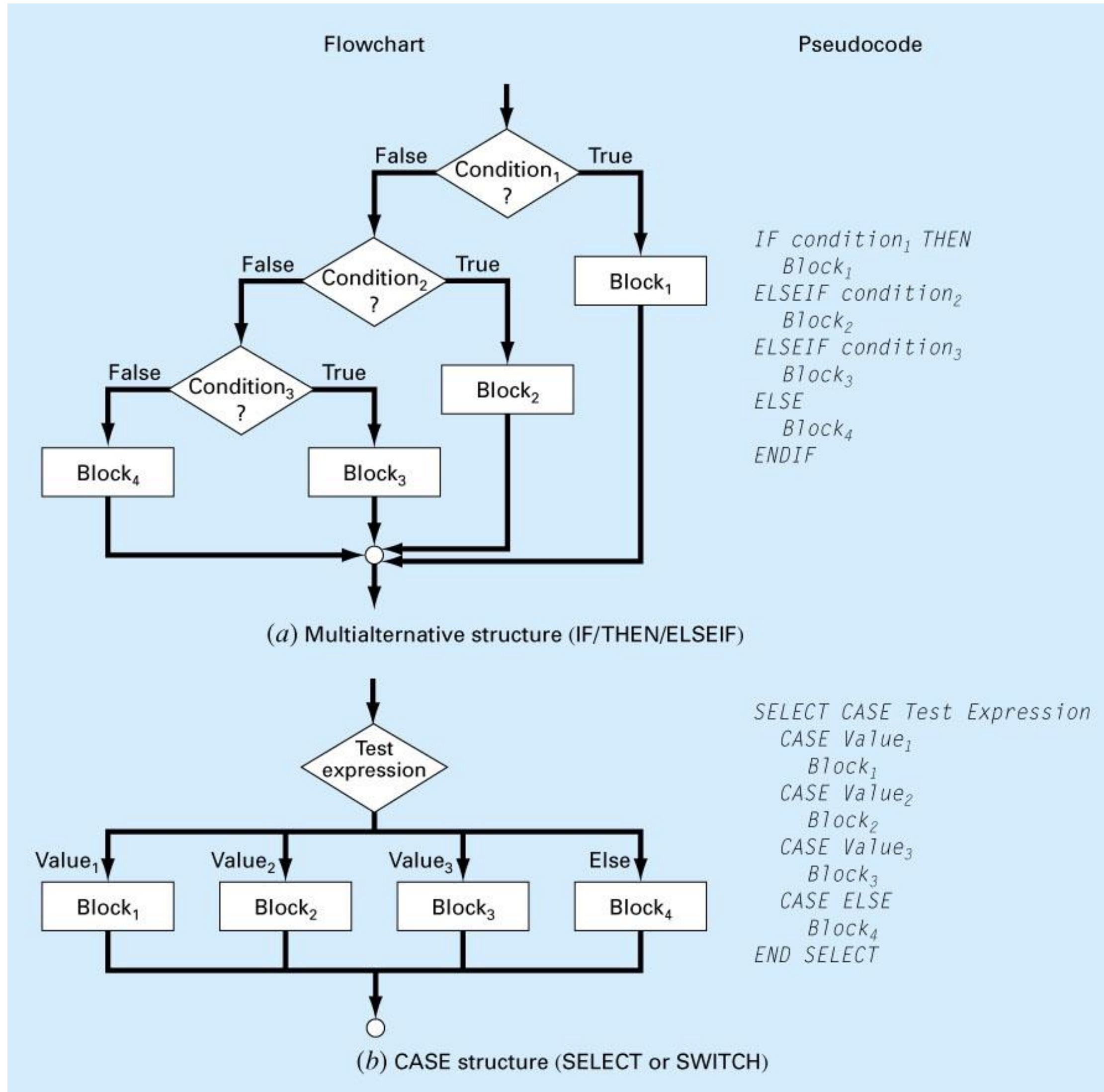


(*a*) Single-alternative structure (IF/THEN)

(*b*) Double-alternative structure (IF/THEN/ELSE)

LOGICAL REPRESENTATION (2)

- **Selection (2)**



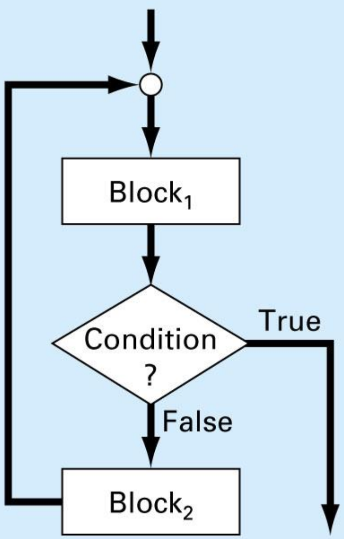
(*a*) Multialternative structure (IF/THEN/ELSEIF)

(*b*) CASE structure (SELECT or SWITCH)

LOGICAL REPRESENTATION (3)

- **Repetition**

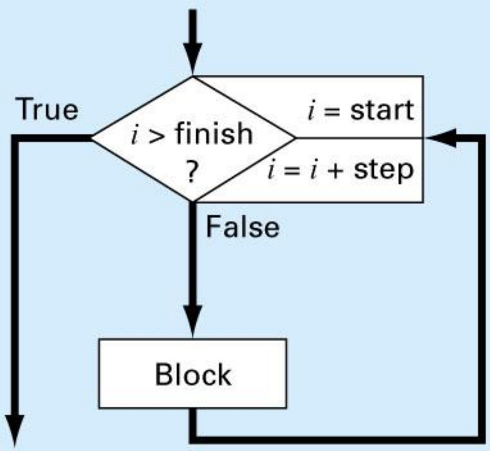
Flowchart



Pseudocode

```
DO
  Block1
  IF condition EXIT
  Block2
ENDDO
```

Flowchart

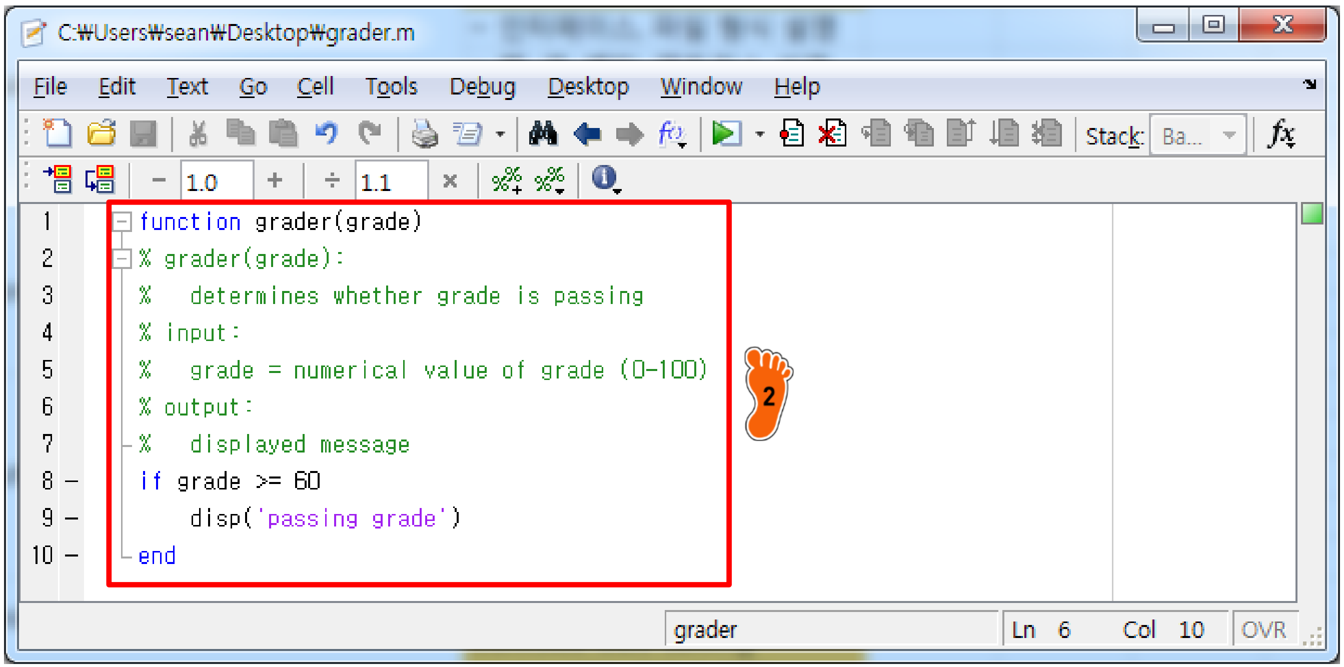


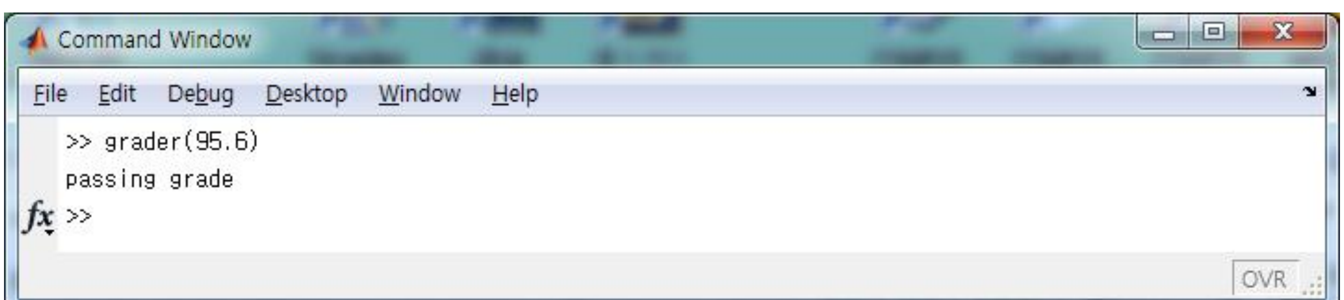
Pseudocode

```
DOFOR i = start, finish, step
  Block
ENDDO
```

IF STRUCTURE

```
if condition
    statements
end
```





1 if 문의 문법

condition: relational operator 를 이용하여 조건을 입력

statements: condition 이 참일 경우 실행

2 성적이 60 점 이상일 경우 ‘passing grade’ 문구를 출력하는 예제

RELATIONAL OPERATORS

논리 연산의 operator 는 다음의 표에 정리 되어 있음

Example	Operator	Relationship
x == 0	==	Equal
unit ~= ‘m’	~=	Not equal
a < 0	<	Less than
s > t	>	Greater than
3.9 <= a/3	<=	Less than or equal to
r >= 0	>=	Greater than or equal to

LOGICAL CONDITIONS

~ (Not): Used to perform logical negation on an expression
~expression

& (And): Used to perform a logical conjuction on two expressions
$expression_1$ & $expression_2$

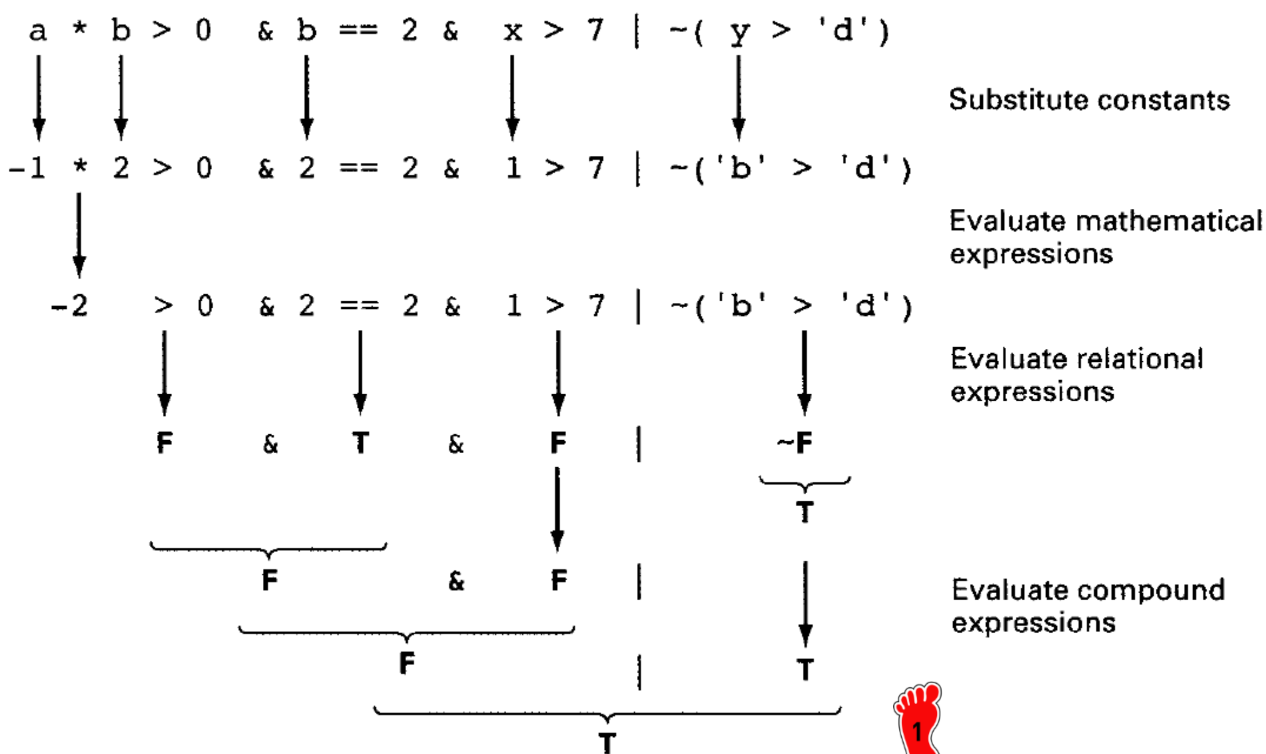
| (Or): Used to perform a logical disjunction on two expressions
$expression_1$ | $expression_2$

x	y	~x	x&y	x|y
T	T	F	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	F

1 논리 조건과 연산기호의 구체적인 예는 다음의 표에 정리

LOGICAL CONDITIONS

$$a = -1,\ b = 2,\ x = 1,\ \text{and}\ y = 'b'$$



논리 연산이 매틀랩 안에서 어떻게 수행이 되는지 알 수 있는 예제

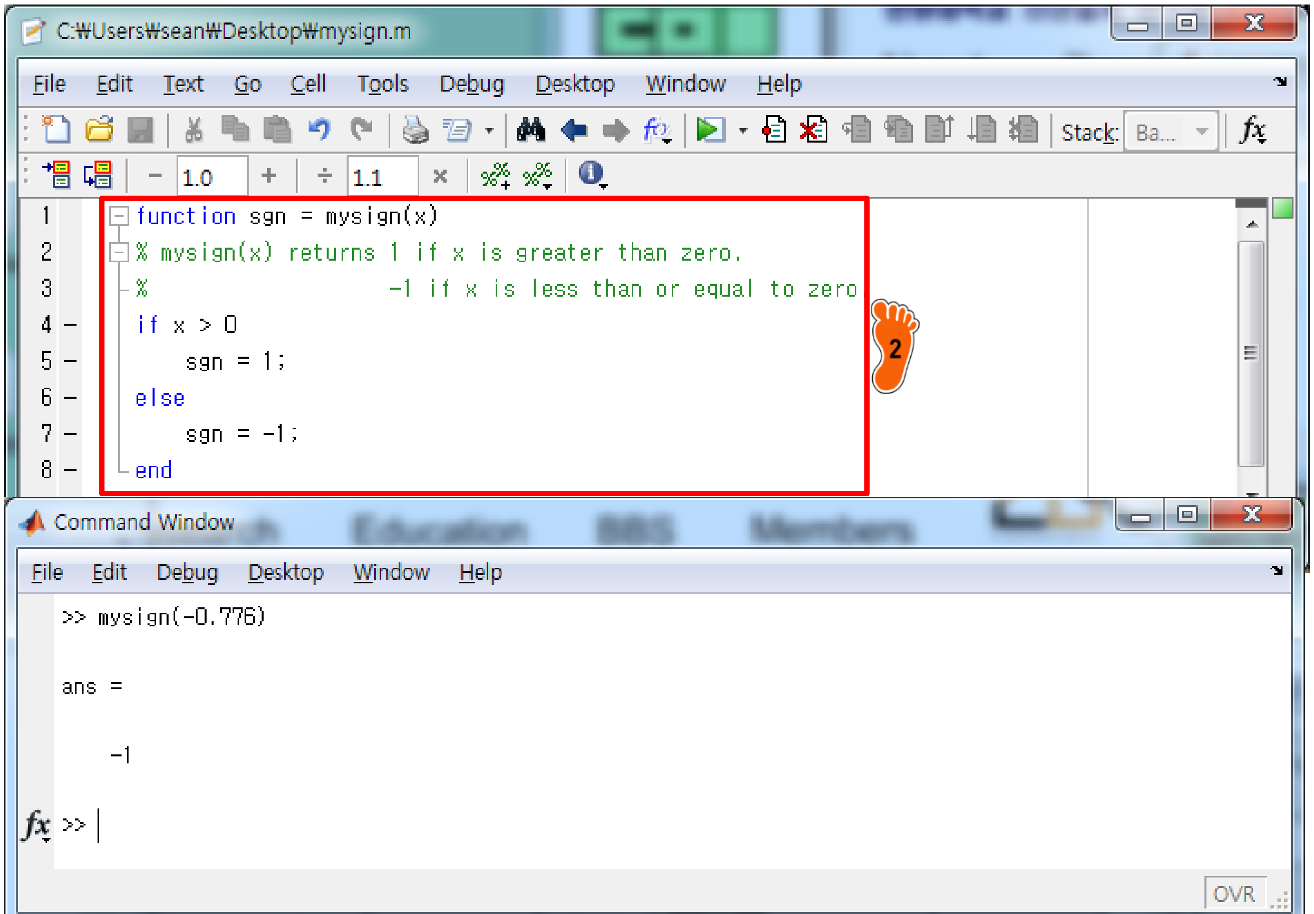
IF... ELSE STRUCTURE

```
if condition
   statements_1
else
   statements_2
end
```

1 if... elseif 문의 문법

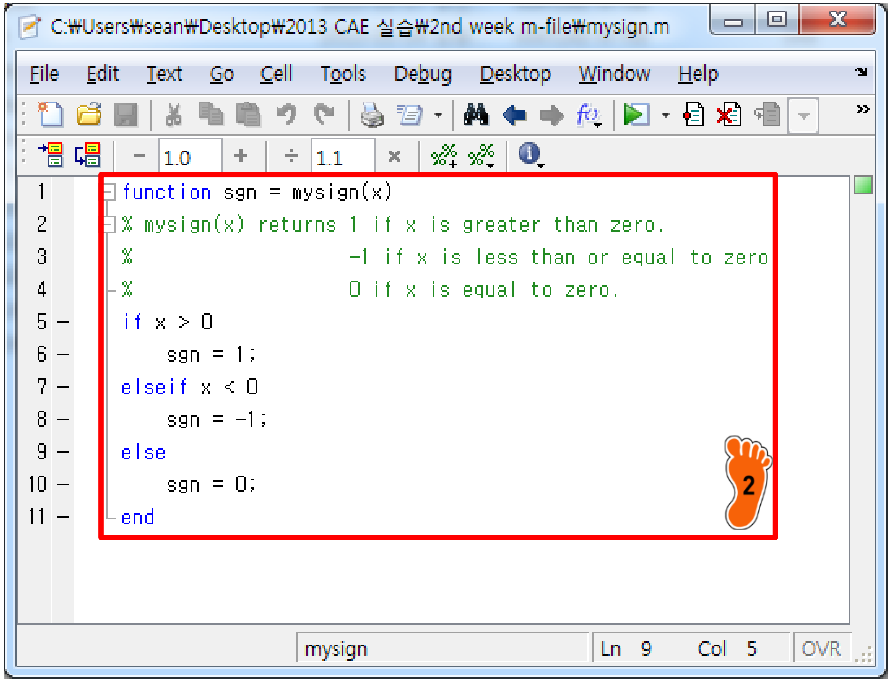
2 개의 statements 를 이용 가능

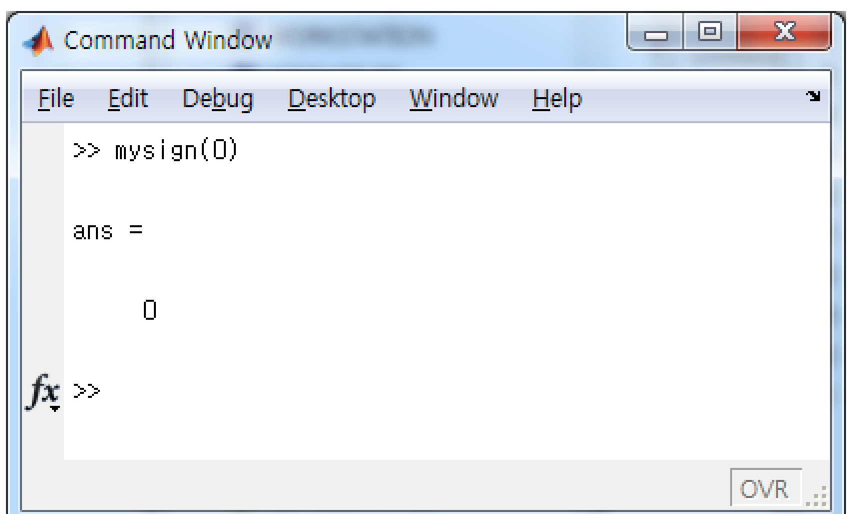
2 입력받은 값의 부호를 판단하는 예제



IF… ELSEIF STRUCTURE

if $condition_1$
  $statements_1$
elseif $condition_2$
  $statements_2$
.
.
.
else
  $statements_{else}$
end





1 if… elseif 문의 문법

여러개의 condition 을 입력 가능

else 명령어를 삽입 가능

2 입력받은 값의 부호를 판단하는 예제
값이 0 일 경우 부호가 아닌 0 을 return

SWITCH STRUCTURE

```
switch testexpression
  case value1
    statements1
  case value2
    statements2
.
.
.
  otherwise
statementsotherwise
end
```

C:\Users\sean\Desktop\Untitled7.m

```matlab
clc; clear all;
grade = 'B';
switch grade
    case 'A'
        disp('Excellent')
    case 'B'
        disp('Good')
    case 'C'
        disp('Mediocre')
    case 'D'
        disp('Whoops')
    case 'F'
        disp('Would like fries with your order?')
    otherwise
        disp('Huh!')
end
```

Command Window

```
Good
>>
```

1. switch 문의 문법

testexpression 을 입력받아 case 를 구분

2. 입력받은 성적을 구분하는 예제

SWITCH STRUCTURE

C:\Users\sean\Desktop\freefall2.m

```
1  function v = freefall2(t,m,cd)
2  % freefall2: bungee velocity with second-order drag
3  %   v=freefall2(t,m,cd) computes the free-fall velocity
4  %                       of an object with second-order drag.
5  %  input:
6  %    t = time (s)
7  %    m = mass (kg)
8  %    cd = drage coefficient (default = 0.27 kg/m)
9  %  output:
10 %    v = downward velocity (m/s)
11 switch nargin
12     case 0
13         error('Must enter time and mass')
14     case 1
15         error('Must enter mass')
16     case 2
17         cd = 0.27;
18 end
19 g = 9.81;        % accerleration of gravity
20 v = sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t);
```

freefall2 Ln 18 Col 4 OVR

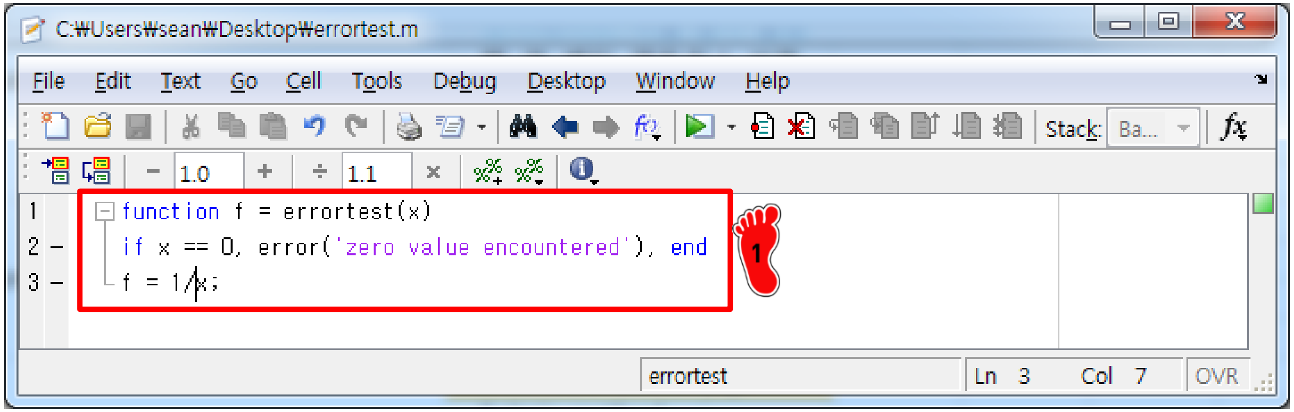
1 nargin 의 의미는 number of arguments in 의 약자로 입력되는 변수의 개수를 의미함

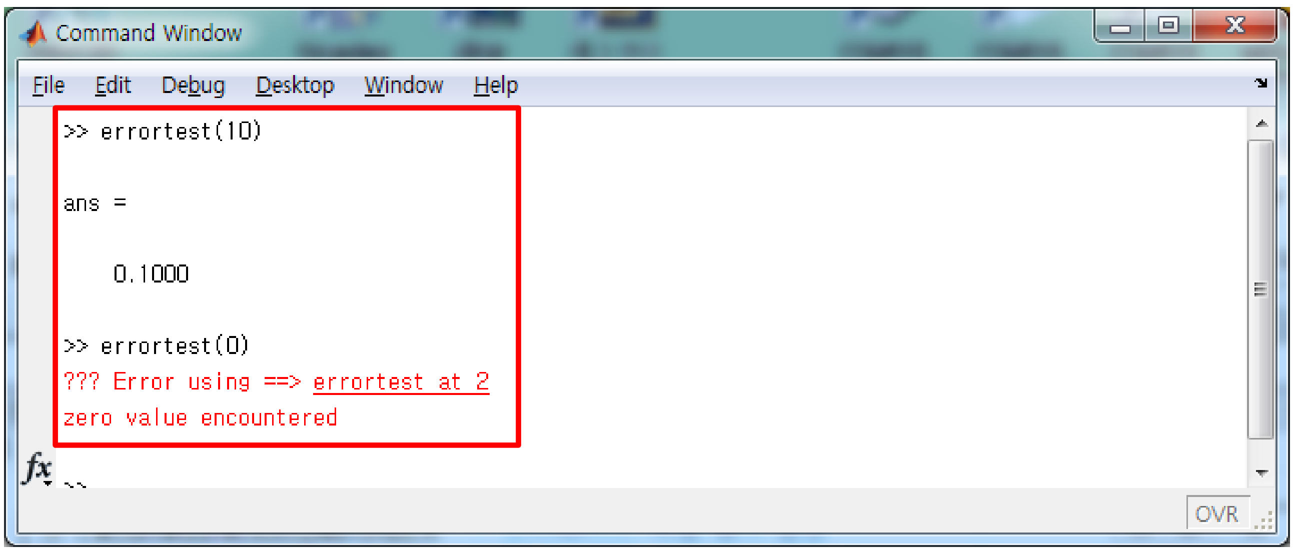
입력되는 변수의 개수에 따라서 결과값 출력하는 예제

Command Window

```
>> freefall2(12,68.1,0.25)

ans =

   50.6175

>> freefall2(12,68.1)

ans =

   48.8747

>> freefall2(12)
??? Error using ==> freefall2 at 15
Must enter mass

>> freefall2()
??? Error using ==> freefall2 at 13
Must enter time and mass

>>
```

ERROR FUNCTION





1 error 함수는 원하는 값을 입력받지 못할 경우 혹은 프로그래밍 상에서 멈춰야 할 경우 에러 메시지를 출력하면서 프로그램을 멈추는 함수

FOR STRUCTURE

```
for index = start:step:finish
    statements
end
```

1 for 문의 문법

콜론 (:) operator 를 이용하여 index 를 결정

2 factorial 을 계산하는 예제

C:₩Users₩sean₩Desktop₩factor.m

```matlab
function fout = factor(n)
% factor(n) :
%    Computes the product of all the integers from 1 to n.
x = 1;
for i = 1:n
    x = x*i;
end
fout = x;
end
```

Command Window

```
>> factor(5)

ans =

   120

>> factor(0)

ans =

     1

>>
```

Factorial

$0! = 1$

$1! = 1$

$2! = 1 \times 2 = 2$

$3! = 1 \times 2 \times 3 = 6$

$4! = 1 \times 2 \times 3 \times 4 = 24$

$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$

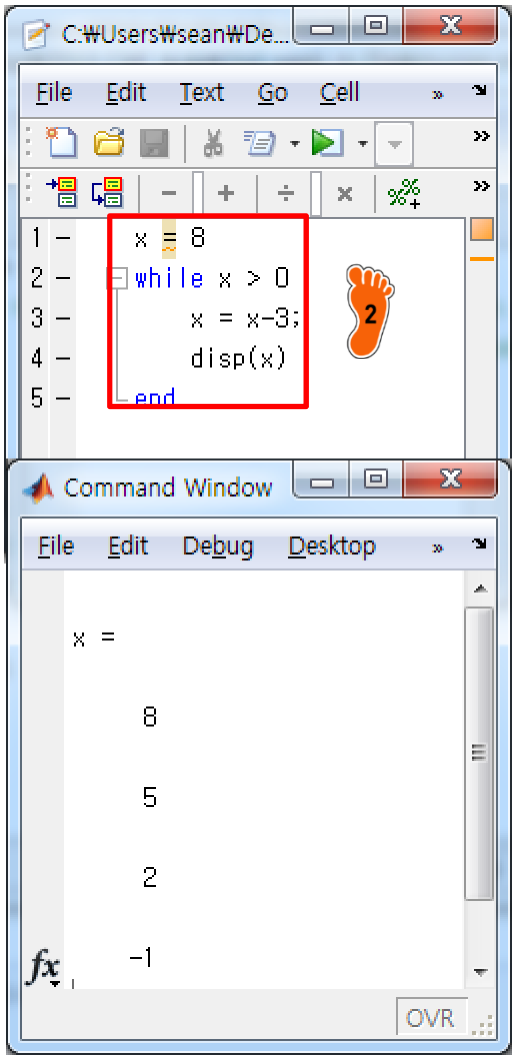
.

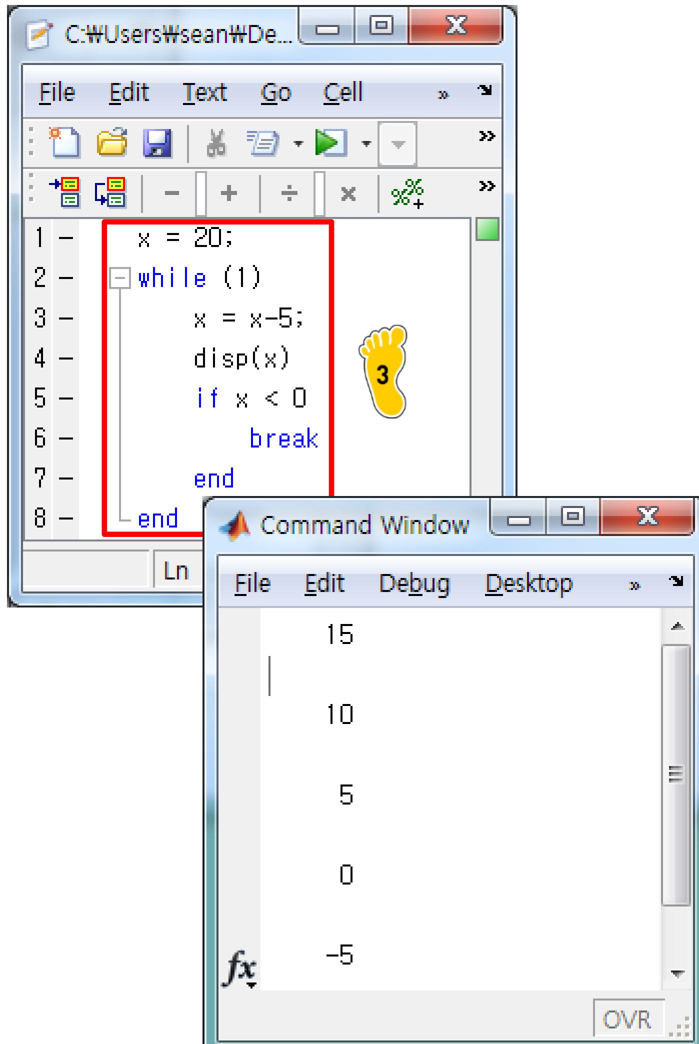
.

.

WHILE STRUCTURE

```
while condition
    statements
end
```





1 while 문의 문법

if 문의 문법과 동일

2 x 값이 8부터 -3 단위로 감소하는 예제

3 condition 을 1로 입력할 경우 무한루프가 실행됨

statements 에 if 문과 break 명령어를 입력하여 프로그램을 중단

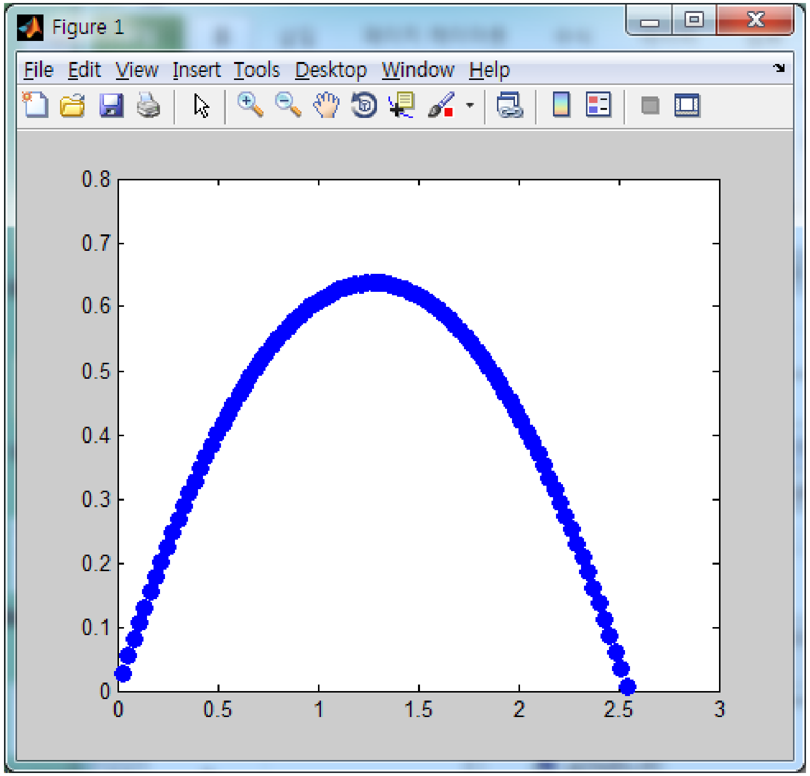
EXAMPLE I

Problem Statement: In the absence of air resistance, the Cartesian coordinates of a projectile launched with an initial velocity (v_0) and angle (θ_0) can be computed with

$$x = v_0 \cos(\theta_0)t$$

$$y = v_0 \sin(\theta_0)t - 0.5gt^2$$

where g = 9.81 m/s^2. Develop a script to generate an animated plot of the projectile trajectory given that v_0 = 5 m/s and θ_0 = 45°.



1

1 주어진 조건대로 독립변수 x 와 종속변수 y 를 animation 으로 plot 을 하는 예제

EXAMPLE I

C:\Users\sean\Desktop\Untitled12.m

```
clc,clf,clear

g = 9.81; theta0 = 45*pi/180; v0 = 5;

t(1) = 0; x = 0; y = 0;

plot(x,y,'o','MarkerFaceColor','b','MarkerSize',8)

axis([0 3 0 0.8])

M(1) = getframe;

dt = 1/128;

for j = 2:10000
    t(j) = t(j-1)+dt;
    x = v0*cos(theta0)*t(j);
    y = v0*sin(theta0)*t(j)-0.5*g*t(j)^2;
    plot(x,y,'o','MarkerFaceColor','b','MarkerSize',8)
    axis([0 3 0 0.8])
    M(j)=getframe;
    if y <=0, break, end
end

pause

movie(M,1)
```

1 clc: command window 창에 표기 되어있는 것을 초기화 (clear command)

clf: 현재 pop up 되어있는 모든 plot 창을 초기화 (clear figure)

clear: workspace 에 저장되어 있는 변수들을 초기화

2 연산에 사용하는 변수들 선언, 주의할 점은 매틀랩에서 모든 각도는 radian 단위로 입력 해야함

3 Animation 을 실행하기 위해 frame 을 저장하는 명령어

EXAMPLE I

C:\Users\sean\Desktop\Untitled12.m

```matlab
clc,clf,clear

g = 9.81; theta0 = 45*pi/180; v0 = 5;

t(1) = 0; x = 0; y = 0;

plot(x,y,'o','MarkerFaceColor','b','MarkerSize',8)

axis([0 3 0 0.8])

M(1) = getframe;

dt = 1/128;

for j = 2:10000
    t(j) = t(j-1)+dt;
    x = v0*cos(theta0)*t(j);
    y = v0*sin(theta0)*t(j)-0.5*g*t(j)^2;
    plot(x,y,'o','MarkerFaceColor','b','MarkerSize',8)
    axis([0 3 0 0.8])
    M(j)=getframe;
    if y <=0, break, end
end

pause

movie(M,1)
```

1. for 문을 이용하여 단계별로 시간을 증가시키면서 독립 변수 x 와 종속변수 y 값을 저장 및 plot

   그 후 animation 을 생성하기 위한 frame 을 매 iteration 마다 저장

   for 문이지만 break 명령어를 이용하여 중간에 중단할 수 있음

2. pause 명령어는 매틀랩 실행을 잠시 중단시키는 명령어

   pause on 혹은 pause off 로 조절 가능

3. movie(m,n,fps) 명령어는 현재 저장된 frame m 을 n 번 단위 초당 몇 개의 frame 을 재생하는 함수

EXAMPLE II

$$f\left(x\right)=ax^{2}+bx+c$$

$$x=\frac{-b\pm\sqrt{b^{2}-4ac}}{2a}$$

C:₩Users₩sean₩Desktop₩quadroots.m

```
function quadroots(a,b,c)
% quadroots: roots of quadratic equation
%    quadroots (a,b,c): real and complex roots
%                       of quadratic equation
% input:
%   a = second-order coefficient
%   b = first-order coefficient
%   a = zero-order coefficient
% output:
%   r1 = real part of first root
%   i1 = imaginary part of first root
%   r2 = real part of second root
%   i2 = imaginary part of second root
if a == 0
    %special cases
else
    %quadratic formula
end
```

coding 작업은 기본적으로 top-down design 방식으로 진행

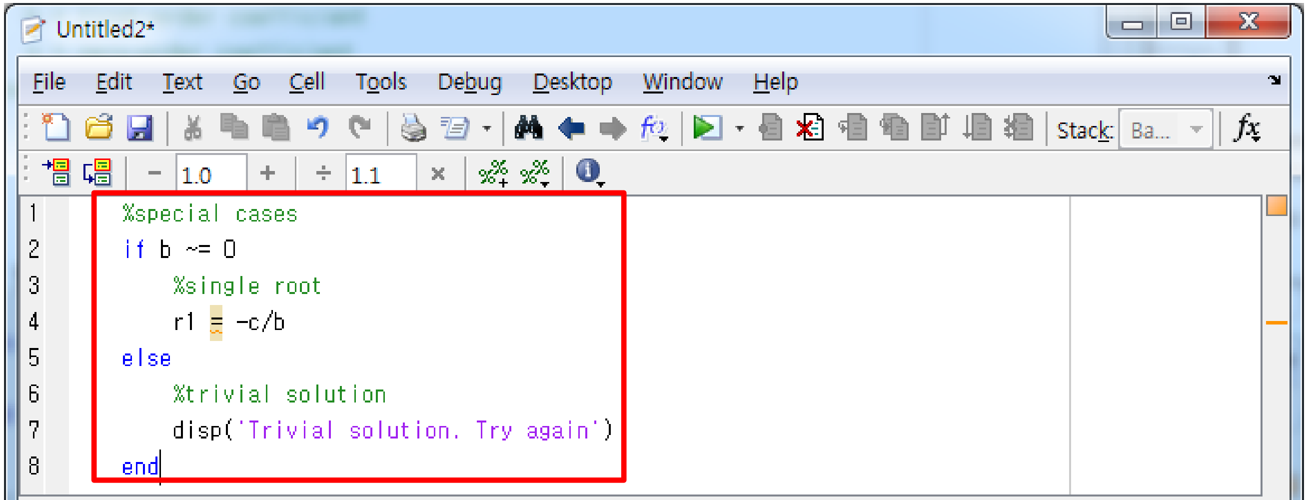
1. 프로그램은 위부터 아래로 한줄 씩 실행되며 input 및 output 과 예상되는 오류를 생각하여 알고리즘을 구성

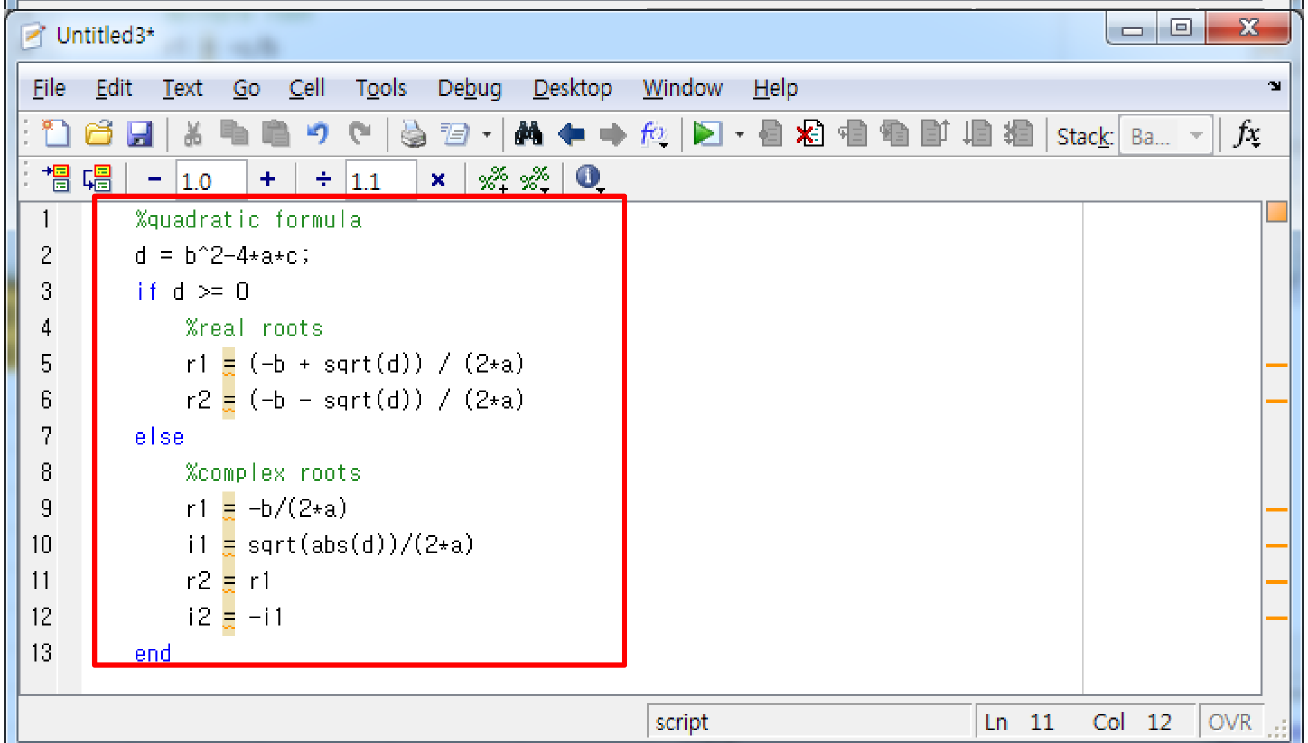
2. 본 예제는 근의 공식을 이용하여 2차 방정식을 푸는 script를 코딩하는 방법을 보여줌

Input: 계수
Output: 근

3. 계수 a 가 0 일 경우에는 무한대이므로 논리 구조를 special case 와 quadratic formula 로 삽입

EXAMPLE II





1 다음 과정은 세부 structure를 코딩

EXAMPLE II

C:₩Users₩sean₩Desktop₩quadroots.m

```
function quadroots(a,b,c)
% quadroots: roots of quadratic equation
%    quadroots (a,b,c): real and complex roots
%                       of quadratic equation
% input:
%   a = second-order coefficient
%   b = first-order coefficient
%   a = zero-order coefficient
% output:
%   r1 = real part of first root
%   i1 = imaginary part of first root
%   r2 = real part of second root
%   i2 = imaginary part of second root
if a == 0
    %special cases
    if b ~= 0
        %single root
        r1 = -c/b
    else
        %trivial solution
        disp('Trivial solution. Try again')
    end
else
    %quadratic formula
    d = b^2-4*a*c;
    if d >= 0
        %real roots
        r1 = (-b + sqrt(d)) / (2*a)
        r2 = (-b - sqrt(d)) / (2*a)
    else
        %complex roots
        r1 = -b/(2*a)
        i1 = sqrt(abs(d))/(2*a)
        r2 = r1
        i2 = -i1
    end
end
```

Command Window

```
>> quadroots(1,1,1)

r1 =

   -0.5000


i1 =

    0.8660


r2 =

   -0.5000


i2 =

   -0.8660

>> quadroots(1,5,1)

r1 =

   -0.2087


r2 =

   -4.7913

>>
```

마지막으로 코딩한 script 를 본 코드에 옮겨서 확인 작업을 마치면 structured programming 작업이 끝남

- **Passing functions to m-files**
  - ✓ **Anonymous functions**
  - ✓ **Function functions**
  - ✓ **Example**
  - ✓ **Passing parameters**

ANONYMOUS FUNCTIONS

1 @ 기호를 이용하여 간단하게 함수를 지정할 수 있다.

지정하는 방식

fhandle = @(arglist) expression

```
>> f1 = @(x,y) x^2 + y^2;
>> f1(3,4)

ans =

    25
```

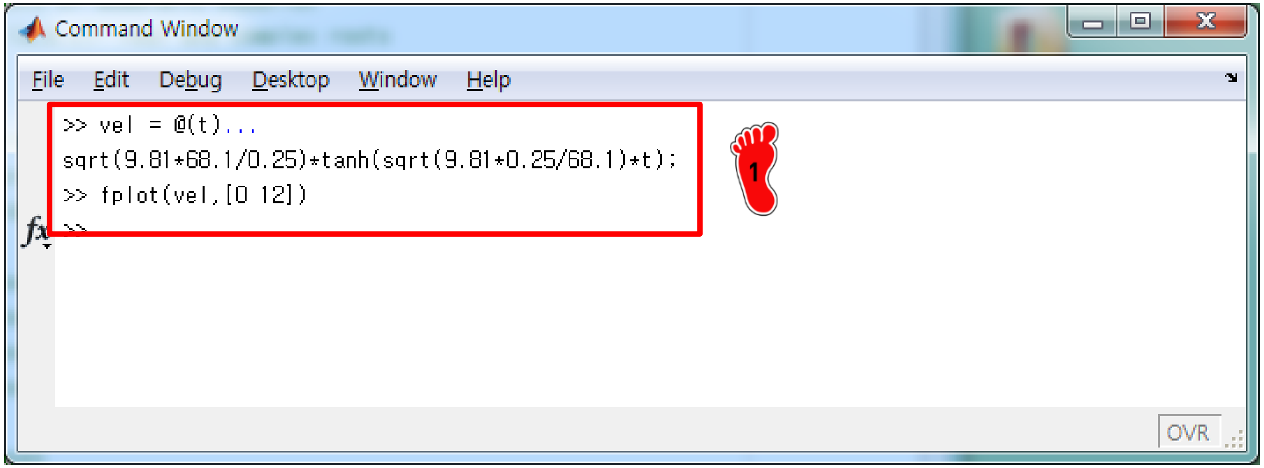
2 상수가 미리 저장되어 있으면 상수를 이용하여 함수를 지정할 수 있음
이 때 상수 값을 변경하면 자동으로 함수값도 변경됨

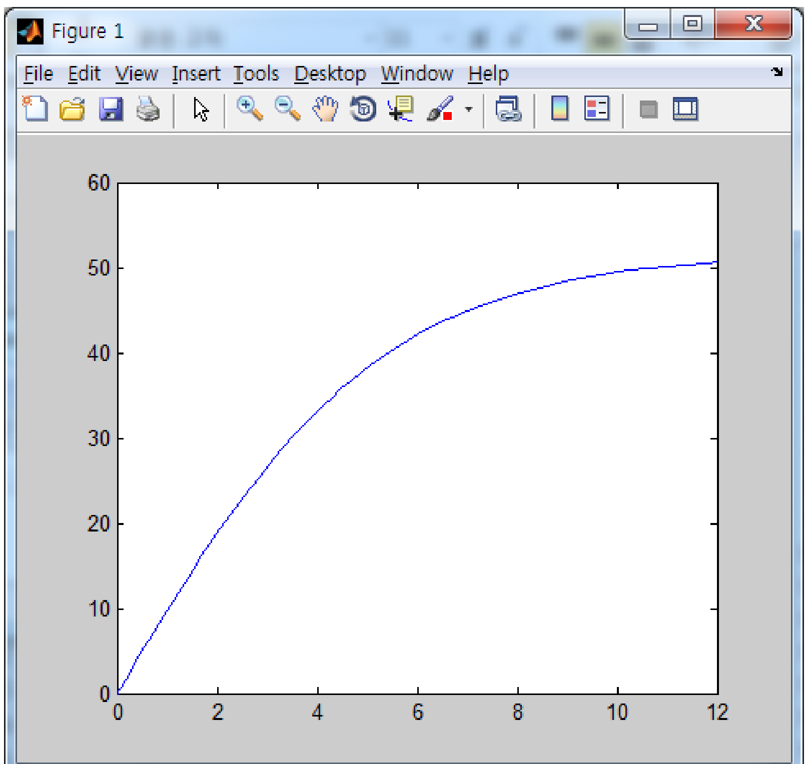
```
>> a = 4;
>> b = 2;
>> f2 = @(x) a+x^b;
>> f2(3)

ans =

    36
```

3 inline 명령어의 경우 @ 기호처럼 함수를 저장 가능

```
>> f1 = inline('x^2+y^2','x','y');
>> f1(2,4)

ans =

    20

>>
```

FUNCTION FUNCTIONS





1

fplot: function 을 이용하여 그래프를 그리는 함수

fplot(func,lims)

func: fplot 에 입력되는 함수

lims: 입력되는 function 의 argument limit

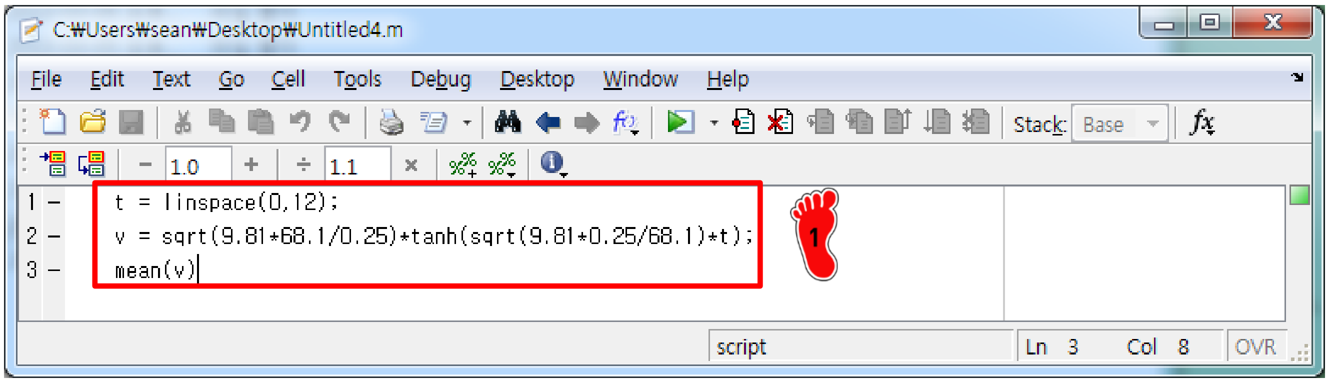
lims = [xmin,xmax]

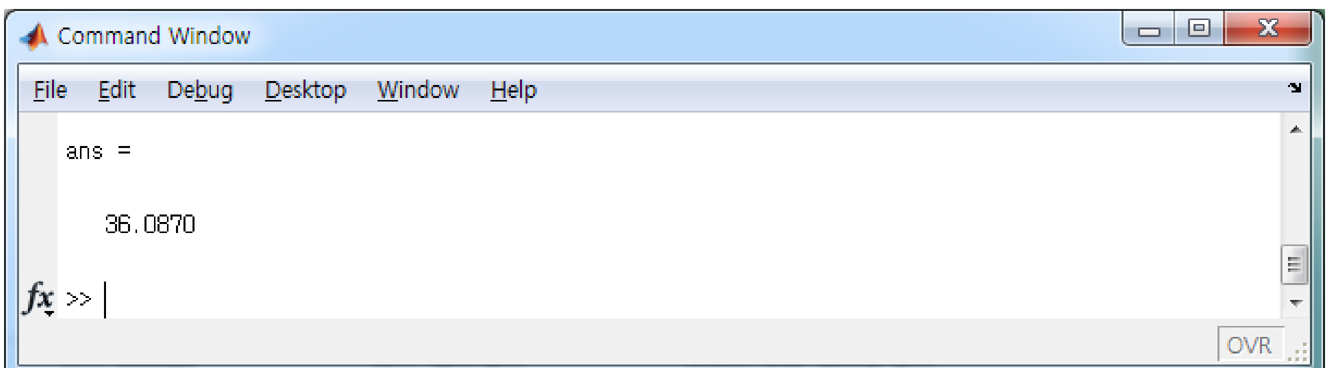
EXAMPLE

Problem statement: Develop an M-file function to determine the average of a function over a range. Illustrate its use for the bungee jumper velocity over the range from t = 0 to 12 s:

$$v = \sqrt{\frac{gm}{c}} \tanh\left(\sqrt{\frac{gc}{m}}t\right)$$

where g = 9.81, m = 68.1, and c = 0.25





1주차에서 강의했던 수식을 이용하여 평균 속도를 구하는 예제

기본적으로 다음과 같이 script 를 구성하여 답을 구할 수 있지만 function function 원리를 이용하여 일반적으로 코드를 구성할 수 있음

EXAMPLE

C:₩Users₩sean₩Desktop₩funcavg.m

```
function favg = funcavg(a,b,n)
% funcavg: average function height
%   favg = funcavg(a,b,n): computs average value
%                          of function over a range
% input:
%    a = lower bound of range
%    b = upper bound of range
%    n = number of intervals
% output:
%    favg = average value of function
x = linspace(a,b,n);
y = func(x);
favg = mean(y);
end

function f = func(t)
f = sqrt(9.81*68.1/0.25)*tanh(sqrt(9.81*0.25/68.1)*t);
end
```

Command Window

```
>> funcavg(0,12,60)

ans =

   36.0127

>>
```

1 Subfunction 으로 구성하여 일반적인 형태로 script 를 구성한 예제

평균값을 구하는 y 함수를 subfunction 으로 지정 평균값을 구하는 mean 함수는 y 함수를 이용하여 계산

따라서 subfunction 을 다른 함수로 입력할 경우 그 함수에 대한 평균값을 구할 수 있음

EXAMPLE

C:₩Users₩sean₩Desktop₩funcavg2.m

```
function favg = funcavg2(f,a,b,n)
% funcavg: average function height
%   favg = funcavg(a,b,n): computs average value
%                          of function over a range
% input:
%    f = function to be evaluated
%    a = lower bound of range
%    b = upper bound of range
%    n = number of intervals
% output:
%    favg = average value of function
x = linspace(a,b,n);
y = f(x);
favg = mean(y);
end
```

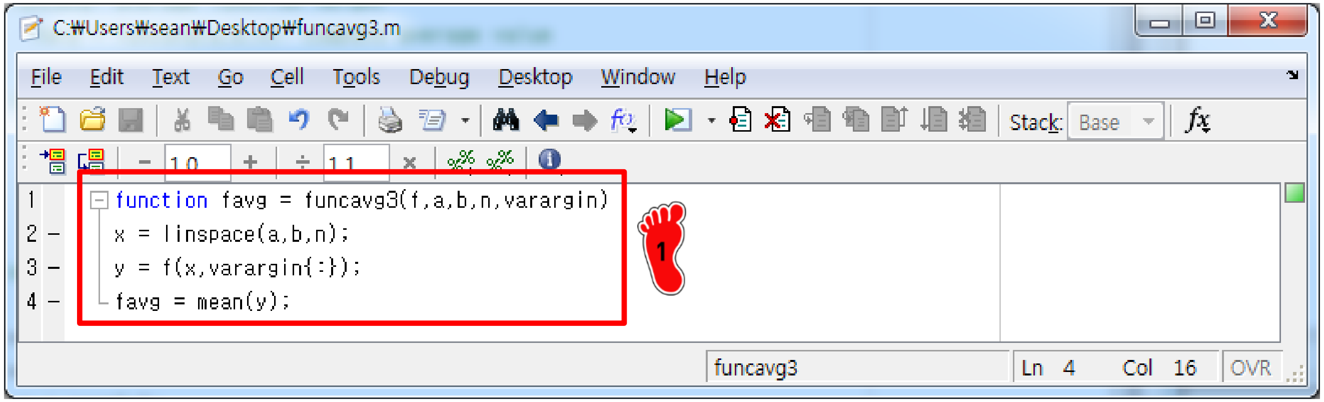
Command Window

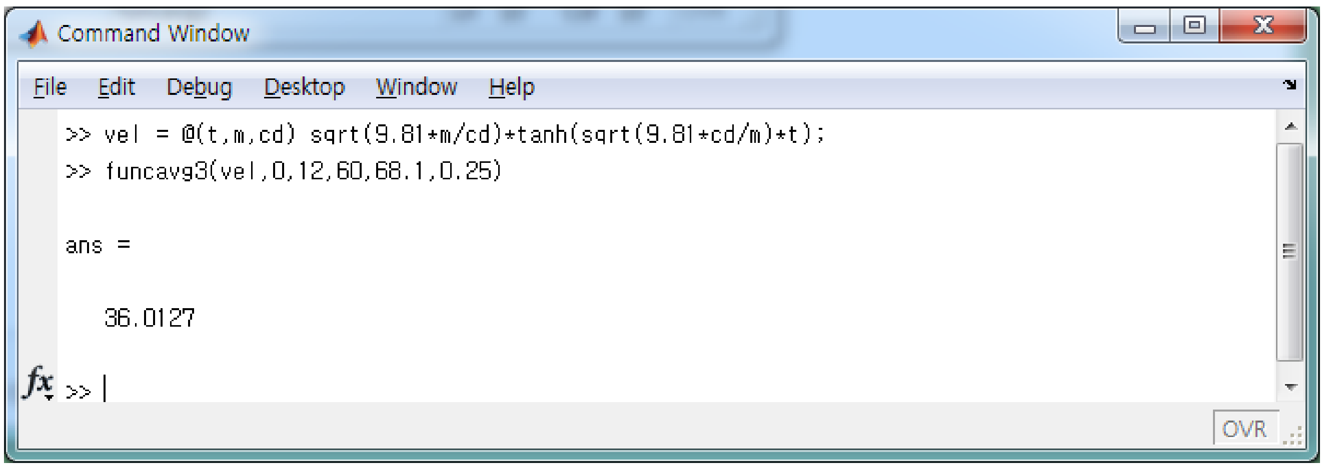
```
>> vel = @(t)...
sqrt(9.81*68.1/0.25)*tanh(sqrt(9.81*0.25/68.1)*t);
>> funcavg2(vel,0,12,60)

ans =

   36.0127
```

평균값을 계산할 함수를 argument 로 입력받아 계산하는 예제

@ 기호를 이용하여 다양한 함수의 평균값을 계산 가능

PASSING PARAMETERS





앞의 예제들은 함수에 들어가는 파라미터값을 미리 정하여 입력

하지만 파라미터값이 변할 경우, 매번 함수를 다시 정의해야 하는 불편함이 있으므로 파라미터값을 varargin 명령어로 입력받을 수 있게 script 를 구성한 예제

- **Case study**

- **Assignment**

CASE STUDY

Background. In this section, we will use MATLAB to solve the free-falling bungee jumper problem we posed at the beginning of this chapter. This involves obtaining a solution of

$$\frac{dv}{dt} = g - \frac{c_d}{m} v|v|$$

Recall that, given an initial condition for time and velocity, the problem involved iteratively solving the formula,

$$v_{i+1} = v_i + \frac{dv_i}{dt} \Delta t$$

Now also remember that to attain good accuracy, we would employ small steps. Therefore, we would probably want to apply the formula repeatedly to step out from our initial time to attain the value at the final time. Consequently, an algorithm to solve the problem would be based on a loop.

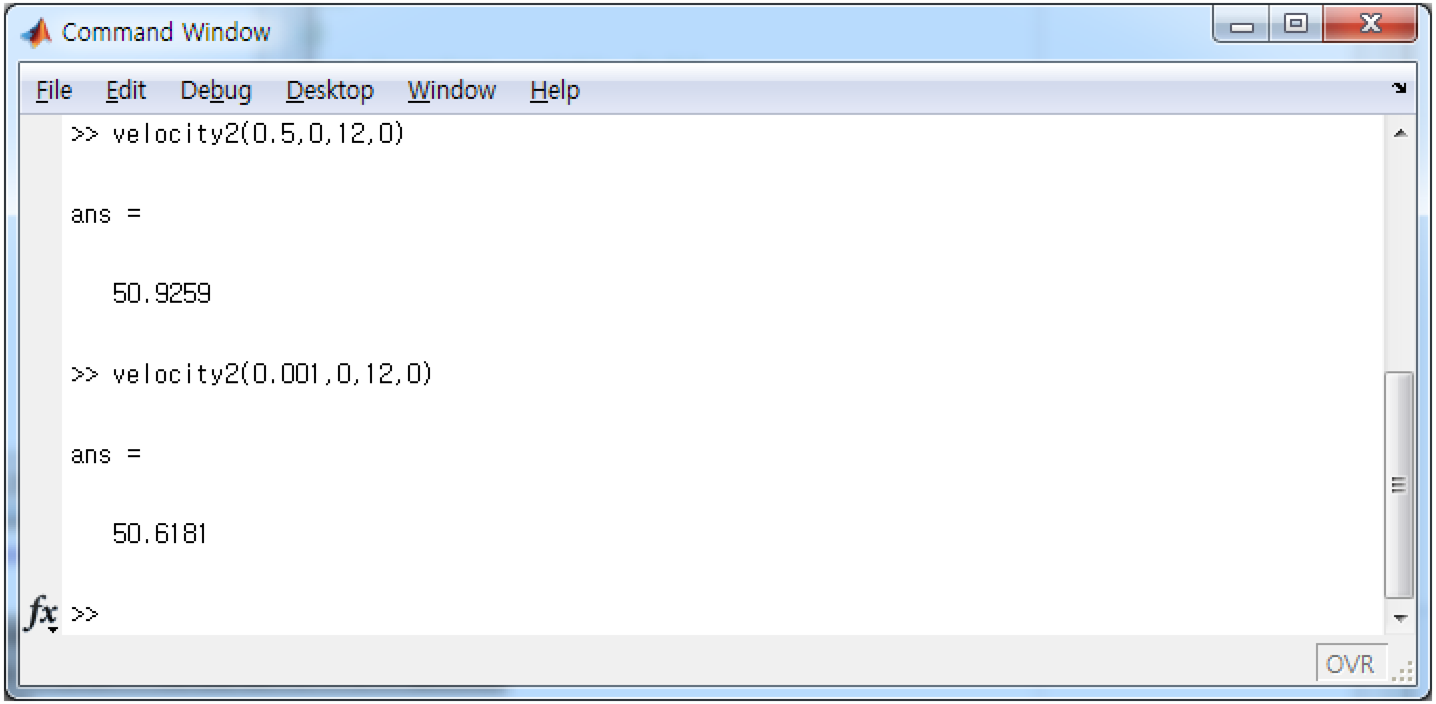
<Hint>

initial time = 0 s

final time = 12 s

time step = 0.25 s

SOLUTION OF CASE STUDY



ASSIGNMENT

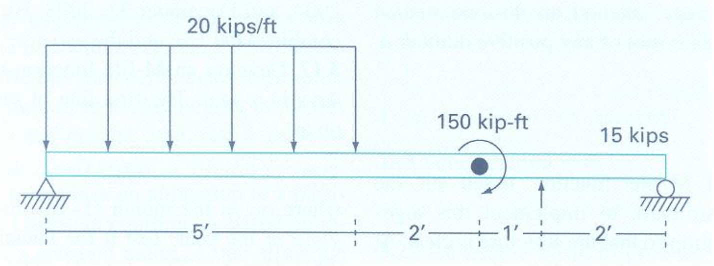


FIGURE P3.10

3.9 Manning's equation can be used to compute the velocity of water in a rectangular open channel:

$$U = \frac{\sqrt{S}}{n}\left(\frac{BH}{B+2H}\right)^{2/3}$$

where U = velocity (m/s), S = channel slope, n = roughness coefficient, B = width (m), and H = depth (m). The following data are available for five channels:

n	S	B	H
0.036	0.0001	10	2
0.020	0.0002	8	1
0.015	0.0012	20	1.5
0.030	0.0007	25	3
0.022	0.0003	15	2.6

Write an M-file that computes the velocity for each of these channels. Enter these values into a matrix where each column represents a parameter and each row represents a channel. Have the M-file display the input data along with the computed velocity in tabular form where velocity is the fifth column. Include headings on the table to label the columns.

3.10 A simply supported beam is loaded as shown in Fig. P3.10. Using singularity functions, the displacement along the beam can be expressed by the equation:

$$u_y(x) = \frac{-5}{6}[\langle x-0\rangle^4 - \langle x-5\rangle^4] + \frac{15}{6}\langle x-8\rangle^3 + 75\langle x-7\rangle^2 + \frac{57}{6}x^3 - 238.25x$$

By definition, the singularity function can be expressed as follows:

$$\langle x-a\rangle^n = \begin{Bmatrix} (x-a)^n & \text{when } x > a \\ 0 & \text{when } x \le a \end{Bmatrix}$$

Develop an M-file that creates a plot of displacement (dashed line) versus distance along the beam, x. Note that $x = 0$ at the left end of the beam.

3.11 The volume V of liquid in a hollow horizontal cylinder of radius r and length L is related to the depth of the liquid h by

$$V = \left[r^2\cos^{-1}\left(\frac{r-h}{r}\right) - (r-h)\sqrt{2rh-h^2}\right]L$$

Develop an M-file to create a plot of volume versus depth. Here are the first few lines:

```
function cylinder(r, L, plot_title)
% volume of horizontal cylinder
% inputs:
% r = radius
% L = length
% plot_title = string holding plot title
```

Test your program with

```
>> cylinder(3,5,...
'Volume versus depth for horizontal...
cylindrical tank')
```