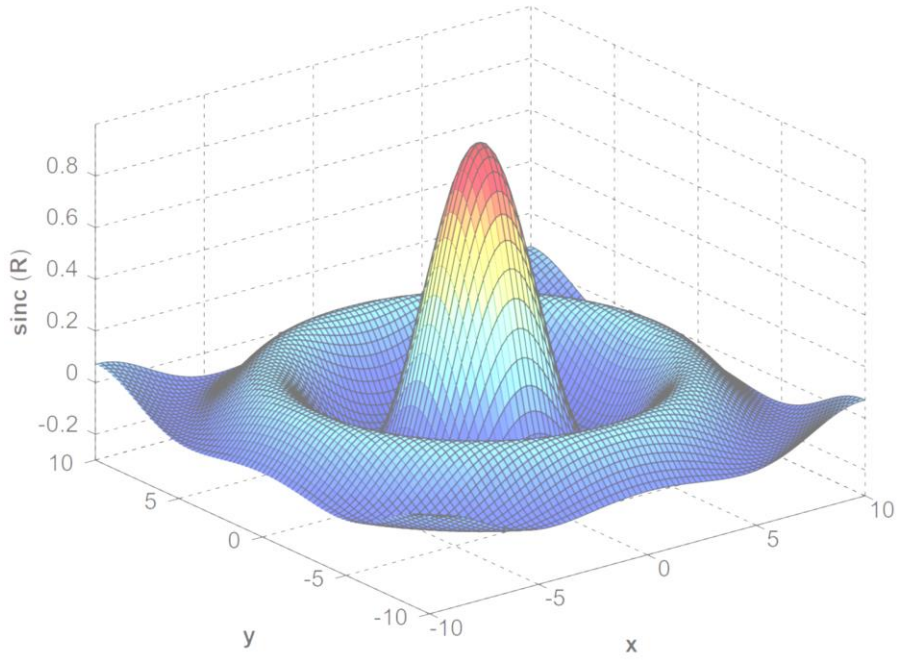
응용전산및실습 I

(강의자료 #6)



교과목명 : 응용전산 및 실습 I

담당교수 : 이 수 형

E-mail : soohyong@uu.ac.kr

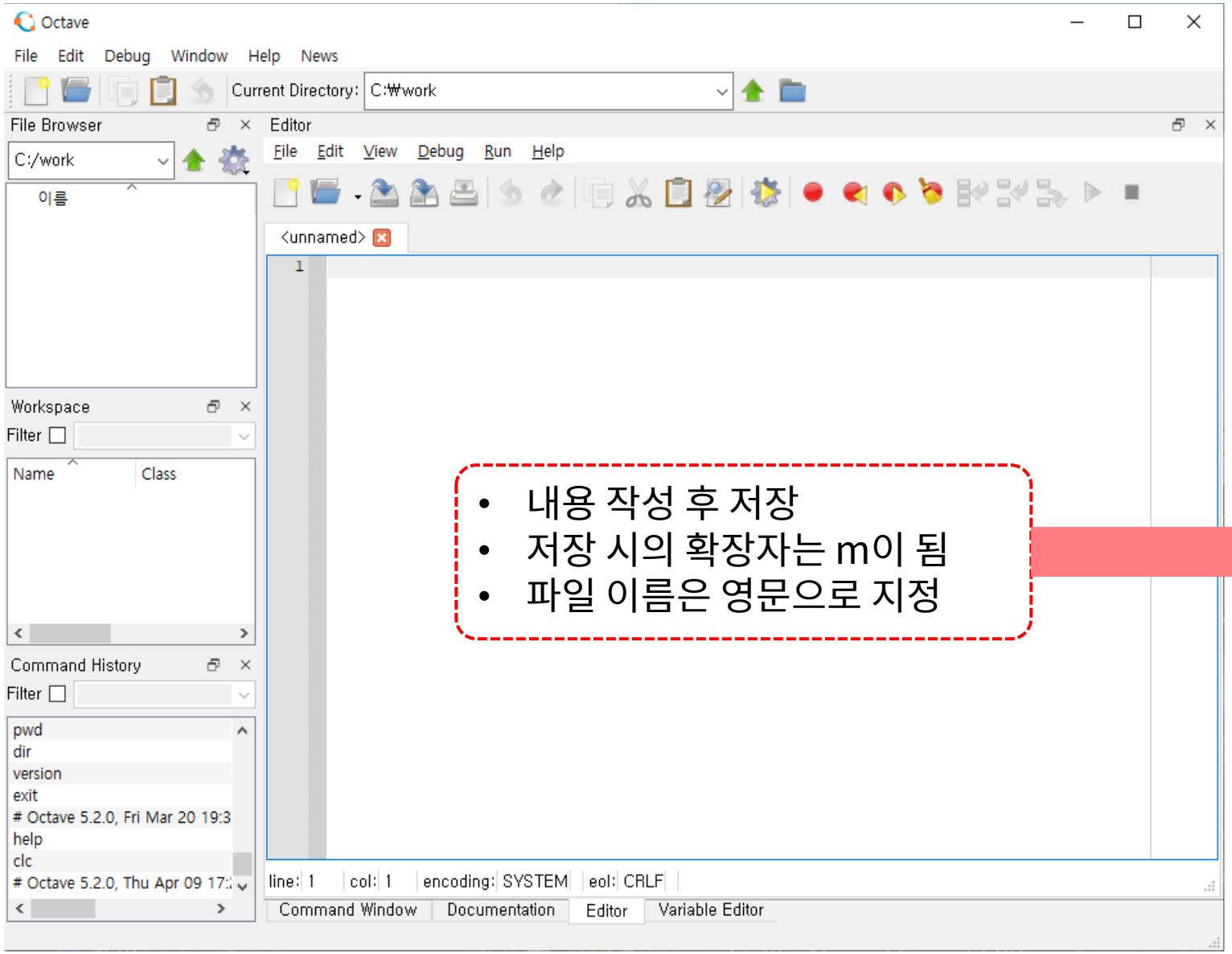
교재명 : 유인물

M-파일 프로그래밍 #1

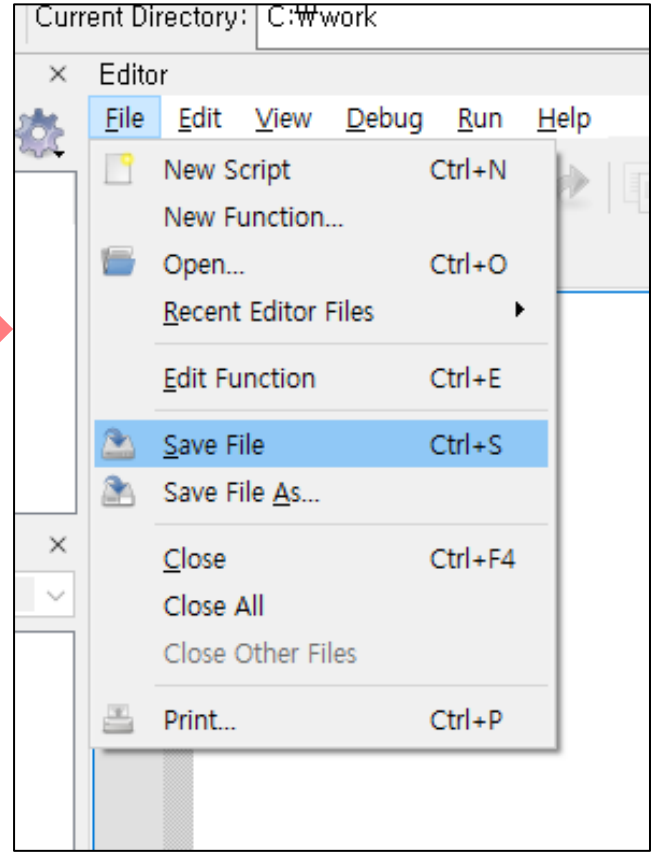
M 파일의 작성

- M파일 : Matlab에서 사용하는 프로그램 저장용 파일
  - 확장자가 'm'으로 되어 있음
  - 스크립트(script) 파일과 함수(function) 파일로 나누어짐
- 스크립트 파일
  - 명령어들을 미리 모아놓은 파일
  - 명령창(command window)에서 호출 → 저장된 명령어들을 순차적으로 실행
- 함수 파일
  - 함수의 형식으로 저장 → 함수를 사용하는 방법으로 호출 (입력변수 → 출력값)
- 새파일 작성
  - 메뉴에서 [File → New → New Script]를 선택하여 실행

M 파일의 작성



- 내용 작성 후 저장
- 저장 시의 확장자는 m이 됨
- 파일 이름은 영문으로 지정



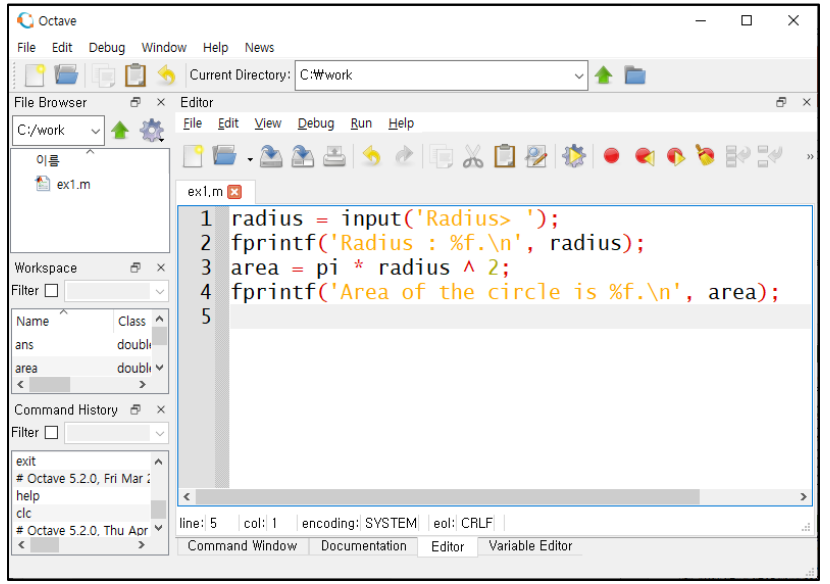
M 파일의 작성

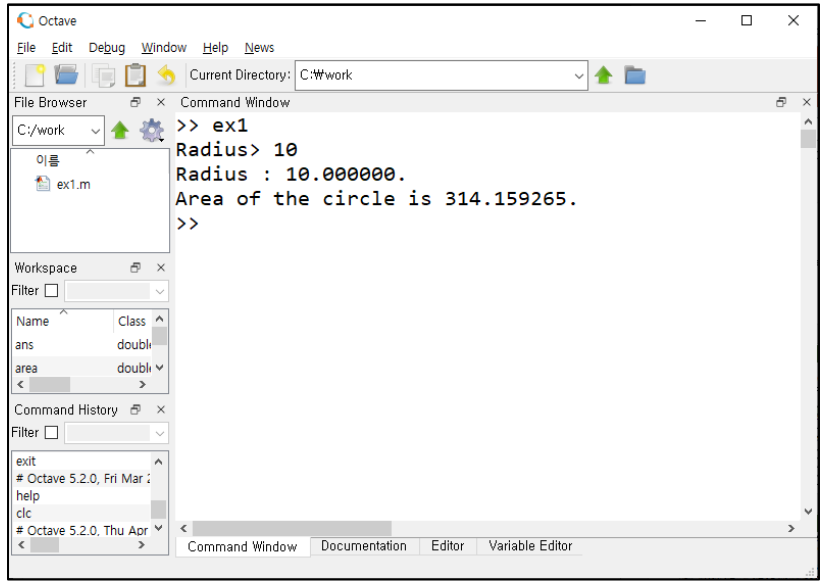
- M-file 작성 (스크립트 파일)
  - 현재 폴더에 저장 (파일 이름은 영문으로) 명령창에서 한글입력 불가능
  - 파일이름
- 수행 (스크립트)
  - 명령창에서 파일 이름 (확장자 제외) 입력

```
>> ex1
```

ex1.m

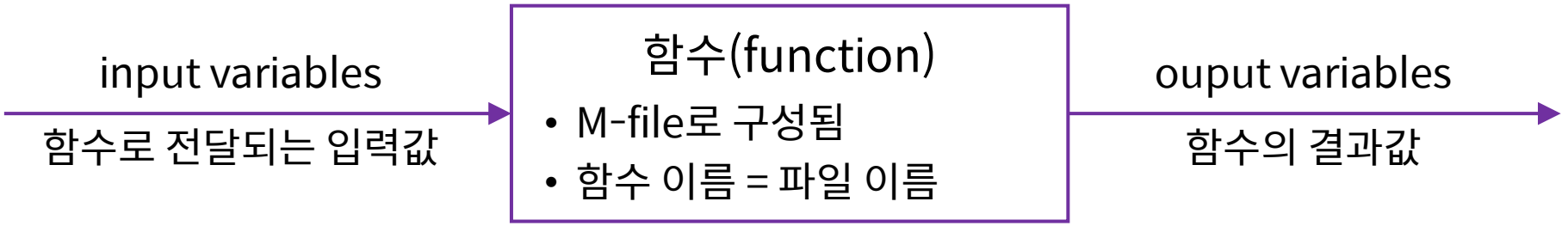
```
radius = input('Radius> ');
fprintf('Radius : %f.\n', radius);
area = pi * radius ^ 2;
fprintf('Area of the circle is %f.\n', area);
```





사용자 정의 함수

- 함수(function)
  - 일반적인 프로그래밍 언어의 함수와 거의 같은 기능을 수행
    - C언어 : 함수(function), Basic & FORTRAN : subroutine, Pascal : procedure
  - 프로그램의 모듈 역할을 하는 기능으로 자주 사용하는 기능을 구현한 뒤 함수로 만들어놓고 간단하게 불러서 사용하는 것



사용자 정의 함수

- 함수(function)의 구조

```
% ------------------------------------------------------------
% Contents of 'help' command
% ------------------------------------------------------------

function [output_variables] = function_name(input_variables)
    % do MATLAB commands
end
```

Item	Description
function	M 파일이 함수를 정의한다는 것을 의미하는 키워드
output_variables	함수의 결과를 저장할 변수명 (생략 가능하며, 두 개 이상인 경우 [] 사용)
function_name	함수 이름 – 파일 이름과 반드시 같게 한다
input_variables	해당 함수의 입력 변수 (단일 혹은 복수 개)

사용자 정의 함수

- 예제

```
%-----------------------------------------------------------
% function sum = intsum(a, b)
% return sum of two input variables
%-----------------------------------------------------------

function sum = intsum(a, b)
    sum = a + b
end
```

```
>> intsum(2, 3)
sum =  5
ans =  5
```

사용자 정의 함수

- 도움말 (help)

```
%-----------------------------------------------------------------
% function sum = intsum(a, b)
% return sum of two input variables
%-----------------------------------------------------------------

%

function sum = intsum(a, b)
    sum = a + b
end
```

첫번째 주석문 그룹을
도움말(help)에 표시

```
>> help intsum
'intsum' is a function from the file c:\work\intsum.m

-----------------------------------------------------------------
 function sum = intsum(a, b)
 return sum of two input variables
-----------------------------------------------------------------
...
```

사용자 정의 함수

- 스크립트 파일 vs. 함수 파일
  - 두 경우 모두 확장자가 *.m
  - 스크립트/함수의 구분
    - 첫번째 실행문이 `function ...` 으로 시작 → 함수 파일
    - 아니면 → 스크립트 파일
  - 파일의 내부 변수
    - 함수 파일의 내부 변수들 → 지역 변수 (함수 내부에서만 사용 가능한 변수)
    - 스크립트 파일의 내부 변수들 → 전역 변수 (command window에서 수행하는 것과 동일)
  - Workspace의 변수 사용
    - 함수 파일 내부 → 사용이 불가능 (필요한 경우 인수/입력 변수로 전달하여야 함)
    - 스크립트 파일 내부 → 사용 가능 (command window에서 수행하는 것과 동일)

사용자 정의 함수

- 예제 : radian → degree

```
%-------------------------------------------------------------
% ret = rad2deg(r)
% Converts r (in radians) value to ret (in degree) value
%-------------------------------------------------------------

function ret = rad2deg(r)
    ret = r * 180 / pi;
end
```

```
>> rad2deg(pi / 2)
ans =  90
>> rad2deg(pi / 6)
ans =  30.000
>> deg = rad2deg(pi / 4)
deg =  45
>>
```

Matlab 프로그래밍

- 관계연산자 (relational operator)
  - 두 피 연산자의 관계를 판단하는 연산자로 조건문에서 비교문(compare statement)에 주로 사용

Operator	Meaning	Example	C언어
<	Less than	A < B	<
<=	Less or equal than	A <= B	<=
>	Greater than	A > B	>
>=	Greater or equal than	A >= B	>=
==	Equal	A == B	==
~=	Not equal	A ~= B	!=

  - 관계연산자의 결과값 : 참 → 1, 거짓 → 0

Matlab 프로그래밍

- 관계연산자
  - 벡터/행렬의 경우 원소 단위로 비교하여 벡터/행렬로 반환

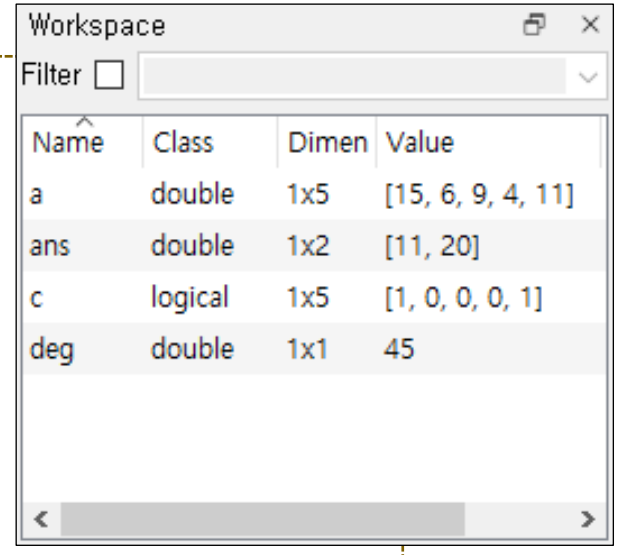
```
>> 5 > 8
ans = 0
>> a = 5 < 10
a = 1
>> x = (6 < 10) + (7 > 8) + (5 * 3 == 60 / 4)
x =  2
>> a = [15 6 9 4 11];
>> b = [8 20 9 2 19];
>> b >= a
ans =

  0  1  1  0  1

>>
```

```
>> a == b
ans =

  0  0  1  0  0

>> a ~= b
ans =

  1  1  0  1  1

>> a > 10
ans =

  1  0  0  0  1
```

Matlab 프로그래밍

- 관계연산자
  - 벡터/행렬의 경우 관계연산자의 결과는 논리 벡터/행렬로 반환 → 주소지정에 사용되어 참(1)인 위치의 원소만 추출 가능

```
>> a = [15 6 9 4 11];
>> c = a > 10
c =

  1   0   0   0   1

>> a(c)
ans =

   15   11

>>
```



벡터 a에서 10보다 큰 값만 추출하기

Matlab 프로그래밍

- 논리연산자 (logical operator)
  - 관계연산자의 결과값(참/거짓)과 같은 논리값의 연산으로 두 개 이상의 복합 조건을 판단할 때 사용

Operator	Meaning	Example	C언어
&	Logical AND	A & B	&&
|	Logical OR	A | B	||
~	Logical NOT	~A	!

Matlab 프로그래밍

- 논리연산자 (logical operator)
  - 벡터/행렬의 경우 각 요소별로 계산됨

```
>> 3 & 7
ans = 1
>> a = 5 | 0
a = 1
>> ~25
ans = 0
>> a = [9 2  0   1 0 10];
>> b = [2 0 12 -10 0  2];
>> a & b
ans =
  1  0  0  1  0  1
>> z = a | b
z =
  1  1  1  1  0  1
>> ~(a + b)
ans =
  0  0  0  0  1  0
```

Matlab 프로그래밍

- 연산자 우선순위

우선순위	연산자	결합방향
1	( )	→
2	.' .^ ' ^	→
3	(단항)+ (단항)- ~	→
4	.* ./ .\ * / \	→
5	+ -	→
6	:	→
7	< <= > >= == ~=	→
8	&	→
9	|	→
10	&&	→
11	||	→

M-file 프로그래밍

제어문

Matlab 프로그래밍 : if

- if 문

```
if 조건문
    문장;   % 조건문이 만족하는 경우 수행하는 문장들...
end         % Octave 편집기에서는 endif가 자동으로 나타나나
            % Matlab과의 호환성을 위해서 end로 수정할 것
```

- Example

```
k = input('Enter k : ');
if k > 0
    disp('You entered positive number.');
end
```

Matlab 프로그래밍 : if

- if ~ else 문

```
if 조건문
    문장;   % 조건이 만족하는 경우 수행하는 문장
else
    문장;   % 만족하지 않은 경우 수행하는 문장
end
```

- Example

```
k = input('Enter k : ');
if k > 0
    disp('You entered positive number.');
else
    disp('You entered negative number.');
end
```

Matlab 프로그래밍 : if

- if ~ else 문 예제
  - 수학, 영어 과목의 점수들을 입력받아 두 과목 모두 60점 이상인 경우 통과(pass) 아니면 탈락(fail)을 계산하는 스크립트 파일 작성.
  - 관계 연산자 및 논리 연산자 사용
  - 통과 조건을 검사하는 경우 vs. 탈락 조건을 검사하는 경우 (2가지 조건문 가능)

Matlab 프로그래밍 : if

- if ~ elseif ~ else 문

```
if 조건문1
    문장;   % 조건 1이 만족하는 경우 수행하는 문장
elseif 조건문2
    문장;   % 조건 1이 만족하지 않고 조건 2가 만족하는 경우
else
    문장;   % 조건 1, 2가 모두 만족하지 않는 경우
end
```

```
k = input('Enter number : ');
if k > 0
    disp('You entered positive number.');
elseif k < 0
    disp('You entered negative number.');
else
    disp('You entered 0.');
end
```

Matlab 프로그래밍 : if

- if ~ elseif ~ else 문 예제
  - 점수(score)을 입력받아 90점 이상이면 'A', 80점 이상 90점 미만이면 'B', … , 60점 미만이면 'F'학점을 출력하는 스크립트 파일 작성.
  - 알고리즘]
    - 만일 점수가 90 이상이면
      'A' 학점 출력
      그렇지 않고 만일 점수가 80 이상이면
      'B' 학점 출력
      그렇지 않고 만일 점수가 70 이상이면
      'C' 학점 출력
      …
      그렇지 않으면
      'F' 학점 출력

Matlab 프로그래밍 : if

- 중첩 조건문 (nested if)
  - 하나의 if문에 따른 조건문 속에 다른 조건문이 포함되는 경우

```
if 조건문1
    if 조건문2  % 조건 1이 만족하는 경우, 조건 2를 검사
        문장;   % 조건 1이 만족하면서 조건 2도 만족하는 경우 수행
    else
        문장;   % 조건 1이 만족하면서 조건 2는 만족하지 않는 경우
    end
else
    if 조건문3  % 조건 1이 만족하지 않으면서, 조건 3을 검사
        문장;   % 조건 1이 만족하지 않으면서 조건 2를 만족하는 경우 수행
    else
        문장;   % 조건 1이 만족하지 않으면서 조건 2도 만족하지 않은 경우
    end
end
```

Matlab 프로그래밍 : if

- 중첩 조건문 (nested if)
  - 하나의 if문에 따른 조건문 속에 다른 조건문이 포함되는 경우

```
if 조건문1
        조건문1이 만족하는 경우 수행하는 문장들
else
        조건문1이 만족하지 않은 경우 수행하는 문장들
end
```

Matlab 프로그래밍 : if

- 중첩 조건문 예제
  - 1[in] = 2.54[cm], 1[ft]=12[in] 인 관계를 이용하여 cm, ft, in 를 서로 변환시키는 스크립트 파일 작성.
    - 실행 예] (파란색은 입력)

```
Enter value : 10
Enter unit : in          → 현재 단위
Enter new unit : cm      → 변환할 단위
10[in] is 25.4[cm]
```

- 응용
  - 1[in] = 2.54[cm], 1[ft]=12[in] 인 관계를 이용하여 m, cm, in, ft 를 서로 변환시키는 스크립트 파일 작성.

과제

- 출석점검용 과제
  - 수업에 수행한 실습 화면 및 결과 화면을 캡처해서 제출하시오.

- 채점용 과제
  - 정수를 입력 받아 입력된 숫자가 짝수(even number)인지 홀수(odd number)인지 알려주는 스크립트 파일을 작성하시오.
    - Matlab에서 나머지를 구하는 함수 : `mod(x, y)`
    - 예] 9를 4로 나눌때 나머지는? → `mod(9, 4)`

실습과제

- 수업에 수행한 실습 화면 및 결과 화면을 캡쳐해서 제출하시오.

- 중심이 $x = 3, y = 2$ 이며 주축이 $a = 8, y = 6$인 타원의 그래프를 그려라.

- 정수를 입력 받아 입력된 숫자가 짝수(even number)인지 홀수(odd number)인지 알려주는 스크립트 파일을 작성하시오.
  - Matlab에서 나머지를 구하는 함수 : `mod(x, y)`
  - 예] 9를 4로 나눌때 나머지는? → `mod(9, 4)`

Matlab 프로그래밍 : for

- for 문

```
for 변수=초기값:최종값
    문장;   % 변수가 초기값에서 최종값까지 1씩 증가하면서 반복됨
end         % Octave 편집기에서는 endfor가 자동으로 나타나나
            % Matlab과의 호환성을 위해서 end로 수정할 것
```

- Example : sumall.m

```
n = input('Enter number : ');
sum = 0;
for k=1:n
    sum = sum + k;
end
fprintf('Sum from 1 to %d is %d\n', n, sum);
```

Matlab 프로그래밍 : for

- for 문

```
for 변수=초기값:증가값:최종값
    문장;   % 변수가 초기값에서 최종값까지 증가값씩 증가하면서 반복됨
end         % Octave 편집기에서는 endfor가 자동으로 나타나나
            % Matlab과의 호환성을 위해서 end로 수정할 것
```

- Example : sumodd.m

```
n = input('Enter number : ');
sum = 0;
for k=1:2:n
    sum = sum + k;
end
fprintf('Sum of odd numbers from 1 to %d is %d\n', n, sum);
```

Matlab 프로그래밍 : for

- for 문 예제
  - 1부터 주어진 인수(입력변수) n까지의 합을 구하는 함수 AddTo()

```
%-------------------------------------------------------------
% function sum = AddTo(n)
% Calculate sum of the numbers from 1 to n
%-------------------------------------------------------------

function sum = AddTo(n)
  sum = 0;
  for k = 1:n
    sum = sum + k;
  end
end
```

Matlab 프로그래밍 : for

- for문의 반복 범위
  - `1:4 == [1, 2, 3, 4]` ⇒ for문에서도 벡터값을 이용하여 반복

```
for k = 1:2:6
    sum = sum + k;
end
```

```
for k = [1 3 5]
    sum = sum + k;
end
```

Matlab 프로그래밍 : nargin, nargout

- 함수의 입/출력 변수의 개수
  - nargin : 입력 변수의 개수 (number of input arguments)
  - nargout : 출력 변수의 개수 (number of output arguments)
  - 호출할 때 사용된 입/출력 변수의 개수에 알맞은 기능을 제공하는 함수 작성이 가능 : 예] plot 함수

Matlab 프로그래밍 : nargin, nargout

```
% function [res1, res2, res3] = argtest(a, b, c, d)
% Test function for nargin, nargout

function [res1, res2, res3] = argtest(a, b, c, d)
    fprintf('nargin = %d\n', nargin);
    fprintf('nargout = %d\n', nargout);

    res1 = nargin;
    if nargout > 1
      res2 = nargout;
    end
    if nargout > 2
      res3 = 0;
    end
End
```

```
>> argtest
nargin = 0
nargout = 0
ans = 0
>> a = argtest(1, 2)
nargin = 2
nargout = 1
a =  2
>> [a, b] = argtest(1, 2)
nargin = 2
nargout = 2
a =  2
b =  2
>> [a, b] = argtest(1, 2, 3)
nargin = 3
nargout = 2
a =  3
b =  2
```

Matlab 프로그래밍 : for

- for 문 예제 : AddTo()함수의 확장 ⇒ AddToEx()
  - AddToEx(n) : 1~n 까지의 합계
  - AddToEx(a, b) : a~b 까지의 합계

```
%-------------------------------------------------------------
% function sum = AddToEx(n)
% Calculate sum of the numbers from 1 to n
% function sum = AddToEx(a, b)
% Calculate sum of the numbers from a to b
%-------------------------------------------------------------

function sum = AddToEx(a, b)
    if nargin == 1
        b = a;
        a = 1;
    end
    sum = 0;
    for k = a:b
        sum = sum + k;
    end
end
```

Ex : 원주율 (π) 계산하기

- for문을 사용하여 π계산하기 #1 : 라이프니츠(Leibniz)의 원주율공식

$$\pi = 4\left(\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} \cdots\right)$$

➢ From Gregory Series : $\tan^{-1} x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$, where $x = 1$

```
n = 10000;     % iteration number
pisum = 0;     % result

for i=1:n
  pisum = pisum + (-1)^(i+1) / (2*i - 1);
  if mod(i, 1000) == 0
    fprintf('Iter [%d], pi = %.12f\n', i, pisum * 4);
  end
end

fprintf('pi = %.12f\n', pisum * 4);
```

Ex : 원주율 (π) 계산하기

- for문을 사용하여 π계산하기 #1 : 라이프니츠(Leibniz)의 원주율공식

$$\pi = 4\left(\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} \cdots\right) = 4\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1}$$

- 함수로 구현

```
%-------------------------------------------------
% function ret = pi1f(n)
% Calculate pi using Leibniz's equation
% n : iteration number
%-------------------------------------------------

function ret = pi1f(n)
  pisum = 0; % result
  for i=1:n
    pisum = pisum + (-1)^(i+1) / (2*i - 1);
  end
  ret = pisum * 4;
end
```

```
>> pi1f(1)
ans =  4
>> pi1f(5)
ans =  3.3397
>> pi1f(10)
ans =  3.0418
>> pi1f(100)
ans =  3.1316
>> pi1f(1000)
ans =  3.1406
>> pi1f(10000)
ans =  3.1415
>> pi1f(100000)
ans =  3.1416
```

Ex : 원주율 (π) 계산하기

- for문을 사용하여 π계산하기 #2 : 월리스(Wallis)의 공식

$$\frac{\pi}{2} = \frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\cdot\frac{8}{9}\cdot\frac{10}{9}\cdots = \left(\frac{2}{1}\cdot\frac{2}{3}\right)\cdot\left(\frac{4}{3}\cdot\frac{4}{5}\right)\cdot\left(\frac{6}{5}\cdot\frac{6}{7}\right)\cdots = \prod_{k=1}^{\infty}\left(\frac{4\cdot k^2}{4\cdot k^2-1}\right)$$

```
n = 10000;     % iteration number
pival = 1;     % result

for k=1:n
  pival = pival * 4 * k^2 / (4 * k^2 - 1);
  if mod(k, 1000) == 0
    fprintf('Iter [%d], pi = %.12f\n', k, pival * 2);
  end
end

fprintf('pi = %.12f\n', pival * 2);
```

Ex : 원주율 (π) 계산하기

- π계산하기 #3 : 오일러(Euler)의 곱셈 공식

$$\frac{\pi^2}{6} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \frac{1}{6^2} + \cdots$$

- π계산하기 #4 : 프랑수아 비에트(François Viète)의 무한급수

$$\frac{2}{\pi} = \frac{\sqrt{2}}{2} \cdot \frac{\sqrt{2+\sqrt{2}}}{2} \cdot \frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2} \cdots$$

Ex : Taylor Series

- 테일러 급수
  - 임의의 해석함수(analytic function)를 도함수의 함 점에서의 값으로 계산된 항의 무한합으로 나타내는 방식 (유도과정 및 자세한 내용 생략)
  - a 근처에서는 몇 개의 항으로도 근사값 계산

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \frac{1}{3!}f'''(a)(x-a)^3 + \cdots$$

  - $a = 0$인 경우 ⇒ Maclaurin series
- 테일러 급수의 응용?
  - 모델의 단순화 : 복잡한 함수가 있을 때 1~3차의 다항 함수로 근사화
  - 정적분의 계산 : 부정 적분의 계산이 힘든 경우 급수를 활용하여 근사화 후 계산
  - 컴퓨터에서 초월 함수(sin, cos, tan, exp, log, …) 계산
    - ➢ 초월 함수의 H/W 계산 회로 구성은 불가능 ⇒ 근사 다항식을 이용하여 계산

Ex : Taylor Series

- 테일러 급수의 예
  - a 근처에서

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \frac{1}{3!}f'''(a)(x-a)^3 + \cdots$$

  - 예] $a = 0$인 경우 ⇒ 0 근처에서 근사값

$$\sin x = \sin 0 + x \sin' 0 + \frac{x^2}{2!}\sin'' 0 + \frac{x^3}{3!}\sin''' 0 + \cdots$$

$$\sin x = 0 + x \cdot 1 + \frac{x^2}{2!} \cdot 0 + \frac{x^3}{3!} \cdot (-1) + \frac{x^4}{4!} \cdot 0 + \cdots$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!}x^{2n+1}$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!}x^{2n}, \qquad e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Ex : Taylor Series

- 테일러 급수를 이용한 sin 함수

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

```
%-------------------------------------------------------
% function ret = sint(x)
% Calculate sine value using Taylor series expansion
%-------------------------------------------------------

function ret = sint(x)
  ret = 0;
  for n = 0:5
    ret = ret + (-1)^n / factorial(2*n + 1) * x^(2*n + 1);
  end
end
```

```
>> sin(pi / 4)
ans =  0.70711
>> sint(pi / 4)
ans =  0.70711
>> sin(pi)
ans =    1.2246e-16
>> sint(pi)
ans = -0.00044516
```

Ex : Taylor Series

- 테일러 급수를 이용한 sin 함수
  - 다항식의 항의 수를 인수로 전달할 수 있도록 확장

```
%-------------------------------------------------------
% function ret = sinto(x, order)
% Calculate sine value using Taylor series expansion
% x : angle [rad], order : order of series expansion
%-------------------------------------------------------

function ret = sinto(x, order)
    if nargin < 2
        order = 5;
    end
    ret = 0;
    for n = 0:order
        ret = ret + (-1)^n / factorial(2*n+1)*x^(2*n+1);
    end
end
```

```
>> sin(pi)
ans =    1.2246e-16
>> sinto(pi, 3)
ans = -0.075221
>> sinto(pi) ⇒ sinto(pi,5)
ans = -0.00044516
>> sinto(pi, 7)
ans = -0.00000077279
>> sinto(pi, 10)
ans =  0.000000000010348
>>
```

Ex : Taylor Series

- 테일러 급수를 이용한 sin 함수
  - 인수로 벡터/행렬도 처리할 수 있도록 확장

```
%-------------------------------------------------------
% function ret = sintv(x, order)
% Calculate sine value using Taylor series expansion
% x can be either vector or matrix
% x : angle [rad], order : order of series expansion
%-------------------------------------------------------

function ret = sintv(x, order)
  if nargin < 2
    order = 5;
  end
  ret = zeros(size(x));
  for n = 0:5
    ret = ret + (-1)^n / factorial(2*n + 1)*x .^ (2*n + 1);
  end
end
```

Matlab 프로그래밍 : while

- while 문

```
while 조건식
    문장;   % 조건식이 만족하는 동안 문장을 반복
end         % Octave 편집기에서는 endwhile이 자동으로 생성되나
            % Matlab과의 호환성을 위해서 end로 수정할 것
```

- Example : sumall.m

```
n = input('Enter number : ');
sum = 0;
j = 1;
while(j <= n)
    sum = sum + j;
    j = j + 1;
end
fprintf('Sum from 1 to %d is %d\n', n, sum);
```

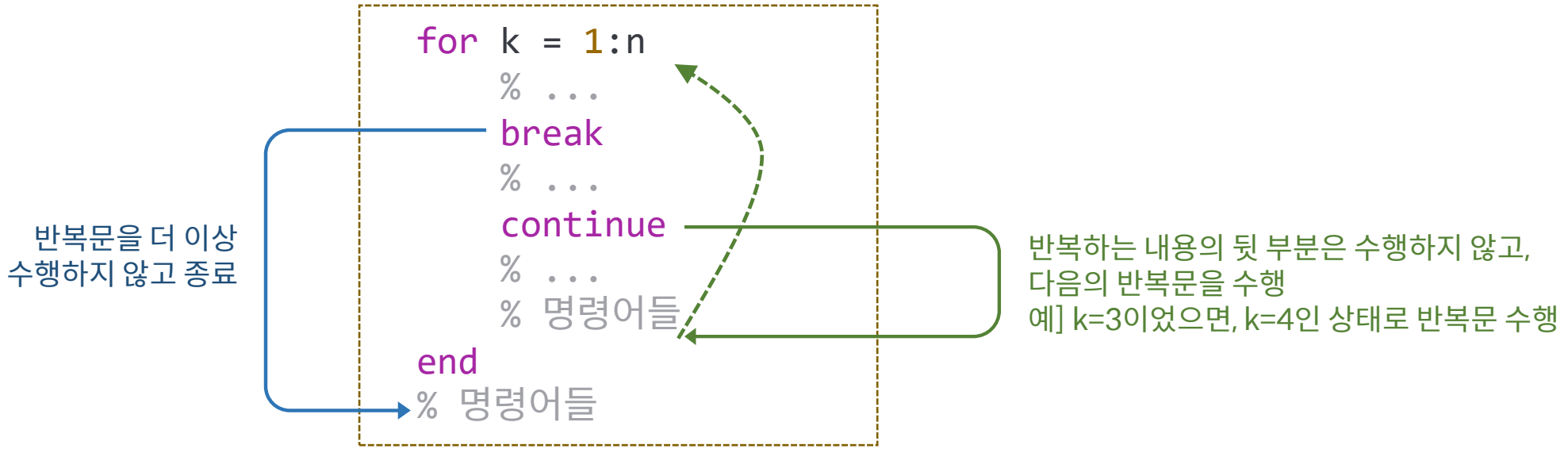
Matlab 프로그래밍 : while

- while 문 예제
  - 1부터 주어진 인수(입력변수) n까지의 합을 구하는 함수 AddTo()

```
%-------------------------------------------------------------
% function sum = AddTo(n)
% Calculate sum of the numbers from 1 to n
%-------------------------------------------------------------

function AddTo(n)
    sum = 0;
    j = 1;
    while(j <= n)
        sum = sum + j;
        j = j + 1;
    end
end
```

Matlab 프로그래밍 : break, continue

- break 문
  - for 또는 while문의 반복문을 종료
- continue 문
  - for 또는 while문의 나머지 반복문을 수행하지 않고 다음 반복문을 수행



Matlab 프로그래밍 : break, continue

- Break, continue 문 예제

```
for k = 1:6
  if k == 4
    break;
  end
  k
end
```

```
>> break1
k =  1
k =  2
k =  3
>>
>>
```

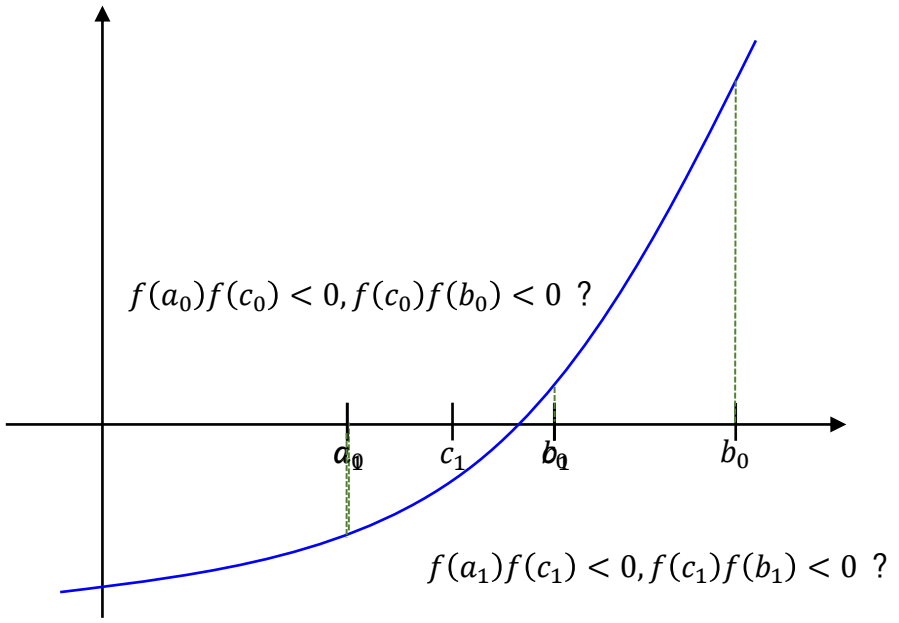
```
for k = 1:5
  if k == 2 | k == 3
    continue;
  end
  k
end
```

```
>> continue1
k =  1
warning: Matlab-style short-circuit…
k =  4
k =  5
>>
```

Matlab에서는 나타나지 않음

Ex: Root-Finding

- 함수의 근 찾기 : Bisection method
  - $f(x) = 0$이 되는 x (roots) 찾기, 초기값 : 근이 존재하는 범위 설정
  - 구간을 절반씩 나누어서 검색$[f(x) \approx 0]$하는 것을 반복함
- 수행 순서 (algorithm)
  - $k = 0$에서 시작 : 초기 구간 $[a_0, b_0]$
    - ➢ 근이 존재하는 조건 : $f(a_0)f(b_0) < 0$
  - k를 증가시키면서 다음을 반복
    - ➢ 구간의 중간값 계산 : $c_k = \frac{a_k + b_k}{c_k}$
    - ➢ $f(a_k)f(c_k), f(c_k)f(b_k)$의 부호 계산
    - ➢ $f(a_k)f(c_k) \Rightarrow a_{k+1} = a_k\ , b_{k+1} = c_k$
    - ➢ $f(c_k)f(b_k) \Rightarrow a_{k+1} = c_k\ , b_{k+1} = b_k$
    - ➢ 오차 : $|a_k - b_k| < \epsilon$ 보다 작으면 중단, 근 $\Rightarrow c_k$
      - ✓ ϵ : 오차 범위, 예] 소수점 5째자리 $\Rightarrow \epsilon = 10^{-6}$

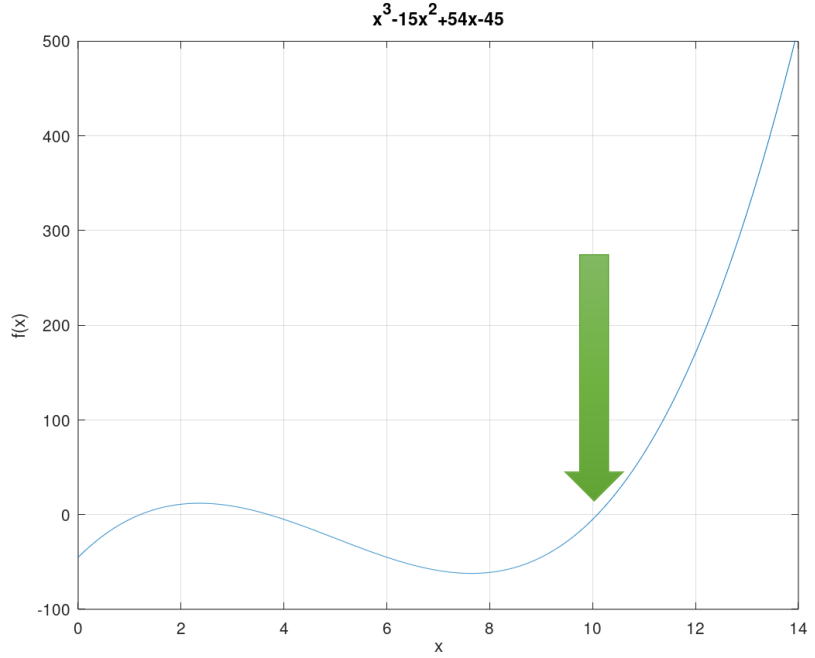


Ex: Root-Finding

- Bisection method example

  - $f(x) = x^3 - 15x^2 + 54x - 45$, 초기 구간 : $[8, 13]$

```
% set initial interval [8, 13]
a = 8;
b = 13;

eps = 1e-6; % epsilon
iter = 1;   % iteration number

fprintf('   k        a           b         f(c)\n');
fprintf('-----------------------------------------\n');
while abs(a - b) > eps
  c = (a + b) / 2;
  if func(a) * func(c) < 0
    b = c;
  else
    a = c;
  end
  fprintf('%5d %10.7f %11.7f  %11.7f\n', iter, a, b, func(c));
  iter = iter + 1;
end
```



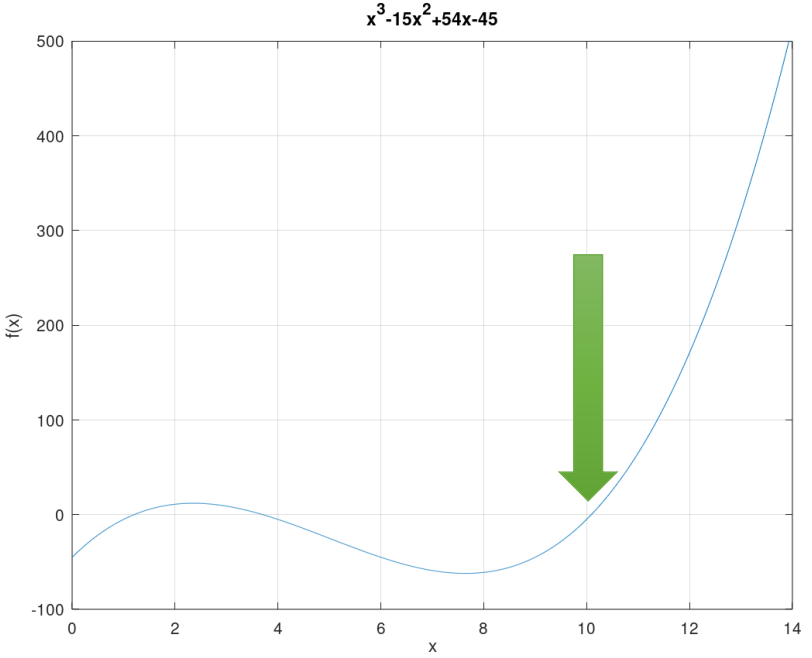

```
function y = func(x)
  y = x.^3 - 15 * x.^2 + 54 * x - 45;
end
```

Ex: Root-Finding

- Bisection method example

  - $f(x) = x^3 - 15x^2 + 54x - 45$, 초기 구간 : $[8, 13]$

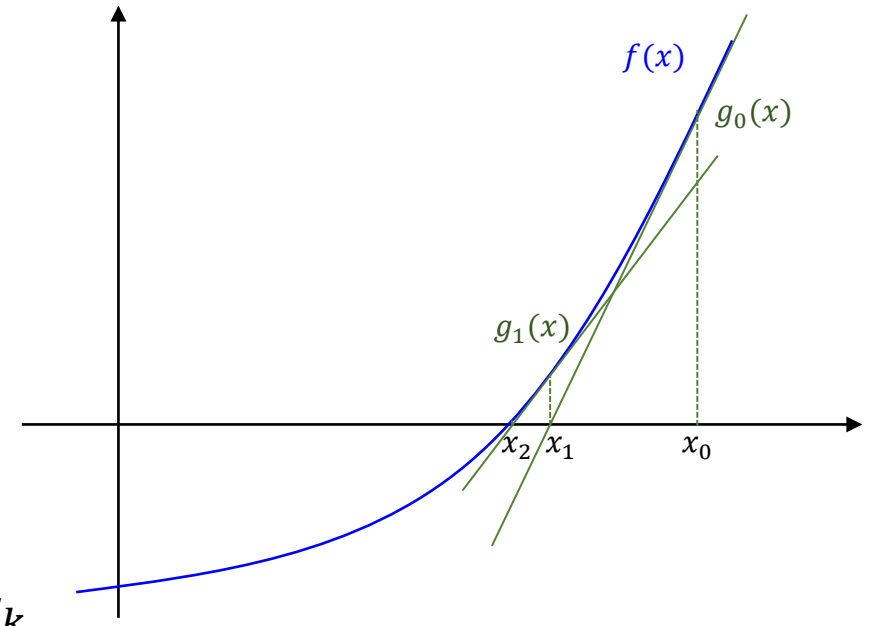
```
% set initial interval [8, 13]
a = 8;
b = 13;

eps = 1e-6; % epsilon
iter = 1;   % iteration number

fprintf('   k        a           b         f(c)\n');
fprintf('-----------------------------------------\n');
while 1
  c = (a + b) / 2;
  if func(a) * func(c) < 0
    b = c;
  else
    a = c;
  end
  fprintf('%5d %10.7f %11.7f  %11.7f\n', iter, a, b, func(c));
  if abs(a - b) < eps
    break
  end
  iter = iter + 1;
end
```



Ex: Root-Finding

- 함수의 근 찾기 : Newton-Raphson method
  - $f(x) = 0$이 되는 근 근처에서 초기값 x_0 설정
  - $x = x_k$에서 $f(x)$의 선형 근사 함수 $g(x)$의 해를 구하여 x_{k+1}로 설정한 후 반복

- 수행 순서 (algorithm)
  - $k = 0$에서 시작 : 초기 해 x_0 설정
  - k를 증가시키면서 다음을 반복
    - ➢ $x = x_k$에서 선형 근사 함수 $g_k(x)$계산
      - ✓ $g_k(x) = f(x_k) + f'(x_k)(x - x_k)$
    - ➢ $g_k(x) = 0$이 되는 x계산
      - ✓ $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$: $f'(x)$가 필요함 [단점]
    - ➢ 오차 : $|x_k - x_{k+1}| < \epsilon$ 보다 작으면 중단, 근 $\Rightarrow x_k$
      - ✓ ϵ : 오차 범위, 예] 소수점 5째자리 $\Rightarrow \epsilon = 10^{-6}$

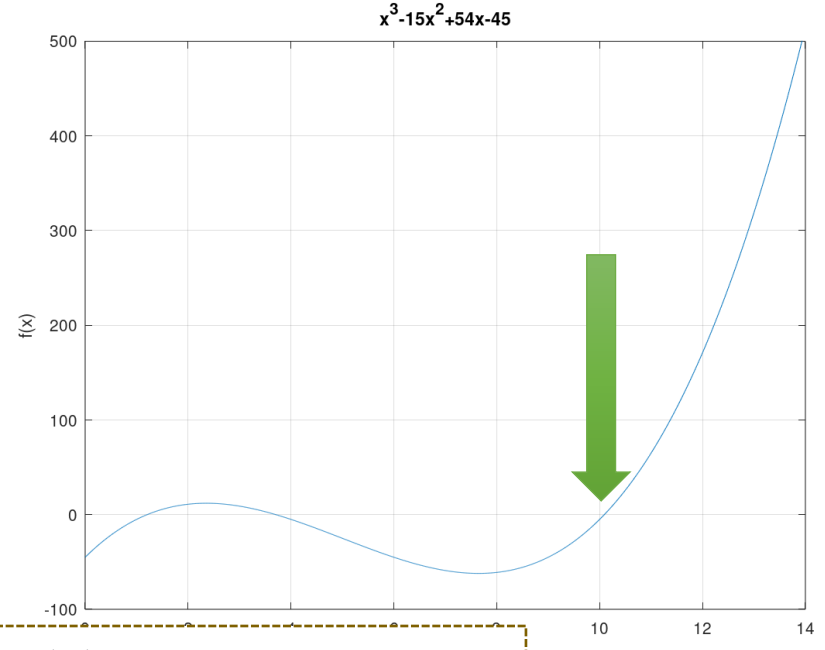


Ex: Root-Finding

- Newton-Raphson method example

  - $f(x) = x^3 - 15x^2 + 54x - 45$, 초기값 : $f_0 = 12$

```
x_cur = 8;  % initial root
eps = 1e-6; % epsilon
iter = 1;   % iteration number
max_iter = 30; % maximum iteration number

fprintf('   k        x         f(c)\n');
fprintf('-----------------------------\n');
while 1
  x_next = x_cur - func(x_cur) / funcd(x_cur);
  fprintf('%5d %11.7f  %11.7f\n', iter, x_next, func(x_next));
  if abs(x_next - x_cur) < eps
    break
  end
  x_cur = x_next;
  iter = iter + 1;
  if iter > max_iter
    break
  end
end
```




```
function y = func(x)
  y = x.^3 - 15 * x.^2 + 54 * x - 45;
end
```

```
function y = funcd(x)
  y = 3 * x.^2 - 30 * x + 54;
end
```

Root-Finding : Comparison

```
>> bisect1
   k        a           b          f(c)
-----------------------------------------
    1  8.0000000  10.5000000   25.8750000
    2  9.2500000  10.5000000  -37.4843750
    3  9.8750000  10.5000000  -11.5175781
    4  9.8750000  10.1875000    5.6589355
    5 10.0312500  10.1875000   -3.2978210
    6 10.0312500  10.1093750    1.0870018
    7 10.0703125  10.1093750   -1.1286197
    8 10.0898438  10.1093750   -0.0266338
    9 10.0898438  10.0996094    0.5287250
   10 10.0898438  10.0947266    0.2506812
   11 10.0898438  10.0922852    0.1119326
   12 10.0898438  10.0910645    0.0426267
   13 10.0898438  10.0904541    0.0079907
   14 10.0901489  10.0904541   -0.0093230
   15 10.0903015  10.0904541   -0.0006665
   16 10.0903015  10.0903778    0.0036621
   17 10.0903015  10.0903397    0.0014978
   18 10.0903015  10.0903206    0.0004157
   19 10.0903111  10.0903206   -0.0001254
   20 10.0903111  10.0903158    0.0001451
   21 10.0903111  10.0903134    0.0000099
   22 10.0903122  10.0903134   -0.0000578
   23 10.0903128  10.0903134   -0.0000240
```

```
>> nr
   k        x          f(c)
------------------------------
    1  18.1666667  1981.0879630
    2  14.1972134  559.8391617
    3  11.7920568  145.6982188
    4  10.5509747   29.4734893
    5  10.1384116    2.7642441
    6  10.0909240    0.0346554
    7  10.0903134    0.0000057
    8  10.0903133    0.0000000
```

inline 함수

- inline() 함수
  - 입력 변수를 함수 형태의 문자열을 받아 인라인 함수로 변환
- Bisection method example

```
a = input('Enter function of x : ', 's');
func = inline(a);
a = input('Enter first point of interval : ');
b = input('Enter second point of interval : ');

eps = 1e-6; % epsilon
iter = 1;   % iteration number

% 나머지는 동일
% ...
```

```
>> bisect2
Enter function of x : cos(x) + 2*sin(x) + x^2
Enter first point of interval : -1
Enter second point of interval : 1
   k         a           b          f(c)
-------------------------------------------
    1 -1.0000000   0.0000000    1.0000000
    2 -1.0000000  -0.5000000    0.1687315
    3 -0.7500000  -0.5000000   -0.0690887
…
   21 -0.6592665  -0.6592655   -0.0000004
```

find 함수

- find()
  - 입력 배열 중에서 조건을 충족하는 배열의 요소들만 반환

```
>> nn = 1:10
nn =
    1    2    3    4    5    6    7    8    9   10

>> find(x > 6)
ans =
    7    8    9   10

>> find(mod(nn, 2) == 0)
ans =
    2    4    6    8   10

>> find(mod(nn, 2))
ans =
   1   3   5   7   9
```

Ex: 소수 구하기

- 소수(prime number) 구하기 알고리즘 : 에라토스테네스의 체
  - 2~100사이의 모든 소수를 구하는 경우
    - ➢ 배열에 2:100 = [2, 3, 4, 5, ⋯ , 100]을 저장
    - ➢ 배열의 처음 요소 2는 소수에 포함하고 배열 요소 중에서 2의 배수를 모두 제거한다.
      - ✓ [2, 3, 4, 5, ⋯ , 100] ⇒ [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, ⋯ , 99], 소수 : [2]
    - ➢ 배열의 처음 요소 3은 소수에 포함하고 배열 요소 중에서 3의 배수를 모두 제거한다.
      - ✓ [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, ⋯ , 99] ⇒ [5, 7, 11, 13, 17, 19, 23, ⋯ , 97], 소수 : [2, 3]
    - ➢ 배열의 처음 요소 5는 소수에 포함하고 배열 요소 중에서 5의 배수를 모두 제거한다.
      - ✓ [5, 7, 11, 13, 17, 19, 23, ⋯ , 97] ⇒ [7, 11, 13, 17, 19, 23, ⋯ , 97], 소수 : [2, 3, 5]
    - ➢ 배열의 남은 요소가 없을 때까지 반복한다.
      - ✓ ⋯
      - ✓ [97] ⇒ [], 소수 : [2, 3, 5, 7, 11, 13, 17, 19, 23, ⋯ , 89, 97]

Ex: 소수 구하기

- 소수(prime number) 구하기 알고리즘 : 에라토스테네스의 체

```
%-------------------------------------------------------------
% function primes = getprimes(n)
% Get prime numbers between 1 to n
%-------------------------------------------------------------

function primes = getprimes(n)
  nos = 2:n;   % 전체 숫자 배열 만들기
  primes = []; % 소수저장용 배열
  while length(nos) > 0
    % 숫자 배열의 처음 요소를 소수에 추가
    primes = [primes nos(1)];
    % 숫자 배열의 처음 요소로 나누어지는 모든 수 삭제
    nos = nos(find(mod(nos, nos(1))));
  end
end
```

```
>> getprimes(10)
ans =  2   3   5   7
>> getprimes(50)
ans =  2    3    5    7   11   13   17   19   23   29   31   37   41   43   47
```