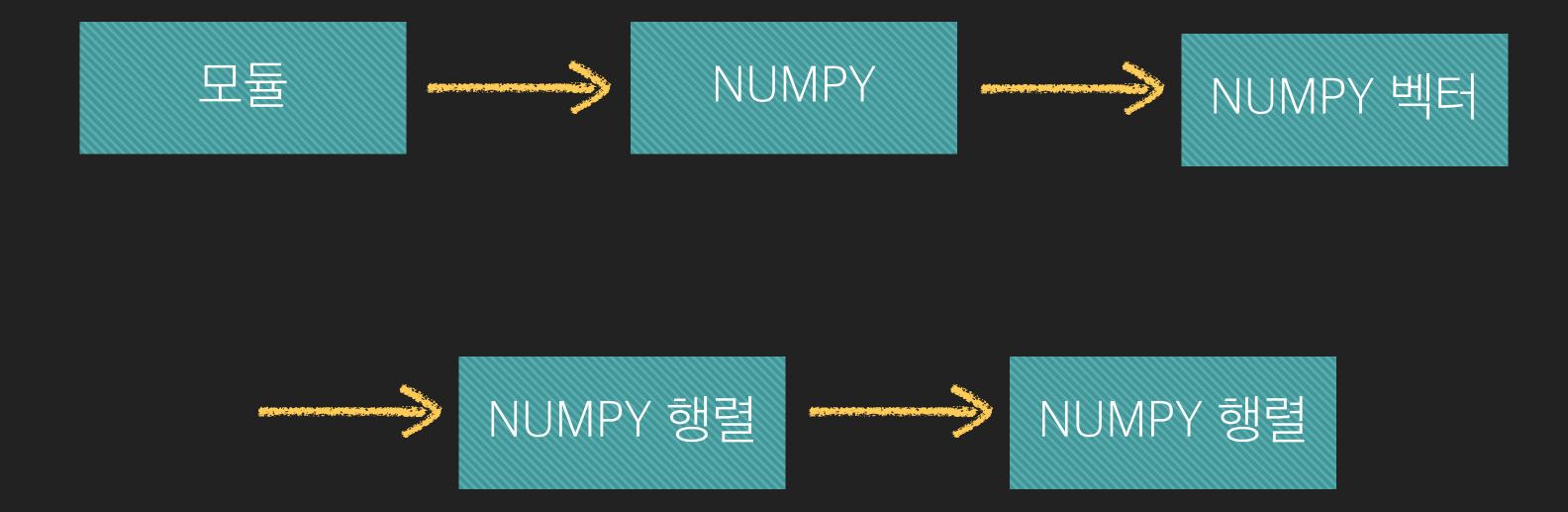
국민대학교 인공지능 연구실

# 파이썬기초4주차

# 4주차 목차



#### INTRO

- ▶ 1주차 colab 실습 링크
  - https://colab.research.google.com/drive/1wgHQKrv2Q6bAjhfKZFItB785nGuEVw-Q
- ▶ 2주차 colab 실습 링크
  - https://colab.research.google.com/drive/1dQ9VYzAuNrHVtF2XOshEztpToIZMF0hz
- ▶ 3주차 colab 실습 링크
  - https://colab.research.google.com/drive/1YdKdD3Y08Bz08\_xG1Ext00ywJQCzp1fJ
- ▶ 4주차 colab 실습 링크
  - https://colab.research.google.com/drive/1xGMQV0wSyl0N7chTGhA1hiyrsH200Ky2

#### 모듈2

- ▶ 모듈 만들기
  - ▶ 실습 자료 참고
- ▶ 작성한 코드를 my\_vector.py로 저장하면 해당 파일 자체가 my\_vecotr모듈
  - ▶ 모듈은? 파이썬 코드를 가지고 있는 텍스트파일(\*.py)
- ▶ 해당 자료를 terminal을 사용해서 인터프리터를 실행해서 동일하게 사용할 수 있음
- ▶ 패키지란?
  - ▶ 모듈들을 넣어둔 디렉토리명

#### 모듈2

#### [(base) inkyungkim@InKyungui-MacBookPro 특수대학원 파이썬 보강 % vi my\_vector.py

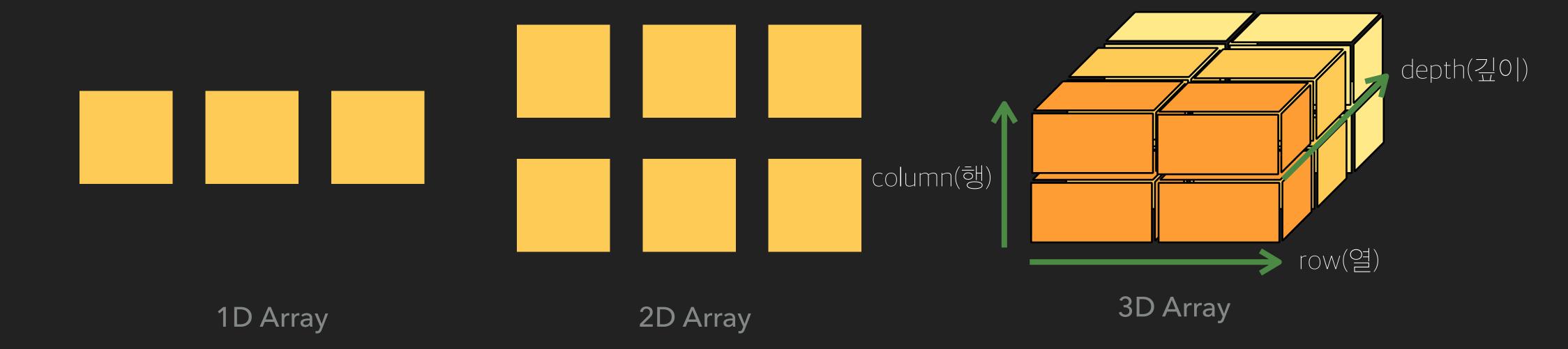
```
1 class vec2:
 x = 0.0
 y = 0.0
    def __init__(self, x, y):
      self.x = x
      self.y = y
    def __str__(self):
      msg = "(" + str(self.x) + ', ' + str(self.y) + ")"
10
      return msg
11
13
    def __add__(self, other):
      return vec2(self.x + other.x, self.y + other.y)
```

```
(base) inkyungkim@InKyungui-MacBookPro 특수대학원 파이션 보강 % python Python 3.7.6 (default, Jan 8 2020, 13:42:34)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin Type "help", "copyright", "credits" or "license" for more information.
>>> import my_vector
>>> a = my_vector.vec2(3, 4)
>>> b = my_vector.vec2(-1, 2)
>>> c = a + b
>>> print(a
...)
(3, 4)
>>> print(b)
(-1, 2)
```

#### 모듈2

- if \_\_name\_\_ == "\_\_main\_\_"
  - ▶ 모듈로 import되는 경우가 아니라, 스크립트로 실행되는 경우에만 해당 부분이 동작하도록 변경
- ▶ 스크립트로 실행
  - python my\_vector.py
- ▶ 모듈로 실행
  - python
  - import my\_vector

#### NUMPY



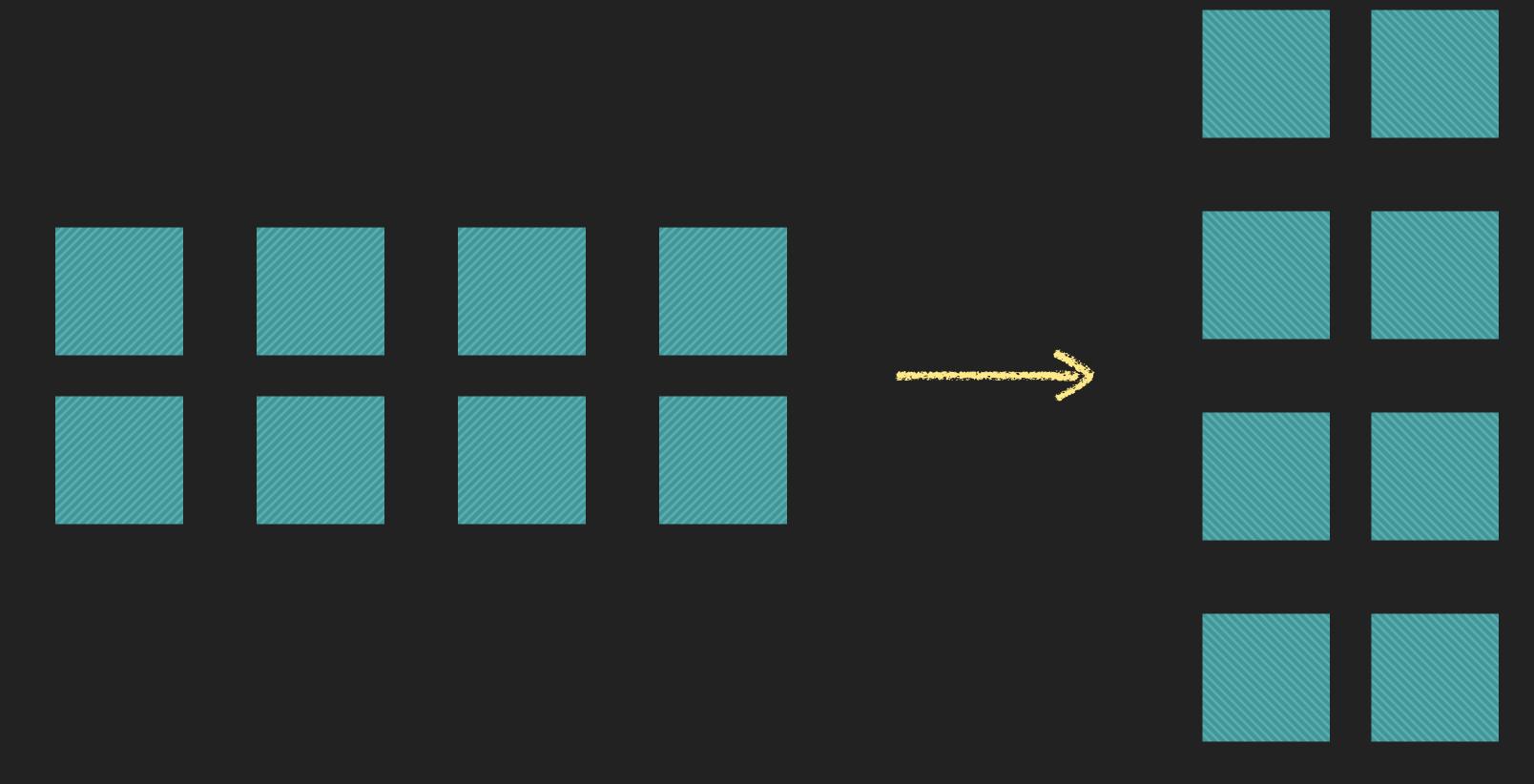
- ▶ 행렬 연산을 위한 핵심 라이브러리
- ▶ 대규모 다차원 배열과 행렬 연산에 필요한 다양함 함수를 제공
- ▶ 배열구조는 "Shape"으로 표현됨
  - ▶ Shape은 배열 구조를 파이썬의 튜플 자료형을 이용하여 정의함
- ▶ 다차원 배열은 입체적인 데이터 구조를 가지며, 데이터 차원은 여러 갈래의 데이터 방향을 가짐
  - ▶ 데이터 방향을 axis로 표현할 수 있음
  - ▶ 높이, 폭, 방향을 각각 axis=0, axis=1, axis=2로 지정

#### NUMPY

- ▶ Numpy array와 python list의 차이점
  - Numpy array
    - > 동일 타입의 원소
    - ▶ 메모리 최적화, 계산 속도 향상
    - ▶ 크기(dimension)이 명확하게 정의
    - ▶ 전체 연산 가능
  - Python list
    - ▶ 여러가지 타입의 원소
    - ▶ 메모리 용량이 크고 속도가 느림
    - nesting가능
    - ▶ 전체 연산 불가

- ▶ 요소 참조
  - ▶ list형과 마찬가지로 [ ]를 사용해서 요소 참조
- ▶ 요소 수정
  - ▶ list형과 마찬가지로 <x[수정할 요소 인덱스 번호] = 수정할 값> 을 이용해서 수정
- ▶ 연속된 정주 벡터 생성
  - ▶ 사용법: np.arange(n) 으로 요소값이 1씩 증가하는 벡터 배열 생성 가능
  - list형의 range(n)과 동일하게 사용가능

- ▶ 선형대수의 데이터 유형
  - ▶ 스칼라(scalar) : 하나의 숫자만으로 이루어진 데이터
  - ▶ 벡터(vector): 여러 숫자가 순서대로 모여 있는 것, 일반적인 일차원 배열이 벡터
    - > x축만을 갖는 자료형
  - ▶ 행렬(matrix) : 복수의 차원을 가지는 데이터가 다시 여러개 있는 경우, 일반적으로 2차원 배열이 행렬
    - ▶ x축과 y축의 컬럼을 갖는 경우
  - ▶ 텐서(tensor): 3차원 이상의 배열을 의미
    - ▶ 3차원 배열 형태의 텐서: 행과 열을 갖고, 각 컬럼은 벡터 형태를 가짐(depth) 즉, x, y, z축의 형태
    - ▶ 4차원 이상의 배열: depth요소가 스칼라가 아니라 벡터 이상의 자료형을 갖는 경우



- 전치행렬
  - ▶ 행렬의 행과 열을 바꾸는 것 즉, 행렬의 축을 바꾸는 것

- ▶ 벡터와 행렬의 연산
  - ▶ 덧셈과 뺄셈
    - ▶ 같은 <u>크기를 가진 벡터와 행렬은 덧셈과 뺄셈을 할 수 있음</u>
    - ▶ 같은 위치에 있는 원소끼리 덧셈과 뺄셈을 하면 됨
    - ▶ 이를 요소별 연산이라 함
  - ▶ 스칼라와 벡터/ 행렬의 곱셈

- 벡터와 행렬의 연산
  - ▶ 곱셈(내적/ inner product)
    - 조건
      - 앞 벡터/ 행렬의 열의 수가 뒤의 벡터/ 행렬의 행의 수와 일치해야 함
    - > 모두 dot()을 사용
    - 행렬의 경우, 교환법칙이 성립하지 않음

# R A

# 참고자료

- https://datascienceschool.net/view-notebook/24e43f9df9ec4abab15e32e68e982cc1/
- https://doorbw.tistory.com/171
- http://taewan.kim/post/numpy\_sum\_axis/
- https://datascienceschool.net/view-notebook/3f44cfdda2874080a9aa6b034c71d5ec/
- https://rfriend.tistory.com/289