

# Image Processing

## 실습 1.

2019. 03. 11

김 혁 진

gurwls9628@naver.com

# 실습 소개

- 과목 홈페이지

- 충남대학교 사이버 캠퍼스 ( <http://e-learn.cnu.ac.kr> )

- TA 연락처

- 김혁진
- 공대 5호관 627호 정보보호연구실
- [gurwls9628@naver.com](mailto:gurwls9628@naver.com)
  - [IP] 를 메일 제목에 붙여주세요
  - 과제 질문은 메일로만 해주세요

# 개요

- 실습
  - Matlab 기본
- 목표
  - Matlab에 대한 기본적인 이해
  - RGB 및 GrayScale 이미지에 대한 이해
- 구현
  - Rgb2Gray

# MatLab

**MATLAB**(매트랩)은 MathWorks사에서 개발한 수치 해석 및 프로그래밍 환경을 제공하는 공학용 소프트웨어이다. 행렬을 기반으로 한 계산 기능을 지원하며, 함수나 데이터를 그림으로 그리는 기능 및 프로그래밍을 통한 알고리즘 구현 등을 제공한다. MATLAB은 수치 계산이 필요한 과학 및 공학 분야에서 다양하게 사용된다. 30일간의 무료 체험판을 사용해 볼 수도 있다.

- wikipedia

# MatLab

Mar 2019	Mar 2018	Change	Programming Language	Ratings	Change
1	1		Java	14.880%	-0.06%
2	2		C	13.305%	+0.55%
3	4	⬆	Python	8.262%	+2.39%
4	3	⬇	C++	8.126%	+1.67%
5	6	⬆	Visual Basic .NET	6.429%	+2.34%
6	5	⬇	C#	3.267%	-1.80%
7	8	⬆	JavaScript	2.426%	-1.49%
8	7	⬇	PHP	2.420%	-1.59%
9	10	⬆	SQL	1.926%	-0.76%
10	14	⬆	Objective-C	1.681%	-0.09%
11	18	⬆	MATLAB	1.469%	+0.06%
12	16	⬆	Assembly language	1.413%	-0.29%

# MatLab

- 유료
- 학생 약 67,000 원 정도
- 실습실에서 과제를 다 하고 가는 것을 추천

## MATLAB and Simulink Student Suite

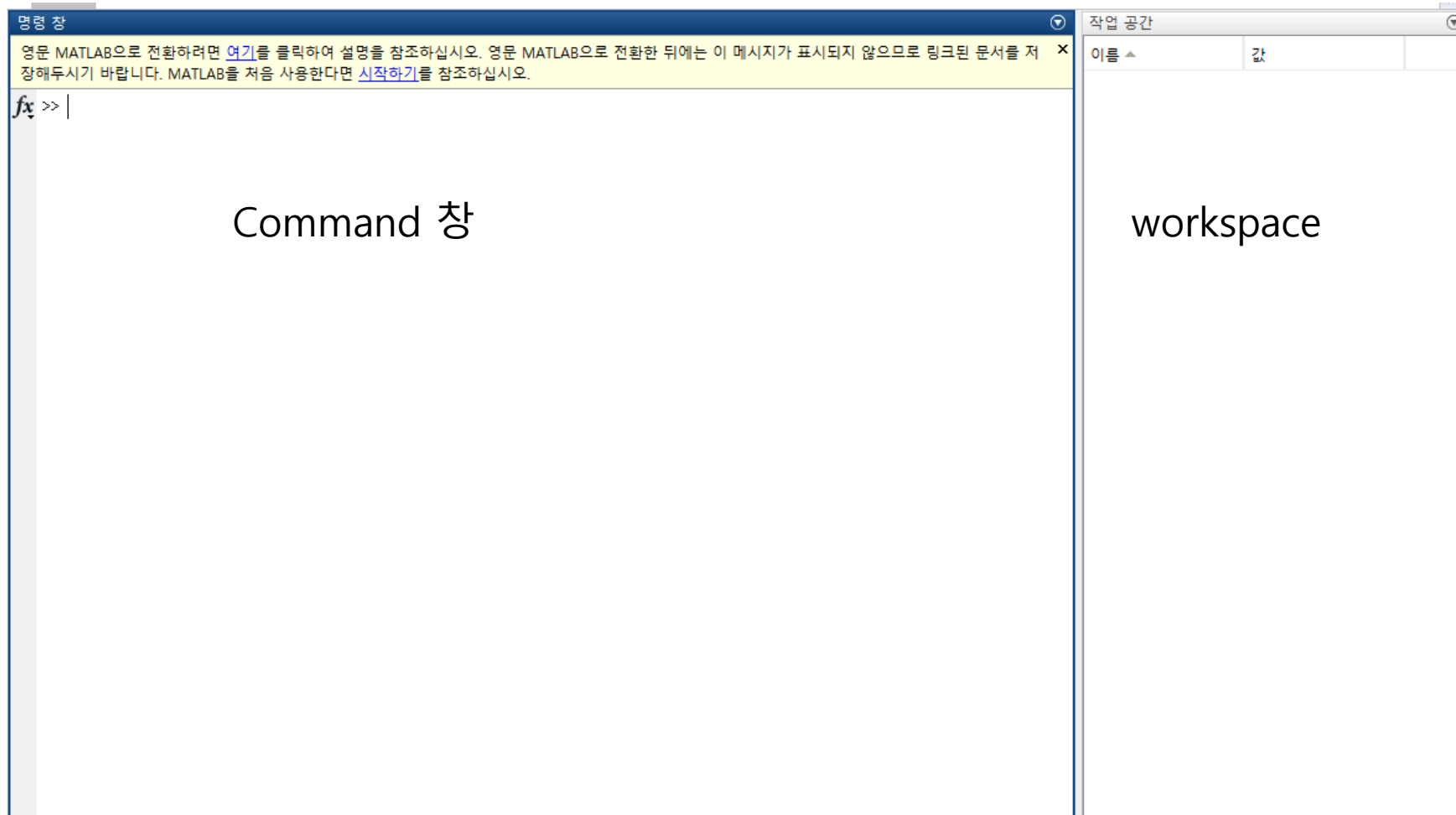
MATLAB, Simulink를 비롯하여 가장 널리 쓰이는 10가지 애드온 제품 제공은 물론, 저렴한 타겟 하드웨어에서의 프로토타이핑, 테스트 및 모델 실행을 기본 지원합니다.

지금 USD 59

# 과제 제출 시

- 다른 언어로 제출 가능
  - C, C++, Java, Python, Go, JavaScript 까지는 사용 가능
  - 이 외의 언어는 받지 않습니다 (Ruby, R, Rust, kotlin, scala, C# 등 안 받습니다)
- 실습은 MatLab으로 진행하지만 위의 언어로 구현해서 제출해도 인정
- 되도록이면 MatLab으로 진행할 것을 추천

# Matlab





# Matlab 기본 - 정의문

- **a = 1**
  - 변수에 값을 정의
  - 변수의 타입을 지정하지 않아도 됨
  - 타입은 자동으로 구분
  - 정의 끝에 ';' 을 붙이면 print를 막을 수 있음
- **A = [1 2 3], B = [1;2;3]**
  - 행렬 값을 정의
  - ';' 을 통해 행과 열을 구분
  - 띄어쓰기가 아닌 ,로 구분해도 됨
- **ans**
  - 결과를 변수로 받지 않으면 ans라는 변수가 받음

```
>> a = 2.1
```

```
a =
```

```
2.1000
```

```
>> A = [1 2 3], B = [1;2;3]
```

```
A =
```

```
1     2     3
```

```
B =
```

```
1  
2  
3
```

# Matlab 기본 - 행렬 1

- **A = 1:10**
  - Python의 range함수처럼 범위의 행렬을 생성 가능
  - For문을 돌릴 때 주로 사용
  - []로 감싸도 되고 아니어도 됨
- **A = [1:10; 2:2:20]**
  - Python range와 거의 흡사
  - 반복할 때 간격도 정할 수 있음
  - 숫자 없이 ':'만 사용할 경우 해당 행 혹은 열을 모두 지정

```
>> A = 1:10
```

```
A =
```

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

```
>> A = [1:10; 2:2:20]
```

```
A =
```

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20

## Matlab 기본 - 행렬 2

- **A(2,5)**
  - 배열과 비슷하지만 Matlab은 1부터 시작
  - 행과 열을 지정해서 값을 얻을 수 있음
  - 값을 해당 위치에 넣는 것도 가능
- **A(1:2, 1:5)**
  - 연속적인 값도 뽑아낼 수 있음
  - 값 정의도 가능하지만 차원 수를 맞춰야 함
  - ➔ 행렬이 크기가 작아지게 할 수 없음  
2x5 -> 2x3 같은 경우는 불가능

```
>> A(2,5)
```

```
ans =
```

```
10
```

```
>> A(1:2, 1:5)
```

```
ans =
```

1	2	3	4	5
2	4	6	8	10

## Matlab 기본 - 행렬 3

- **A > 5**
  - 논리 연산자를 사용 시 행렬에 모두 적용
  - 각 논리연산자를 적용한 값이 행렬에 0 혹은 1로 표시됨
  - 1은 참
  - 0은 거짓
- **A(A>5)**
  - A의 조건문에 맞는 값을 가져오게 됨

```
>> A = 1:10

A =

     1     2     3     4     5     6     7     8     9    10

>> A > 5

ans =

     0     0     0     0     0     1     1     1     1     1

>> A(A>5)

ans =

     6     7     8     9    10
```

# Matlab 기본 - 행렬 4

- **A \* B**
  - 행렬곱으로 연산
- **A .\* B**
  - 연산 앞에 ' .' 을 붙일 경우 각 위치의 값끼리 연산
  - 값 정의도 가능하지만 차원 수를 맞춰야 함

$m \times n$ 행렬과  $n \times o$ 행렬을 곱하면  $m \times o$ 행렬이 된다.<sup>[2]</sup>

$$\begin{array}{c} 4 \times 2 \text{행렬} \\ \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \end{array} \times \begin{array}{c} 2 \times 3 \text{행렬} \\ \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \end{array} = \begin{array}{c} 4 \times 3 \text{행렬} \\ \begin{bmatrix} 1a + 2d & 1b + 2e & 1c + 2f \\ 3a + 4d & 3b + 4e & 3c + 4f \\ 5a + 6d & 5b + 6e & 5c + 6f \\ 7a + 8d & 7b + 8e & 7c + 8f \end{bmatrix} \end{array}$$

```
>> A = [1 0; 0 1]
```

```
A =
```

```
1    0
0    1
```

```
>> B = [2 2; 2 2]
```

```
B =
```

```
2    2
2    2
```

```
>> A*B
```

```
ans =
```

```
2    2
2    2
```

```
>> A.*B
```

```
ans =
```

```
2    0
0    2
```

# Matlab 기본 – 내장 함수 1

- $I = \text{eye}(4)$ ,  $I = \text{eye}(2, 5)$ 
  - 단위 행렬을 만드는 함수
  - Identity Matrix
  - 단위 행렬을 주로  $I$ 로 사용해서 `eye`라는 함수로 만들어짐
  - 행렬의 곱셈에 대한 항등원 → 곱해도 같은 값이 나옴

$$\begin{aligned}
 AE &= \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
 &= \begin{pmatrix} a \times 1 + b \times 0 & a \times 0 + b \times 1 \\ c \times 1 + d \times 0 & c \times 0 + d \times 1 \end{pmatrix} \\
 &= \begin{pmatrix} a & b \\ c & d \end{pmatrix}
 \end{aligned}$$

```
>> I = eye(4)
```

```
I =
```

```

1     0     0     0
0     1     0     0
0     0     1     0
0     0     0     1

```

```
>> I = eye(2, 5)
```

```
I =
```

```

1     0     0     0     0
0     1     0     0     0

```

## Matlab 기본 – 내장 함수 2

- **A = zeros(5), A = zeros(2, 5)**
  - 0으로 채워진 행렬을 얻는 함수
- **A = ones(5), A = ones(2, 5)**
  - 1으로 채워진 행렬을 얻는 함수
- **size(A)**
  - 행렬의 크기를 얻는 함수
  - 변수의 경우 1, 1의 크기

```
>> A = zeros(5)

A =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0

>> A = zeros(2, 5)

A =

     0     0     0     0     0
     0     0     0     0     0
```

```
>> size(A)

ans =

      2      5

>> a = 1

a =

      1

>> size(a)

ans =

      1      1
```

# Matlab 기본 – 내장 함수 3

- **A = magic(5)**
  - 마방진을 만드는 함수
  - 행, 열, 대각선의 합이 모두 같은 값

2	7	6	→15	
9	5	1	→15	
4	3	8	→15	
↙15	↓15	↓15	↓15	↘15

```
>> magic(5)
```

```
ans =
```

```
17    24     1     8    15
23     5     7    14    16
 4     6    13    20    22
10    12    19    21     3
11    18    25     2     9
```

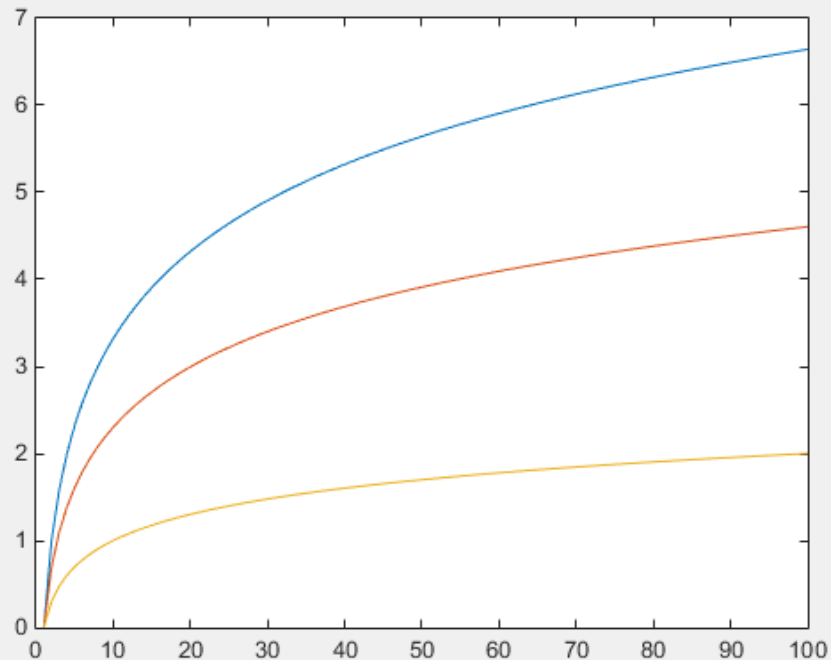
```
>> magic(5, 2)
```

다음 사용 중 오류가 발생함: **magic**  
입력 인수가 너무 많습니다.



## Matlab 기본 - 내장 함수 4

- **A = log(2), log2(2), log10(2)**
  - Log 연산 함수
  - Log의 밑을 e, 2, 10으로 지정 가능
  - 행렬에도 적용 가능



```
>> log(10)
```

```
ans =
```

```
2.3026
```

```
>> log(2)
```

```
ans =
```

```
0.6931
```

```
>> log2(2)
```

```
ans =
```

```
1
```

```
>> log10(2)
```

```
ans =
```

```
0.3010
```

# Matlab 기본 – 내장 함수 5

- **A = sin(x), cos(x), tan(x)**
  - sin, cos, tan의 삼각함수를 적용하는 함수
- **A = asin(x), acos(x), atan(x)**
  - arcsin, arccos, arctan의 역삼각함수를 적용하는 함수
  - 삼각함수의 역함수
- **A = sqrt(x)**
  - 제곱근을 적용하는 함수
- **A = abs(x)**
  - 절대값으로 만드는 함수

## Matlab 기본 – 내장 함수 6

- **A = exp(2)**
  - 자연로그의 지수함수를 구하는 함수
- **A = floor(3.5)**
  - 내림 함수
- **A = ceil(3.2)**
  - 올림 함수
- **A = round(x)**
  - 반올림 함수
- **A = mod(x)**
  - 나머지연산 함수 (%를 사용하지 않음) % 는 주석

```
>> 5%2
```

```
ans =
```

```
5
```

```
>> 2%5
```

```
ans =
```

```
2
```

# Matlab 기본 - 내장 함수 7

- **length(x)**
  - X 행렬의 길이를 구하는 함수
- **disp('Hello World')**
  - Print함수
  - Command 창에 입력받은 문자열 혹은 숫자를 출력해주는 함수
- **sprintf**
  - 문자열을 formatting을 해서 문자열을 내보내는 함수
- **fprintf**
  - fprintf의 값을 출력하는 함수

```
>> sprintf('Hello, %s', 'Image Processing')
```

```
ans =
```

```
Hello, Image Processing
```

```
>> fprintf('Hello, %s', 'Image Processing')
```

```
Hello, Image Processing>>
```

```
>> disp('Hello, %s', 'Image Processing')
```

다음 사용 중 오류가 발생함: **disp**

입력 인수가 너무 많습니다.

# Matlab 기본 - 조건문

- If 조건문

- If, elseif, else 로 조건문을 여러 개 나열 가능
- 조건문이 모두 끝나고 end로 끝내야 함

```
if a < 0
    a = -a;
elseif a > 0
    a = +a;
else
    a = 0;
end
```

# Matlab 기본 - 반복문

- **For 반복문**

- 행렬만큼 반복하게 할 수 있음
- ':' 와 함께 사용하여 foreach문처럼 사용 가능

- **While 반복문**

- 조건을 만족할 때까지 반복
- 무한 루프 조심
- 무한 루프 발생 시 ctrl+c로 캔슬 가능

```
a = zeros(1, 10);  
  
for i = 1:10  
    if mod(i, 2) == 1  
        a(1, i) = i+1;  
    else  
        a(1, i) = i;  
    end  
end
```

# Matlab 기본 – plotting 1

- Plotting

- 만들어진 행렬 값을 이용해 plotting
  - 여러가지 값을 동시에 출력도 가능
  - Figure를 호출 시 새 창이 뜸
- ➔ 이를 이용해 여러 창에 plotting 가능  
plot(x축, y축, x축2, y축2, ...) 로 사용

```
x = 1:100;
```

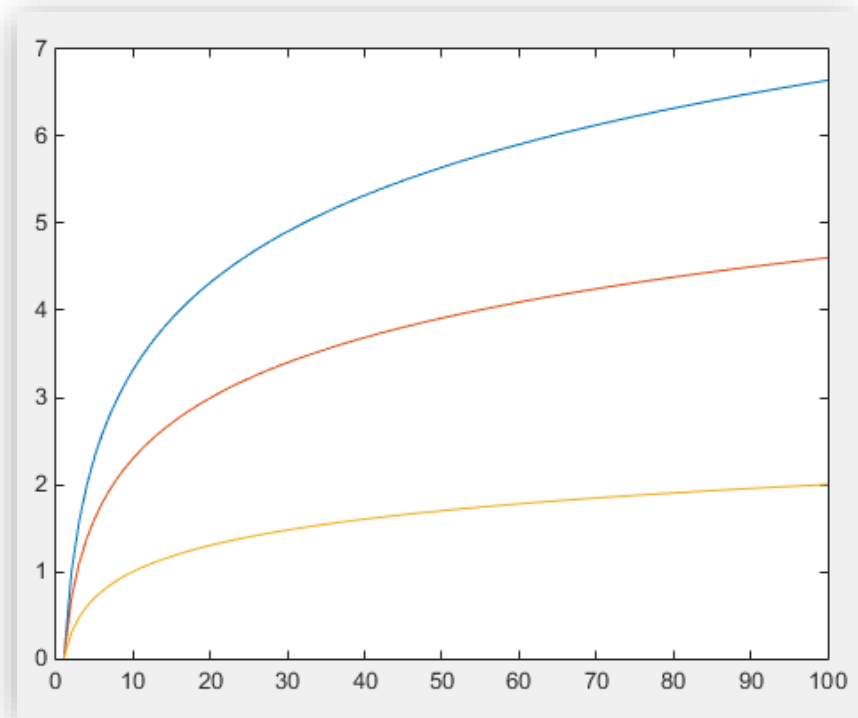
```
y1 = log2(x);
```

```
y2 = log(x);
```

```
y3 = log10(x);
```

```
figure % plotting할 window 생성
```

```
plot(x, y1, x, y2, x, y3)
```



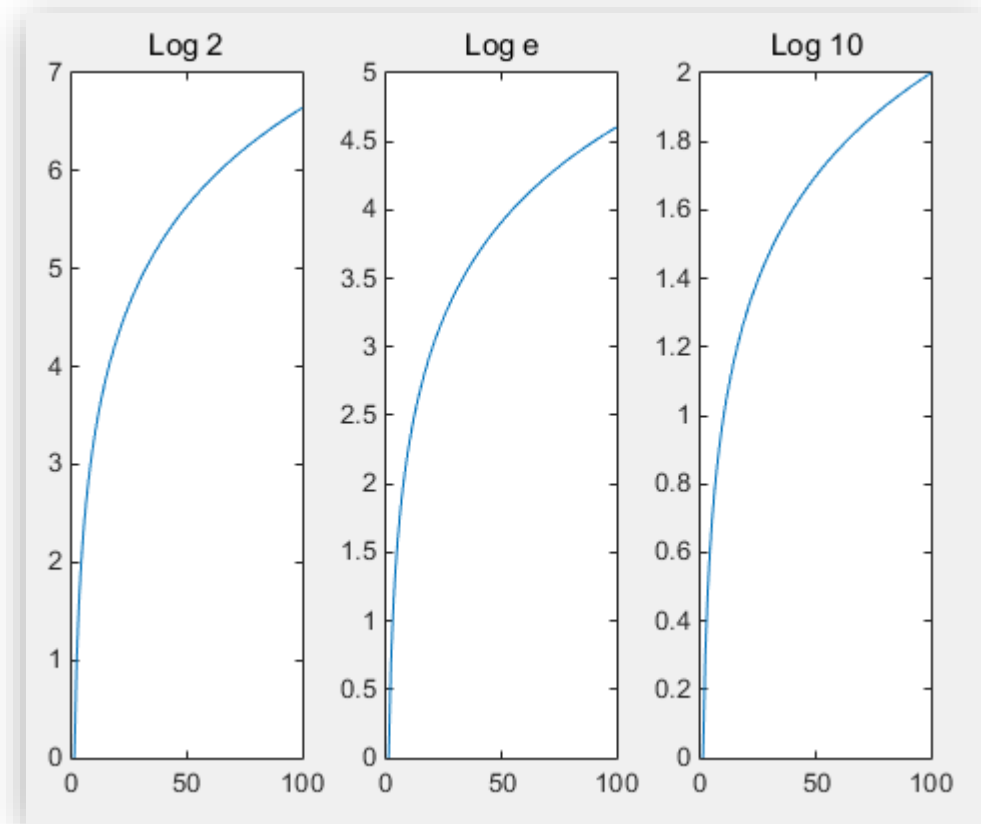
# Matlab 기본 – plotting 2

- **subPlotting**

- Window를 grid로 나누어서 plotting
- Subplot(행, 열, index)로 사용 가능
- 전체 window를 입력한 행, 열로 나누고 index번째 창에 plotting

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

```
>> subplot(1,3, 1);  
>> plot(x, y1);  
>> title('Log 2')  
>>  
>> subplot(1, 3, 2);  
>> plot(x, y2);  
>> title('Log e')  
>>  
>> subplot(1, 3, 3);  
>> plot(x, y3);  
>> title('Log 10')
```

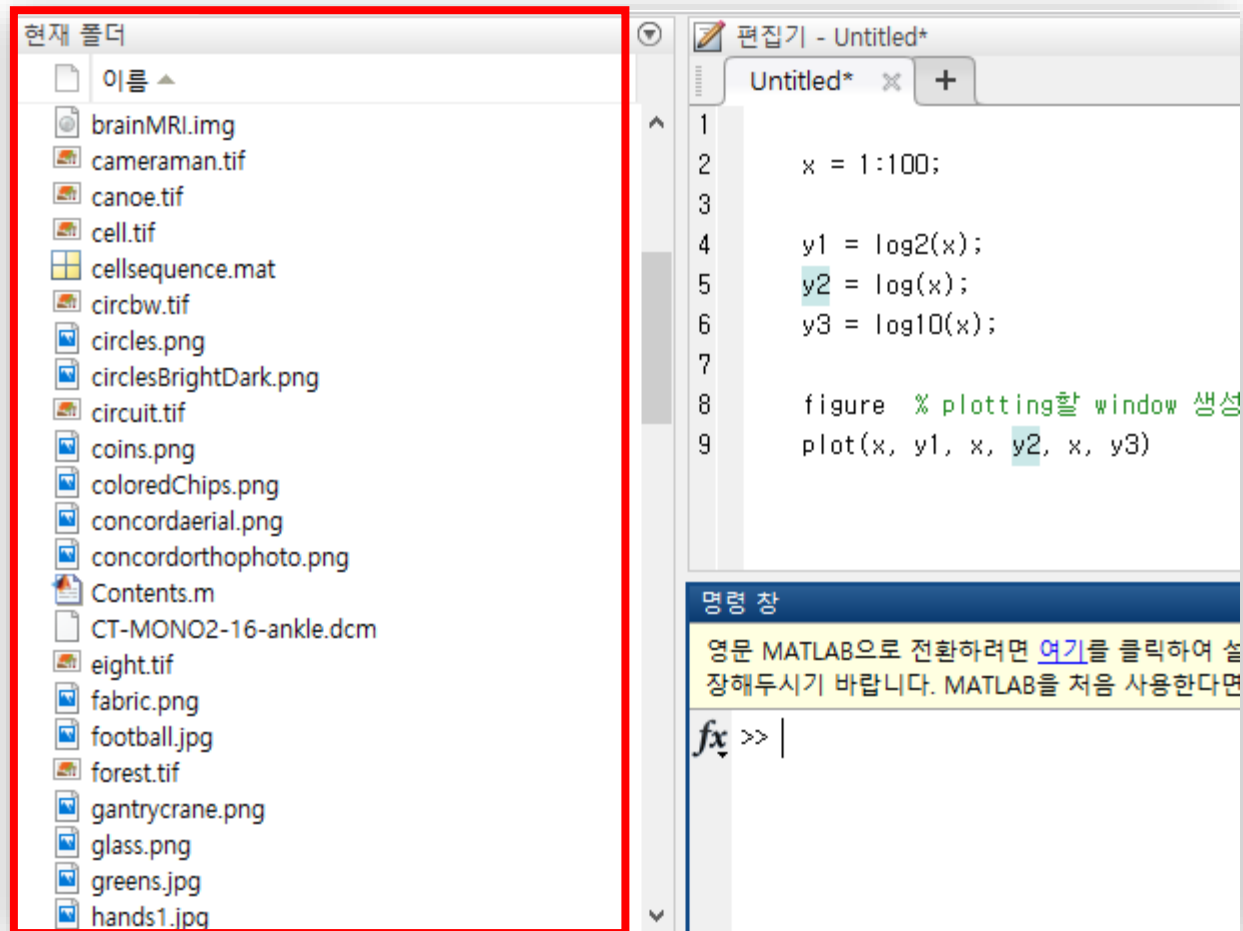




# Matlab 기본 – image 1

- **imread**

- 이미지를 읽는 함수
- 현재 지정된 폴더를 기준으로 경로설정하여 이미지를 읽음
- ';'을 붙이지 않으면 이미지의 행렬 값이 모두 출력됨
- MATLAB 내부 image 폴더에서 이미지가 있다면 내부 image 폴더에서 이미지를 찾음



# Matlab 기본 – image 2

- **imshow**

- 행렬을 이미지로 보고 이미지 출력을 하는 함수
- 행렬을 입력으로 함

```
> RGB = imread('peppers.png');  
> imshow(RGB)
```

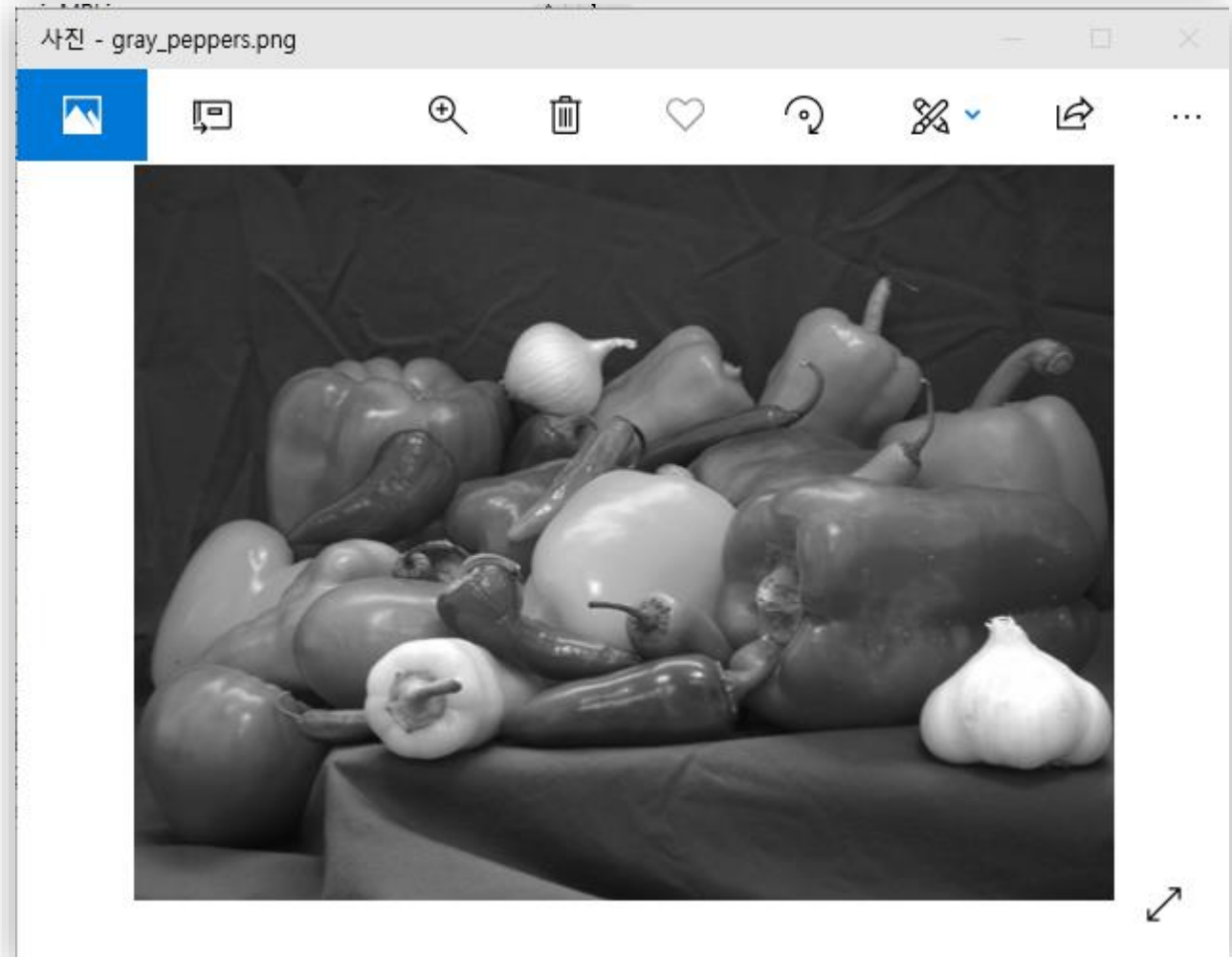


# Matlab 기본 – image 3

- **imwrite**

- 행렬을 이미지로 저장함
- 행렬을 입력으로 함
- 이미지 Format 지정 가능

```
gray = rgb2gray(RGB);  
imwrite(gray, 'gray_peppers.png')
```



# Matlab 기본 – image 4

- **imfinfo**

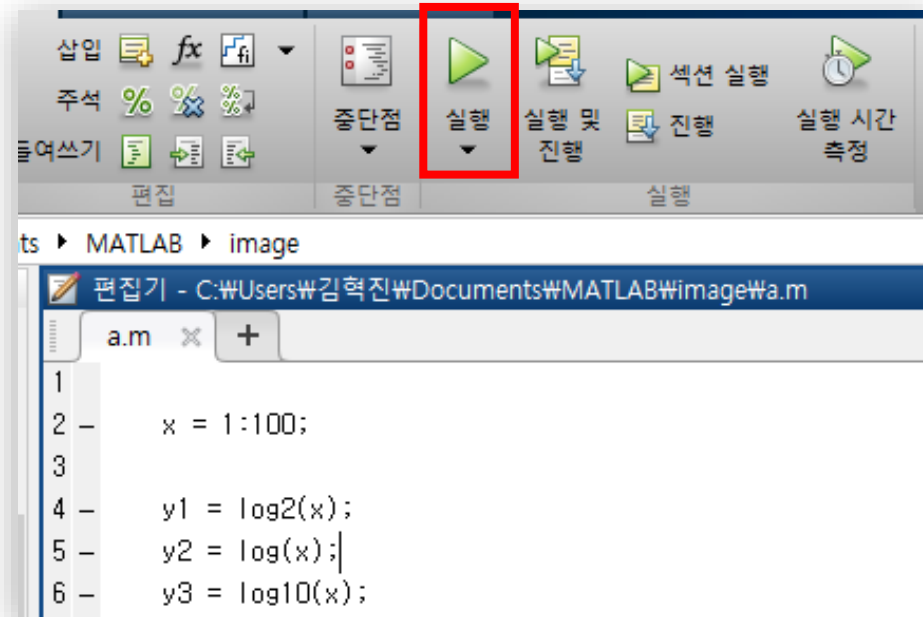
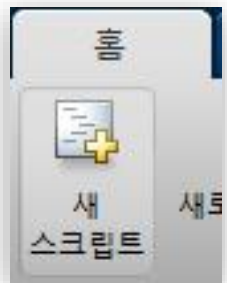
- 이미지의 정보를 보는 함수
- Format, filesize 등 여러 정보 출력

```
>> imfinfo('gray_peppers.png')  
  
ans =  
  
          Filename: 'C:\Users\#;  
    FileModDate: '09-Mar-20'  
      FileSize: 89147  
        Format: 'png'  
   FormatVersion: []  
         Width: 512  
        Height: 384  
      BitDepth: 8  
     ColorType: 'grayscale'
```

```
>> imfinfo('peppers.png')  
  
ans =  
  
          Filename: 'C:\Program  
    FileModDate: '16-Dec-2002  
      FileSize: 287677  
        Format: 'png'  
   FormatVersion: []  
         Width: 512  
        Height: 384  
      BitDepth: 24  
     ColorType: 'truecolor'
```

# Matlab 기본 - m 파일

- 홈/새 스크립트 버튼을 눌러 m 파일 작성 가능
- Command 창에서 실행하는 것은 script로 진행됨
  - 다시 실행하기 어렵고, 한번 잘못했을 때 처음부터 재실행 해야함
  - .m파일을 만들어 .m파일을 실행
  - Python, JavaScript를 파일로 만들어 실행하는 것과 같은 이유
  - 실행버튼으로 m파일 실행 가능



# Matlab 기본 – function

- **function**

- 함수를 만들 때에 사용
- 함수 결과와 인자를 지정
- .m파일을 만들고 저장 시 스크립트에서 함수 호출 가능  
.m파일에서만 function 선언 가능. 스크립트에서는 불가능
- 함수의 인자는 reference되지 않음

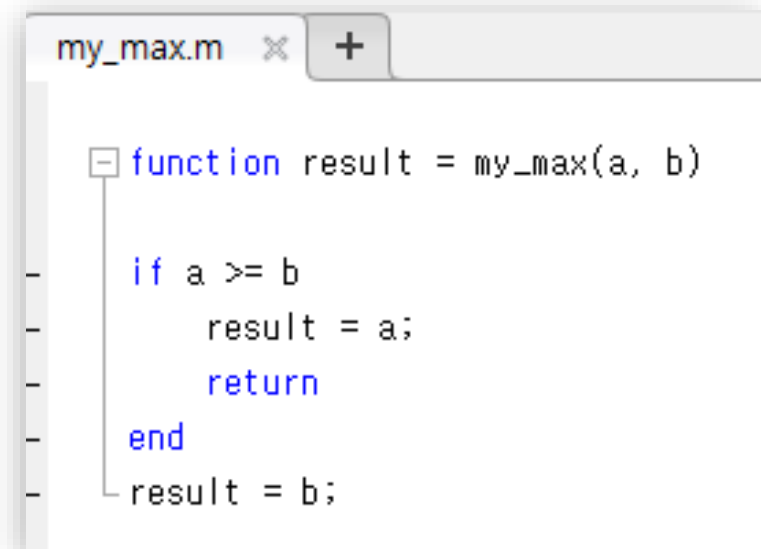
- **return**

- C, Java, python 등의 return과 달리  
함수를 종료하는 역할만 함
- return할 인자는 함수의 result를 지정한 인자로  
값을 정의해 사용

```
>> my_max(2, 3)
```

```
ans =
```

```
3
```



```
my_max.m x +  
  
function result = my_max(a, b)  
    if a >= b  
        result = a;  
        return  
    end  
    result = b;
```

# Matlab 기본 – 기타

- **clc**
  - Command창을 초기화
  - 화면 정리용
- **clear**
  - Workspace를 초기화함
  - 다시 처음부터 하고싶을 때 사용

이외의 다른 것은 google에

<https://kr.mathworks.com/help/matlab/getting-started-with-matlab.html>

## rgb2gray

- RGB 영상의 R, G, B 값을 이용하면 gray scale 영상을 만들 수 있음
  - R, G, B 값을 계산하여 하나의 값으로 만듦
- 단, 구현에 따라 영상의 품질이 다르게 보일 수 있음
  - 사람의 눈이 R, G, B 값에 반응하는 정도가 각각 다름





# rgb2gray

- **rgb2gray 내장 함수를 사용하지 않고 rgb2gray 함수를 구현**
  - .m파일로 my\_rgb2gray 함수를 만들어 제출
    - 입력 값은 (x, y, 3)인 RGB 이미지 행렬
    - 출력 결과는 (x, y)인 grayscale 이미지 행렬
  - my\_rgb2gray.m 파일을 만들어 제출
  - 예외처리 하지 않아도 됨. 입력은 항상 RGB 이미지로 가정
  - 주석 굳이 달지 않아도 됨 (보고서에 정리)
- **채점 기준**
  - Gray Scale 영상의 품질
  - 내장 함수 사용시 과제 1점 (과제를 진행하기 어렵다면 내장함수라도 사용해 제출 )  
미구현으로 제출 시 0점
  - 다른 언어로 구현하더라도 구현 기능의 핵심 부분은 직접 계산하여 구현해야 함  
(라이브러리 혹은 내장 함수를 통해 핵심기능 구현 시 1점)

# 과제

- 보고서

- 내용:

- 이름, 학번, 학과
    - 구현 내용: 구현한 내용과 방법에 대한 설명
    - 구현 이유: 구현한 방법에 대한 이유
    - 느낀 점: 구현하면서 느낀 점, 혹은 어려운 점
    - 과제 난이도: 개인적으로 생각하는 난이도 및 이유 (과제가 너무 쉬운 것 같다 등..)

- .pdf 파일로 제출 (이 외의 파일 형식일 경우 감점)
  - 다른 언어로 제출 시 조금 더 자세하게 설명

- 파일 이름:

- [IP]20xxxxxxx\_이름\_1주차\_과제.pdf

# 과제

- **제출 기한**

- 3월 17일 23시 59분까지

- **추가 제출 기한**

- 3월 18일 0시 10분까지 (최대 점수 9점, 과제 총점 계산 후 -1점)
- 3월 18일 0시 20분까지 (최대 점수 8점, 과제 총점 계산 후 -2점)
- 3월 18일 0시 30분까지 (최대 점수 7점, 과제 총점 계산 후 -3점)
- 3월 18일 0시 40분까지 (최대 점수 6점, 과제 총점 계산 후 -4점)
- 3월 18일 0시 50분까지 (최대 점수 5점, 과제 총점 계산 후 -5점)
- 3월 24일 23시 59분까지 (최대 점수 4점, 과제 총점 계산 후 -6점)

- Ex: 과제 점수가 1점인데 과제 3월 18일 24분 제출시 0점

## 과제 요약

### 1. **rgb2gray** 내장 함수를 사용하지 않고 **rgb2gray** 함수를 구현

- .m파일로 my\_rgb2gray 함수를 만들어 .m파일 제출

### • 채점 기준

- Rgb2gray 채점 기준 + 보고서

### • 제출 파일

- my\_rgb2gray.m 파일
- .pdf 보고서 파일
- 위의 파일을 압축해서 [IP]20xxxxxxx\_이름\_1주차\_과제.zip로 제출