Python 수업 2회차

오늘의 목표

- 딥러닝의 기초를 이해한다.
- 파이썬의 기초를 배운다.
- 본 세미나에서 사용된 책
  - 밑바닥부터 시작하는 딥러닝

1. Python에서 Numpy란?

- 과학 계산을 위한 라이브러리, 다차원 배열을 처리하는데 필요한 여러 유용한 기능을 제공
- 딥러닝의 배열이나 행렬 계산에 많이 등장.

- Numpy의 배열클래스인 numpy.array에는 관련된 메서드가 많이 존재하여 사용한다.

- numpy 모듈 가져오기

```
>> import numpy as np
```

- 많은 곳에서 np로 사용한다.

2. Python에서 Matplotlib란?

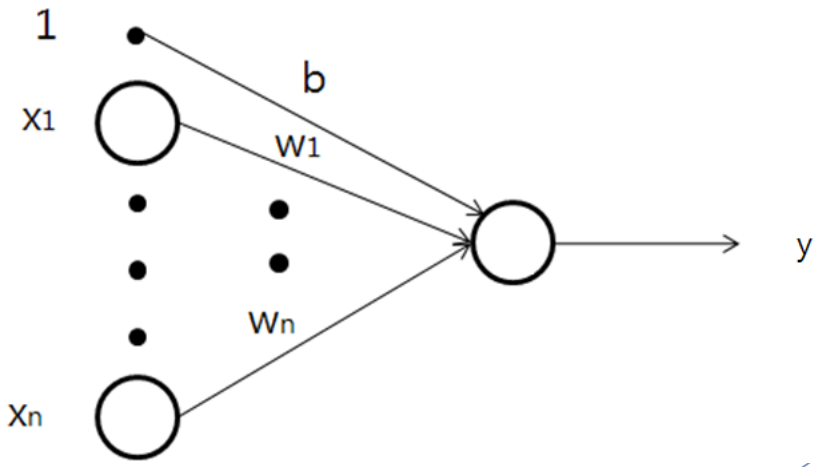
- Matplotlib는 파이썬에서 자료를 차트(chart)나 플롯(plot)으로 시각화(visulaization)하는 패키지.

- Ex) 라인 플롯(line plot), 스캐터 플롯(scatter plot), 컨투어 플롯(contour plot), 서피스 플롯(surface plot), 바 차트(bar chart), 히스토그램(histogram), 박스 플롯(box plot)

- 참고 : https://matplotlib.org/gallery.html

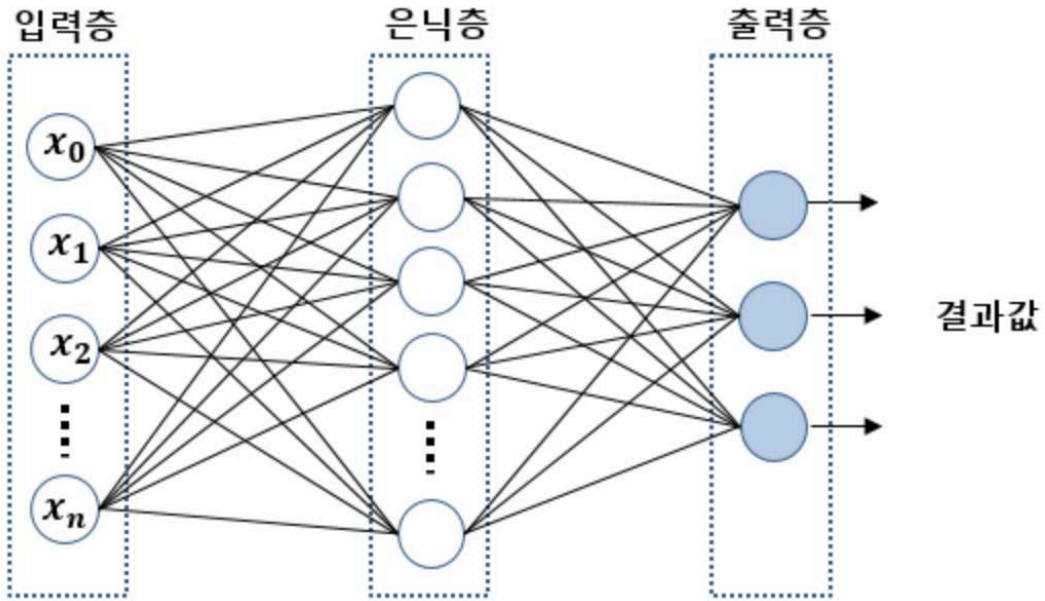
3. 퍼셉트론이란?

- 신경망(딥러닝)의 기원이 되는 알고리즘
- 가중치와 편향을 매개변수로 설정함
- 다수의 입력을 받아 그 가중치 값을 곱해 총합을 구함. 그 총합이 정해진 한계를 넘을 때 1을 출력한다(= 뉴런이 활성화 한다). = w1x1 + w2x2 > **θ 일때 1이다.**
- 이때 위의 식을 b+ w1x1 + w2x2 >0 일때 1 로 표현하면 이때의 b가 편향 이라 한다.



4. 신경망의 층을 간단히 그림으로

가중치 매개변수의 적절한 값을 자동으로 학습하는 성질을 갖게됨.



5. 일반적인 신경망 학습의 과정은?

- 1. 미니배치
  - 훈련 데이터 무작위 선발
- 2. 기울기 산출
  - 손실함수의 기울기를 산출
- 3. 매개변수 갱신
  - 더 최적의 매개변수로 갱신
- 4. 1~3반복.

6. 활성화 함수가 무엇인지, 그 예시

- 입력신호의 총합을 출력 신호로 변환하는 함수
- 입력신호의 총합이 활성화를 일으키는지를 정하는 역할을 함.

- Ex) 계단함수, 시그모이드 함수, ReLU 함수,

7. 기계학습의 분류와 회귀, 각각 어떤 활성화 함수 사용?

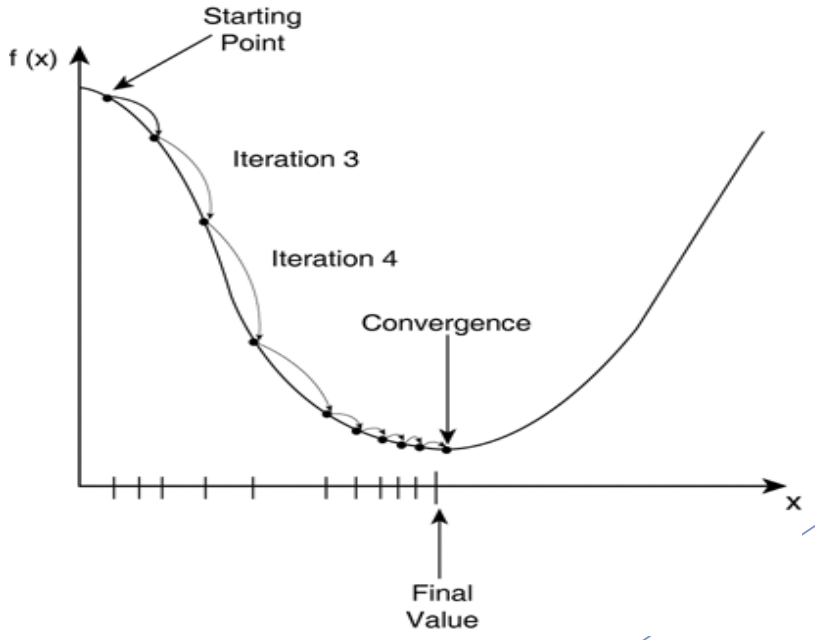
- 분류(어느 클래스에 속하느냐) : 소프트맥스
  - 0 ~ 1.0 사이의 실수가 나오기에 확률적 해석이 가능.

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

- 회귀(연속적 수치 예측) : 항등함수
  - 입력과 출력이 항상 같음.

8. 경사 하강법

- 기계학습의 학습단계에서 최적의 매개변수를 찾아내기 위한 방법.
- 손실함수의 기울기를 이용해 최솟값이 될때의 매개변수 값을 찾으려는 방법.
- 현 위치에서 기울어진 방향으로 일정 거리만큼 이동하여 그곳의 기울기를 구하고, 또 그만큼씩 이동해 나아가는 방법.



9. 오차역전파법

- # 신경망 학습은 다음의 과정을 겪는다.
  - 1. 미니배치-> 2. 기울기 산출-> 3. 매개변수 갱신-> 1~3반복.
- 위의 과정 중 2번을 위한 방법.
- 가중치 매개변수의 기울기를 효율적이고 빠르게 계산하는 방법.
- 데이터를 역방향으로 전파하여 계산하기에 수치 미분 보다 계산이 훨씬 빠름.

10. 오버피팅 이란?

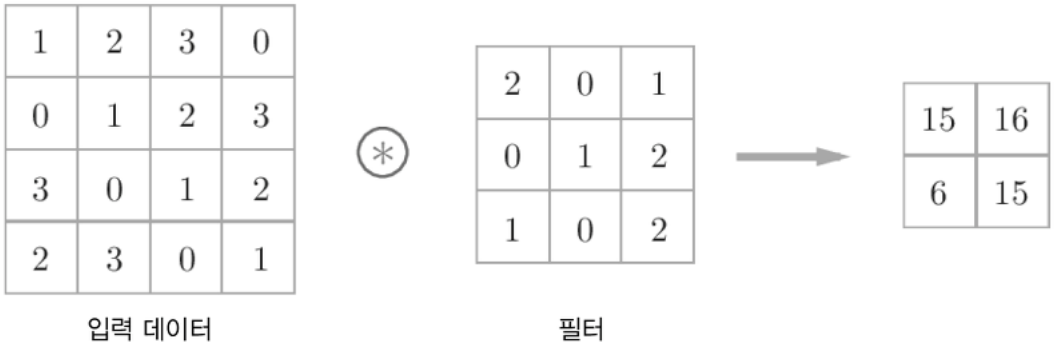
- 신경망이 훈련 데이터에만 지나치게 적응되어 그 외의 데이터에는 제대로 대응하지 못하는 상태
- 범용 성능이 떨어지므로 가치를 잃음
- 매개변수가 많고 표현력이 높거나, 훈련 데이터가 적은 모델에서 주로 일어남.

- 가중치 감소, 드롭아웃으로 억제 가능.

11. CNN의 핵심인 두 계층

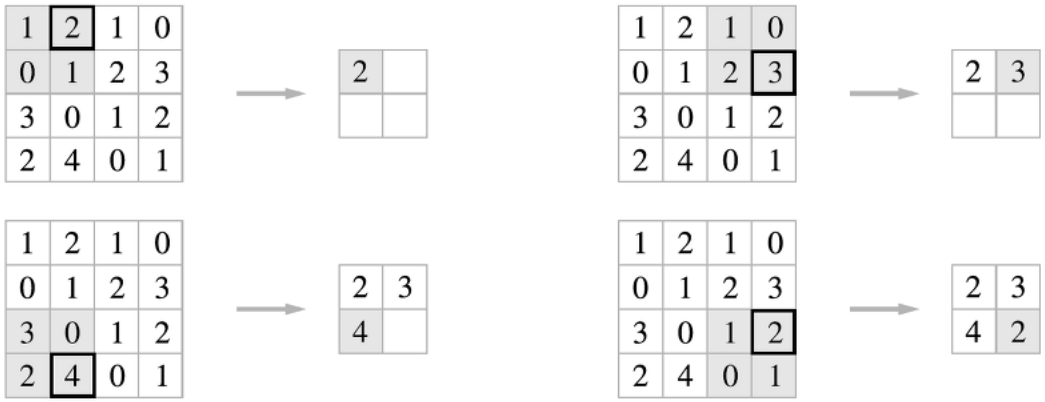
- CNN은 주로 이미지 인식, 음성 인식에 사용.

- 합성곱(convolutional) 계층
  - 필터를 이용해 합성곱 연산을 수행한다. 필터(커널)의 윈도우를 일정 간격으로 이동해가며 입력 데이터에 적용한다.
  - 딥러닝과 달리 데이터 형상이 유지됨.

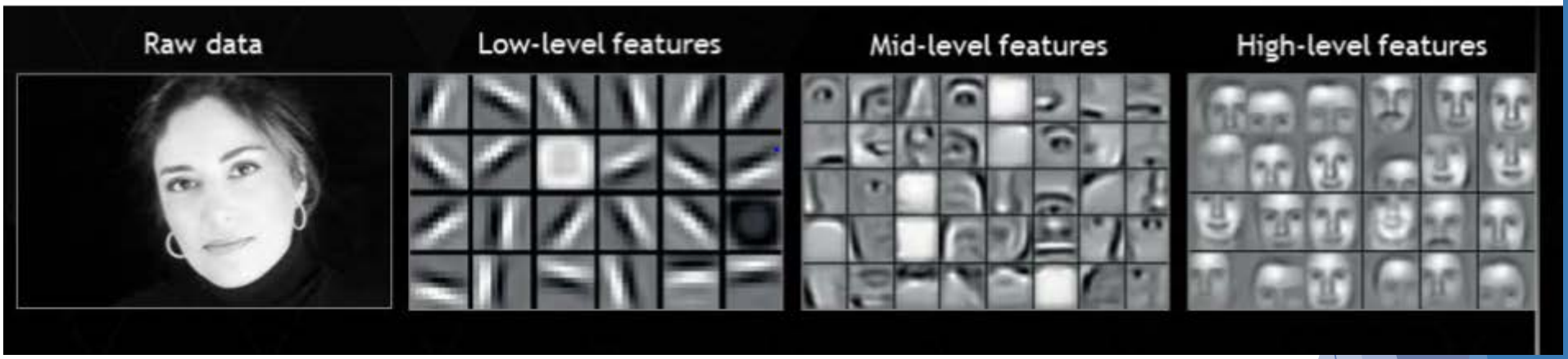


11. CNN의 핵심인 두 계층

- 풀링(pooling) 계층
  - 풀링은 2차원 데이터의 세로 및 가로 방향의 공간을 줄이는 연산
  - 풀링에는 최대 풀링(Max Pooling), 평균 풀링(Average Pooling) 등이 있다.
  - 최대 풀링은 대상 영역에서 최댓값을 취하는 연산, 평균 풀링은 대상 영역의 평균을 계산한다. 이미지 인식 분야에서는 주로 최대 풀링을 사용한다.



11. CNN의 핵심인 두 계층



- Convolutional Neural Network
  - 초기 layer에서는 edge나 line을 학습하고, 그 다음 이미지의 high-level 표현을 습득한다.

12. 교사, 비교사, 강화학습이란?

종류	학습	테스트
교사학습	데이터와 답을 입력.	다른 데이터의 답을 예측. Ex) 고양이
비교사 학습	데이터는 입력. 답은 입력하지 않음.	다른 데이터의 규칙성을 찾음. Ex) 클러스터링
강화학습	부분적으로 답을 입력.	데이터를 기반으로 최적의 답을 찾아냄. Ex) 알파고

https://www.youtube.com/watch?v=V1eYniJ0Rnk

기초-파이썬

- 설치
  - https://www.python.org/
  - python 3버전을 사용한다.
  - Windows x86-64 executable installer
  - 위 링크를 통해 설치하면 된다.
  - 설치시 path를 추가하는 옵션을 체크하고 설치한다.

기초-파이썬

- 산술연산

```
>>> 1-2
-1
>>> 4*5
20
>>> 7/5
1.4
>>> 3**2
9
>>> 1+2
3
>>> 7//5
1
>>> 7%5
2
```

기초-파이썬

- 자료형

```
>>> type(10)
<class 'int'>
>>> type(2.5)
<class 'float'>
>>> type("hello")
<class 'str'>
```

기초-파이썬

- 변수
- 파이썬은 동적언어이다.

```
>>> x=10
>>> print(x)
10
>>> x=100
>>> print(x)
100
>>> y=3.14
>>> x*y
314.0
```

기초-파이썬

- 리스트
- 원소에 접근할 때는 a[index]로 한다.
- index는 0부터 시작한다
- 리스트 내에는 어떤 type이라도 들어올 수 있다.

```
>>> x=10
>>> print(x)
10
>>> x=100
>>> print(x)
100
>>> y=3.14
>>> x*y
314.0
>>> a=[1,2,3,4,5]
>>> print(a)
[1, 2, 3, 4, 5]
>>> len(a)
5
>>> a[0]
1
>>> a[4]
5
>>> a[4]=99
>>> print(a)
[1, 2, 3, 4, 99]
```

기초-파이썬

- 리스트-슬라이싱

```
>>> print(a)
[1, 2, 3, 4, 99]
>>> a[0:2]
[1, 2]
>>> a[1:]
[2, 3, 4, 99]
>>> a[:3]
[1, 2, 3]
>>> a[:-1]
[1, 2, 3, 4]
>>> a[:-2]
[1, 2, 3]
```

기초-파이썬

- 딕셔너리
  - key와 value를 가진다.

```
>>> me = {'height':160}
>>> me['height']
160
>>> me['weight']=40
>>> print(me)
{'height': 160, 'weight': 40}
```

기초-파이썬

- bool
  - True(참)와 False(거짓)
  - and, or, not 연산자를 사용할 수 있다.
  - 또한 +, -, /, *등이 가능하며, 숫자로 인식한다.

```
>>> hungry = True
>>> sleepy = False
>>> type(hungry)
<class 'bool'>
>>> not hungry
False
>>> hungry and sleepy
False
>>> hungry or sleepy
True
>>>
```

기초-파이썬

- if문
  - 조건문, 조건에 따라 처리한다.
  - 들여쓰기(tab or 공백 4개)는 내부 블록을 만든다.
  - pep 8에 따르면, 공백 4개를 사용하도록 권장한다.

```
>>> hungry = True
>>> if hungry:
...     print("I'm hungry")
...
I'm hungry
>>> hungry = False
>>> if hungry:
...     print("I'm hungry")
... else:
...     print("I'm not hungry")
...     print("I'm sleepy")
...
I'm not hungry
I'm sleepy
```

기초-파이썬

- while문

  while(조건문)

    실행문장1

    실행문장2

- 조건문이 참이면 실행문장을 실행한다.
- 조건문에는 비교문같은 것이 들어가거나, 무한 반복일 경우 True를 사용한다.
- break는 반복문을 나가게 만든다.
- continue는 반복문을 한 번 건너뛰게 한다.

기초-파이썬

- for문

```
>>> for i in [1, 2, 3]:
...     print(i)
...
1
2
3
```

기초-파이썬

- 함수

```
>>> def hello():
...     print("Hello World!")
...
>>> hello()
Hello World!
```

기초-파이썬

- class

```
class 클래스 이름:
    def __init__(self, 인수, ...):
        ...
    def 메서드 이름 1(self, 인수, ...):
        ...
    def 메서드 이름 2(self, 인수, ...):
        ...
```

기초-파이썬

- 더 자세한 내용은 https://wikidocs.net/book/1 에서 공부.

기초-IDE

- IDE(Integrated Development Environment)
  - 코딩, 디버그, 컴파일, 배포등 프로그램 개발과 관련된 작업을 하나의 프로그램 안에서 처리하는 환경을 제공하는 소프트웨어이다.
- pycharm
  - https://www.jetbrains.com/pycharm/
  - 파이썬에서 자주 쓰이는 IDE이다.
- Ipython
  - https://ipython.org/
  - 계산 과학에서 많이 사용된다. 대화용으로 되어있다.

기초-IDE

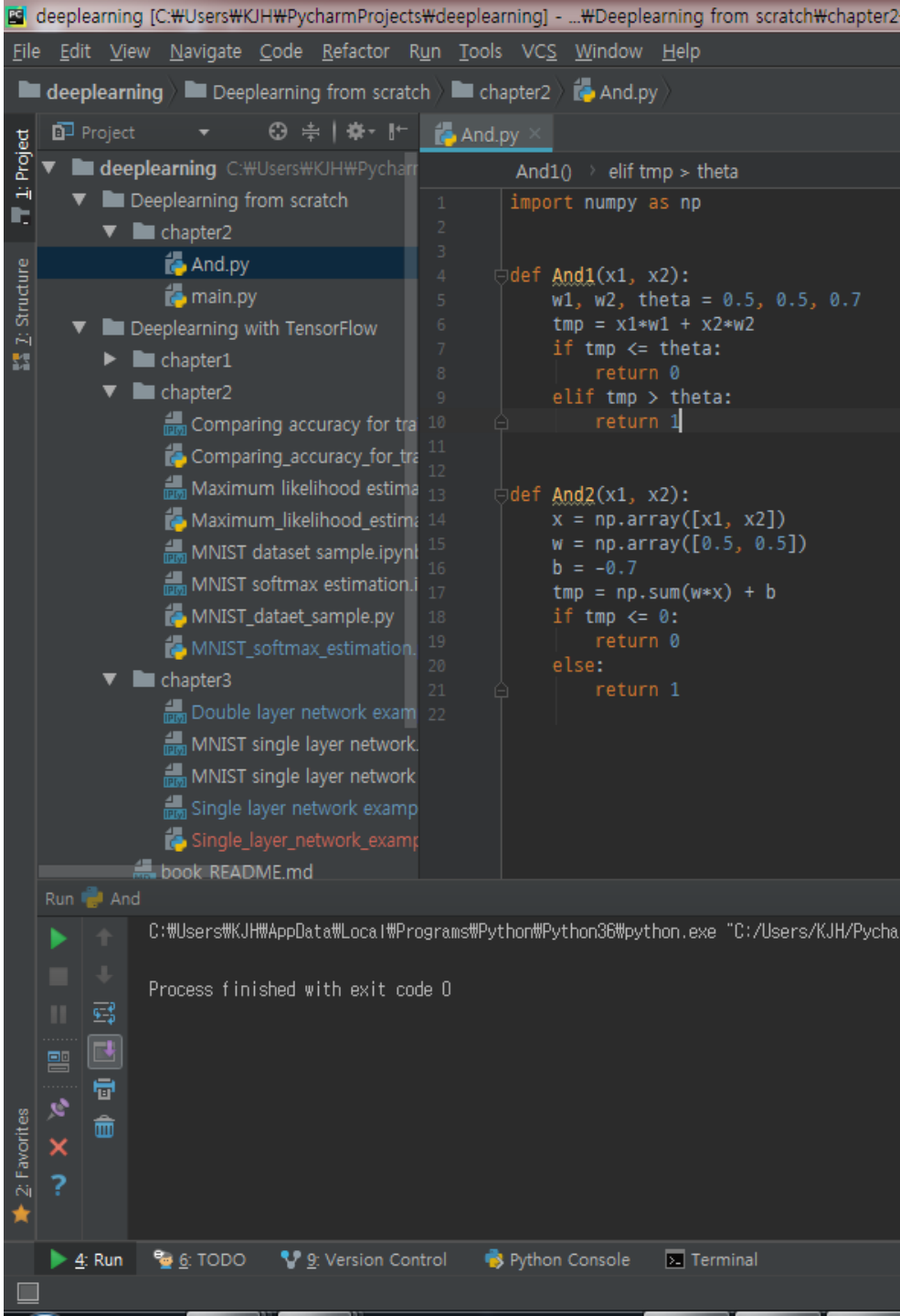
- Jupyter
  - http://jupyter.org/
  - 대화용 도구이다.
  - 서버를 만들어서 접근하는 방식을 취한다.
- notepad++ & cmd
  - https://notepad-plus-plus.org/download/v7.5.4.html
  - 정 느리면 이걸 쓰자.
  - notepad++의 언어를 python으로 변경해주면, python문법에 맞춰서 오류를 잡아준다.
  - 실행은 명령프롬프트에서 cd명령어로 이동하여 .py를 통해 실행한다.
- IDLE
  - 파이썬에서 기본적으로 주는 대화형 프롬프트이다.

기초-IDE

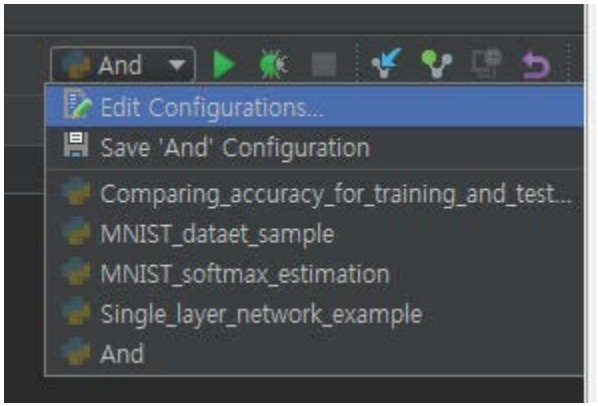
- D2coding
  - coding때 사용하기 좋은 font
  - I(대문자 I), l(소문자 L), 0(영)과 O(대문자 오) 등을 구분하기 쉽게 해준다.
  - 또한 고정폭 글자여서 코딩에선 보기 좋다.

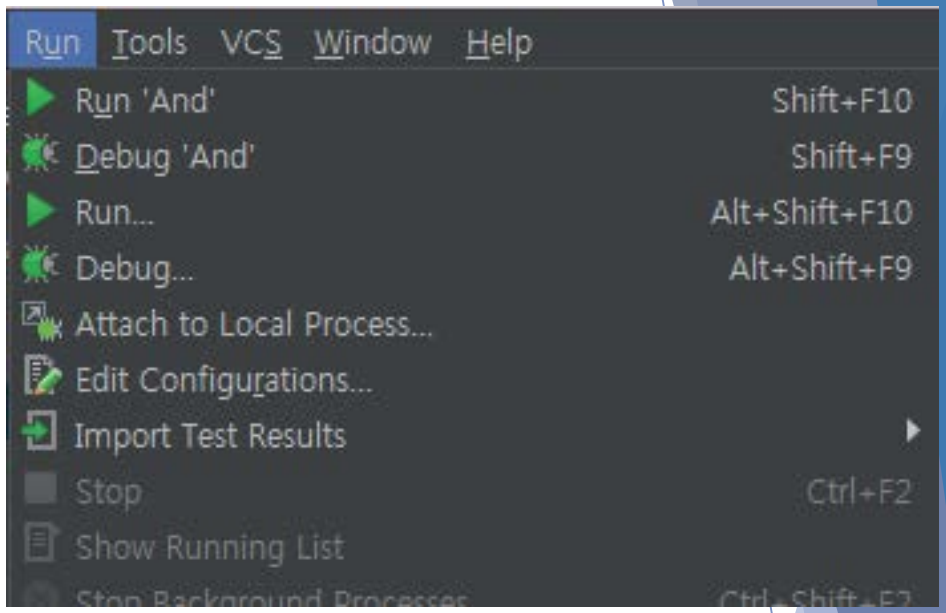
기초-pycharm

- pycharm은 프로젝트 단위로 관리하며, 프로젝트는 내부에 폴더 단위로 관리할 수 있다.
- 또한, 여기서 관리하는 폴더 단위는 탐색기에서 보는 폴더와 같다.
- 왼쪽과 아래쪽에 있는 창은 줄일 수 있고, 아래의 네모를 눌러 각 창을 열 수 있는 버튼을 없앨 수 있다.



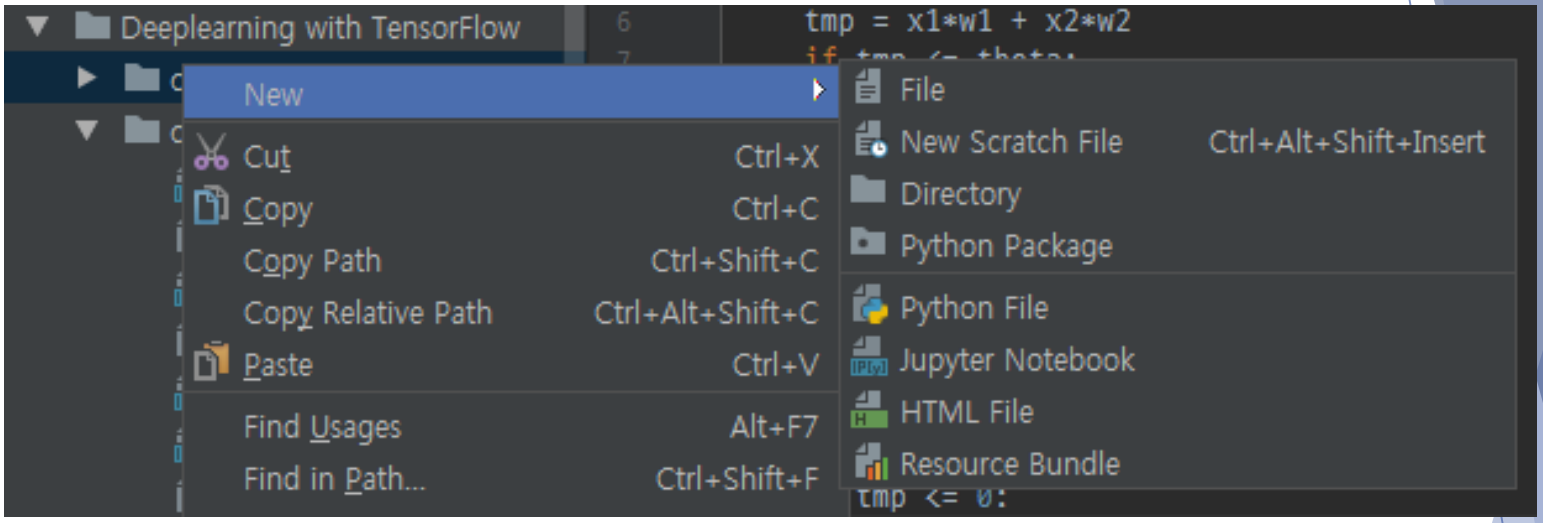
기초-pycharm





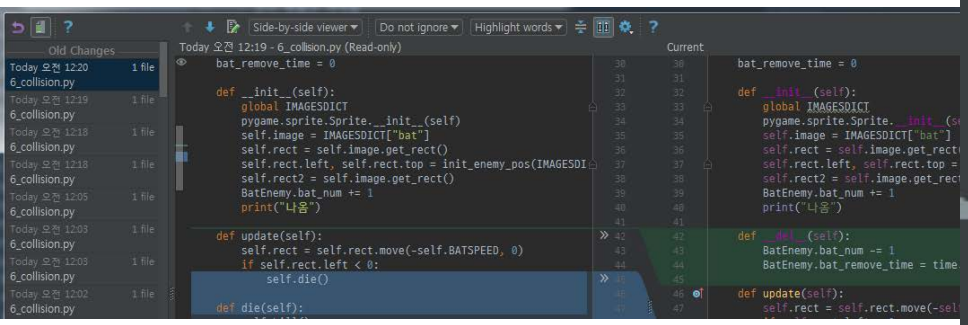
- run 부분에서 실행할 수 있다. Run...을 통해 실행할 python파일을 선택할 수 있으며, run부분은 이전에 실행한 파일을 기억해서 바로 실행하게 만든다.
- 오른쪽 위에 있는 실행버튼은, 버튼 왼쪽에 있는 파일을 바로 실행하며, 이전에 실행한 5개의 파일을 기억한다.
- 파일이 열린 상태에서 오른쪽 버튼을 누르면 있는 run은 바로 실행할 수 있게 해준다.
- 디버깅을 위한 break포인트는 vs와 같이 행을 더블클릭해서 추가할 수 있다.

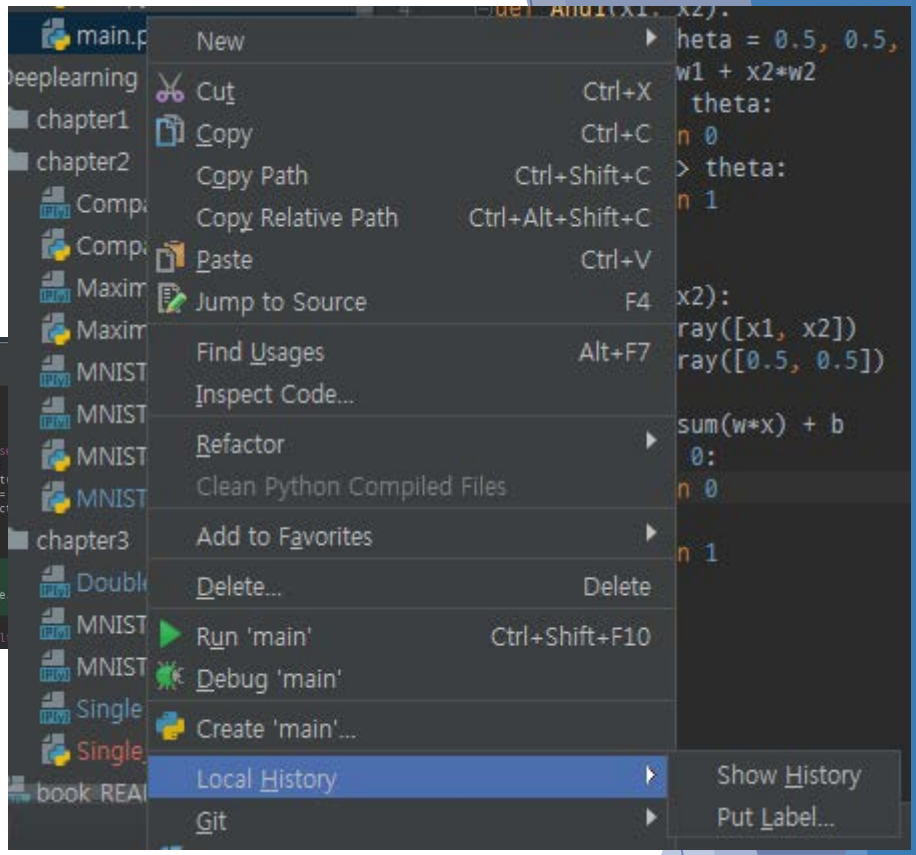
기초-IDE



- 폴더나 프로젝트, 오른쪽 버튼 New- python file을 통해서 python파일을 만들 수 있다.

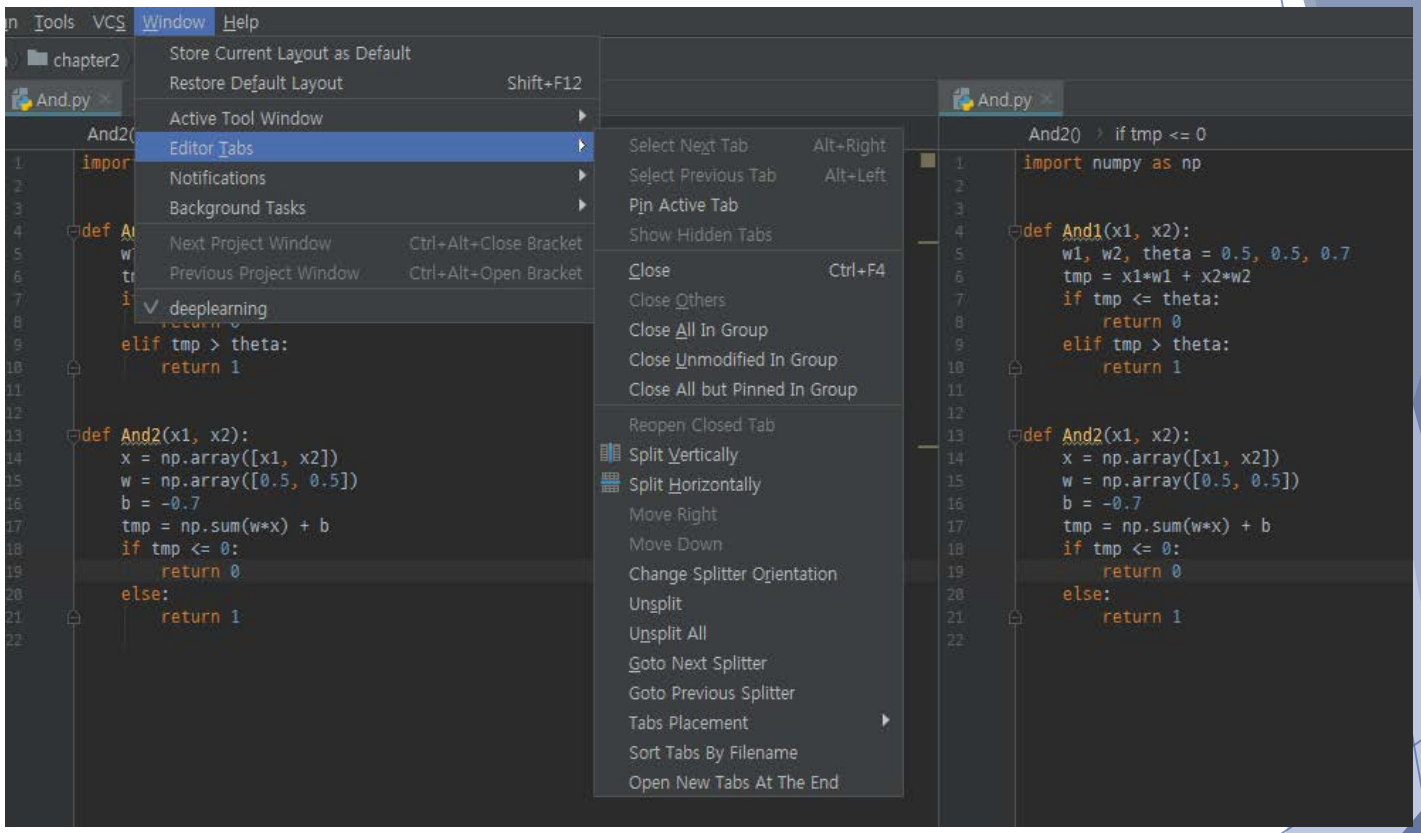
기초-IDE





- 파일-오른쪽버튼-local History에서 파일의 변화를 일부 저장하고 있다. 이를 통해 이전 파일로 되돌릴 수 있다. (아마 일정 날짜가 지나가면 삭제되는 것으로 예상된다.)

기초-IDE



- windows-editor Tabs의 split vertically, horizontally를 통해 여러 창을 동시에 열어 비교하며 코딩이 가능하다.

기초-PIP

- pip란?
  - python으로 작성된 패키지를 설치 및 관리하는 시스템
  - 업데이트도 관리해준다.
- pip 업데이트
  - 패키지를 설치할 때, 오류를 방지하기 위해서 pip를 먼저 업데이트 해야 된다.

기초-PIP

- 모든 명령어는 명령프롬프트에서 입력한다
- pip list
  - 설치되어 있는 패키지 리스트가 나타난다.

```
C:\Users\KJH>pip list
DEPRECATION: The default format will switch to columns in the future. You can us
e --format=(legacy|columns) (or define a format=(legacy|columns) in your pip.con
f under the [list] section) to disable this warning.
pip (9.0.1)
setuptools (28.8.0)
```

기초-PIP

- pip list의 목록 확인방식
  - pip list --format=legacy
  - pip list --format=columns

```
C:₩Users₩KJH>pip list --format=legacy
pip (9.0.1)
setuptools (38.4.0)

C:₩Users₩KJH>pip list --format=columns
Package    Version
---------- -------
pip        9.0.1
setuptools 38.4.0
```

기초-PIP

- pip 업데이트

  pip install --upgrade pip

- 설치된 패키지 업데이트 확인

```
C:\Users\KJH>pip list --outdated
DEPRECATION: The default format will switch to columns in the future. You can us
e --format=(legacy|columns) (or define a format=(legacy|columns) in your pip.con
f under the [list] section) to disable this warning.
setuptools (28.8.0) - Latest: 38.4.0 [wheel]
```

기초-PIP

- 설치된 패키지의 업그레이드
  - pip install --upgrade <패키지명>

ex) setuptools의 업그레이드

```
C:₩Users₩KJH>pip install --upgrade pip
Requirement already up-to-date: pip in c:₩users₩kjh₩appdata₩local₩programs₩pytho
n₩python36₩lib₩site-packages

C:₩Users₩KJH>pip install --upgrade setuptools
Collecting setuptools
  Downloading setuptools-38.4.0-py2.py3-none-any.whl (489kB)
    100% |████████████████████████████████| 491kB 669kB/s
Installing collected packages: setuptools
  Found existing installation: setuptools 28.8.0
    Uninstalling setuptools-28.8.0:
      Successfully uninstalled setuptools-28.8.0
Successfully installed setuptools-38.4.0
```

기초-PIP

- pip를 통한 패키지 설치

  pip install <패키지명>

  ex)

  pip install Numpy

  pip install matplotlib

```
C:₩Users₩KJH>pip install Numpy
Collecting Numpy
  Using cached numpy-1.14.0-cp36-none-win_amd64.whl
Installing collected packages: Numpy
Successfully installed Numpy-1.14.0
```

기초-PIP

- pip를 통한 패키지 설치

  pip uninstall <패키지명>

  삭제할 내용을 전부 출력한 뒤에 Proceed (y/n)? 를 물어본다.

  y를 누르면 삭제한다.

```
  c:\users\kjh\appdata\local\programs\python\python36\lib\site-packages\numpy\ve
rsion.py
  c:\users\kjh\appdata\local\programs\python\python36\scripts\__pycache__\f2py.c
python-36.pyc
  c:\users\kjh\appdata\local\programs\python\python36\scripts\f2py.py
Proceed (y/n)? y
  Successfully uninstalled numpy-1.14.0
```

기초-Numpy

- 딥러닝에서는 배열이나 행렬 계산이 많이 등장한다.
- Numpy의 배열클래스인 numpy.array에는 관련된 메서드가 많이 존재하여 사용한다.
- numpy 모듈 가져오기

```
>> import numpy as np
```

- 외부 라이브러리를 사용하기위해 import한다.
- numpy라는 모듈을 np라는 이름으로 가져오기 위해 사용한다.
- 많은 곳에서 np로 사용한다..

기초-Numpy

```
>>> x = np.array([1.0, 2.0, 3.0])
>>> print(x)
[1. 2. 3.]
>>> type(x)
<class 'numpy.ndarray'>
>>>
```

- numpy배열 생성하기
  - np.array()를 사용한다.
  - np.array()는 리스트를 인수로 받아 numpy가 제공하는 특수한 형태의 배열(numpy.ndarray)를 반환한다.

기초-Numpy

```
>>> x = np.array([1.0, 2.0, 3.0])
>>> y = np.array([2.0, 4.0, 6.0])
>>> x + y
array([3., 6., 9.])
>>> x - y
array([-1., -2., -3.])
>>> x * y
array([ 2.,  8., 18.])
>>> x / y
array([0.5, 0.5, 0.5])
```

- numpy산술 연산
  - 배열 x와 y의 원소 수가 같으면, 산술 연산은 각 원소에 대해서 행해진다.
  - 원소수가 다르면, 오류가 발생한다.
  - 원소별(element-wise)계산을 한다.
  - numpy 배열의 산술연산은 배열 한 개와 수치 하나(스칼라값)의 조합으로 된 산술연산을 수행할 수 있다.
  - 이를 브로드캐스트라 한다.

기초-Numpy

```
>>> x = np.array([1.0, 2.0, 3.0])
>>> y = np.array([2.0, 4.0, 6.0])
>>> x + y
array([3., 6., 9.])
>>> x - y
array([-1., -2., -3.])
>>> x * y
array([ 2.,  8., 18.])
>>> x / y
array([0.5, 0.5, 0.5])
```

- numpy산술 연산
  - 배열 x와 y의 원소 수가 같으면, 산술 연산은 각 원소에 대해서 행해진다.
  - 원소수가 다르면, 오류가 발생한다.
  - 원소별(element-wise)계산을 한다.
  - numpy 배열의 산술연산은 배열 한 개와 수치 하나(스칼라값)의 조합으로 된 산술연산을 수행할 수 있다.
  - 이를 브로드캐스트라 한다.
  - 딥러닝에서 편리한 개념이다.

```
>>> x = np.array([1.0, 2.0, 3.0])
>>> x / 2.0
array([0.5, 1. , 1.5])
```

기초-Numpy

```
>>> A = np.array([[1, 2], [3, 4]])
>>> print(A)
[[1 2]
 [3 4]]
>>> A.shape
(2, 2)
>>> A.dtype
dtype('int32')
```

- numpy의 N차원 배열
  - 행렬의 모양(shape)는 shape
  - 행렬내의 자료형은 dtype
  - 행렬의 산술연산
    - 모양이 같으면, 행렬의 산술 연산도 대응하는 원소별로 계산된다.
    - 행렬과 스칼라값의 산술연산도 가능하다.

```
>>> B = np.array([[3, 0], [0, 6]])
>>> A + B
array([[ 4,  2],
       [ 3, 10]])
>>> A * B
array([[ 3,  0],
       [ 0, 24]])
```

```
>>> print(A)
[[1 2]
 [3 4]]
>>> A * 10
array([[10, 20],
       [30, 40]])
```

기초-Numpy

- np.array는 N차원 배열을 작성할 수 있다.
- 수학에서 0차원을 스칼라 1차원 배열은 백터, 2차원 배열은 행렬이라 하고, 이를 일반화 한 것(N차원)을 텐서라 한다. 본 세미나에서는 책을 따라, 3차원 이상의 배열을 다차원 배열이라고 한다.
- tensorflow의 이름이 여기서 유래했다.
  tensor가 신경망을 따라 흐른다.

기초-Numpy

$$\begin{matrix}1 & 2\\ 3 & 4\end{matrix} * 10 = \begin{matrix}1 & 2\\ 3 & 4\end{matrix} * \begin{matrix}10 & 10\\ 10 & 10\end{matrix} = \begin{matrix}10 & 20\\ 30 & 40\end{matrix}$$

- 브로드캐스트
  - 모양이 다른 배열끼리 계산할 수 있게 한다.
  - 2X2 행렬에 A에 스칼라값 10을 곱했다.
  - 이 때, 위와 같이 10이라는 스칼라 값이 2X2로 확대된 후 연산이 이뤄진다.
  - 이를 브로드캐스트라 한다.

기초-Numpy

$$\begin{matrix}1 & 2\\ 3 & 4\end{matrix} * \begin{matrix}10 & 20\end{matrix} = \begin{matrix}1 & 2\\ 3 & 4\end{matrix} * \begin{matrix}10 & 20\\ 10 & 20\end{matrix} = \begin{matrix}10 & 40\\ 30 & 80\end{matrix}$$

- 브로드캐스트
  - 2차원 배열과 1차원 배열의 연산은 B가 브로드캐스팅 되어 A과 같은 모양이 된다.
  - 하지만, 행이나 열의 크기는 맞아야 된다.

```
>>> A = np.array([[1, 2], [3, 4]])
>>> B = np.array([10, 20])
>>> A * B
array([[10, 40],
       [30, 80]])
>>> C = np.array([10, 20, 30])
>>> A * C
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: operands could not be broadcast together with shapes (2,2) (3,)
```

기초-Numpy

- 원소 접근
  - 원소의 index는 0부터 시작한다.

```
>>> X = np.array([[51, 55], [14,19], [0, 4]])
>>> print(X)
[[51 55]
 [14 19]
 [ 0  4]]
>>> X[0]
array([51, 55])
>>> X[0][1]
55
```

기초-Numpy

- 원소 접근
  - for문으로 접근

```
>>> for row in X:
...     print(row)
...
[51 55]
[14 19]
[0 4]
```

기초-Numpy

- 원소 접근
  - 인덱스를 배열로 지정에 한 번에 접근
  - flatten()은 X를 1차원 배열로 변환한다(평탄화).

```
>>> X = X.flatten()
>>> print(X)
[51 55 14 19  0  4]
>>> X[np.array([0, 2, 4])]
array([51, 14,  0])
```

기초-Numpy

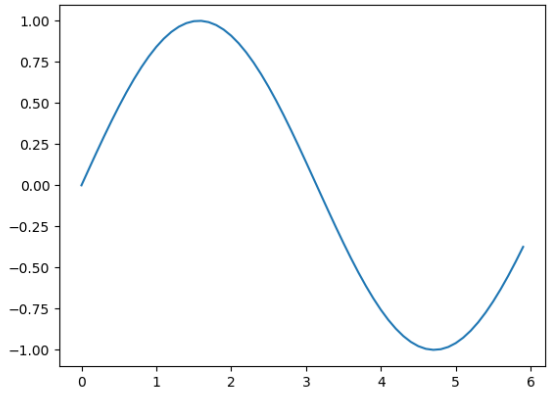
- 특정 조건을 만족하는 원소 얻기

```
>>> X > 15
array([ True,  True, False,  True, False, False])
>>> X[X>15]
array([51, 55, 19])
>>>
```

기초-matplotlib

- 그래프를 그려보자.
- matplotlib에 있는 pyplot 모듈을 사용하며, 보통 plt로 줄인다.

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3
4   # 데이터 준비
5   x = np.arrange(0, 6, 0.1)
6   y = np.sin(x)
7
8   # 그래프 그리기
9   plt.plot(x, y)
10  plt.show()
```



기초-matplotlib

- 그래프를 그려보자.
- matplotlib에 있는 pyplot 모듈을 사용하며, 보통 plt로 줄
  인다.

```python
import numpy as np
import matplotlib.pyplot as plt

# 데이터 준비
x = np.arange(0, 6, 0.1)
y1 = np.sin(x)
y2 = np.cos(x)

# 그래프 그리기
plt.plot(x, y1, label="sin")
plt.plot(x, y2, linestyle="--", label="cos")
plt.xlabel("x")
plt.ylabel("y")
plt.title('sin & cos')
plt.legend()
plt.show()
```