Programming Session 2

Jan Tekülve

jan.tekuelve@ini.rub.de

Computer Science and Mathematics
Preparatory Course

07.10.2021

Overview

1. **Programming**
   - Utilities
2. **Tasks**
3. **Outlook: Matrices and Scientific Programming**
   - Matrices Quick Summary
   - The Numpy Module
   - Matrix Calculation with Numpy

Ask for a correct user input

- Sometimes a specific user input is required

```
userIn = input("Please type exit! ")
while not userIn == "exit" :
      userIn = input("Please type exit! ")
```

Ask for a correct user input

- Sometimes a specific user input is required

```
userIn = input("Please type exit! ")
while not userIn == "exit" :
       userIn = input("Please type exit! ")
```

- The input might allow a range of options

```
userIn = input("Please choose Left or Right: ")
while not (userIn == "Left" or userIn == "Right"):
       userIn = input("Please choose Left or Right: ")
```

Variations of the For-Loop

- The range function has an optional stepsize parameter

```
myList = ["A","B","C","D","E","F"]
#Print every second element of a list
for i in range(0,len(myList),2):
      print(myList[i])
#This prints A C E
```

Variations of the For-Loop

- The range function has an optional stepsize parameter

```
myList = ["A","B","C","D","E","F"]
#Print every second element of a list
for i in range(0,len(myList),2):
      print(myList[i])
#This prints A C E
```

- One can even go through the list in reverse

```
#From len(myList)-1 to 0 with stepsize -1
for i in range(len(myList)-1,-1,-1):
      print(myList[i])
#This prints F E D C B A
```

Dissecting Strings

- Split a sentence into words

```
mySentence = "Hello I am a Sentence"
words = mySentence.split(" ") # words is a list
# ["Hello", "I", "am" , "a", "Sentence"]
```

Dissecting Strings

- Split a sentence into words

```
mySentence = "Hello I am a Sentence"
words = mySentence.split(" ") # words is a list
# ["Hello", "I", "am" , "a", "Sentence"]
```

- Split a word into letters

```
word = "Hello"
#The list typecast converts strings to lists
letters = list(word) #["H","e","l","l","o"]
```

Dissecting Strings

- Split a sentence into words

```
mySentence = "Hello I am a Sentence"
words = mySentence.split(" ") # words is a list
# ["Hello", "I", "am" , "a", "Sentence"]
```

- Split a word into letters

```
word = "Hello"
#The list typecast converts strings to lists
letters = list(word) #["H","e","l","l","o"]
```

- Use the “in” operator to check if an element is in a list

```
if "e" in letters:
        print("The letter 'e' is in the list.")
```

Exchange Variable Values

- How to exchange two variable values?

```
FirstPlace = "Schumacher"
SecondPlace = "Lauda"
```

Exchange Variable Values

- How to exchange two variable values?

```
FirstPlace = "Schumacher"
SecondPlace = "Lauda"
```

- Now Lauda overtakes Schumacher

```
FirstPlace = SecondPlace # FirstPlace = "Lauda"
SecondPlace = Firstplace # SecondPlace = "Lauda" !!!
```

Exchange Variable Values

- How to exchange two variable values?

```
FirstPlace = "Schumacher"
SecondPlace = "Lauda"
```

- Now Lauda overtakes Schumacher

```
FirstPlace = SecondPlace # FirstPlace = "Lauda"
SecondPlace = Firstplace # SecondPlace = "Lauda" !!!
```

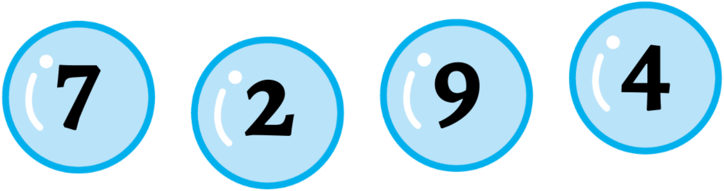
- A helper variable is required

```
helper = FirstPlace # helper = "Schumacher"
FirstPlace = SecondPlace # FirstPlace = "Lauda"
SecondPlace = helper # SecondPlace = "Schumacher"
```

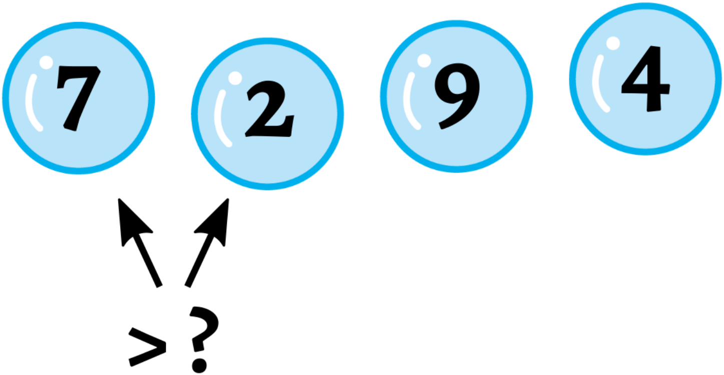
The Bubble Sort Algorithm

List = [7,2,9,4]

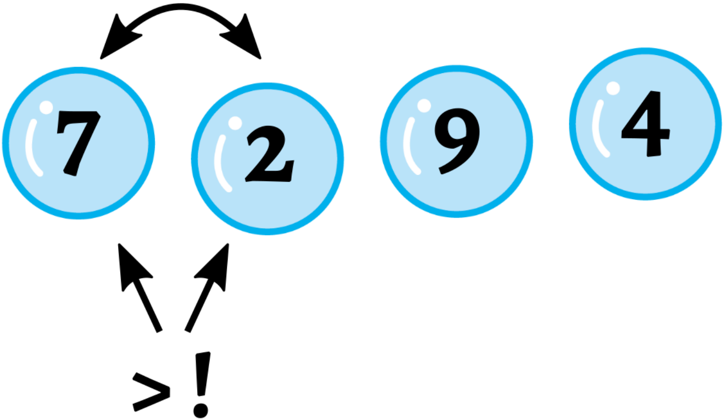
The Bubble Sort Algorithm



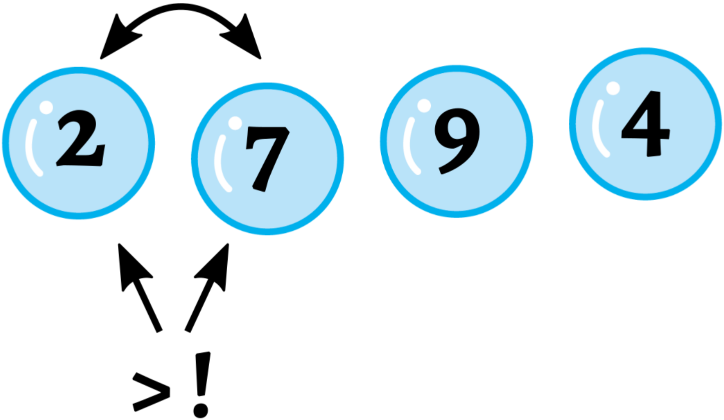
The Bubble Sort Algorithm



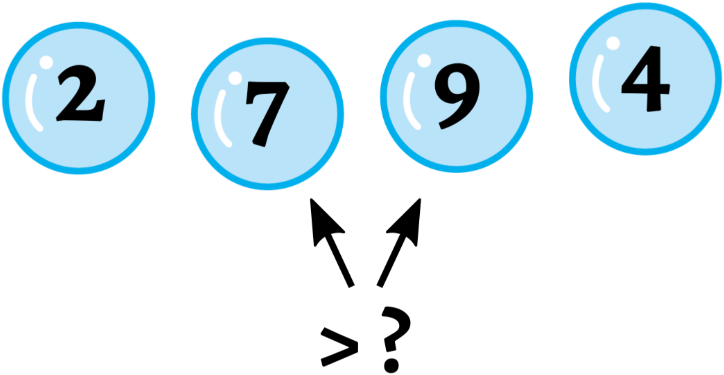
The Bubble Sort Algorithm



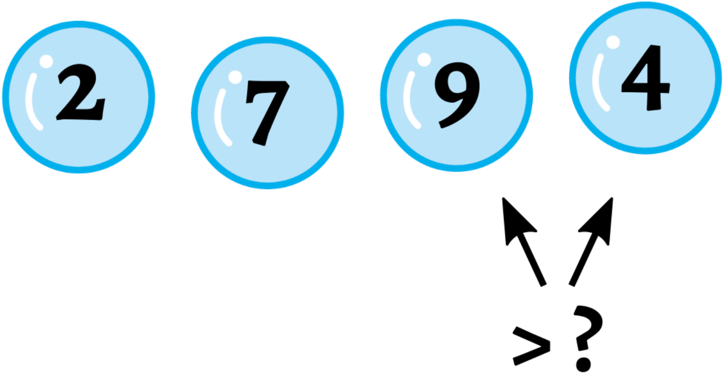
The Bubble Sort Algorithm



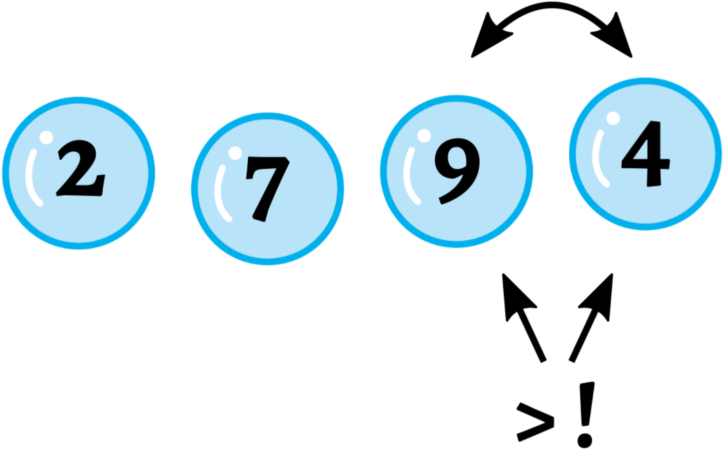
The Bubble Sort Algorithm



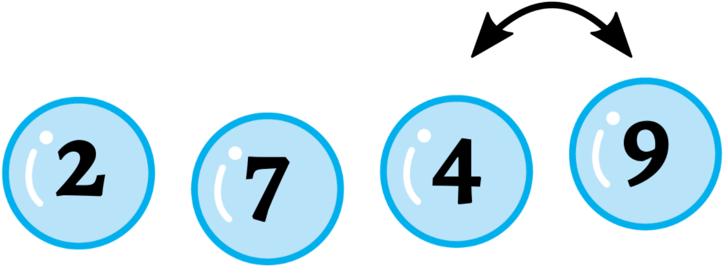
The Bubble Sort Algorithm



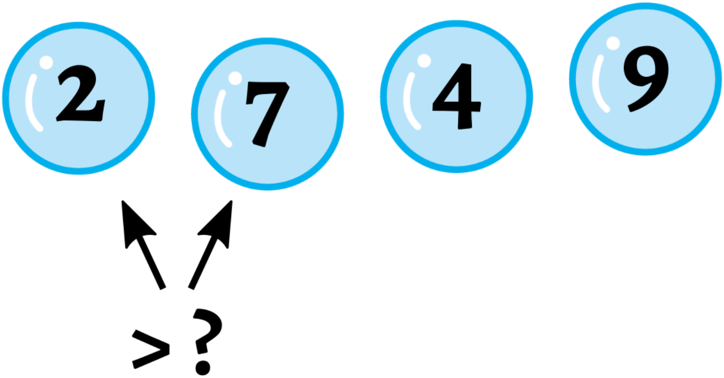
The Bubble Sort Algorithm



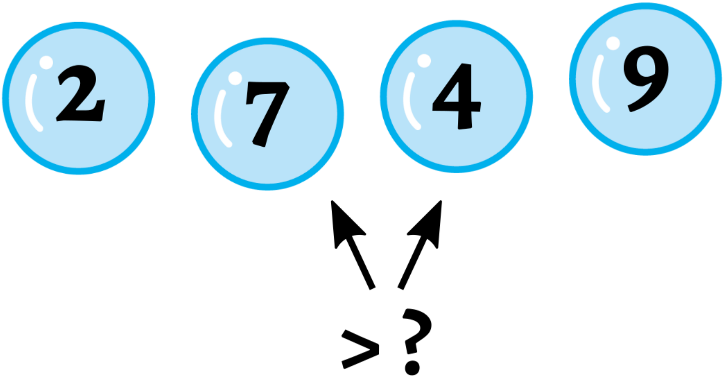
The Bubble Sort Algorithm



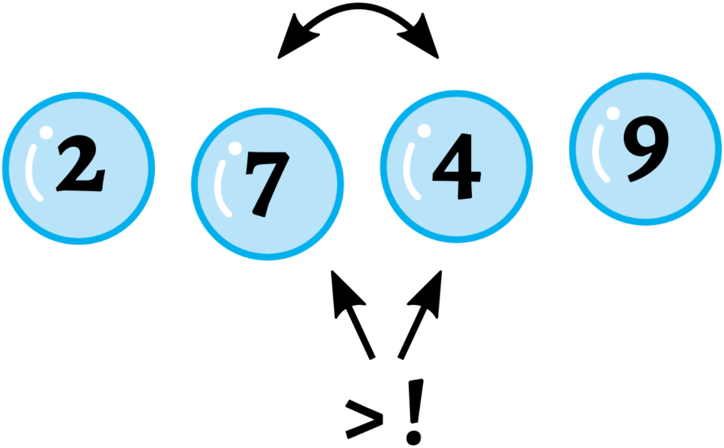
The Bubble Sort Algorithm



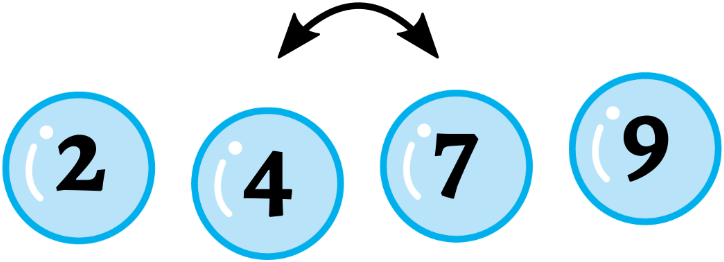
The Bubble Sort Algorithm



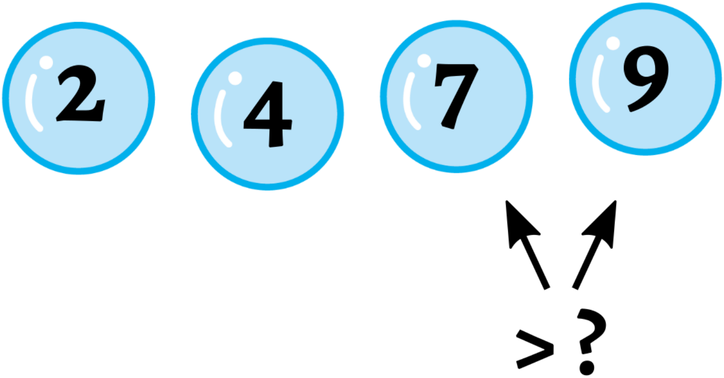
The Bubble Sort Algorithm



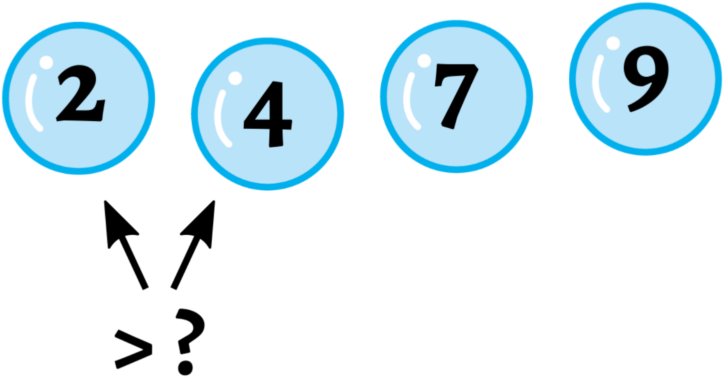
The Bubble Sort Algorithm



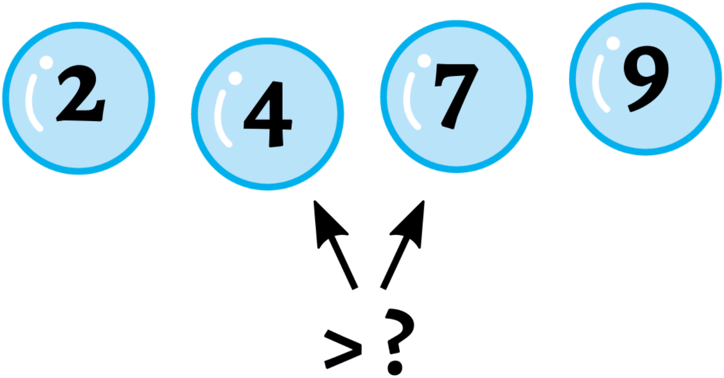
The Bubble Sort Algorithm



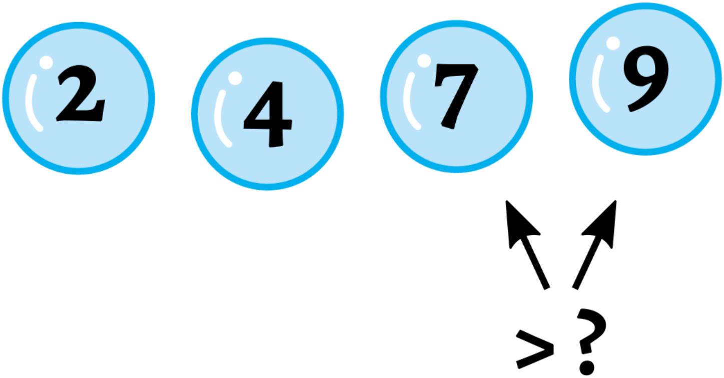
The Bubble Sort Algorithm



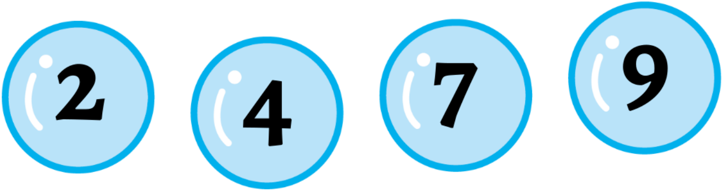
The Bubble Sort Algorithm



The Bubble Sort Algorithm



The Bubble Sort Algorithm



All pairs are in correct order!
Done!

Bubble Sort in Words

- Input: An unsorted list
- Do the following until nothing is changed anymore:
  - Iterate through the complete list
    1. Compare the current element with the next element
    2. If the current element is greater than the next element, switch their positions
    3. Notify whether a change was made
- The list is now sorted.

Helpful Functions

- The random module

```
import random #import the module similar to import math
#assigns dice_roll a number between 1 and 6
dice_roll = random.randint(1,6)
#random list item
myList = ["Rock","Paper","Scissors"]
random_item = myList[random.randint(0,len(myList)-1)]
```

- Convert a string to uppercase

```
name = "Peter"
upname = name.upper()
print(upname) # "PETER"
```

Task: Reverse a sentence

1. Write a script that reverts the word order in a given sentence
    - Let the user type in any sentence via the *input()* method
    - Split the sentence into a list of words
    - Use a for loop to go through the list in reverse order
    - During each iteration add the current word to a string variable *sentence*
    - Print the *sentence* variable

This is an example sentence $\rightarrow$ *sentence example an is This*

Task: Hangman

2. Write a Hangman computer game. The computer secretly chooses a word and the user may guess letters until the word is found.
   - Choose a random word from the words list and store it in variable
   - For each letter of the Word print an underscore "_"
   - Start a while loop that runs until the whole word is found
   - In the loop let the user guess a character and store the guessed character in a list
   - Run a second loop through each letter of the word and check whether this letter has been guessed already. If it has been guessed, print it otherwise print an underscore "_".
   - If you still had to replace a word by "_" the while loop continues

$$TASK \rightarrow \_\,\_\,\_\,\_$$

Task: Bubble Sort

3. Implement the Bubbling Sort Algorithm to sort a list of numbers
    - Start a while loop
    - In the while loop iterate through the list and compare the current and the next element
    - If the next element is smaller than the current one swap them
    - If you swap, make sure that the while loop is continued
    - If you did not swap at all, make sure the while loop ends

Matrix Definition

A Matrix $\boldsymbol{A}_{m,n}$ is a rectangular array arranged in m rows and n columns.

- Example:

$$\boldsymbol{A}_{3,4} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

Matrix Definition

A Matrix $\boldsymbol{A}_{m,n}$ is a rectangular array arranged in m rows and n columns.

- Example:

$$\boldsymbol{A}_{3,4} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

- A single element in a matrix is usually denoted by $a_{i,j}$, where i is the row and j the column index. For example $a_{2,3} = 7$.

Matrix Definition

A Matrix $\boldsymbol{A}_{m,n}$ is a rectangular array arranged in m rows and n columns.

- Example:

$$\boldsymbol{A}_{3,4} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

- A single element in a matrix is usually denoted by $a_{i,j}$, where i is the row and j the column index. For example $a_{2,3} = 7$.

- A matrix A_m, n, where $m = n$ is called a **square matrix**

Matrix Definition

A Matrix $\boldsymbol{A}_{m,n}$ is a rectangular array arranged in m rows and n columns.

- Example:

$$\boldsymbol{A}_{3,4} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

- A single element in a matrix is usually denoted by $a_{i,j}$, where i is the row and j the column index. For example $a_{2,3} = 7$.
- A matrix A_m, n, where $m = n$ is called a **square matrix**
- A matrix that has only entries on the diagonal is called a **diagonal matrix**

$$\boldsymbol{D}_{3,3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 4 \end{pmatrix} \text{ Special case } \textbf{identity matrix } \boldsymbol{I}_{3,3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Matrix Addition/Subtraction

- It is possible to add two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ together, if they have the same number of rows and columns.

Matrix Addition/Subtraction

- It is possible to add two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ together, if they have the same number of rows and columns.
- Addition is carried out element-wise:

$$\boldsymbol{A}_{3,2} + \boldsymbol{B}_{3,2} = \begin{pmatrix} 1 & 2 \\ 5 & 6 \\ 9 & 10 \end{pmatrix} + \begin{pmatrix} 4 & 2 \\ 3 & 1 \\ 8 & 2 \end{pmatrix} = \begin{pmatrix} 1+4 & 2+2 \\ 5+3 & 6+1 \\ 9+8 & 10+2 \end{pmatrix} = \begin{pmatrix} 5 & 4 \\ 8 & 7 \\ 17 & 12 \end{pmatrix}$$

Matrix Addition/Subtraction

- It is possible to add two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ together, if they have the same number of rows and columns.
- Addition is carried out element-wise:

$$\boldsymbol{A}_{3,2} + \boldsymbol{B}_{3,2} = \begin{pmatrix} 1 & 2 \\ 5 & 6 \\ 9 & 10 \end{pmatrix} + \begin{pmatrix} 4 & 2 \\ 3 & 1 \\ 8 & 2 \end{pmatrix} = \begin{pmatrix} 1+4 & 2+2 \\ 5+3 & 6+1 \\ 9+8 & 10+2 \end{pmatrix} = \begin{pmatrix} 5 & 4 \\ 8 & 7 \\ 17 & 12 \end{pmatrix}$$

- Subtraction works analogously:

$$\boldsymbol{A}_{3,2} - \boldsymbol{B}_{3,2} = \begin{pmatrix} 1 & 2 \\ 5 & 6 \\ 9 & 10 \end{pmatrix} - \begin{pmatrix} 4 & 2 \\ 3 & 1 \\ 8 & 2 \end{pmatrix} = \begin{pmatrix} 1-4 & 2-2 \\ 5-3 & 6-1 \\ 9-8 & 10-2 \end{pmatrix} = \begin{pmatrix} -3 & 0 \\ 2 & 5 \\ 1 & 8 \end{pmatrix}$$

Scalar Multiplication and Transposition

- Multiplication with scalar values is also applied element-wise:

$$\boldsymbol{A}_{3,2} \cdot 3 = \begin{pmatrix} 1 & 2 \\ 5 & 6 \\ 9 & 10 \end{pmatrix} \cdot 3 = \begin{pmatrix} 1 \cdot 3 & 2 \cdot 3 \\ 5 \cdot 3 & 6 \cdot 3 \\ 9 \cdot 3 & 10 \cdot 3 \end{pmatrix} = \begin{pmatrix} 3 & 6 \\ 15 & 18 \\ 27 & 30 \end{pmatrix}$$

Scalar Multiplication and Transposition

- Multiplication with scalar values is also applied element-wise:

$$\boldsymbol{A}_{3,2} \cdot 3 = \begin{pmatrix} 1 & 2 \\ 5 & 6 \\ 9 & 10 \end{pmatrix} \cdot 3 = \begin{pmatrix} 1 \cdot 3 & 2 \cdot 3 \\ 5 \cdot 3 & 6 \cdot 3 \\ 9 \cdot 3 & 10 \cdot 3 \end{pmatrix} = \begin{pmatrix} 3 & 6 \\ 15 & 18 \\ 27 & 30 \end{pmatrix}$$

- The transposition $\boldsymbol{A}^T$ of a matrix switches the roles of row and columns
  Example:

$$\boldsymbol{A}_{3,2}^T = \begin{pmatrix} 1 & 2 \\ 5 & 6 \\ 9 & 10 \end{pmatrix}^T = \begin{pmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \end{pmatrix}$$

  The transposition turns a $m \times n$ matrix into a $n \times m$ matrix.

Matrix Multiplication

- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be multiplied with each other, if the number of columns of $\boldsymbol{A}_{m,\boldsymbol{n}}$ matches the number of rows in $\boldsymbol{B}_{\boldsymbol{n},o}$.
- The resulting matrix $\boldsymbol{C}_{m,o}$ shares the number of rows from $\boldsymbol{A}$ and the number of columns from $\boldsymbol{B}$.

Matrix Multiplication

- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be multiplied with each other, if the number of columns of $\boldsymbol{A}_{m,\boldsymbol{n}}$ matches the number of rows in $\boldsymbol{B}_{\boldsymbol{n},o}$.
- The resulting matrix $\boldsymbol{C}_{m,o}$ shares the number of rows from $\boldsymbol{A}$ and the number of columns from $\boldsymbol{B}$.
- Matrix multiplication is carried out by multiplying the row-vector of the first matrix with the column-vector of the second matrix.
  Multiply Row by Column

$$\boldsymbol{A}_{2,3} \cdot \boldsymbol{B}_{3,3} = \begin{pmatrix} 3 & 6 & 5 \\ 4 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 & 3 & 8 \\ 1 & 2 & 10 \\ 7 & 3 & 2 \end{pmatrix} = \begin{pmatrix} - & - & - \\ - & - & - \end{pmatrix}$$

Matrix Multiplication

- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be multiplied with each other, if the number of columns of $\boldsymbol{A}_{m,\boldsymbol{n}}$ matches the number of rows in $\boldsymbol{B}_{\boldsymbol{n},o}$.
- The resulting matrix $\boldsymbol{C}_{m,o}$ shares the number of rows from $\boldsymbol{A}$ and the number of columns from $\boldsymbol{B}$.
- Matrix multiplication is carried out by multiplying the row-vector of the first matrix with the column-vector of the second matrix.
  Multiply Row by Column

$$\boldsymbol{A}_{2,3} \cdot \boldsymbol{B}_{3,3} = \begin{pmatrix} \mathbf{3} & \mathbf{6} & \mathbf{5} \\ 4 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{4} & 3 & 8 \\ \mathbf{1} & 2 & 10 \\ \mathbf{7} & 3 & 2 \end{pmatrix} = \begin{pmatrix} ? & - & - \\ - & - & - \end{pmatrix}$$

Matrix Multiplication

- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be multiplied with each other, if the number of columns of $\boldsymbol{A}_{m,\boldsymbol{n}}$ matches the number of rows in $\boldsymbol{B}_{\boldsymbol{n},o}$.
- The resulting matrix $\boldsymbol{C}_{m,o}$ shares the number of rows from $\boldsymbol{A}$ and the number of columns from $\boldsymbol{B}$.
- Matrix multiplication is carried out by multiplying the row-vector of the first matrix with the column-vector of the second matrix.
  Multiply Row by Column

$$\boldsymbol{A}_{2,3} \cdot \boldsymbol{B}_{3,3} = \begin{pmatrix} \mathbf{3} & \mathbf{6} & \mathbf{5} \\ 4 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{4} & 3 & 8 \\ \mathbf{1} & 2 & 10 \\ \mathbf{7} & 3 & 2 \end{pmatrix} = \begin{pmatrix} (3 * 4 + 6 * 1 + 5 * 7) & \_ & \_ \\ \_ & \_ & \_ \end{pmatrix}$$

Matrix Multiplication

- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be multiplied with each other, if the number of columns of $\boldsymbol{A}_{m,\boldsymbol{n}}$ matches the number of rows in $\boldsymbol{B}_{\boldsymbol{n},o}$.
- The resulting matrix $\boldsymbol{C}_{m,o}$ shares the number of rows from $\boldsymbol{A}$ and the number of columns from $\boldsymbol{B}$.
- Matrix multiplication is carried out by multiplying the row-vector of the first matrix with the column-vector of the second matrix.
  Multiply Row by Column

$$\boldsymbol{A}_{2,3} \cdot \boldsymbol{B}_{3,3} = \begin{pmatrix} \mathbf{3} & \mathbf{6} & \mathbf{5} \\ 4 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{4} & 3 & 8 \\ \mathbf{1} & 2 & 10 \\ \mathbf{7} & 3 & 2 \end{pmatrix} = \begin{pmatrix} \mathbf{53} & - & - \\ - & - & - \end{pmatrix}$$

Matrix Multiplication

- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be multiplied with each other, if the number of columns of $\boldsymbol{A}_{m,\boldsymbol{n}}$ matches the number of rows in $\boldsymbol{B}_{\boldsymbol{n},o}$.
- The resulting matrix $\boldsymbol{C}_{m,o}$ shares the number of rows from $\boldsymbol{A}$ and the number of columns from $\boldsymbol{B}$.
- Matrix multiplication is carried out by multiplying the row-vector of the first matrix with the column-vector of the second matrix.
  Multiply Row by Column

$$\boldsymbol{A}_{2,3} \cdot \boldsymbol{B}_{3,3} = \begin{pmatrix} \mathbf{3} & \mathbf{6} & \mathbf{5} \\ 4 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 & \mathbf{3} & 8 \\ 1 & \mathbf{2} & 10 \\ 7 & \mathbf{3} & 2 \end{pmatrix} = \begin{pmatrix} 53 & ? & - \\ - & - & - \end{pmatrix}$$

Matrix Multiplication

- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be multiplied with each other, if the number of columns of $\boldsymbol{A}_{m,\boldsymbol{n}}$ matches the number of rows in $\boldsymbol{B}_{\boldsymbol{n},o}$.
- The resulting matrix $\boldsymbol{C}_{m,o}$ shares the number of rows from $\boldsymbol{A}$ and the number of columns from $\boldsymbol{B}$.
- Matrix multiplication is carried out by multiplying the row-vector of the first matrix with the column-vector of the second matrix.
  Multiply Row by Column

$$\boldsymbol{A}_{2,3} \cdot \boldsymbol{B}_{3,3} = \begin{pmatrix} \mathbf{3} & \mathbf{6} & \mathbf{5} \\ 4 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 & \mathbf{3} & 8 \\ 1 & \mathbf{2} & 10 \\ 7 & \mathbf{3} & 2 \end{pmatrix} = \begin{pmatrix} 53 & (3 * 3 + 6 * 2 + 5 * 3) & \_ \\ \_ & \_ & \_ \end{pmatrix}$$

Matrix Multiplication

- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be multiplied with each other, if the number of columns of $\boldsymbol{A}_{m,\boldsymbol{n}}$ matches the number of rows in $\boldsymbol{B}_{\boldsymbol{n},o}$.
- The resulting matrix $\boldsymbol{C}_{m,o}$ shares the number of rows from $\boldsymbol{A}$ and the number of columns from $\boldsymbol{B}$.
- Matrix multiplication is carried out by multiplying the row-vector of the first matrix with the column-vector of the second matrix.
  Multiply Row by Column

$$\boldsymbol{A}_{2,3} \cdot \boldsymbol{B}_{3,3} = \begin{pmatrix} \mathbf{3} & \mathbf{6} & \mathbf{5} \\ 4 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 & \mathbf{3} & 8 \\ 1 & \mathbf{2} & 10 \\ 7 & \mathbf{3} & 2 \end{pmatrix} = \begin{pmatrix} 53 & \mathbf{36} & \_ \\ \_ & \_ & \_ \end{pmatrix}$$

Matrix Multiplication

- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be multiplied with each other, if the number of columns of $\boldsymbol{A}_{m,\boldsymbol{n}}$ matches the number of rows in $\boldsymbol{B}_{\boldsymbol{n},o}$.
- The resulting matrix $\boldsymbol{C}_{m,o}$ shares the number of rows from $\boldsymbol{A}$ and the number of columns from $\boldsymbol{B}$.
- Matrix multiplication is carried out by multiplying the row-vector of the first matrix with the column-vector of the second matrix.
  Multiply Row by Column

$$\boldsymbol{A}_{2,3} \cdot \boldsymbol{B}_{3,3} = \begin{pmatrix} 3 & 6 & 5 \\ \mathbf{4} & \mathbf{2} & \mathbf{1} \end{pmatrix} \cdot \begin{pmatrix} 4 & 3 & \mathbf{8} \\ 1 & 2 & \mathbf{10} \\ 7 & 3 & \mathbf{2} \end{pmatrix} = \begin{pmatrix} 53 & 36 & \_ \\ \_ & \_ & ? \end{pmatrix}$$

Matrix Multiplication

- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be multiplied with each other, if the number of columns of $\boldsymbol{A}_{m,\boldsymbol{n}}$ matches the number of rows in $\boldsymbol{B}_{\boldsymbol{n},o}$.
- The resulting matrix $\boldsymbol{C}_{m,o}$ shares the number of rows from $\boldsymbol{A}$ and the number of columns from $\boldsymbol{B}$.
- Matrix multiplication is carried out by multiplying the row-vector of the first matrix with the column-vector of the second matrix.
  Multiply Row by Column

$$\boldsymbol{A}_{2,3} \cdot \boldsymbol{B}_{3,3} = \begin{pmatrix} 3 & 6 & 5 \\ \mathbf{4} & \mathbf{2} & \mathbf{1} \end{pmatrix} \cdot \begin{pmatrix} 4 & 3 & \mathbf{8} \\ 1 & 2 & \mathbf{10} \\ 7 & 3 & \mathbf{2} \end{pmatrix} = \begin{pmatrix} 53 & 36 & - \\ - & - & (4 * 8 + 2 * 10 + 1 * 2) \end{pmatrix}$$

Matrix Multiplication

- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be multiplied with each other, if the number of columns of $\boldsymbol{A}_{m,\boldsymbol{n}}$ matches the number of rows in $\boldsymbol{B}_{\boldsymbol{n},o}$.
- The resulting matrix $\boldsymbol{C}_{m,o}$ shares the number of rows from $\boldsymbol{A}$ and the number of columns from $\boldsymbol{B}$.
- Matrix multiplication is carried out by multiplying the row-vector of the first matrix with the column-vector of the second matrix.
  Multiply Row by Column

$$\boldsymbol{A}_{2,3} \cdot \boldsymbol{B}_{3,3} = \begin{pmatrix} 3 & 6 & 5 \\ \mathbf{4} & \mathbf{2} & \mathbf{1} \end{pmatrix} \cdot \begin{pmatrix} 4 & 3 & \mathbf{8} \\ 1 & 2 & \mathbf{10} \\ 7 & 3 & \mathbf{2} \end{pmatrix} = \begin{pmatrix} 53 & 36 & \_ \\ \_ & \_ & \mathbf{54} \end{pmatrix}$$

Matrix Multiplication

- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be multiplied with each other, if the number of columns of $\boldsymbol{A}_{m,\boldsymbol{n}}$ matches the number of rows in $\boldsymbol{B}_{\boldsymbol{n},o}$.
- The resulting matrix $\boldsymbol{C}_{m,o}$ shares the number of rows from $\boldsymbol{A}$ and the number of columns from $\boldsymbol{B}$.
- Matrix multiplication is carried out by multiplying the row-vector of the first matrix with the column-vector of the second matrix.
  Multiply Row by Column

$$\boldsymbol{A}_{2,3} \cdot \boldsymbol{B}_{3,3} = \begin{pmatrix} 3 & 6 & 5 \\ \mathbf{4} & \mathbf{2} & \mathbf{1} \end{pmatrix} \cdot \begin{pmatrix} 4 & 3 & \mathbf{8} \\ 1 & 2 & \mathbf{10} \\ 7 & 3 & \mathbf{2} \end{pmatrix} = \begin{pmatrix} 53 & 36 & 94 \\ 25 & 19 & 54 \end{pmatrix}$$

The Numpy Module

- Numpy is part of SciPy **the** module for scientific programming
- It should have been installed with matplotlib
- It is usually imported like this:

```
import numpy as np
```

The Numpy Array

- Numpy brings its own data structure the numpy array

```
import numpy as np
#Arrays can be created from lists
array_example = np.array([1,6,7,9])
#Arrays can be created with arange
#An array with numbers from 4 to 5 and step size 0.2
array2 = np.arange(4,5,0.2) #5 is not in the array
print(array2) # [4.0 4.2 4.4 4.6 4.8]
```

- Elements of an array can be manipulated simultaneously

```
array3 = array2*array2 #For example with multiplication
print(array3)# [16.0 16.64 19.36 21.16 23.04]
```

Matplotlib and Numpy

- Plotting $sin(x)$ from 0 to π with lists

```
listX=[]
listY=[]
step_size = 0.5
for i in range(0,int(math.pi/step_size)):
        xValue = i*step_size
        listX.append(xValue)
        listY.append(math.sin(xValue))
plt.plot(listX,listY)
```

- Plotting $sin(x)$ from 0 to π with numpy

```
xValues = np.arange(0,math.pi,0.5)
yValues = np.sin(xValues)
plt.plot(xValues,yValues)
```

Numpy Arrays as Matrices

- Creating the following matrix: $\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$

Numpy Arrays as Matrices

- Creating the following matrix: $\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$
- In numpy a matrix can be created from a multi-dimensional list

```
# This creates a 3x4 Matrix
A = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
```

Numpy Arrays as Matrices

- Creating the following matrix: $\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$
- In numpy a matrix can be created from a multi-dimensional list

```
# This creates a 3x4 Matrix
A = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
```

- Numpy treats such an array as a matrix

```
arr_dim = A.shape #Gives you the shape of your matrix
print(arr_dim) #Prints (3,4)
# Access elements with indexing
single_number = A[1,3] #8, 2nd list,4th element
num2 = A[0,1] #2, 1st list, 2nd element
```

Matrix Operations in Numpy

- Matrix Addition: $\begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \end{pmatrix} + \begin{pmatrix} 3 & 5 & 1 \\ 5 & -3 & 1 \end{pmatrix} = \begin{pmatrix} 4 & 7 & 4 \\ 10 & 3 & 8 \end{pmatrix}$
- In numpy code:

```
A = np.array([[1,2,3], [5,6,7]])
B = np.array([[3,5,1], [5,-3,1]])
C = A + B
D = A - B #Subtraction works analogously
print(D) #[[-2 -3 2],[0 9 6]]
```

Matrix Operations in Numpy

- Matrix Multiplication: $\begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \end{pmatrix} * \begin{pmatrix} 3 & 5 \\ 5 & -3 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 16 & 2 \\ 52 & 14 \end{pmatrix}$
- In numpy code:

```
A = np.array([[1,2,3], [5,6,7]])
E  = np.array([[3,5], [5,-3],[1,1]])
F = np.matmul(A,E)
print(F) # [[16,2],[52,14]]
```

Matrix Operations in Numpy

- Matrix Multiplication: $\begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \end{pmatrix} * \begin{pmatrix} 3 & 5 \\ 5 & -3 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 16 & 2 \\ 52 & 14 \end{pmatrix}$
- In numpy code:

```
A = np.array([[1,2,3], [5,6,7]])
E  = np.array([[3,5], [5,-3],[1,1]])
F = np.matmul(A,E)
print(F) # [[16,2],[52,14]]
```

- Do not confuse with element-wise multiplication

```
A = np.array([[1,2,3], [5,6,7]])
B = np.array([[3,5,1], [5,-3,1]])
G = A*B # [[3,10,3],[25,-18,7]]
```

Matrix Operations in Numpy

- It also works for vectors:

$$< v_1, v_2 >= {v_1}^T v_2 = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} * \begin{pmatrix} 3 \\ 5 \\ 1 \end{pmatrix} = 16$$

- In numpy code:

```
V1 = np.array([1,2,3])
V2  = np.array([3,5,1])
R = np.matmul(V1,V2)
print(R) # 16
```

Matrix Operations in Numpy

- It also works for vectors:

$$< v_1, v_2 >= {v_1}^T v_2 = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} * \begin{pmatrix} 3 \\ 5 \\ 1 \end{pmatrix} = 16$$

- In numpy code:

```
V1 = np.array([1,2,3])
V2  = np.array([3,5,1])
R = np.matmul(V1,V2)
print(R) # 16
```

- Or vectors and matrices if you want to

Other helpful Operations

- Transpose Matrices: $\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \end{pmatrix} \quad \boldsymbol{A}^T = \begin{pmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \end{pmatrix}$

- In numpy:

```
A = np.array([[1,2,3], [5,6,7]])
H = A.T # [[1,5],[2,6],[3,7]]
```

- Element-wise summing across arrays:

```
sum = np.sum(H) #24,
V1 = np.array([1,2,3]) #works also for 1D-arrays
sum_v = np.sum(V1) # 6
```

Images as Matrices

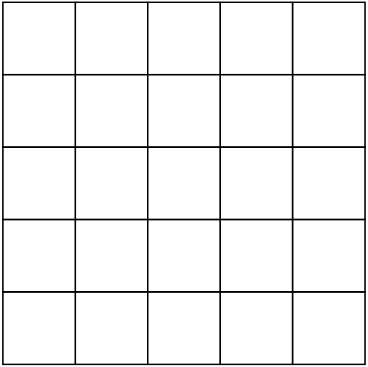
$$\begin{pmatrix} x_{0,0} & x_{0,1} & x_{0,2} & x_{0,3} & x_{0,4} \\ x_{1,0} & x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,0} & x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,0} & x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,0} & x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{pmatrix}$$

Images as Matrices

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

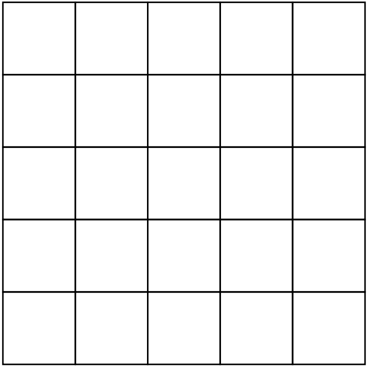
Images as Matrices

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



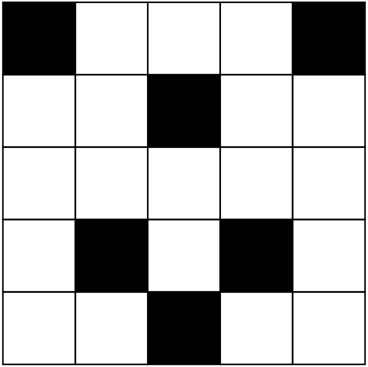
Images as Matrices

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$



Images as Matrices

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$



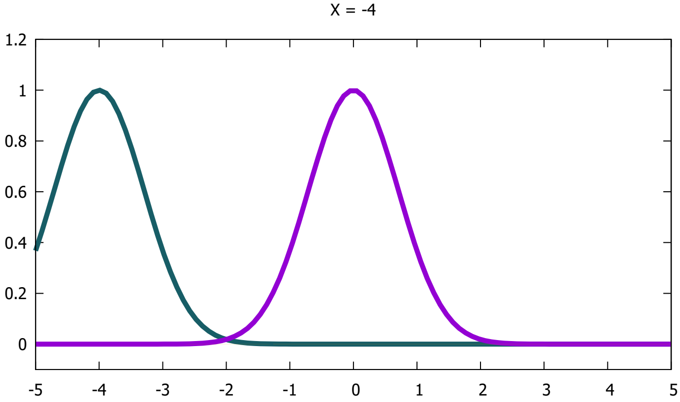
Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

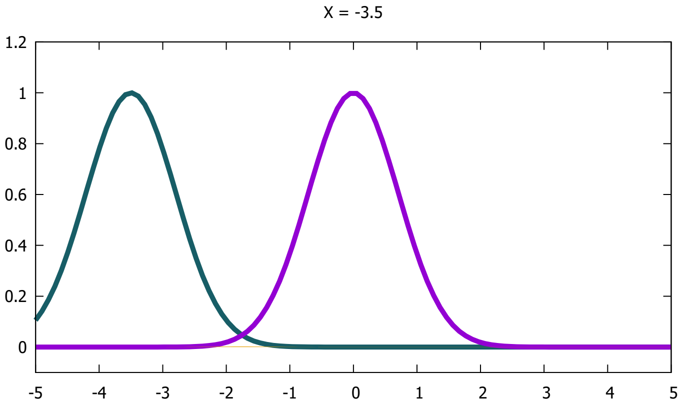
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

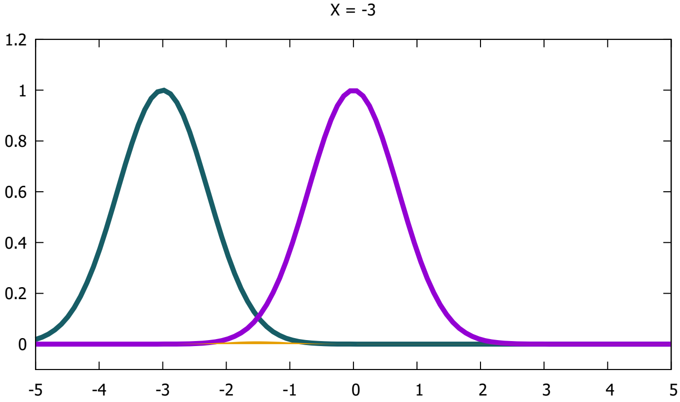
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

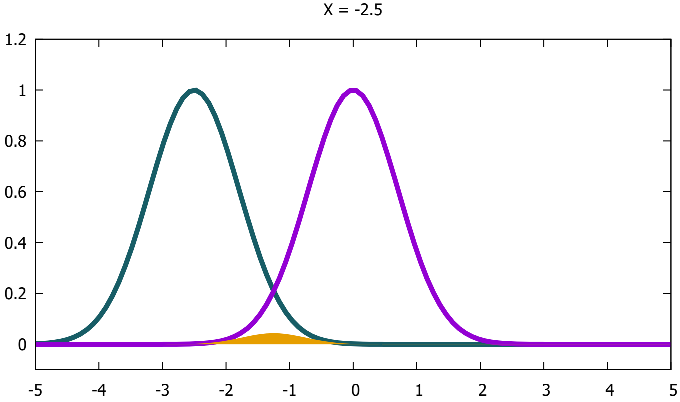
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

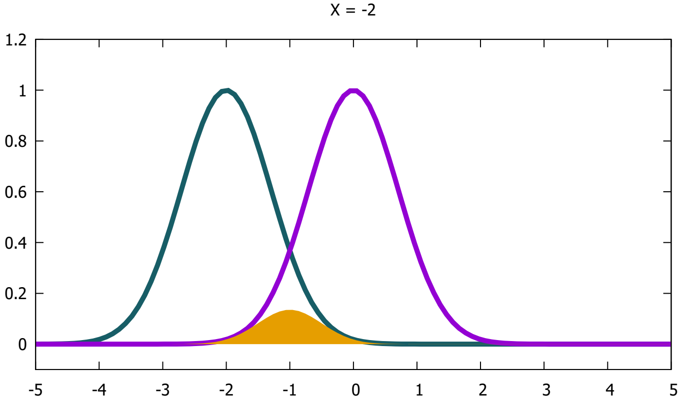
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

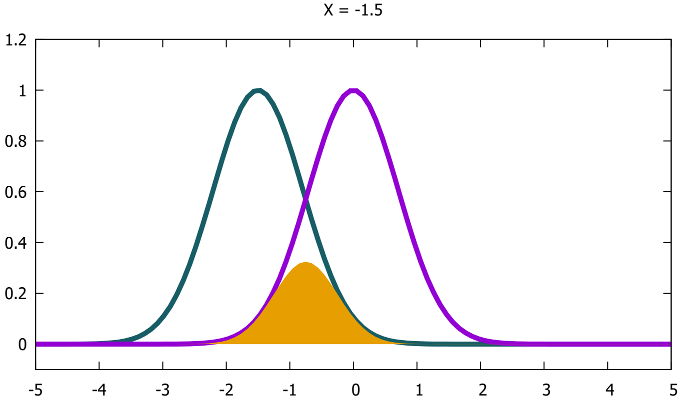
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

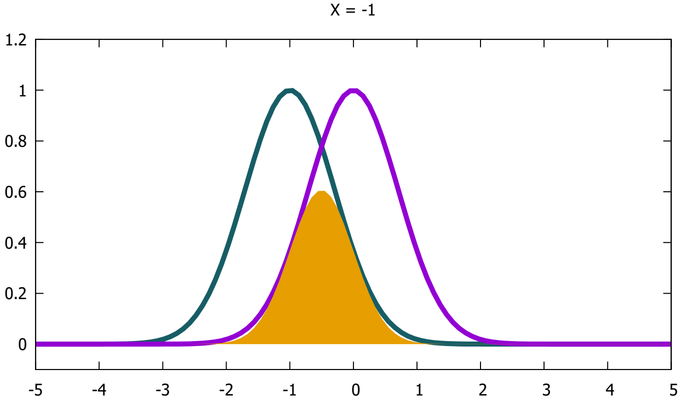
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

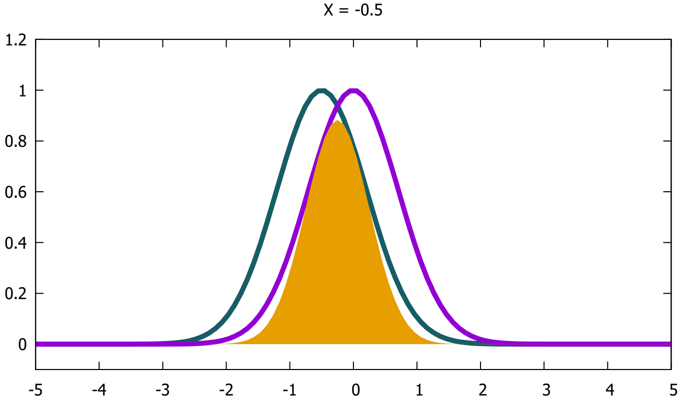
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

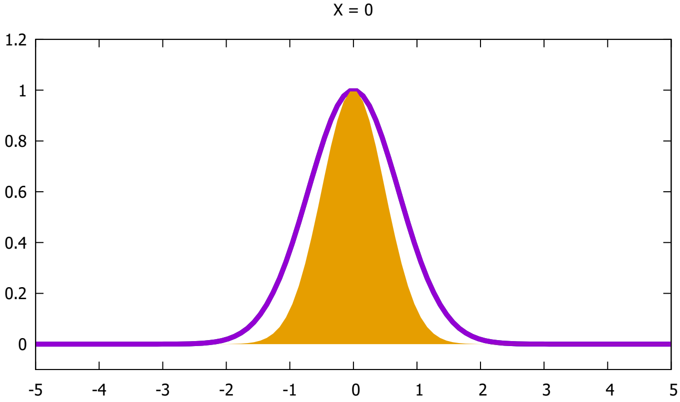
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

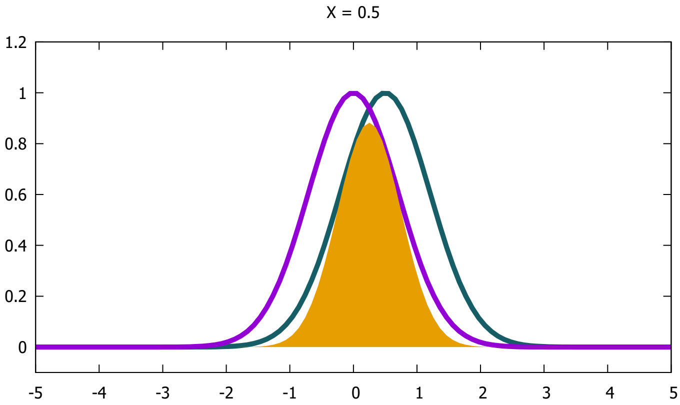
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

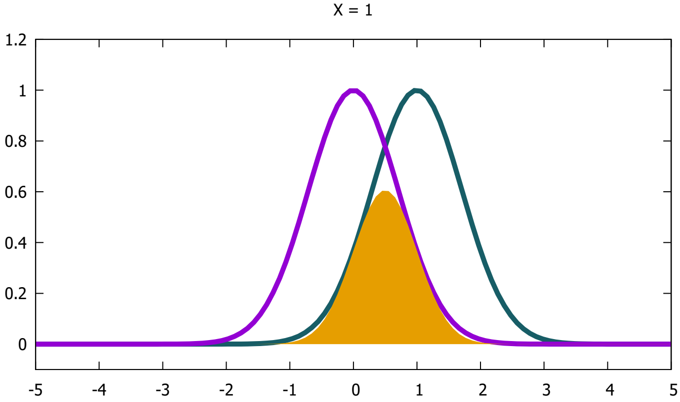
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

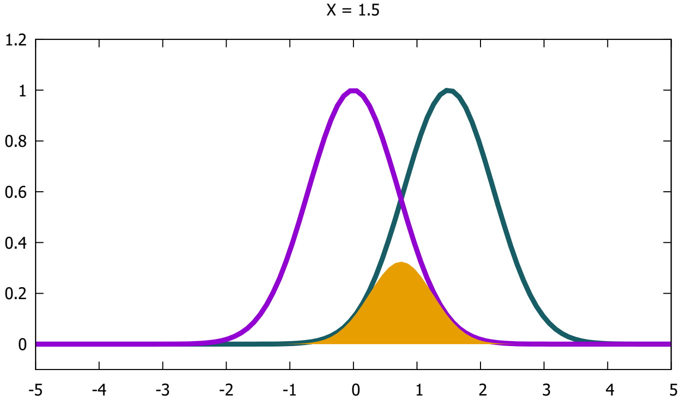
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

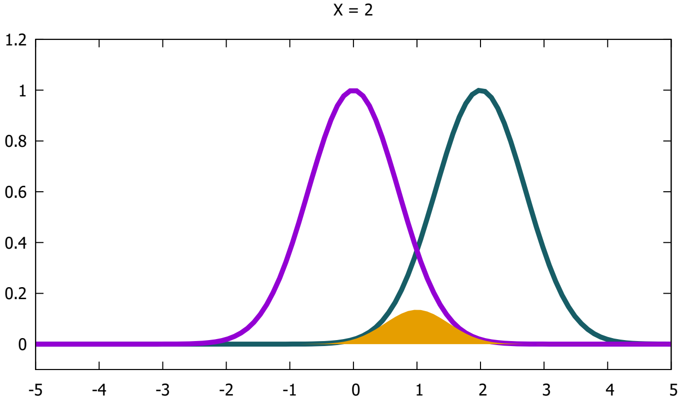
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

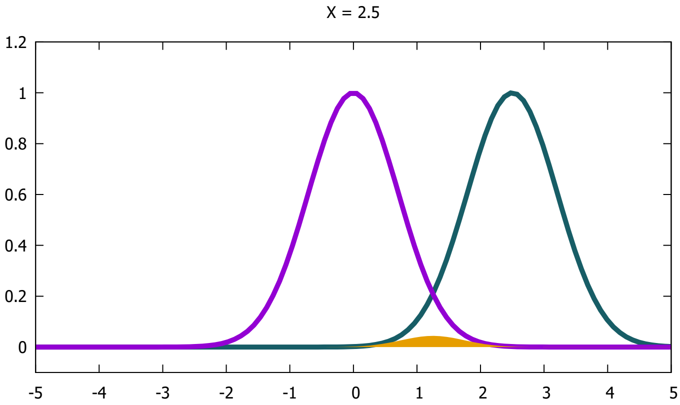
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

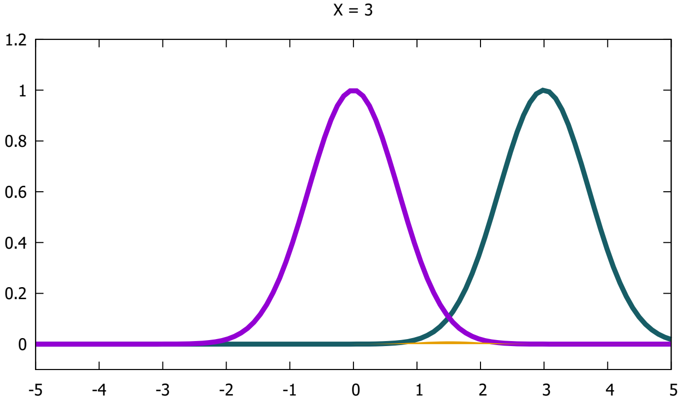
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

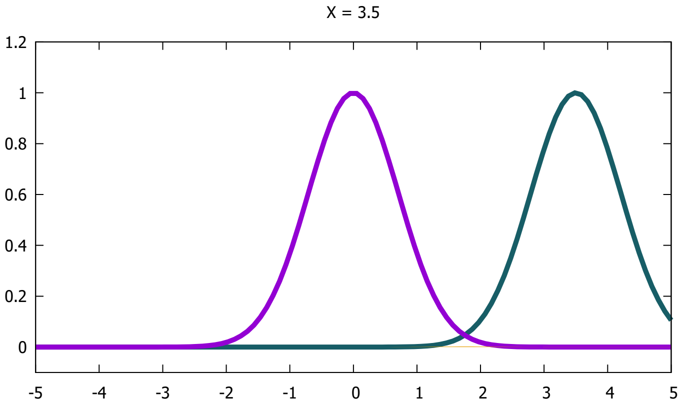
Convolution of the Gaussian function with itself



Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

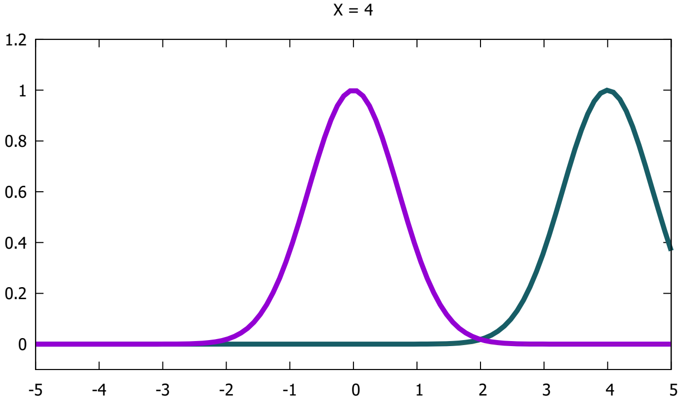
Convolution of the Gaussian function with itself



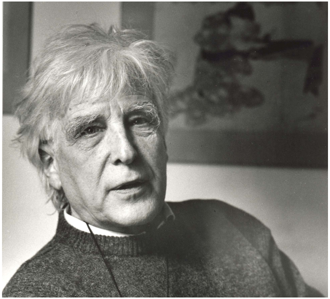
Convolution of Functions

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x')g(x - x')dx'$$

Convolution of the Gaussian function with itself



Applying Filters to Images

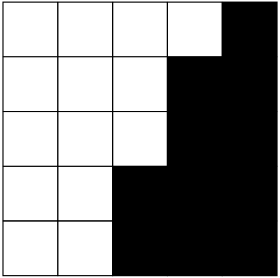


Convolution with Matrices

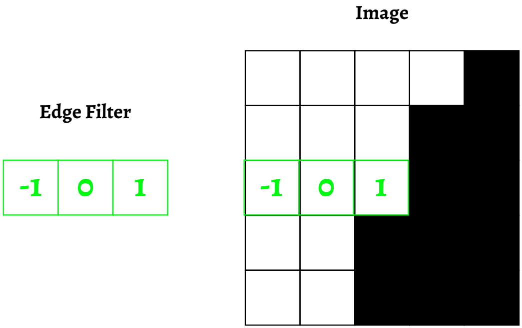
Edge Filter

-1	0	1

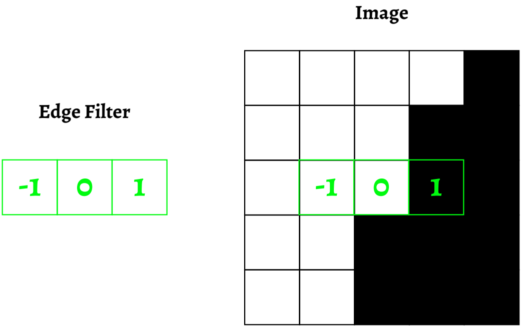
Image



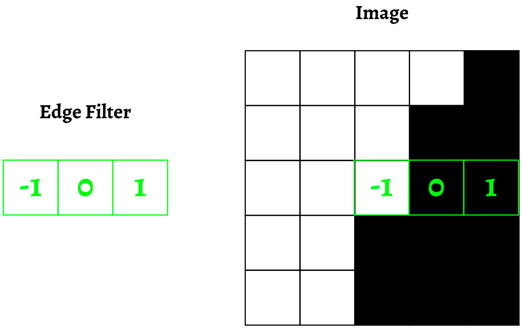
Convolution with Matrices



Convolution with Matrices



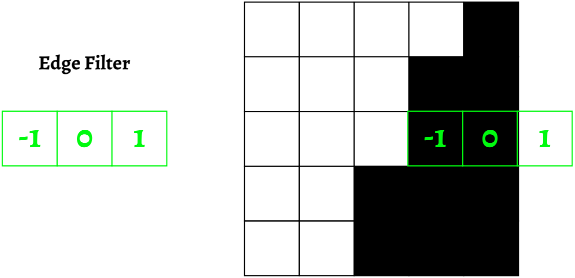
Convolution with Matrices



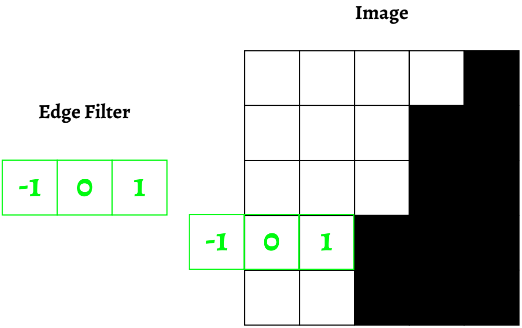
Convolution with Matrices

Image

Edge Filter



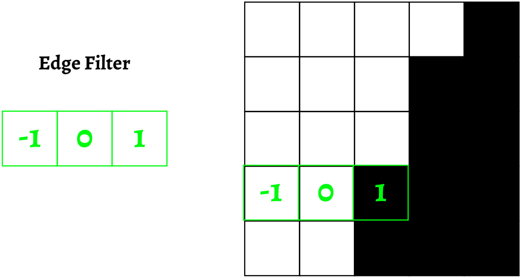
Convolution with Matrices



Convolution with Matrices

Image

Edge Filter

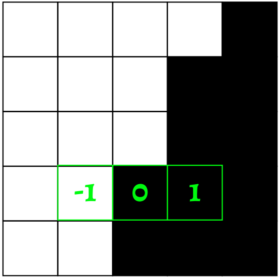


Convolution with Matrices

Edge Filter

-1	0	1

Image



Convolution with Matrices

Edge Filter

-1	0	1

Image