UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

Deep Learning

Session 2

Deep Learning Fundamentals with Python

Applied Data Science

2024/2025

Deep Learning Fundamentals with Python

Don't memorize these formulas. If you understand the concepts, you can invent your own notation.

—John Cochrane, *Investments Notes* 2006

GitHub SetUp

- Lectures will be provided both on GitHub and Moodle.
  - https://github.com/LCDA-UCP/tac-hands-on

- You will need to fork the repository and then clone it to your local machine.

- Setup a python/conda environment and install the requirements.
  - $ pip install –r requirements.txt

Deep Learning Fundamentals with Python

- **NumPy**
  - NumPy is a fast Python library for numeric computation, primarily using multidimensional arrays (especially two- or three-dimensional) to handle data in neural networks.
  - The *ndarray* class enables intuitive and efficient operations on these arrays.

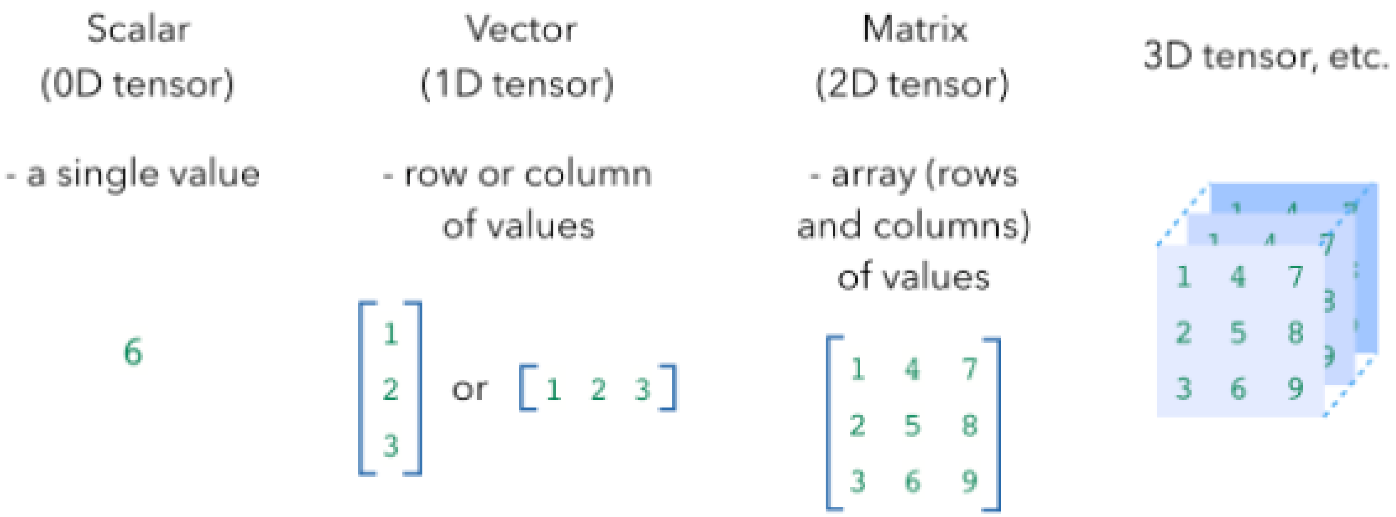
Deep Learning Fundamentals with Python

- **NumPy**
  - First, we need to import the NumPy package and set seeds for reproducibility so that we get the exact same results every time.

```
import numpy as np

# Set seed for reproducibility
np.random.seed(1234)
```

- **NumPy: Basics**

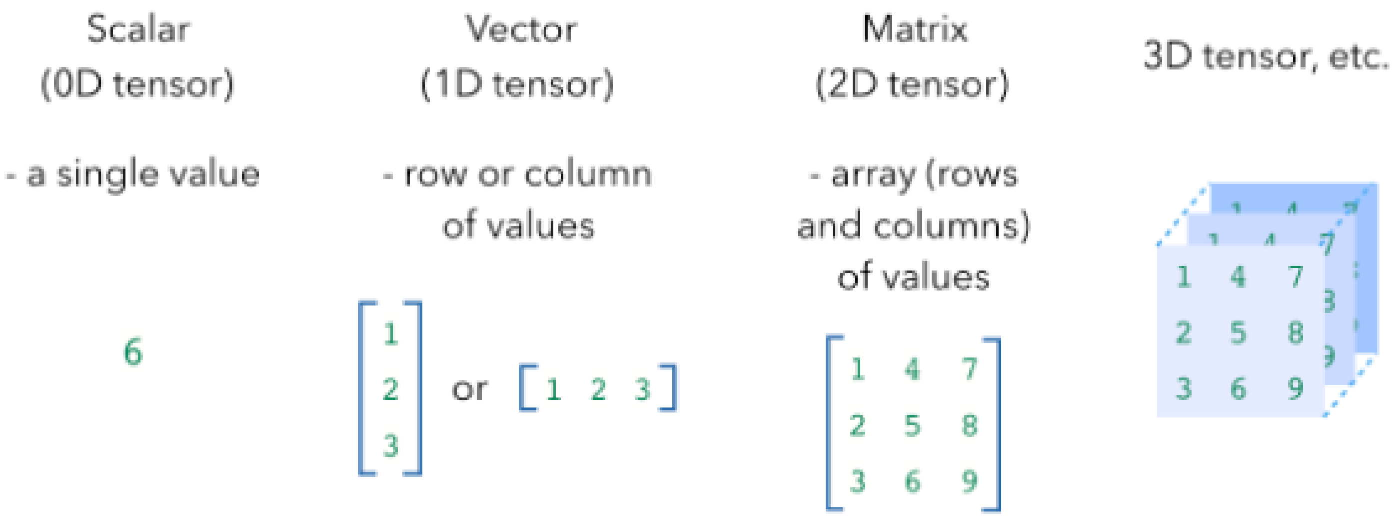



```python
# Scalar
x = np.array(6)
print ("x: ", x)
print ("x ndim: ", x.ndim) # number of dimensions
print ("x shape:", x.shape) # dimensions
print ("x size: ", x.size) # size of elements
print ("x dtype: ", x.dtype) # data type
```

Deep Learning Fundamentals with Python

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

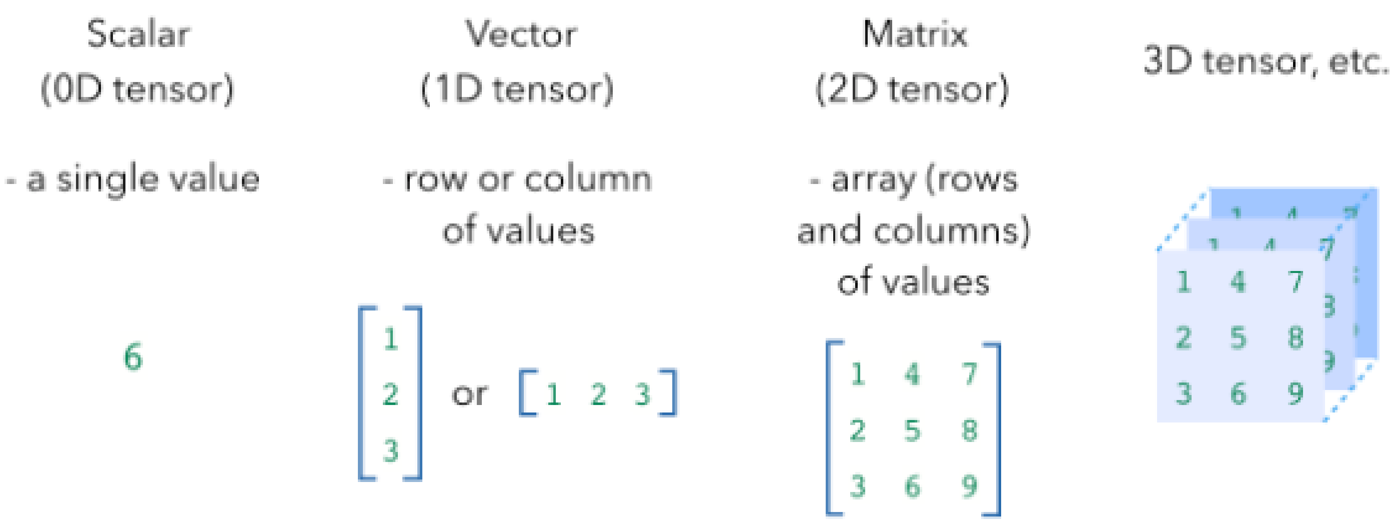
- **NumPy: Basics**




```python
# Vector
x = np.array([1.3 , 2.2 , 1.7])
print ("x: ", x)
print ("x ndim: ", x.ndim)
print ("x shape:", x.shape)
print ("x size: ", x.size)
print ("x dtype: ", x.dtype) # notice the float datatype
```

Deep Learning Fundamentals with Python

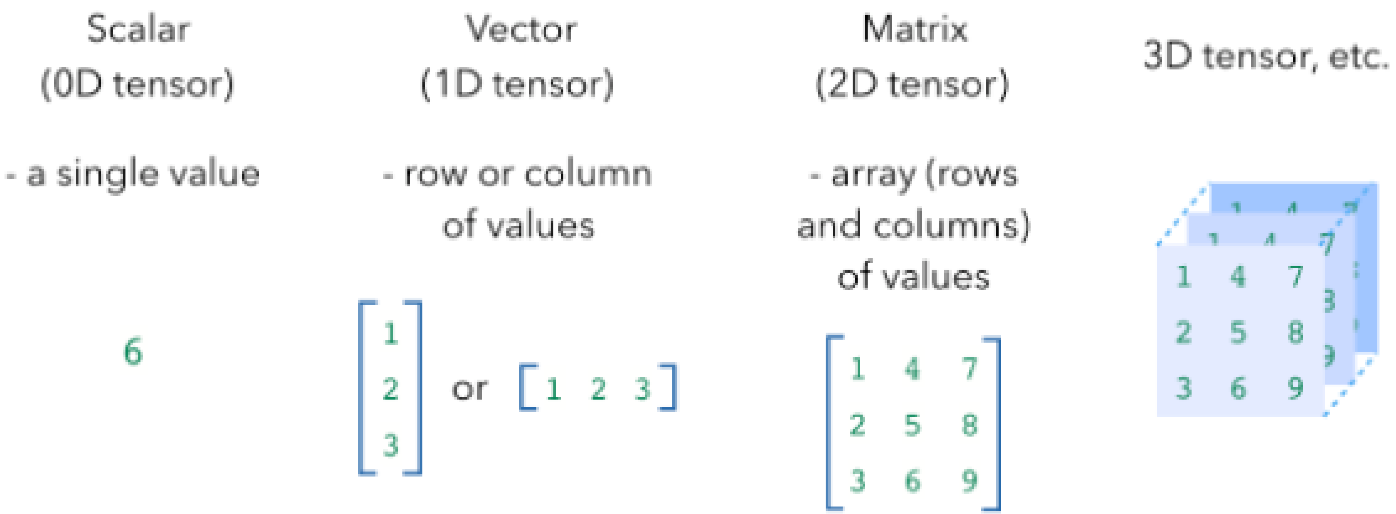
- **NumPy: Basics**




```
# Matrix
x = np.array([[1,2], [3,4]])
print ("x:\n", x)
print ("x ndim: ", x.ndim)
print ("x shape:", x.shape)
print ("x size: ", x.size)
print ("x dtype: ", x.dtype)
```

Deep Learning Fundamentals with Python

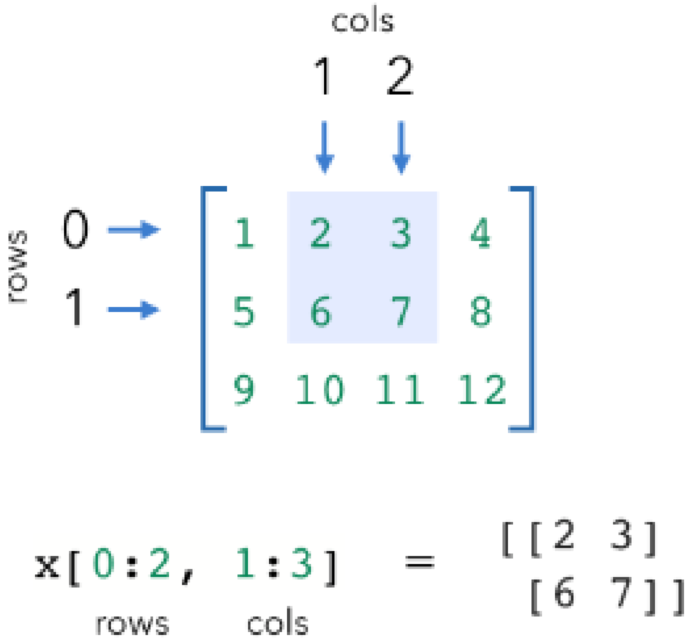
- **NumPy: Basics**




```python
# 3-D Tensor
x = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])
print ("x:\n", x)
print ("x ndim: ", x.ndim)
print ("x shape:", x.shape)
print ("x size: ", x.size)
print ("x dtype: ", x.dtype)
```

UNIVERSIDADE CATÓLICA PORTUGUESA BRAGA

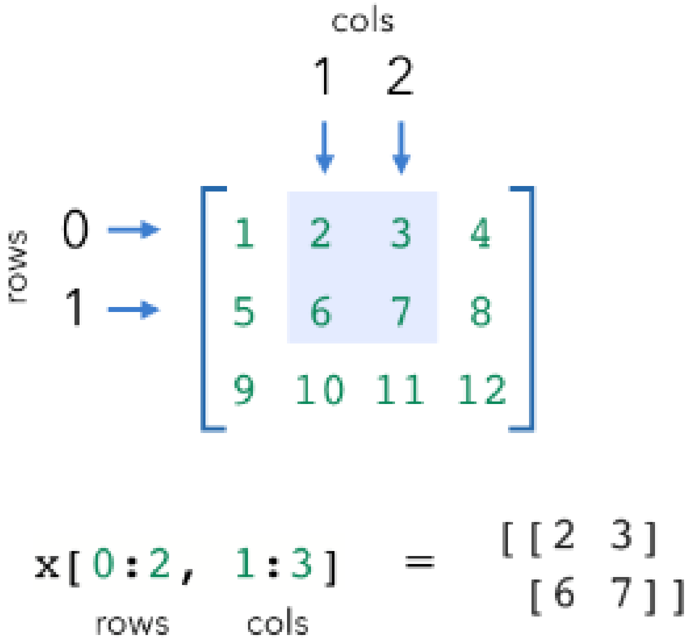
- **NumPy: Indexing**
  - We can extract specific values from arrays using indexing.




```
# Indexing
x = np.array([1, 2, 3])
print ("x: ", x)
print ("x[0]: ", x[0])
x[0] = 0
print ("x: ", x)
```

UNIVERSIDADE CATÓLICA PORTUGUESA BRAGA

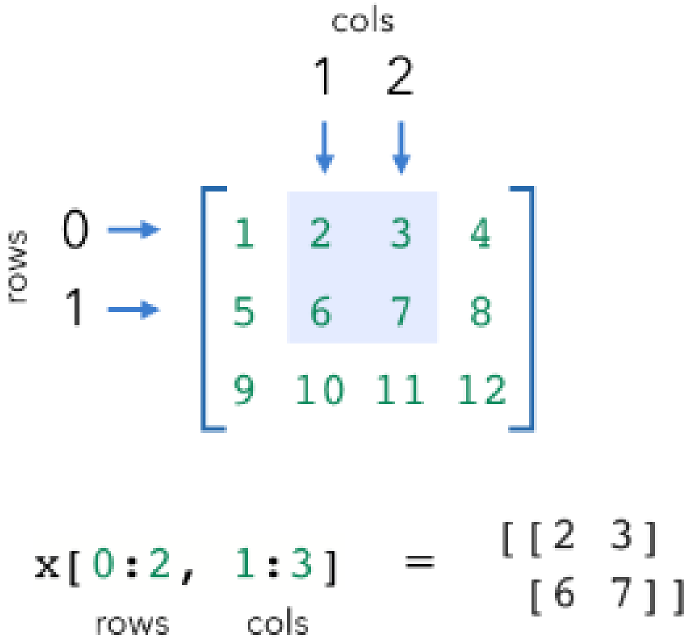
- **NumPy: Indexing**
  - o We can extract specific values from arrays using indexing.




```
# Slicing
x = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
print (x)
print ("x column 1: ", x[:, 1])
print ("x row 0: ", x[0, :])
print ("x rows 0,1 & cols 1,2: \n", x[0:2, 1:3])
```

UNIVERSIDADE CATÓLICA PORTUGUESA BRAGA

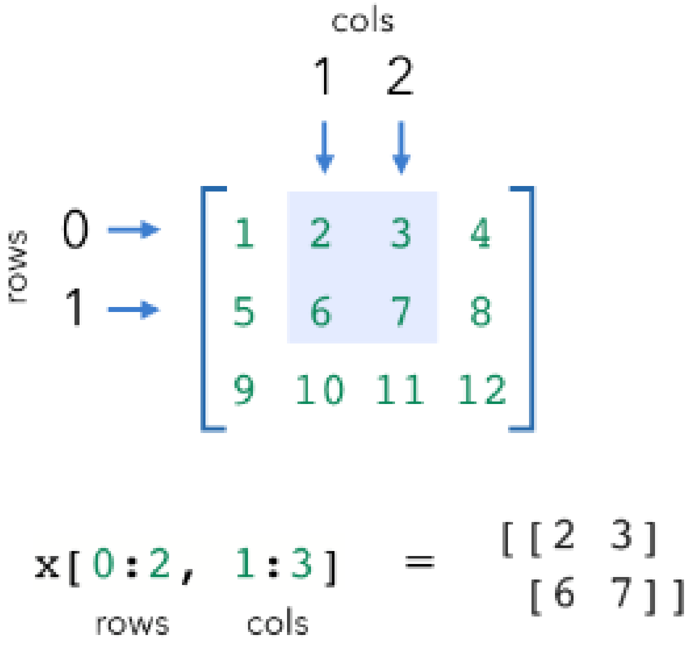
- **NumPy: Indexing**
  - We can extract specific values from arrays using indexing.




```python
# Integer array indexing
print (x)
rows_to_get = np.array([0, 1, 2])
print ("rows_to_get: ", rows_to_get)
cols_to_get = np.array([0, 2, 1])
print ("cols_to_get: ", cols_to_get)
# Combine sequences above to get values to get
print ("indexed values: ", x[rows_to_get, cols_to_get]) # (0, 0), (1, 2), (2, 1)
```

Deep Learning Fundamentals with Python

- **NumPy: Indexing**
  - o We can extract specific values from arrays using indexing.




```
# Boolean array indexing
x = np.array([[1, 2], [3, 4], [5, 6]])
print ("x:\n", x)
print ("x > 2:\n", x > 2)
print ("x[x > 2]:\n", x[x > 2])
```

Deep Learning Fundamentals with Python

- **NumPy: Arithmetic**

```
# Basic math
x = np.array([[1,2], [3,4]], dtype=np.float64)
y = np.array([[1,2], [3,4]], dtype=np.float64)
print ("x + y:\n", np.add(x, y)) # or x + y
print ("x - y:\n", np.subtract(x, y)) # or x - y
print ("x * y:\n", np.multiply(x, y)) # or x * y
```

Deep Learning Fundamentals with Python

- **NumPy: Dot Product**

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \bullet \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} \quad & \quad \end{bmatrix}$$

[2 X 3] [3 X 2] [2 X 2]

```python
# Dot product
a = np.array([[1,2,3], [4,5,6]], dtype=np.float64) # we can specify dtype
b = np.array([[7,8], [9,10], [11, 12]], dtype=np.float64)
c = a.dot(b)
print (f"{a.shape} · {b.shape} = {c.shape}")
print (c)
```

Deep Learning Fundamentals with Python

- **NumPy: Axis Operations**

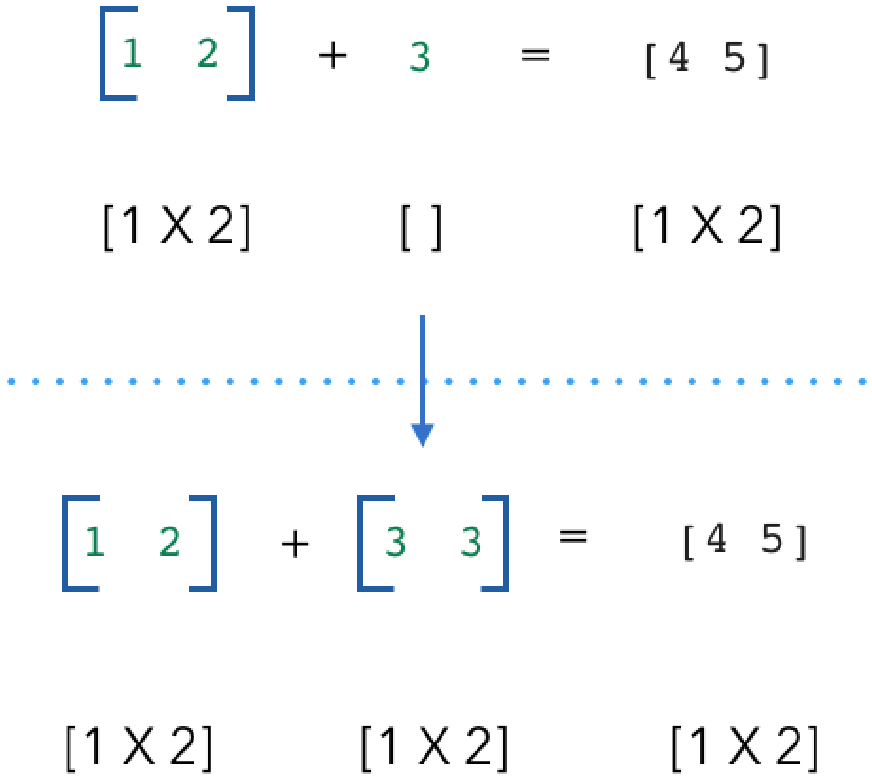
$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$$

```
np.sum(x)          =
np.sum(x, axis=0)  =
np.sum(x, axis=1)  =
```

```python
# Sum across a dimension
x = np.array([[1,2],[3,4]])
print (x)
print ("sum all: ", np.sum(x)) # adds all elements
print ("sum axis=0: ", np.sum(x, axis=0)) # sum across rows
print ("sum axis=1: ", np.sum(x, axis=1)) # sum across columns
```

Deep Learning Fundamentals with Python

- **NumPy: Brodcast**




```python
# Broadcasting
x = np.array([1,2]) # vector
y = np.array(3) # scalar
z = x + y
print ("z:\n", z)
```

Deep Learning Fundamentals with Python

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- **NumPy: Transpose**

```
np.transpose(x, (1,0))
```

$$x = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \qquad \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

[2 X 3] [3 X 2]

```
# Transposing
x = np.array([[1,2,3], [4,5,6]])
print ("x:\n", x)
print ("x.shape: ", x.shape)
y = np.transpose(x, (1,0)) # flip dimensions at index 0 and 1
print ("y:\n", y)
print ("y.shape: ", y.shape)
```

- **NumPy: Reshape**

```
x = [1  2  3 | 4  5  6]                    [1 X 6]
```

$$\texttt{np.reshape(x, (2, 3))} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad [\mathbf{2} \times 3]$$

$$\texttt{np.reshape(x, (2, -1))} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad [2 \times 3]$$

```python
# Reshaping
x = np.array([[1,2,3,4,5,6]])
print (x)
print ("x.shape: ", x.shape)
y = np.reshape(x, (2, 3))
print ("y: \n", y)
print ("y.shape: ", y.shape)
z = np.reshape(x, (2, -1))
print ("z: \n", z)
print ("z.shape: ", z.shape)
```

Deep Learning Fundamentals with Python

- **NumPy: Joining**
  - We can join arrays via concatentation or stacking.

```
x = np.random.random((2, 3))
print (x)
print (x.shape)
```

```
# Concatenation
y = np.concatenate([x, x], axis=0) # concat on a specified axis
print (y)
print (y.shape)
```

```
# Stacking
z = np.stack([x, x], axis=0) # stack on new axis
print (z)
print (z.shape)
```

Deep Learning Fundamentals with Python

- **NumPy: Expanding/Reducing**
  - We can also easily add and remove dimensions to our arrays.

```
# Adding dimensions
x = np.array([[1,2,3],[4,5,6]])
print ("x:\n", x)
print ("x.shape: ", x.shape)
y = np.expand_dims(x, 1) # expand dim 1
print ("y: \n", y)
print ("y.shape: ", y.shape)   # notice extra set of brackets are added
```

```
# Removing dimensions
x = np.array([[[1,2,3]],[[4,5,6]]])
print ("x:\n", x)
print ("x.shape: ", x.shape)
y = np.squeeze(x, 1) # squeeze dim 1
print ("y: \n", y)
print ("y.shape: ", y.shape)  # notice extra set of brackets are gone
```

Deep Learning Fundamentals with Python

- **NumPy Arrays vs Lists**

```
print("Python list operations:")
a = [1,2,3]
b = [4,5,6]
print("a+b:", a+b)
try:
    print(a*b)
except TypeError:
    print("a*b has no meaning for Python lists")
print()
print("numpy array operations:")
a = np.array([1,2,3])
b = np.array([4,5,6])
print("a+b:", a+b)
print("a*b:", a*b)
```

Deep Learning Fundamentals with Python

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- **NumPy**

```python
print("Python list operations:")
a = [1,2,3]
b = [4,5,6]
print("a+b:", a+b)
try:
    print(a*b)
except TypeError:
    print("a*b has no meaning for Python lists")
print()
print("numpy array operations:")
a = np.array([1,2,3])
b = np.array([4,5,6])
print("a+b:", a+b)
print("a*b:", a*b)
```

```
Python list operations:
a+b: [1, 2, 3, 4, 5, 6]
a*b has no meaning for Python lists

numpy array operations:
a+b: [5 7 9]
a*b: [ 4 10 18]
```

Deep Learning Fundamentals with Python

- **NumPy**

```
a = np.array([[1,2],[3,4]])
print('a:')
print(a)
print('a.sum(axis=0):', a.sum(axis=0))
print('a.sum(axis=1):', a.sum(axis=1))
```

```
a = np.array([[1,2,3],
[4,5,6]])
b = np.array([10,20,30])
print("a+b:\n", a+b)
```

Deep Learning Fundamentals with Python

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- **NumPy**

```python
a = np.array([[1,2],[3,4]])
print('a:')
print(a)
print('a.sum(axis=0):', a.sum(axis=0))
print('a.sum(axis=1):', a.sum(axis=1))
```

```
a:
[[1 2]
 [3 4]]
a.sum(axis=0): [4 6]
a.sum(axis=1): [3 7]
```

```python
a = np.array([[1,2,3],
[4,5,6]])
b = np.array([10,20,30])
print("a+b:\n", a+b)
```

```
a+b:
 [[11 22 33]
 [14 25 36]]
```

Deep Learning Fundamentals with Python

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- **Type-hints and docstings in Python**
  - If you are given this function what assumptions can you make?

```
def operation(x, y):
    return x + y
```

Deep Learning Fundamentals with Python

- **Type-hints and docstings in Python**

  - If you are given this function what assumptions can you make?

```
def operation(x, y):
    return x + y
```

  - What about this one?

```
def operation(x: str, y: str) -> str:
    """
    This function concatenates two strings.

    Parameters
    ----------
    x : str
        The first string.
    y : str
        The second string.

    Returns
    -------
    str
        The concatenation of the two input strings.
    """
    return x + y
```

Deep Learning Fundamentals with Python

- **Type-hints and docstings in Python**
  - **Type hints** improve code clarity by indicating expected argument and return types.
  - **Docstrings** explain a function's purpose and usage, aiding readability and maintenance.

Deep Learning Fundamentals with Python

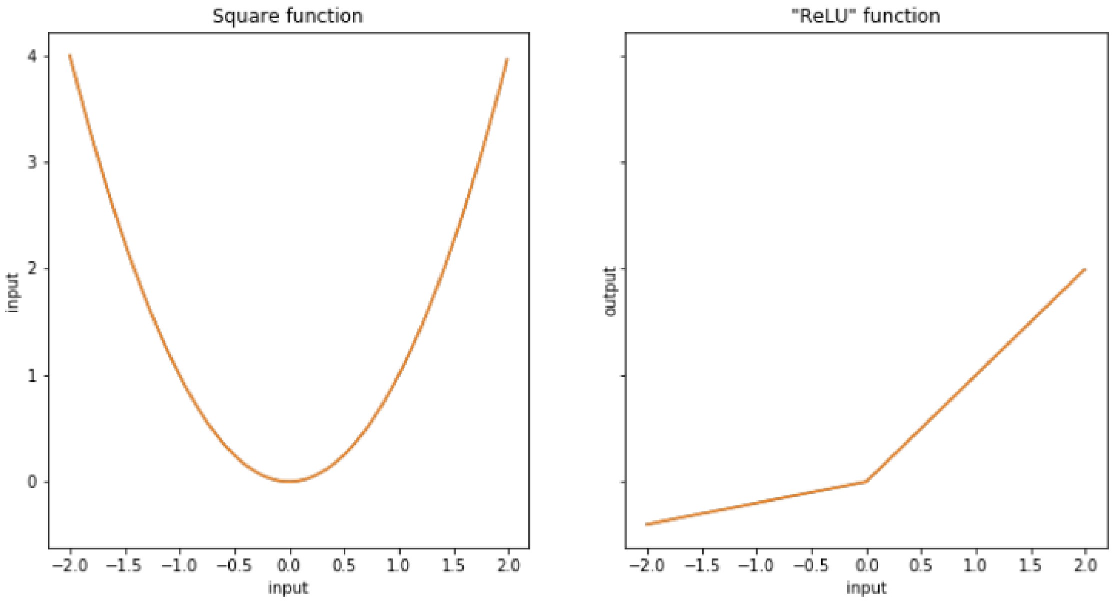
UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- **Functions:** A reusable block of code that performs a specific task.
  - **Math:** $$f_1(x) = x^2$$ $$f_2(x) = max(x, 0)$$
    - These functions will take as input x and transform it to x^2, in the case of f_1, and max(x, 0), in the case of f_2.

- **Exercise 1:** Implement the f_1 and f_2 functions with Python and NumPy (python_intro_for_dl.ipynb)

Deep Learning Fundamentals with Python

- **Functions:** A reusable block of code that performs a specific task.
  - **Diagrams:** To depict a function, we can dra an x-y plane, plot points with x-coordinates as inputs and y-coordinates as outputs.



- **Exercise 2:** Plot these functions using Python and matplotlib.

Deep Learning Fundamentals with Python

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

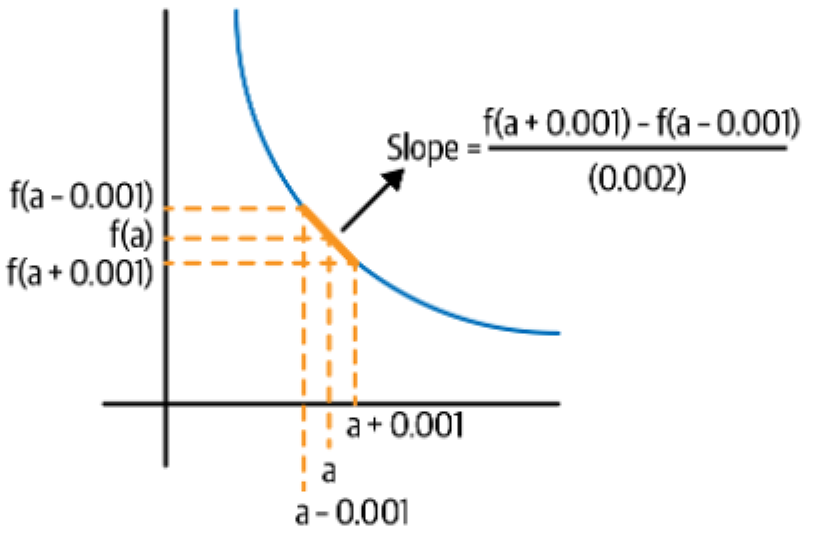
- **Derivatives:** A measure of how a function's **output changes as its input changes**, represented as the **slope of the tangent line at a specific point**.
  - **Math:** $$\frac{df}{du}(a) = \lim_{\Delta \to 0} \frac{f(a+\Delta) - f(a-\Delta)}{2 \times \Delta}$$
    - This limit can be approximated numerically by setting a very small value for Δ, such as 0.001, so we can compute the derivative as:

$$\frac{df}{du}(a) = \frac{f(a+0.001) - f(a-0.001)}{0.002}$$

UNIVERSIDADE CATÓLICA PORTUGUESA BRAGA

- **Derivatives:** A measure of how a function's **output changes as its input changes**, represented as the **slope of the tangent line at a specific point**.

  - **Diagrams:** Like functions, derivatives can be depicted in a graphic.

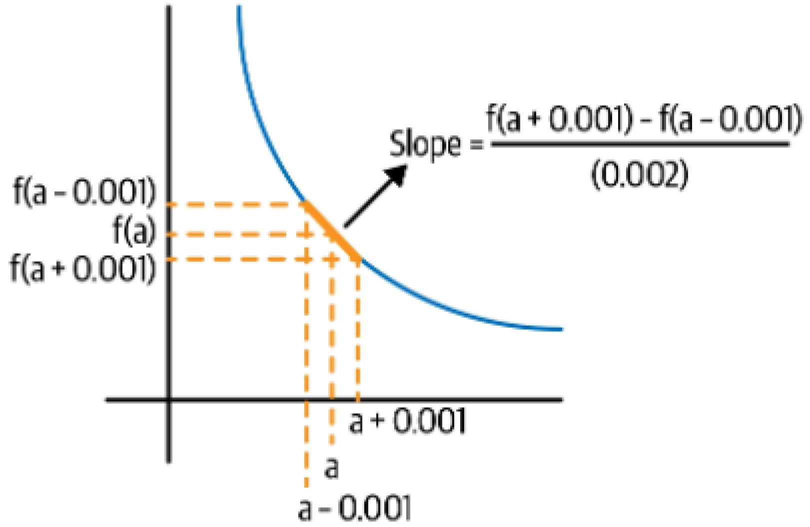


Deep Learning Fundamentals with Python

- **Derivatives:**

$$\frac{df}{du}(a) = \lim_{\Delta \to 0} \frac{f(a+\Delta) - f(a-\Delta)}{2 \times \Delta}$$

$$\frac{df}{du}(a) = \frac{f(a+0.001) - f(a-0.001)}{0.002}$$



- **Exercise:** Implement a function that returns the derivative of the input function at the input value.
- **Exercise:** Test it with the f1 function at the point x=1.0.
- What do you expect the derivative on point x=0.0 to be?