Machine Learning Techniques for Text

Module 0: Python Crash Course

Dr. Nikos Tsourakis

Course outline

- **Module 0: Python Crash Course**
- Module 1: Intro to Machine Learning
- Module 2: Detecting Spam Emails
- Module 3: Classifying Topics of Newsgroup Posts
- Module 4: Extracting Sentiments from Product Reviews
- Module 5: Recommending Music Titles
- Module 6: Teaching Machines to Translate
- Module 7: Summarizing Wikipedia Articles
- Module 8: Detecting Hateful and Offensive Language
- Module 9: Generating Text in Chatbots
- Module 10: Clustering Speech-to-Text Transcriptions

Overview

- ***Python*** is programming language chosen for its versatility, readability, and a vibrant community
- With applications in web development, data science, and artificial intelligence, Python is an excellent language for beginners
- This crash course provides a rapid introduction to Python, emphasizing practical skills through hands-on exercises
- It serves as a foundation for further learning and offers skills applicable across various industries, opening doors to diverse career opportunities

Module objectives

After completing this module, you should be able to:

- Understand Python's fundamental syntax and control flow
- Comprehend essential data structures and their utility
- Define and call functions, grasp function parameters, and return values
- Implement basic error handling
- Read from and write to files in Python
- Request and parse user input
- Work with common Python libraries and frameworks (e.g., NumPy and pandas)

Machine Learning Techniques for Text

Section 1: The Python programming language

What is Python?

- ***Python*** is a high-level, interpreted programming language known for its simplicity, readability, and versatility
- Created by Guido van Rossum in the late 1980s, Python has become one of the most popular languages for both beginners and experienced developers
- Its syntax is designed to be clear and concise, emphasizing readability and reducing the cost of program maintenance
- Python supports multiple programming paradigms, including *procedural, object-oriented, and functional programming, making it* adaptable to various use cases

What is Python?

- With a vast standard library and a thriving ecosystem of third-party packages, Python is widely employed in web development, data science, artificial intelligence, automation, and more
- Its ease of learning and powerful capabilities make Python an ideal choice for individuals and organizations seeking an accessible yet robust language for diverse applications
- Major online platforms like Google, Dropbox, Instagram, and Spotify & YouTube — all had worked with this programming language
- You can write Python programs using a simple text editor, or use an ***Integrated Development Environment*** (IDE)
    - E.g.: Visual Studio Code

Python code for data science

- Useful packages
  - ***NumPy*** provides efficient storage and computation for multidimensional data arrays
  - ***SciPy*** contains a wide array of numerical tools such as numerical integration and interpolation
  - ***Pandas*** provides a DataFrame object along with a powerful set of methods to manipulate, filter, group, and transform data
  - ***Matplotlib*** provides a useful interface for creation of publication-quality plots and figures
  - ***Scikit-Learn*** provides a uniform toolkit for applying common machine learning algorithms to data

Python help

https://docs.python.org/3/

Python » English 3.12.0 3.12.0 Documentation »

Download

Download these documents

Docs by version

Python 3.13 (in development)
Python 3.12 (stable)
Python 3.11 (stable)
Python 3.10 (security-fixes)
Python 3.9 (security-fixes)
Python 3.8 (security-fixes)
Python 3.7 (EOL)
Python 3.6 (EOL)
Python 3.5 (EOL)
Python 3.4 (EOL)
Python 3.3 (EOL)
Python 3.2 (EOL)
Python 3.1 (EOL)
Python 3.0 (EOL)
Python 2.7 (EOL)
Python 2.6 (EOL)

Python 3.12.0 documentation

Welcome! This is the official documentation for Python 3.12.0.

Parts of the documentation:

What's new in Python 3.12?
or all "What's new" documents since 2.0

Tutorial
start here

Library Reference
keep this under your pillow

Language Reference
describes syntax and language elements

Installing Python Mod
installing from the Python Pacl
sources

Distributing Python M
publishing modules for installa

Extending and Embe
tutorial for C/C++ programmer

Python help

https://docs.python.org/3/

Python » English 3.12.0 3.12.0 Documentation »

Download

Download these documents

Docs by version

- Python 3.13 (in development)
- Python 3.12 (stable)
- Python 3.11 (stable)
- Python 3.10 (security-fixes)
- Python 3.9 (security-fixes)
- Python 3.8 (security-fixes)
- Python 3.7 (EOL)
- Python 3.6 (EOL)
- Python 3.5 (EOL)
- Python 3.4 (EOL)
- Python 3.3 (EOL)
- Python 3.2 (EOL)
- Python 3.1 (EOL)
- Python 3.0 (EOL)
- Python 2.7 (EOL)
- Python 2.6 (EOL)

Python 3.12.0 documentation

Welcome! This is the official documentation for Python 3.12.0.

Parts of the documentation:

What's new in Python 3.12?

or all "What's new" documents since 2.0

Tutorial

start here

Library Reference

keep this under your pillow

Language Reference

describes syntax and language elements

Installing Python Mod

installing from the Python Pacl

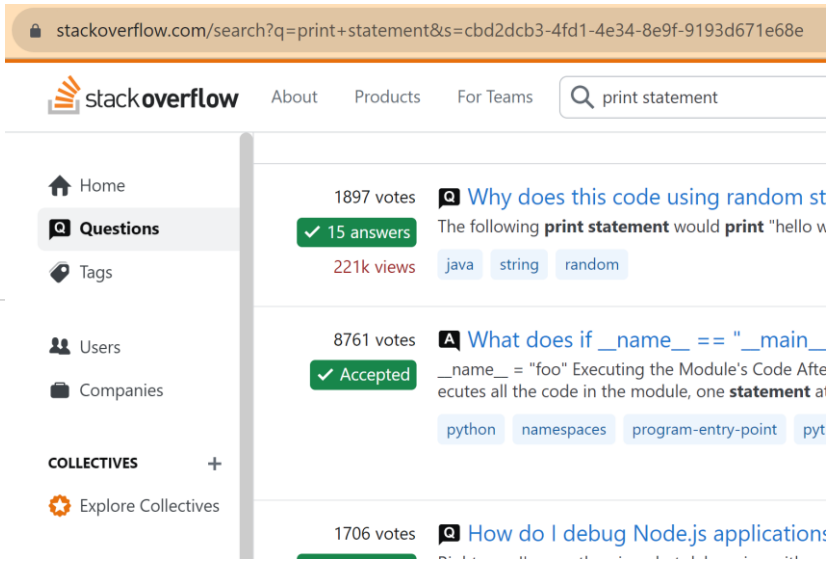
sources

Distributing Python M

publishing modules for installa

Extending and Embe

tutorial for C/C++ programmer



Python help

https://docs.python.org/3/

Python » English 3.12.0 3.12.0 Documentation »

Download

Download these documents

Docs by version

Python 3.13 (in development)
Python 3.12 (stable)
Python 3.11 (stable)
Python 3.10 (security-fixes)
Python 3.9 (security-fixes)
Python 3.8 (security-fixes)
Python 3.7 (EOL)
Python 3.6 (EOL)
Python 3.5 (EOL)
Python 3.4 (EOL)
Python 3.3 (EOL)
Python 3.2 (EOL)
Python 3.1 (EOL)
Python 3.0 (EOL)
Python 2.7 (EOL)
Python 2.6 (EOL)

Python 3.12.0 documentation

Welcome! This is the official documentation for Python 3.12.0.

Parts of the documentation:

What's new in Python 3.12?
or all "What's new" documents since 2.0

Tutorial
start here

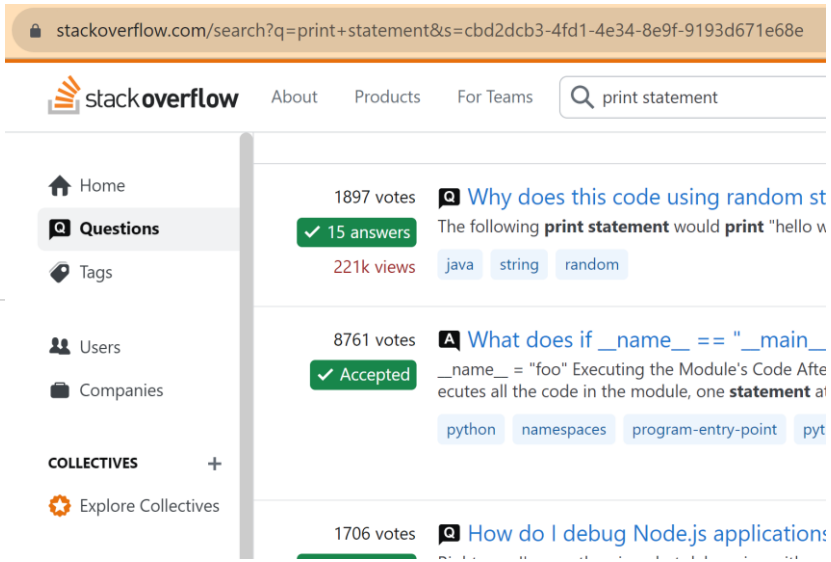
Library Reference
keep this under your pillow

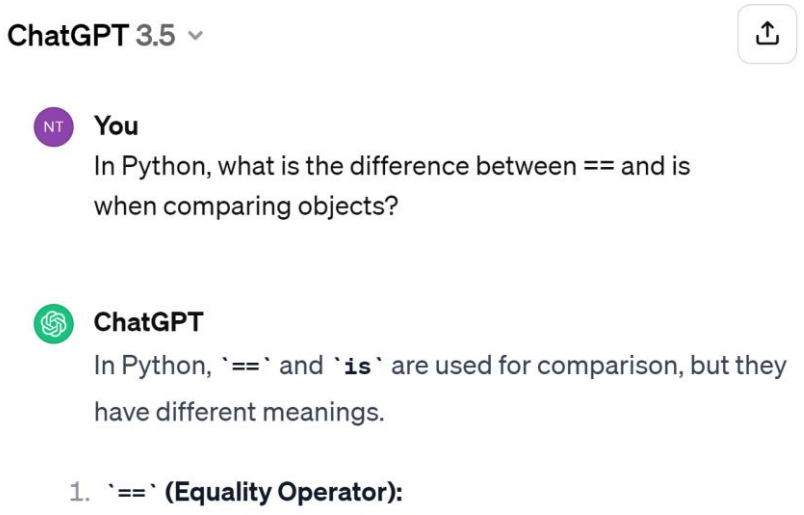
Language Reference
describes syntax and language elements

Installing Python Moc
installing from the Python Pacl
sources

Distributing Python M
publishing modules for installa

Extending and Embe
tutorial for C/C++ programmer

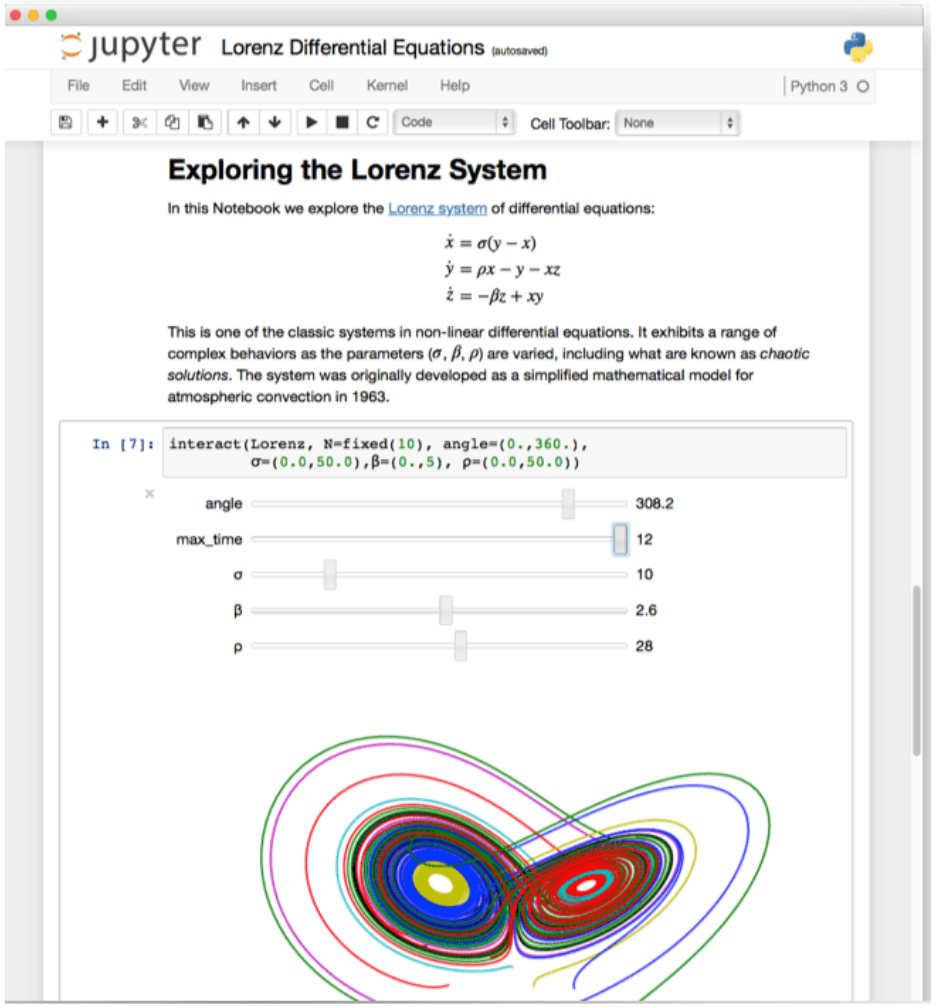




Machine Learning Techniques for Text

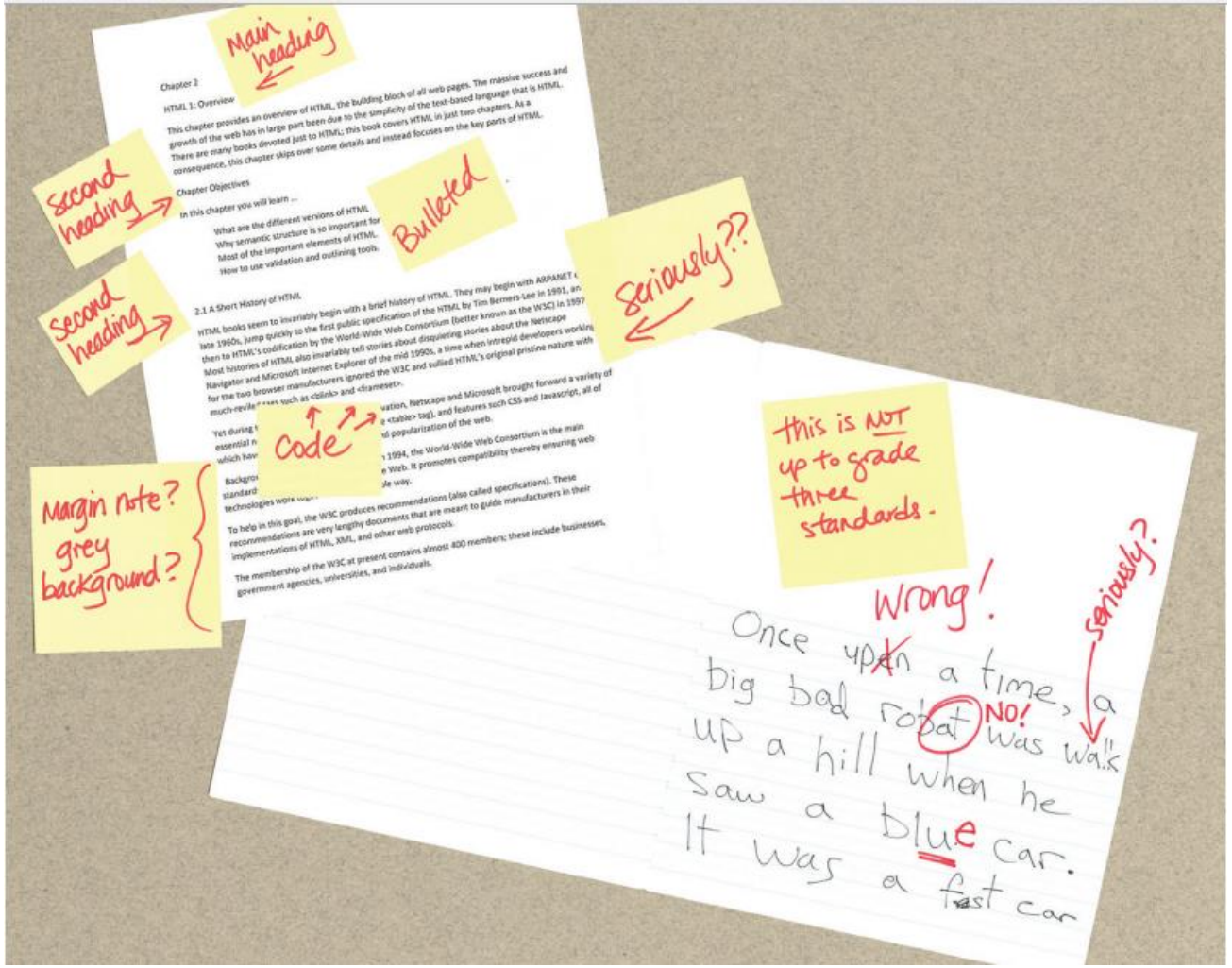
Section 2: Understanding Jupyter notebooks

Jupyter Notebooks



- ***Jupyter Notebooks*** provide an interactive computing environment that allows users to create and share documents containing live code, equations, visualizations, and narrative text
- Jupyter supports various programming languages, including but not limited to Python, R, and Julia
- Notebooks are organized into cells, each of which can contain code, text (formatted using ***Markdown***), or rich media outputs

Ad-hoc markup languages



Markup languages

"In computer text processing, a markup language is a system for annotating a document in a way that is syntactically distinguishable from the text, meaning when the document is processed for display, the markup language is not shown, and is only used to format the text"
[Wikipedia]

Hypertext Markup Language

- ***Hypertext Markup Language*** (HTML) is the standard markup language for documents designed to be displayed in a web browser
- The “information about content” in HTML is implemented via ***HTML tags***


```
<!DOCTYPE html>
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <div>
        <p>Hello world!</p>
    </div>
  </body>
</html>
```

eXtensible Markup Language

- ***eXtensible Markup Language*** (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable
- It is designed to store and transport data
- Uses the same notion of ***tags***


```
<SampleXML>
  <Colors>
    <Color1>White</Color1>
    <Color2>Blue</Color2>
    <Color3>Black</Color3>
    <Color4 Special="Light">Green</Color4>
    <Color5>Red</Color5>
  </Colors>
  <Fruits>
    <Fruits1>Apple</Fruits1>
    <Fruits2>Pineapple</Fruits2>
    <Fruits3>Grapes</Fruits3>
    <Fruits4>Melon</Fruits4>
  </Fruits>
</SampleXML>
```

LaTeX

- LaTeX is a software system for document preparation
- The writer uses plain text as opposed to the formatted text found in "WYSIWYG" word processors like Microsoft Word
- Uses markup ***tagging*** conventions to define the general structure of a document, to stylise text, etc.


```
  % The following shows
typesetting  power of
LaTeX:
  \begin{align}
    E_0 &= mc^2 \\
    E &= \frac{mc^2}
{\sqrt{1-\frac{v^2}
{c^2}}}
  \end{align}
\end{document}
```

$$E_0 = mc^2$$

$$E = \frac{mc^2}{\sqrt{1 - \frac{v^2}{c^2}}}$$

Markdown overview

- ***Markdown*** is a lightweight markup language for creating formatted text
- Markdown is designed to be as easy-to-read and easy-to-write as possible. Benefits:
  - Interactivity and exploration (run code in a step-by-step manner)
  - Mixed content (mix of code, text, and visualizations)
  - Collaboration and sharing (easily shareable)
- Often converted into the corresponding HTML
  - Supported by Jupyter:
    - `Headings, Blockquotes, Code section, Mathematical Symbol, Line Break, Bold and Italic Text, Horizontal Lines, Ordered List, Unordered List, Internal and External Link, Table`

Markdown example syntax

Symbols	*versus*	**HTML**
# (Header 1, title)		Header 1,title
This is bold text		This is bold text
This is italic text		This is italic text
---		<hr>
__[text link](https://www.google.com)__		 Link to Google
$\sqrt{k}$		-

Markdown example cell


```
# Jupyter Notebook Demo

## Introduction

Welcome to this Jupyter Notebook demo. In this notebook, we'll explore
various features, including:
- **Text formatting**
- **Code snippets**
- **Lists**
- **Tables**
- **Images**
- **Equations**

## Code Snippets

Let's start with a simple Python code snippet:

```python
# Python code snippet
def greet(name):
    return f"Hello, {name}!"

print(greet("World"))
```

Jupyter Notebook Demo

Introduction

Welcome to this Jupyter Notebook demo. In this notebook, we'll explore various features, including:

- **Text formatting**
- **Code snippets**
- **Lists**
- **Tables**
- **Images**
- **Equations**

Code Snippets

Let's start with a simple Python code snippet:

```python
# Python code snippet
def greet(name):
    return f"Hello, {name}!"

print(greet("World"))
```
```

Machine Learning Techniques for Text

Section 3: Basic Python

What are variables?

- In Python, a ***variable*** is a named location in the computer's memory that stores a value
- Think of it as a container or a label that you can use to refer to a specific piece of data
- Unlike some other programming languages, Python does not require explicit declaration of the variable type
- You can simply assign a value to a variable, and Python will determine its type dynamically

What are variables?

Bulk Cargo

Corrugated Pads

Easy-Fold Mailers

32 ECT Lightweight

Multi-Depth

Storage File

Moving Boxes

Side Loaders

Haz Mat

Wine Shippers

Insulated Shippers

Wood Crates

What are variables?

integer

float

string

boolean

list

tuple

dictionary

set

array

series

date-time

dataframe

Why does it matter?

Why does it matter?

Why does it matter?

Many “boxes” in a single process!

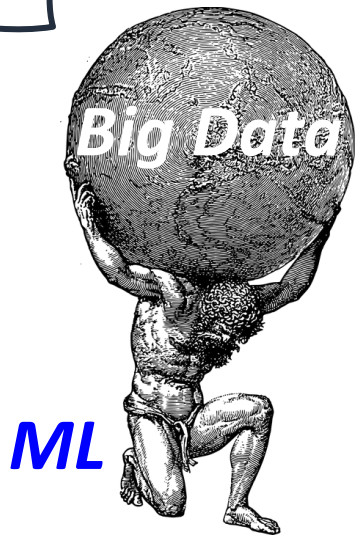
Many “containers” is a single server!

Why does it matter?

Many “boxes” in a single process!

Many “containers” is a single server!

- ✓Understand which data type to use
- ✓Comprehend the type returned by the libraries



- ✓But also assess the impact of your choices on the underlying platform

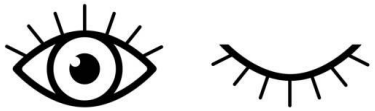
How can we use an eye to indicate ***yes*** or ***no***?

How can we use an eye to indicate ***yes*** or ***no***?

Tip *: You can open to close you eye to represent this information*

How can we use an eye to indicate ***yes*** or ***no***?

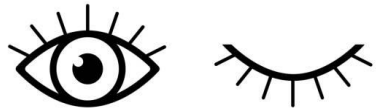
Tip *: You can open to close you eye to represent this information*



Yes No

Tip *: You can open to close you eye to represent this information*

How can we use an eye to indicate ***yes*** or ***no***?

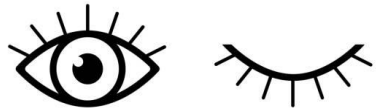


Yes No

How can we represent the four directions ***north***, ***est***, ***south*** and ***west***?

Tip *: You can open to close you eye to represent this information*

How can we use an eye to indicate ***yes*** or ***no***?



Yes No

How can we represent the four directions ***north***, ***est***, ***south*** and ***west***?

north east south west

How many different numbers?

- There are two different binary numbers with one ***bit*** (one eye):

0 

1 

How many different numbers?

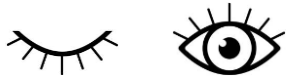
- There are two different binary numbers with one ***bit*** (one eye):

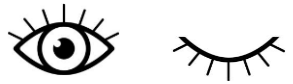
  0 

  1 

- There are four different binary numbers with two bits:

  00 (decimal 0) 

  01 (decimal 1) 

  10 (decimal 2) 

  11 (decimal 3) 

How many different numbers?

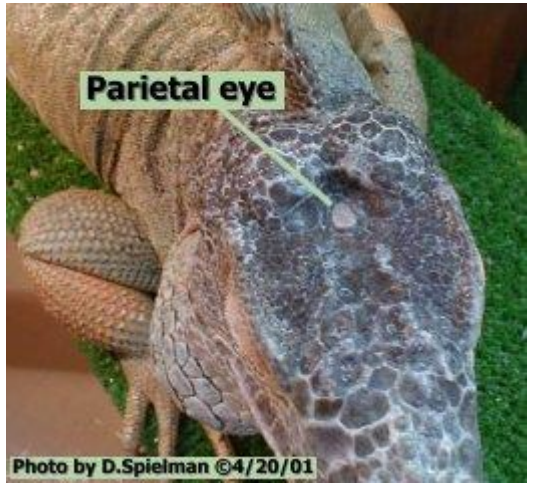
- There are two different binary numbers with three bit:

000	(decimal 0)	
001	(decimal 1)	
010	(decimal 2)	
011	(decimal 3)	
100	(decimal 4)	
101	(decimal 5)	
110	(decimal 6)	
111	(decimal 7)	

How many different numbers?

- There are two different binary numbers with three bit:

000	(decimal 0)	
001	(decimal 1)	
010	(decimal 2)	
011	(decimal 3)	
100	(decimal 4)	
101	(decimal 5)	
110	(decimal 6)	
111	(decimal 7)	



tuatara

How many numbers can we represent with one ***byte***?

1 byte

How many numbers can we represent with one ***byte***?

1 byte

$$2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^8 = 256$$

How many numbers can we represent with one ***byte***?

1 byte

$2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^8 = 256$

0	0	0	0	0	0	0	0

= 0

1	1	1	1	1	1	1	1

= 255

How many numbers can we represent with one ***byte***?

- 1 Bit = Binary Digit
- 8 Bits = 1 Byte
- 1024 Bytes = 1 Kilobyte
- 1024 Kilobytes = 1 Megabyte
- 1024 Megabytes = 1 Gigabyte
- 1024 Gigabytes = 1 Terabyte
- 1024 Terabytes = 1 Petabyte
- 1024 Petabytes = 1 Exabyte
- 1024 Exabytes = 1 Zettabyte
- 1024 Zettabytes = 1 Yottabyte
- 1024 Yottabytes = 1 Brontobyte
- 1024 Brontobytes = 1 Geopbyte

1 byte

$2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^8 = 256$

0	0	0	0	0	0	0	0

= 0

1	1	1	1	1	1	1	1

= 255

Built-in data types

- Python comes with several built-in data types that define the nature of a variable
- Here are some common ones:
  - ***int***: Integer type for whole numbers
  - ***float***: Floating-point type for decimal numbers
  - ***str***: String type for text
  - ***bool***: Boolean type for representing truth values (***True*** or ***False***)
  - ***list***: Ordered collection of items
  - ***tuple***: Immutable ordered collection of items
  - ***dict***: Dictionary type for key-value pairs
  - ***set***: A collection of unique elements with no duplicate values

Type casting

- Transform a floating point number to an integer
  - int(3.7)
- Transform an integer to a floating point number
  - float(2)
- Transform a string to a floating point number
  - float("3.2")
- What is the result of
  - int(float("3.2"))
- Can be applied between certain built-in types, if there is a logical and meaningful conversion between them

Type casting

- Transform a floating point number to an integer
  - int(3.7) → 3
- Transform an integer to a floating point number
  - float(2)
- Transform a string to a floating point number
  - float("3.2")
- What is the result of
  - int(float("3.2"))
- Can be applied between certain built-in types, if there is a logical and meaningful conversion between them

Type casting

- Transform a floating point number to an integer
  - int(3.7) → 3
- Transform an integer to a floating point number
  - float(2) → 2
- Transform a string to a floating point number
  - float("3.2")
- What is the result of
  - int(float("3.2"))
- Can be applied between certain built-in types, if there is a logical and meaningful conversion between them

Type casting

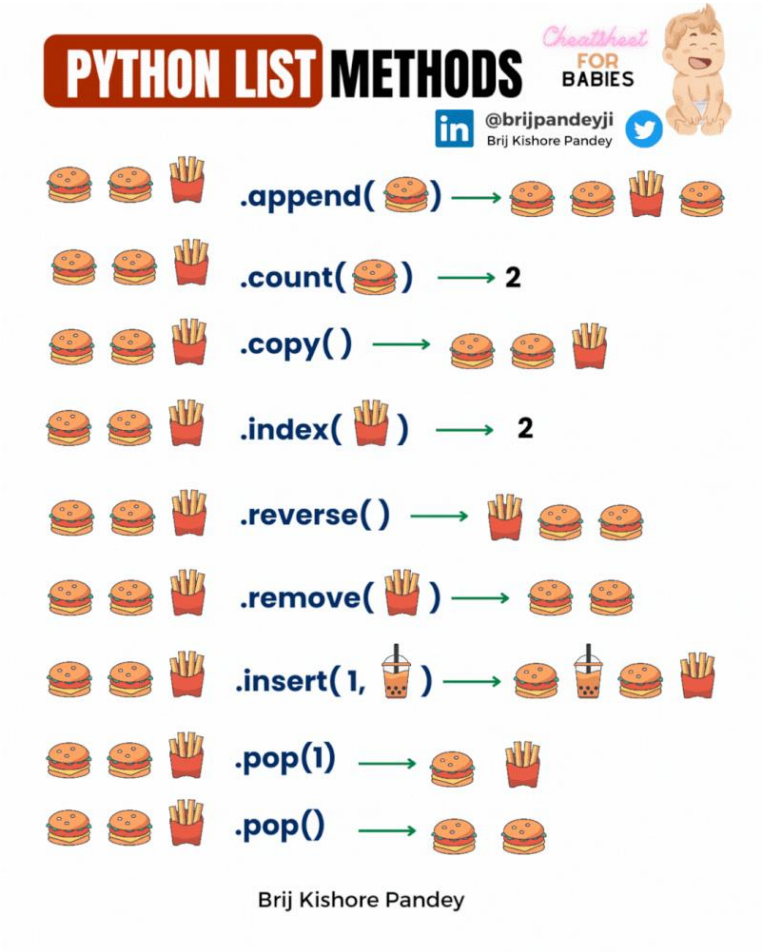
- Transform a floating point number to an integer
  - int(3.7) → 3
- Transform an integer to a floating point number
  - float(2) → 2
- Transform a string to a floating point number
  - float("3.2") → 3.2
- What is the result of
  - int(float("3.2"))
- Can be applied between certain built-in types, if there is a logical and meaningful conversion between them

Type casting

- Transform a floating point number to an integer
  - int(3.7) → 3
- Transform an integer to a floating point number
  - float(2) → 2
- Transform a string to a floating point number
  - float("3.2") → 3.2
- What is the result of
  - int(float("3.2")) → 3
- Can be applied between certain built-in types, if there is a logical and meaningful conversion between them

List methods

list: Ordered collection of items



The **count()** method returns the number of elements with the specified value

- my_list.count(1)

The **append()** method appends an element to the end of the list

- my_list.append(5)

The **remove()** method removes the first occurrence of the element with the specified value

- my_list.remove(5)

The **pop()** method removes the element at the specified position

- my_list.pop(0)

Naming rules

- Variable names
  - Should start with a letter (a-z, A-Z) or an underscore (_)
  - Subsequent characters can be letters, numbers, or underscores
  - Case-sensitive: ***my_var*** and ***My_Var*** are different
- Avoid reserved words
  - Do not use Python reserved words (keywords) as variable names.
  - Examples of reserved words: ***if***, ***else***, ***for***, ***while***, ***class***, etc.
- Descriptive and Readable:
  - Choose names that are descriptive and convey the purpose of the variable
  - Use meaningful words or abbreviations for clarity

Naming rules

- Snake case for *variables*
  - Standard convention is to use snake_case for variable names
  - Example: ***user_name***, ***total_count***, ***average_value***
- Camel Case for *functions* and *methods*
  - Use camelCase for function and method names
  - Example: ***calculateTotal***, ***getUserInfo***, ***processData***
- UPPERCASE for *constants*
  - Constants should be in uppercase with underscores separating words.
  - Example: ***MAX_SIZE***, ***PI_VALUE***, ***DEFAULT_THRESHOLD***
- Consistency matters
  - Maintain consistency in naming styles throughout your codebase

Naming rules

- Snake case for *variables*
  - Standard convention is to use snake_case for variable names
  - Example: ***user_name***, ***total_count***, ***average_value***
- Camel Case for *functions* and *methods*
  - Use camelCase for function and method names
  - Example: ***calculateTotal***, ***getUserInfo***, ***processData***
- UPPERCASE for *constants*
  - Constants should be in uppercase with underscores separating words.
  - Example: ***MAX_SIZE***, ***PI_VALUE***, ***DEFAULT_THRESHOLD***
- Consistency matters
  - Maintain consistency in naming styles throughout your codebase

tip

PEP 8 – Style Guide for Python Code

Conditional statements

Making decisions using ***if*** statement

The logic:

If something is true, then take the action

Example:

```
a=3; b=2

if (a>b):
    print("a is larger than b")
```

```
Output:a is larger than b
```

Conditional statements

The ***if else*** statement

The logic

If something is true, then take action 1. Otherwise, take action 2

Example:

```
a=2; b=2

if (a>b):
    print ("a is larger than b")
else:
    print("a is less than or equal to b")
```

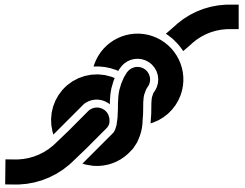
```
Output:a is less than or equal to b
```

for loop

The ***for*** statement

The logic

Use loops to execute a series of repeatable actions



Example:

```
my_list = [1,2,3,4]

for number in my_list:
    print(number)
```

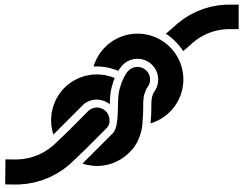
```
Output:
1
2
3
4
```

while loop

The ***while*** statement

The logic

Use loops to execute a series of repeatable actions



Example:

```
a = 0

while (a < 3):
    print("Welcome to Mars!")
    a += 1
```

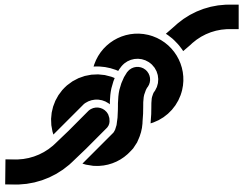
```
Output:
Welcome to Mars!
Welcome to Mars!
Welcome to Mars!
```

while loop

The ***while*** statement

The logic

Use loops to execute a series of repeatable actions



Example:

```
a = 0

while (a < 3):
    print("Welcome to Mars!")
    a += 1
```

```
Output:
Welcome to Mars!
Welcome to Mars!
Welcome to Mars!
```

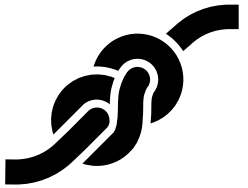
```
while True:
    print("Attack!!!")
```

while loop

The ***while*** statement

The logic

Use loops to execute a series of repeatable actions



Example:

```
a = 0

while (a < 3):
    print("Welcome to Mars!")
    a += 1
```

```
Output:
Welcome to Mars!
Welcome to Mars!
Welcome to Mars!
```

```
while True:
    print("Attack!!!")
```

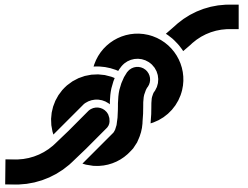
```
Output:
Attack!!! ...
```

while loop

The ***while*** statement

The logic

Use loops to execute a series of repeatable actions



Example:

```
a = 0

while (a < 3):
    print("Welcome to Mars!")
    a += 1
```

```
Output:
Welcome to Mars!
Welcome to Mars!
Welcome to Mars!
```

Infinite loop

```
while True:
    print("Attack!!!")
```

```
Output:
Attack!!! ...
```

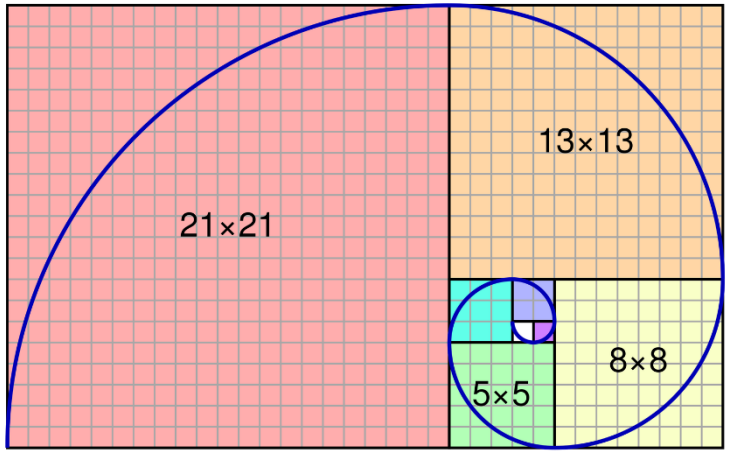
Functions

- Functions are groups of code that have a name and can be called
- They can be defined with the ***def*** statement
- Example:

```
def fibonacci(N):
    nums = []
    a, b = 0, 1
    while len(nums) < N:
        a, b = b, a + b
        nums.append(a)
    return nums

print(fibonacci(10))
```

```
Output:[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```



Know your type

Built-in types

```
v = 1                    # integer
v = 0.5                  # float
v = True                 # boolean
v = 'name'               # string
```

Built-in function:
type(v)

Built-in data structures

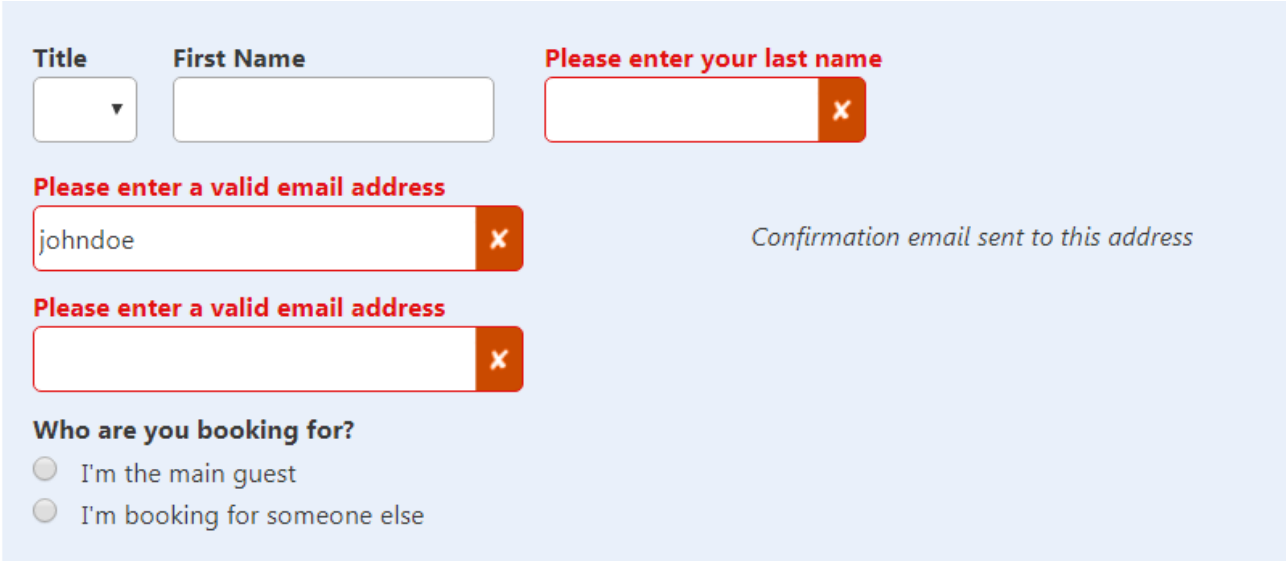
```
v = [1,4,3,2]                    # list
v = (1, 2, 3)                    # tuple
v = {'one':1, 'two':2, 'three':3}  # dictionary
v = {2, 3, 5, 7}                 # set
```

Regular expressions

- ***Regular expressions*** are expressions that are made to match a specific pattern in a piece of text. E.g.:
  - verify the authenticity of emails, phone numbers, and URLs

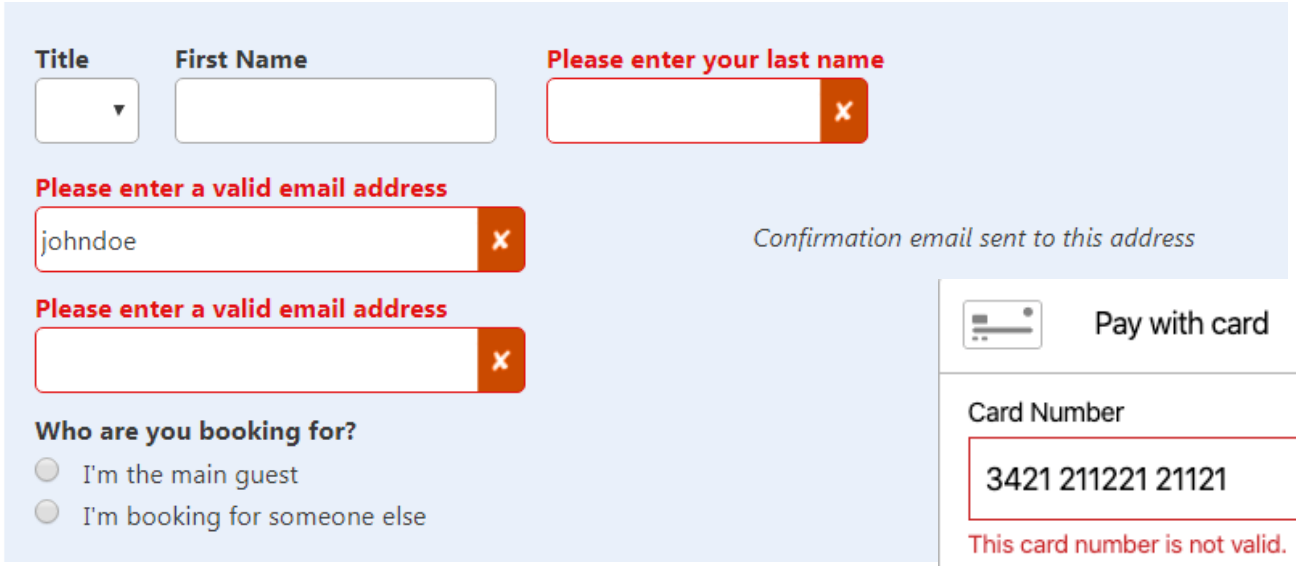
Regular expressions

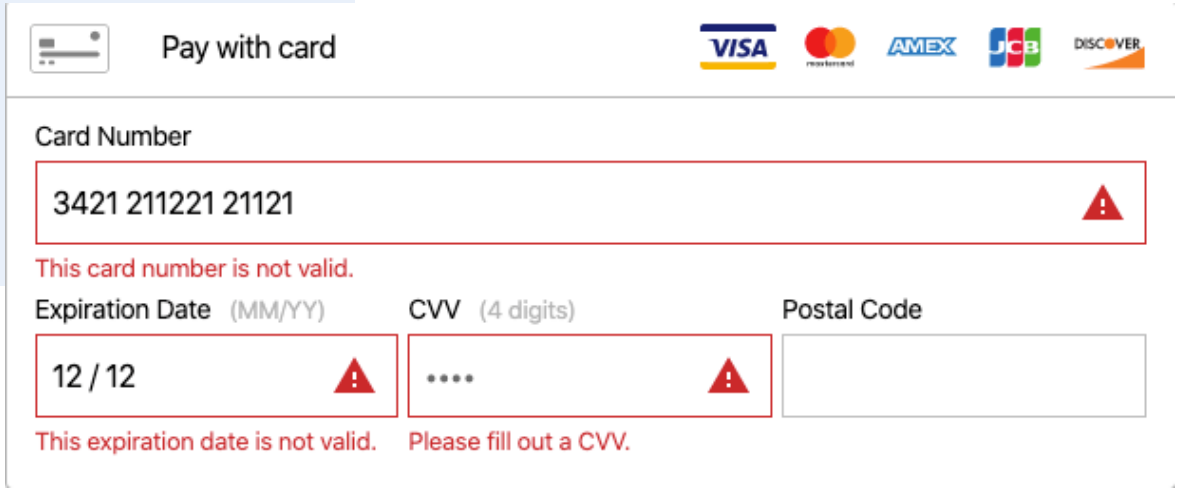
- ***Regular expressions*** are expressions that are made to match a specific pattern in a piece of text. E.g.:
  - verify the authenticity of emails, phone numbers, and URLs



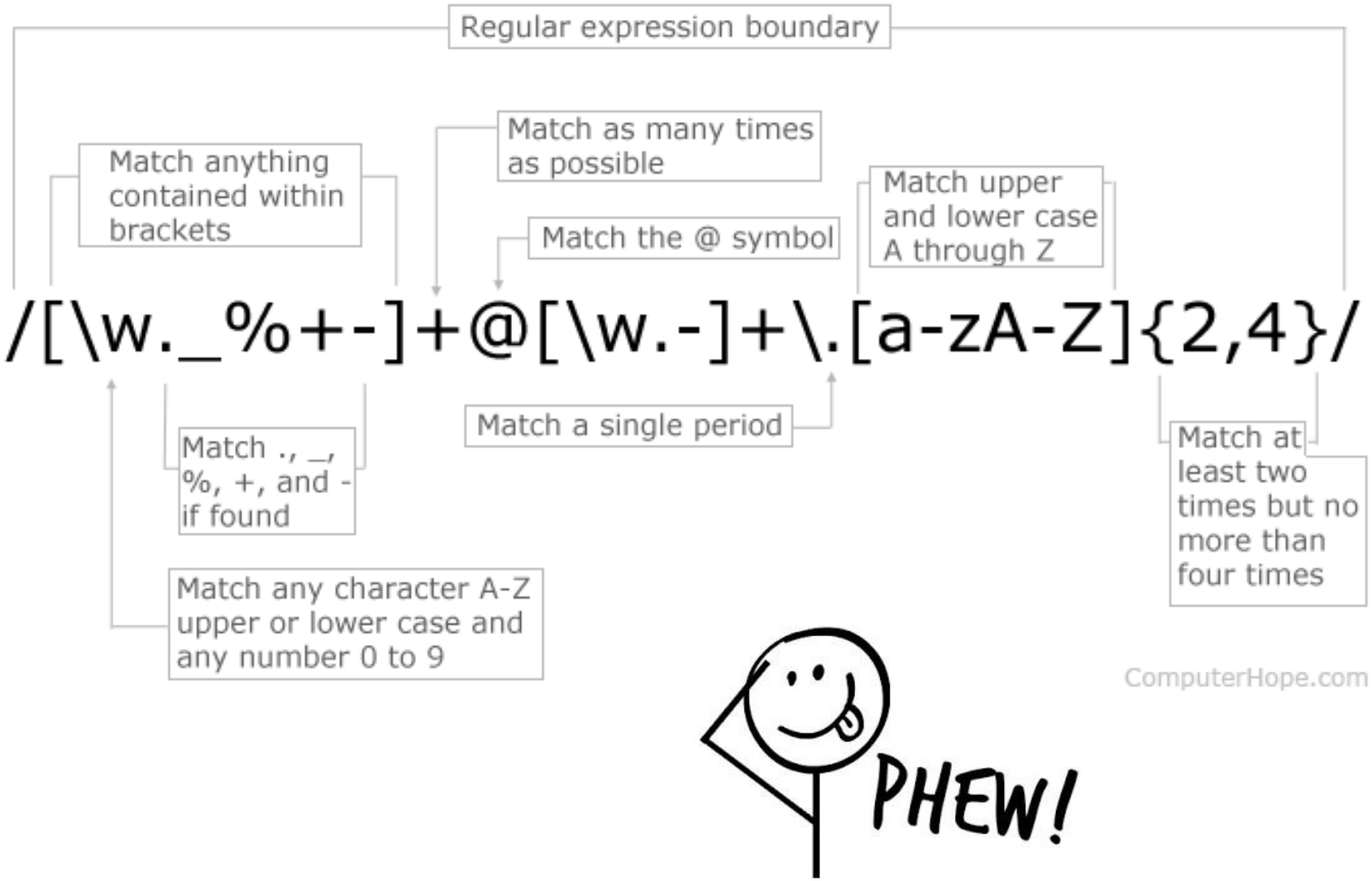
Regular expressions

- ***Regular expressions*** are expressions that are made to match a specific pattern in a piece of text. E.g.:
  - verify the authenticity of emails, phone numbers, and URLs





Email matching



Algorithms

- ***Algorithm*** is a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer
- ***Pseudocode*** is an informal high-level description of a program or an algorithm
- For example: Find the max value in the list = [1,4,3,2]
- Pseudocode:
    1. *Iterate over all elements in the list*
    2. *Check whether the previous maximum value is larger than the current list element*
        a. *If True store this value*

Find the max value in a list


```
list = [1,4,3,2]

max_value = list[0]
for n in list:
    print(n)
    if n > max_value:
        max_value = n

print(max_value)
```

```
Output:
1
4
3
2
4
```

Find the max value in a list


```
list = [1,4,3,2]

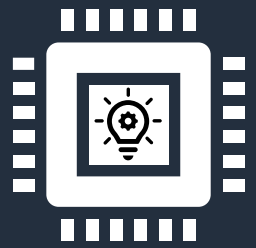
max_value = list[0]
for n in list:
    print(n)
    if n > max_value:
        max_value = n

print(max_value)
```

Built-in function:
max(list)

```
Output:
1
4
3
2
4
```

Let’s practice!



Tasks

- Basic Python

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/python-crash-couse/python-crash-course.ipynb

Machine Learning Techniques for Text

Section 4: The NumPy library


12mm


12mm

Python list

Python list

NumPy ndarray

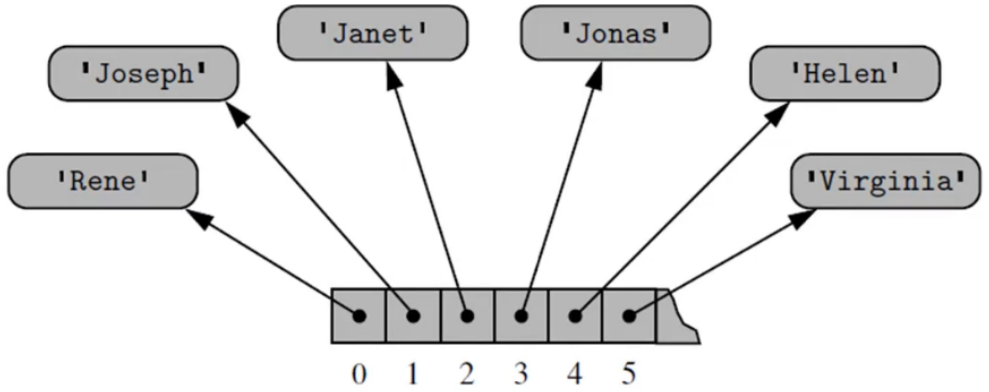
Python list

NumPy ndarray

pandas DataFrame

Arrays in data science

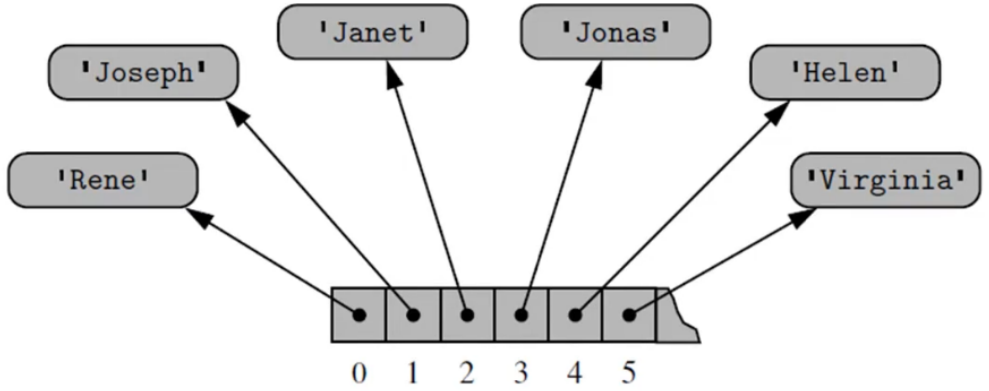
Arrays in data science



Arrays in data science

Images

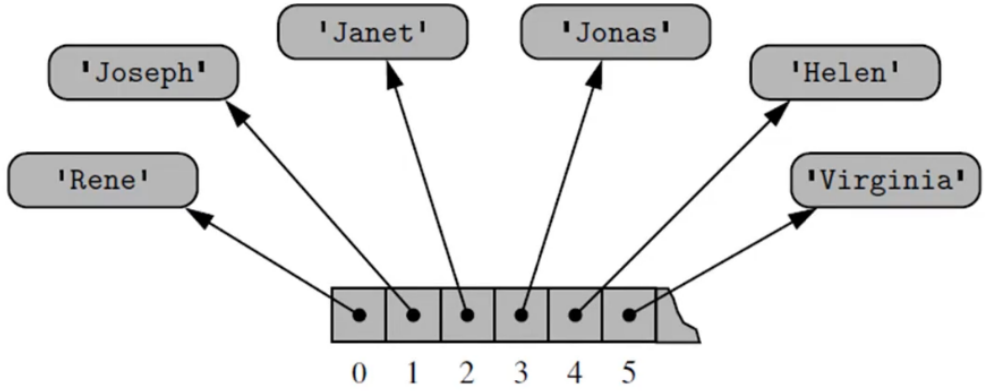
9	9	9	9	9	9	9	9	9	9
9	9	9	5	5	5	5	9	9	9
9	9	5	5	5	5	5	5	9	9
9	5	5	0	5	5	0	5	5	9
9	5	5	5	5	5	5	5	5	9
9	5	5	5	5	5	5	5	5	9
9	5	5	0	0	0	0	5	5	9
9	9	5	5	5	5	5	5	9	9
9	9	9	5	5	5	5	9	9	9
9	9	9	9	9	9	9	9	9	9



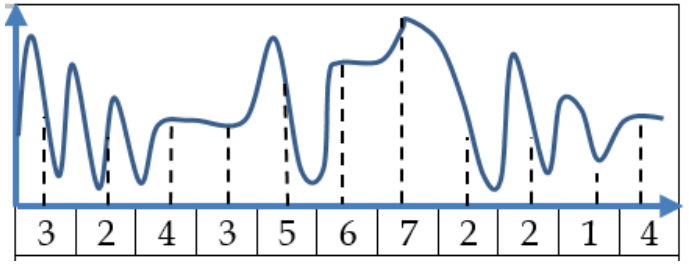
Arrays in data science

Images

9	9	9	9	9	9	9	9	9	9
9	9	9	5	5	5	5	9	9	9
9	9	5	5	5	5	5	5	9	9
9	5	5	0	5	5	0	5	5	9
9	5	5	5	5	5	5	5	5	9
9	5	5	5	5	5	5	5	5	9
9	5	5	0	0	0	0	5	5	9
9	9	5	5	5	5	5	5	9	9
9	9	9	5	5	5	5	9	9	9
9	9	9	9	9	9	9	9	9	9



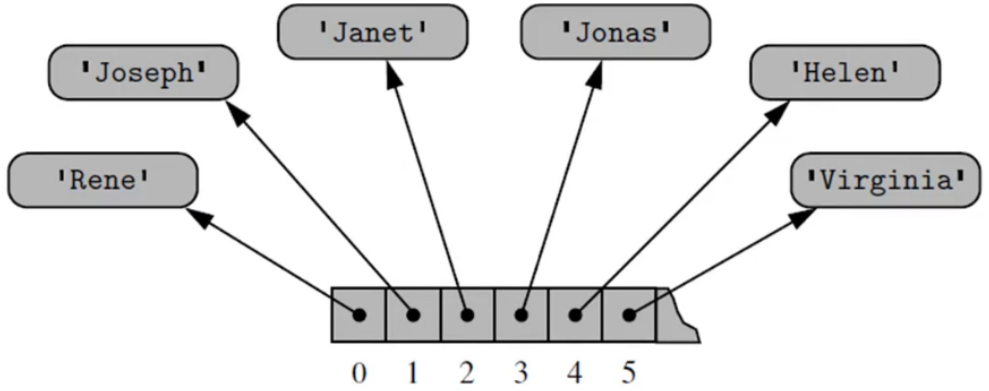
Audio



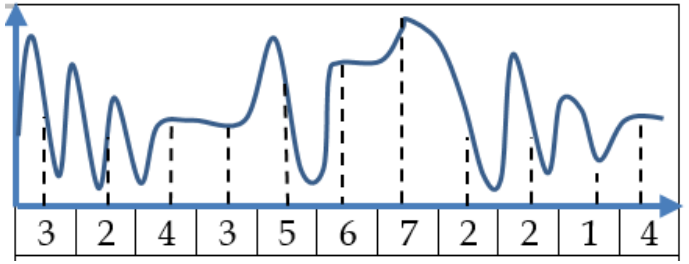
Arrays in data science

Images

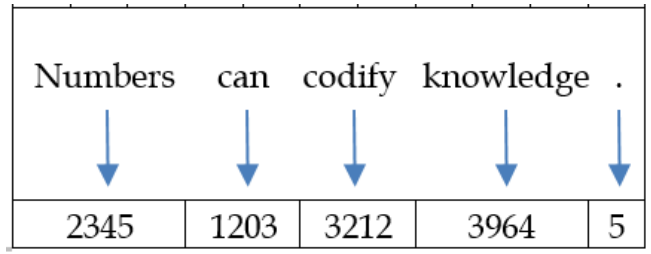
9	9	9	9	9	9	9	9	9	9
9	9	9	5	5	5	5	9	9	9
9	9	5	5	5	5	5	5	9	9
9	5	5	0	5	5	0	5	5	9
9	5	5	5	5	5	5	5	5	9
9	5	5	5	5	5	5	5	5	9
9	5	5	0	0	0	0	5	5	9
9	9	5	5	5	5	5	5	9	9
9	9	9	5	5	5	5	9	9	9
9	9	9	9	9	9	9	9	9	9



Audio



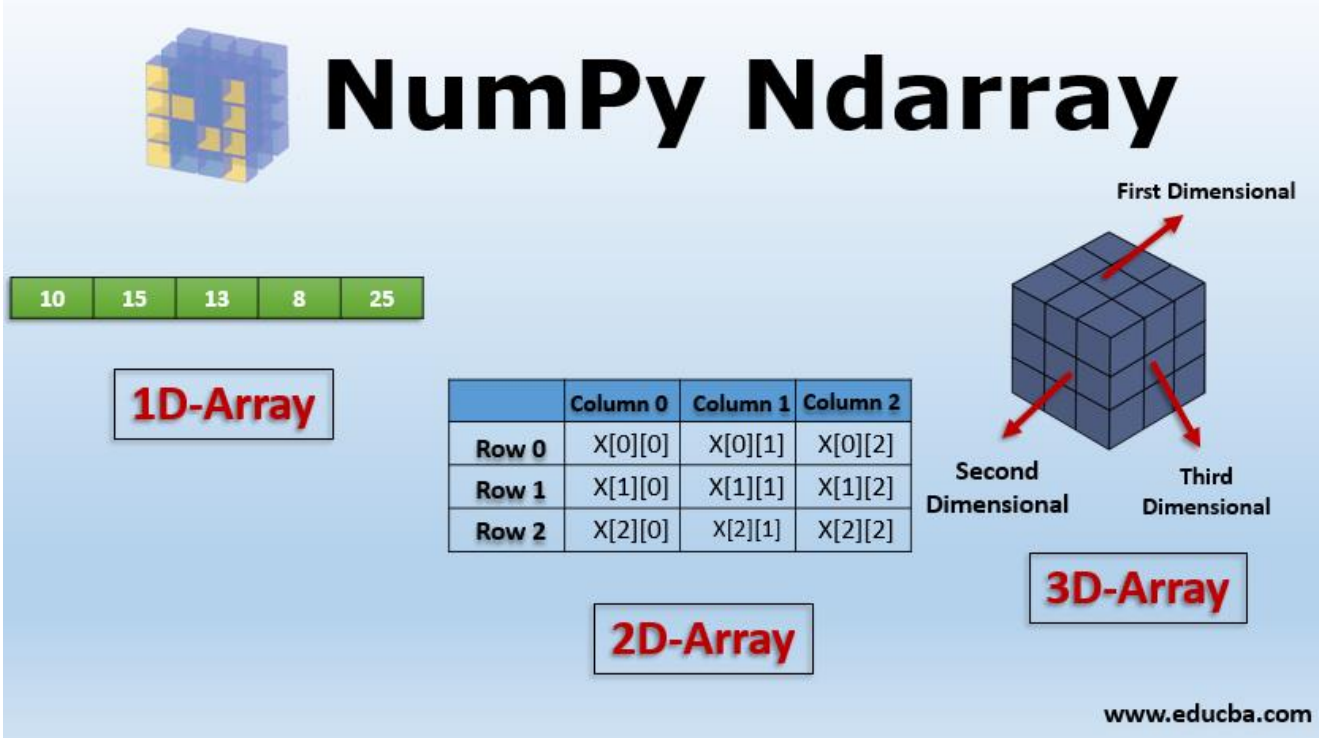
Text



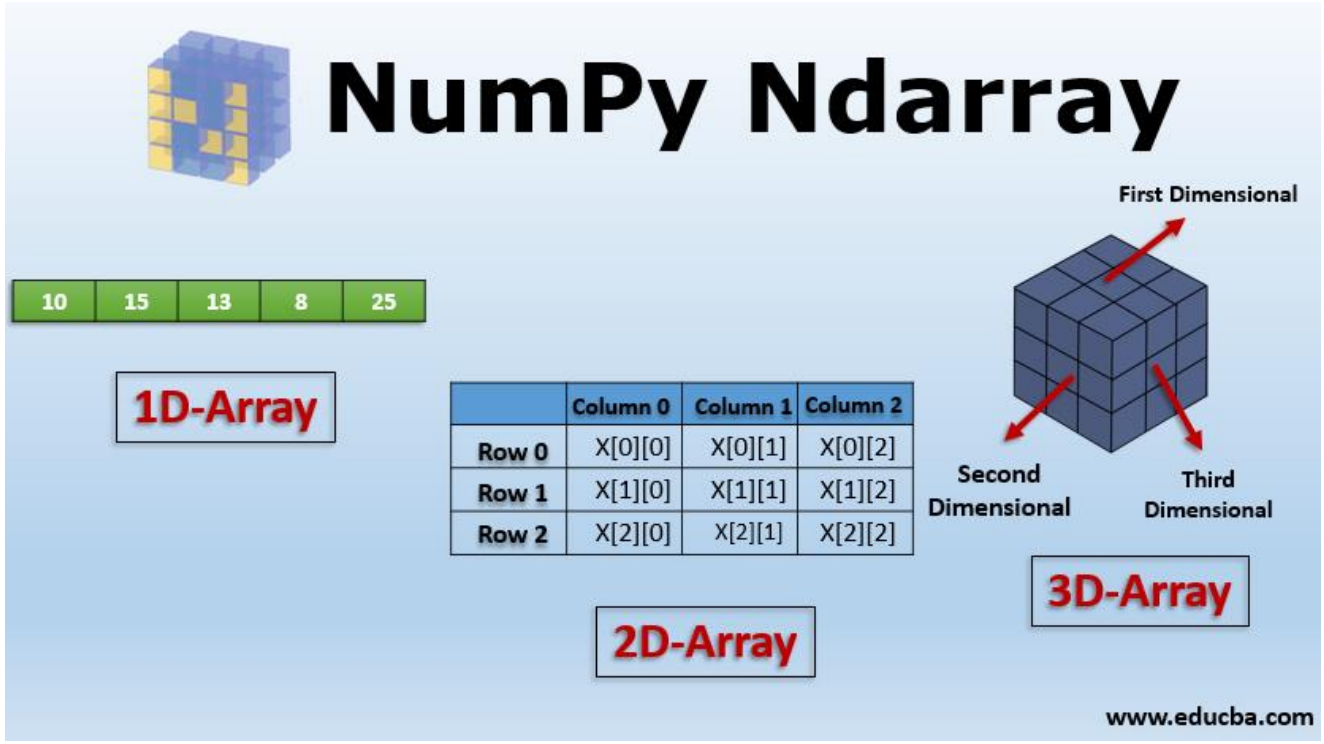
NumPy arrays

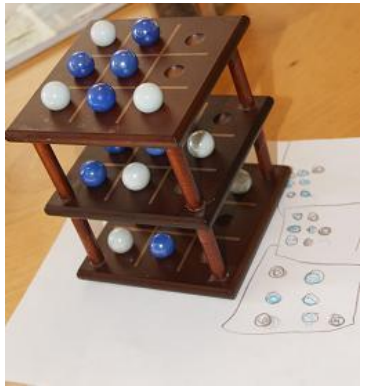
- Data manipulation in Python is nearly synonymous with NumPy array manipulation
- Even newer tools like Pandas are built around the NumPy array
- More efficient for storing and manipulating data
- Arrays are very frequently used in data science, where speed and resources are very important
- The array object in NumPy is called ***ndarray***
  - It provides a lot of supporting functions that make working with ndarray very easy

Array dimensions



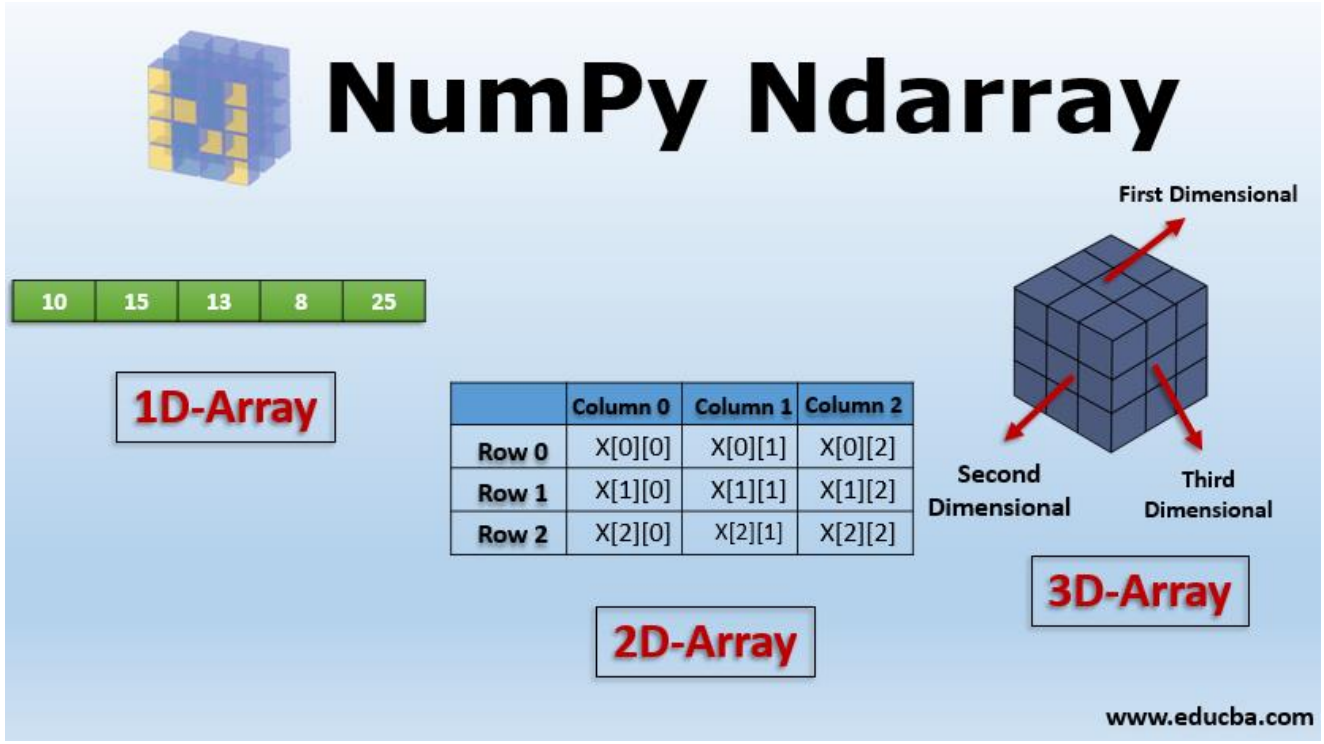
Array dimensions

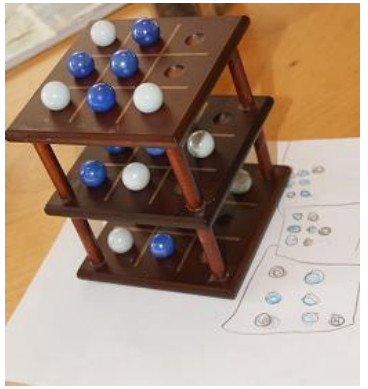




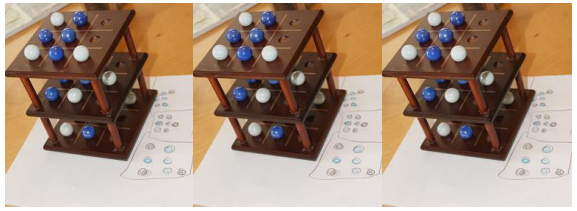
3D tic-tac-toe

Array dimensions





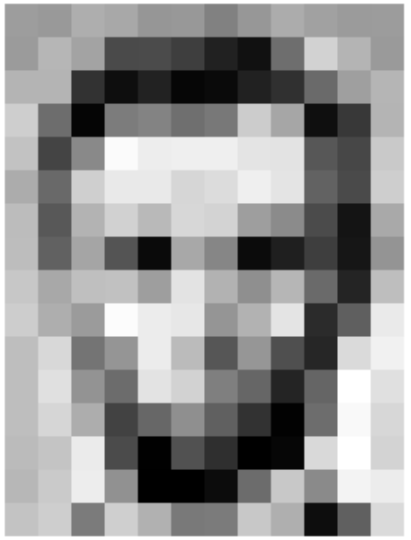
3D tic-tac-toe

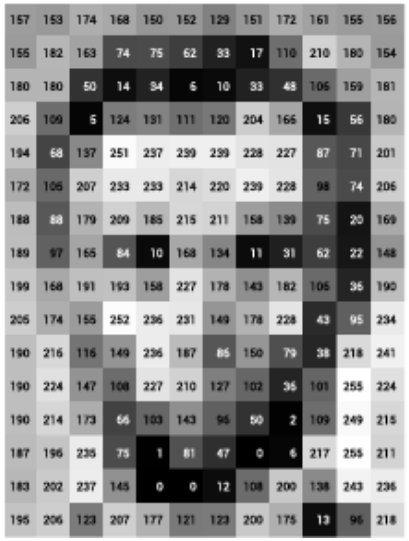


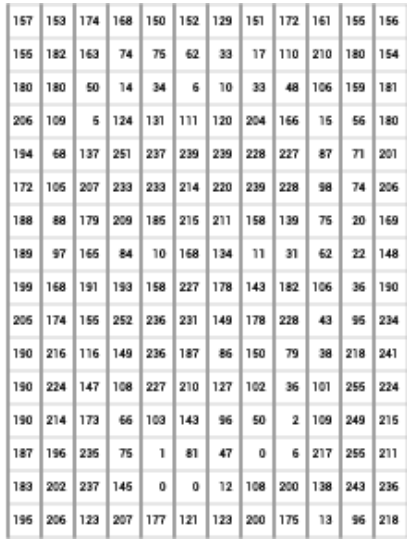
4D tic-tac-toe

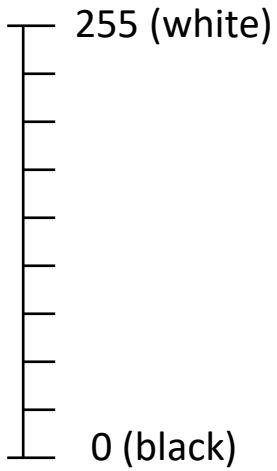
Numerical representation of B/W images

- Each element of the table represents only the quantity of the light
- The intensity values of the pixels vary from 0 (black) to 255 (white)
- All values are saved in the image file



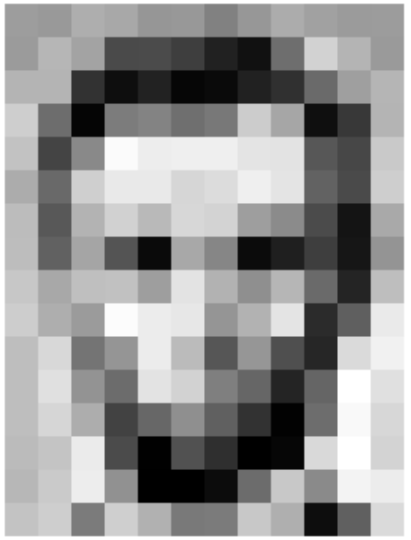


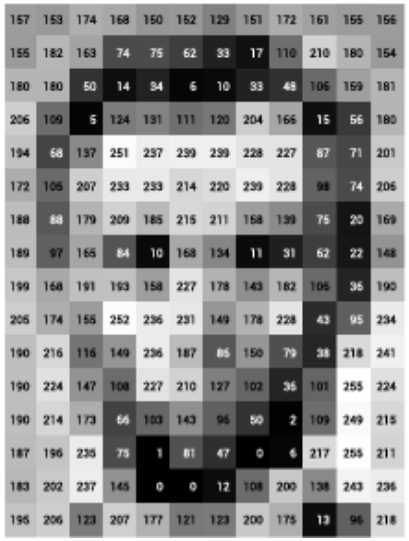


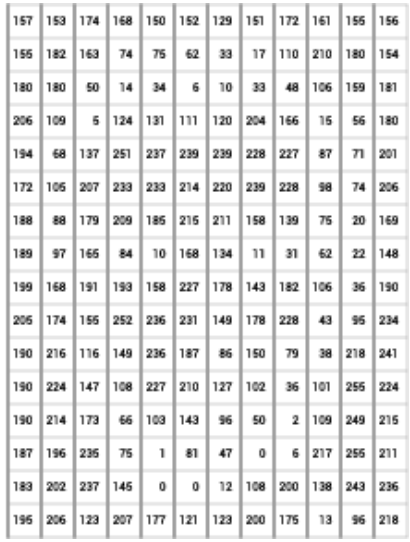


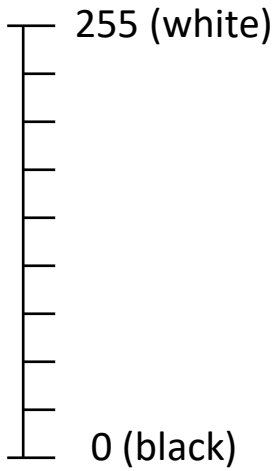
Numerical representation of B/W images

- Each element of the table represents only the quantity of the light
- The intensity values of the pixels vary from 0 (black) to 255 (white)
- All values are saved in the image file





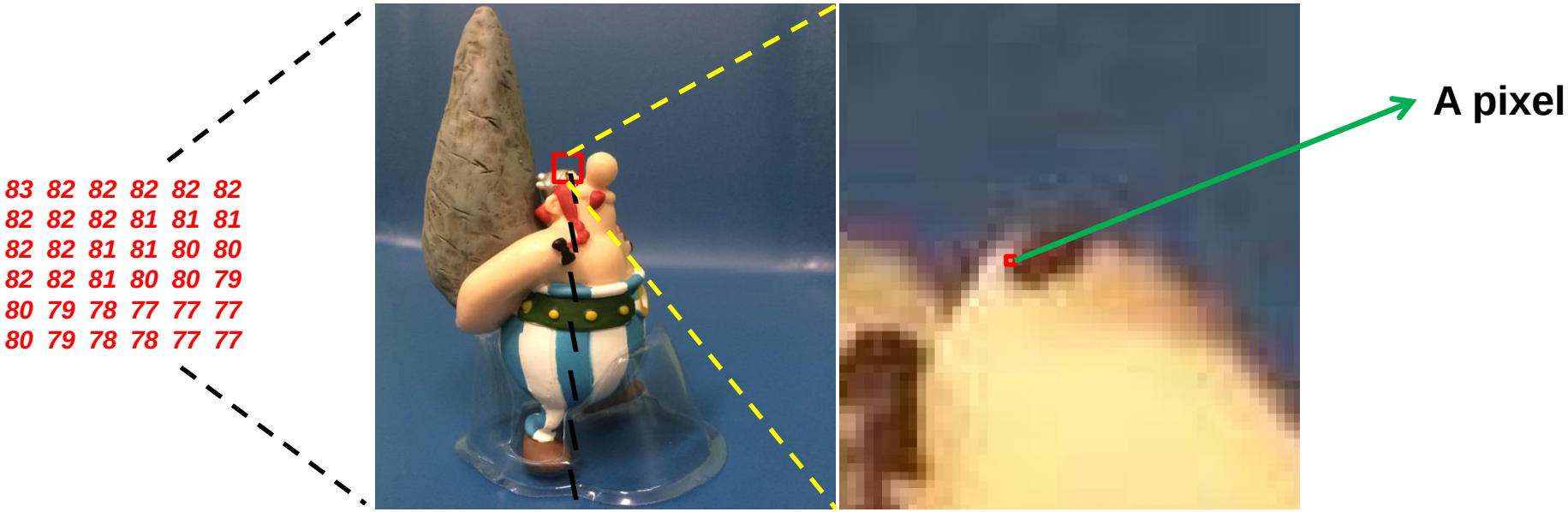




tip *In practice encoding algorithms are used to reduce the image size*

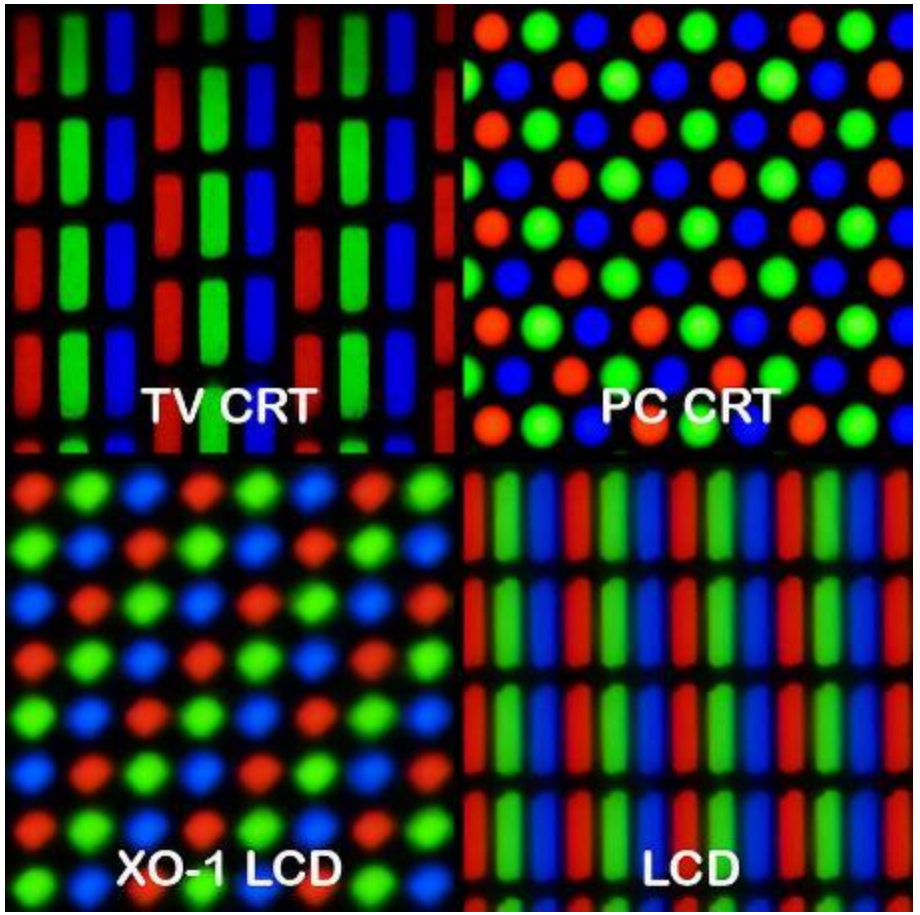
Display the image on the screen

How can we render the image table on the screen?



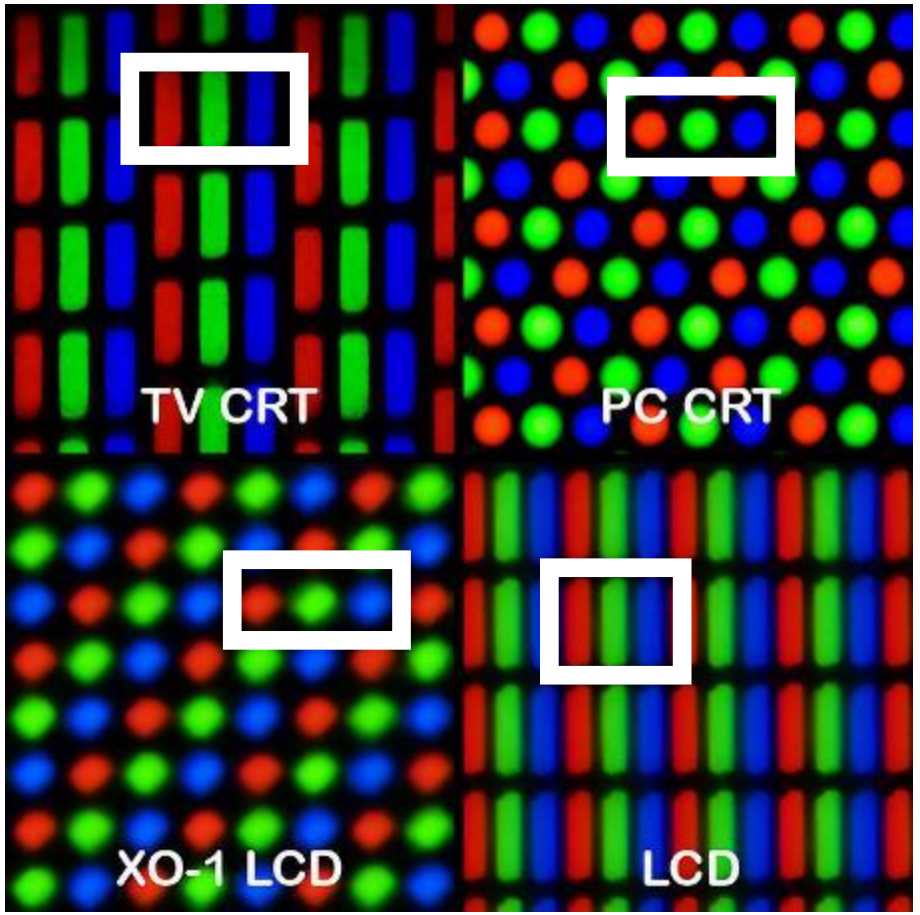
Pixels

- ***Pixels*** are the elements that make up each digital image. They are placed horizontally and vertically
- As they differ in size, each monitor has a different number of ***pixels per inch*** (PPP)



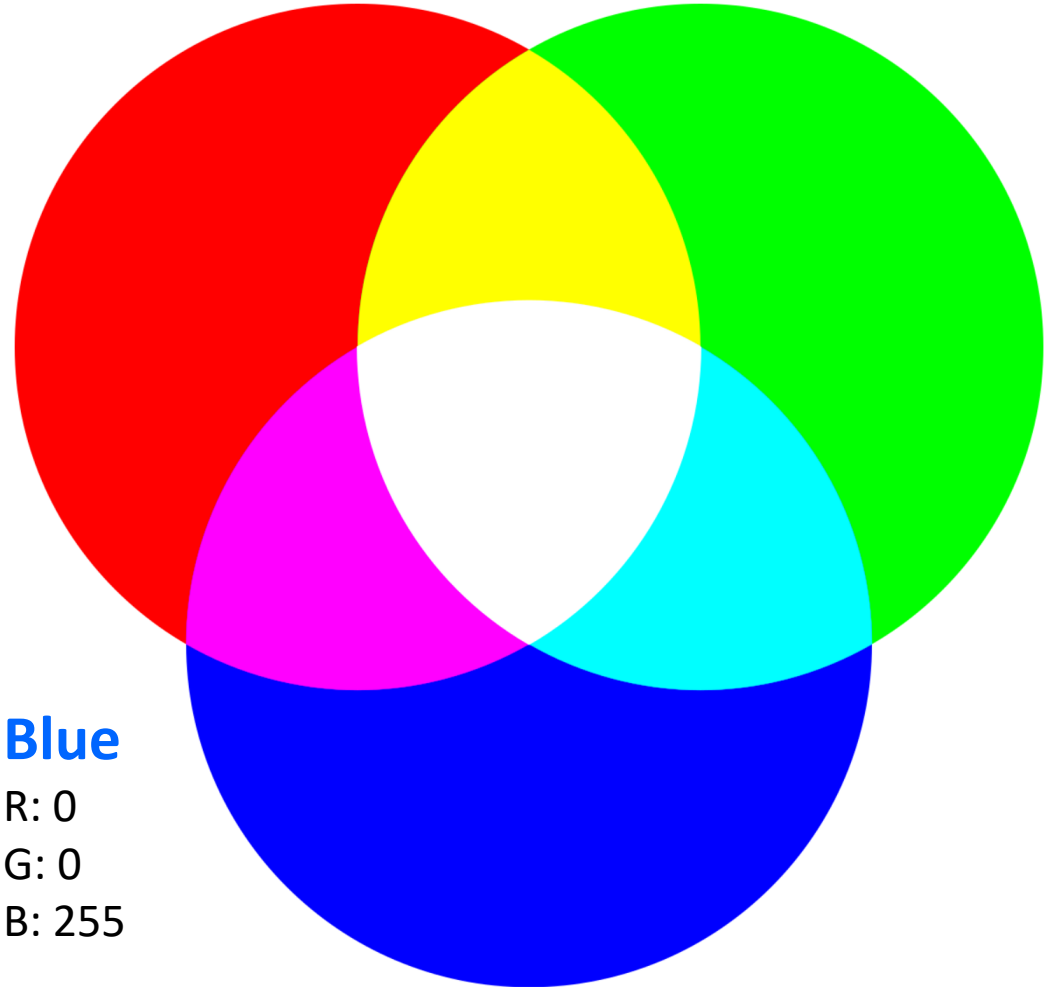
Pixels

- ***Pixels*** are the elements that make up each digital image. They are placed horizontally and vertically
- As they differ in size, each monitor has a different number of ***pixels per inch*** (PPP)



RGB color scheme

Red
R: 255
G: 0
B: 0

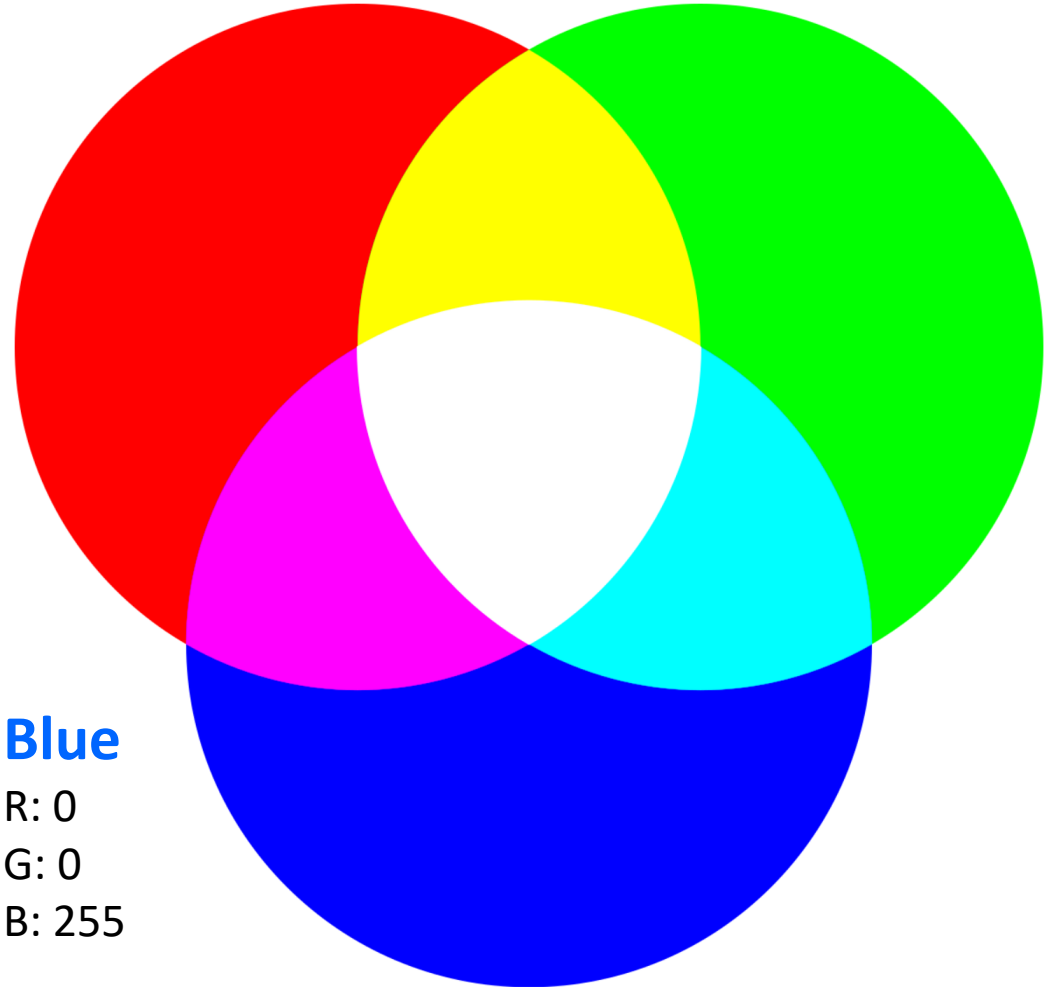


Green
R: 0
G: 255
B: 0

Blue
R: 0
G: 0
B: 255

RGB color scheme

Red
R: 255
G: 0
B: 0



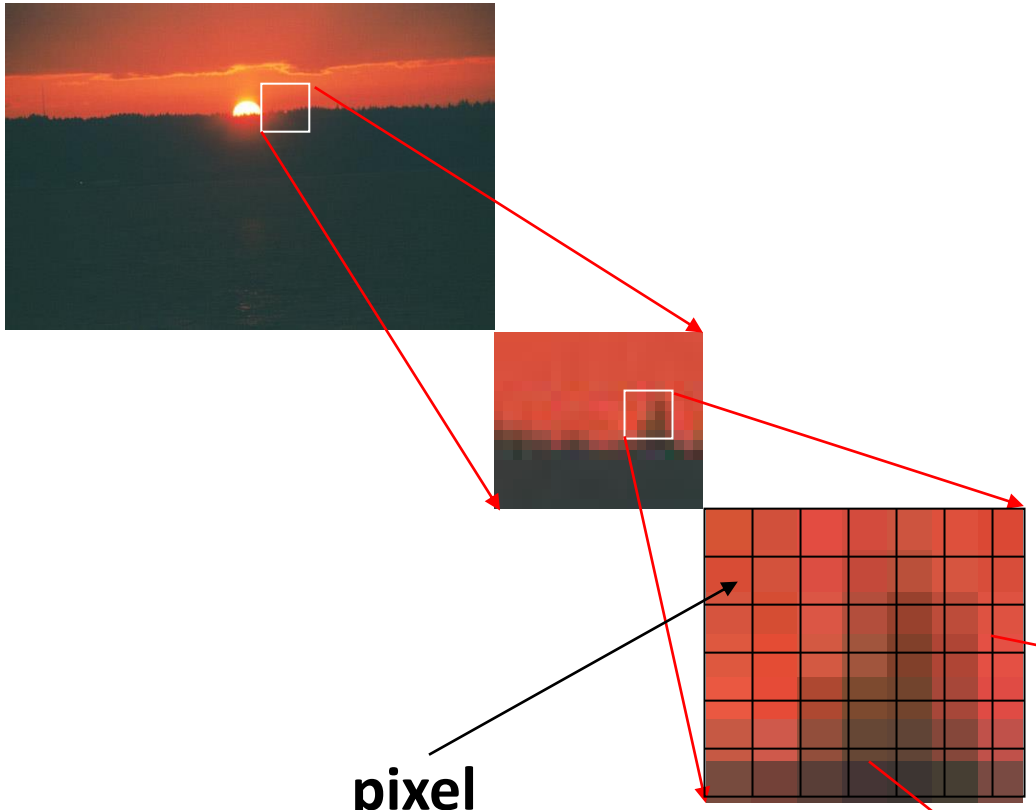
Blue
R: 0
G: 0
B: 255

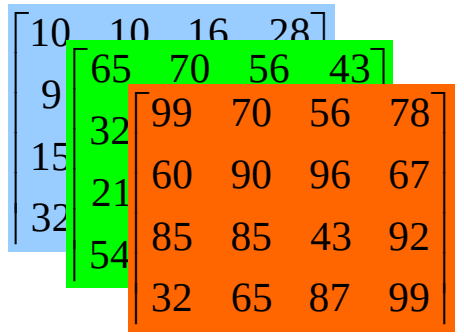
256 x 256 x 256 =16.8M colors

Green
R: 0
G: 255
B: 0

Numerical representation of images

Numerical image

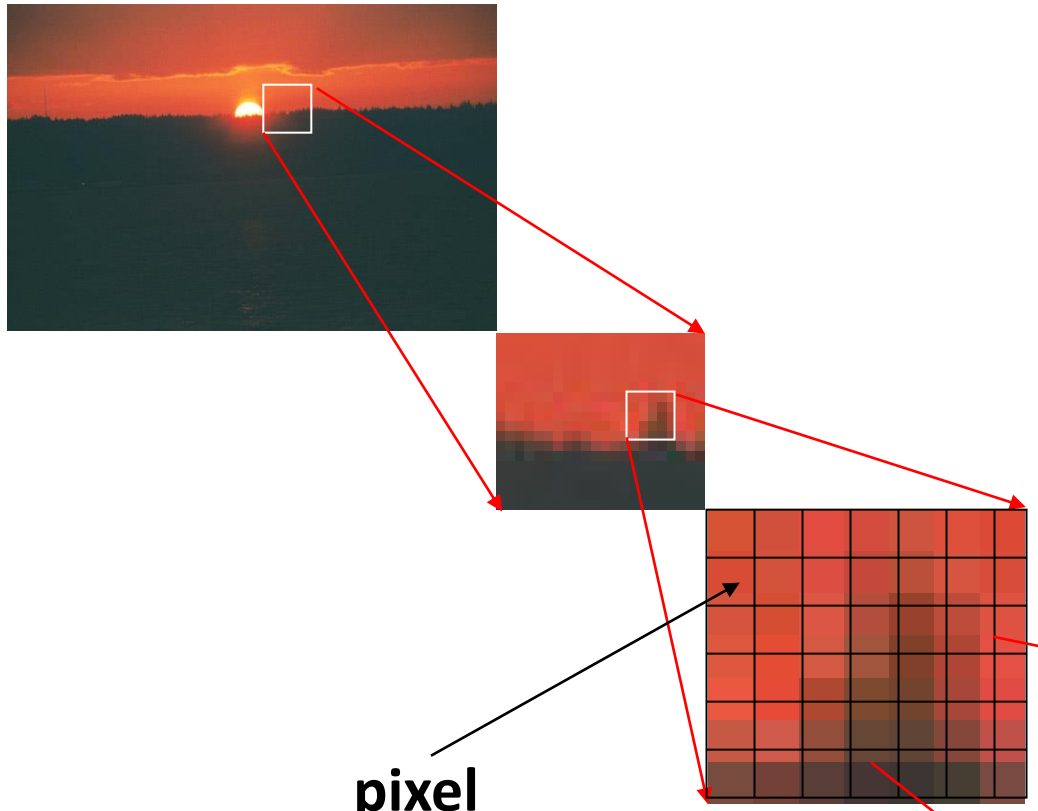




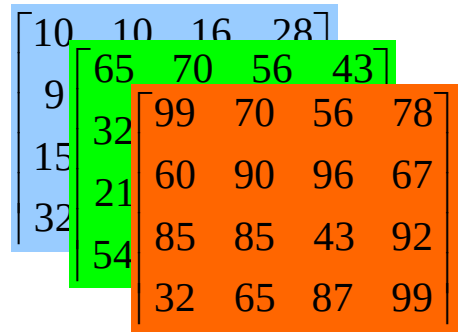
RGB colors

Numerical representation of images

Numerical image



A color image is composed of three tables for each RGB color



RGB colors

Images in Python

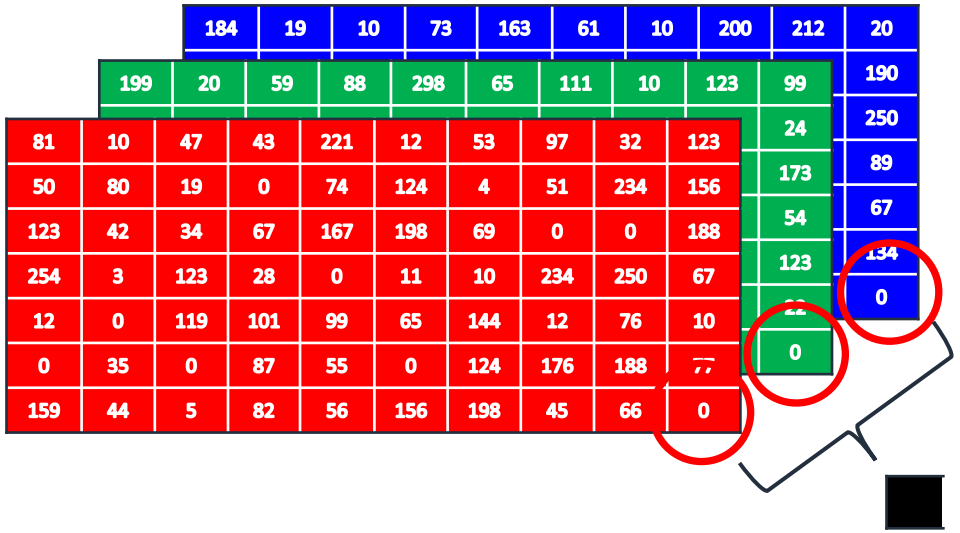
How can we blur the image?

Images in Python

How can we blur the image?

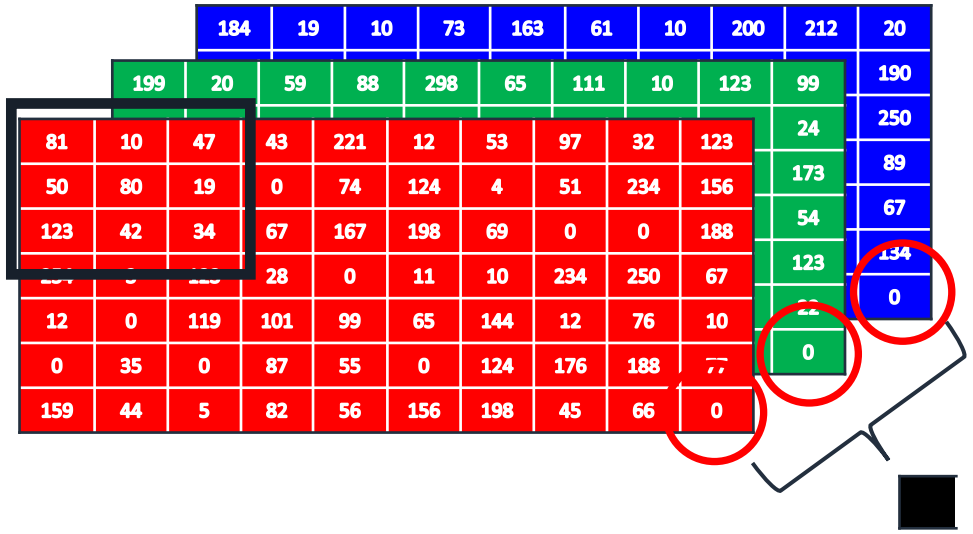
	184	19	10	73	163	61	10	200	212	20
199	20	59	88	298	65	111	10	123	99	190
									24	250
81	10	47	43	221	12	53	97	32	123	89
50	80	19	0	74	124	4	51	234	156	173
123	42	34	67	167	198	69	0	0	188	54
254	3	123	28	0	11	10	234	250	67	123
12	0	119	101	99	65	144	12	76	10	0
0	35	0	87	55	0	124	176	188	77	0
159	44	5	82	56	156	198	45	66	0	

Images in Python



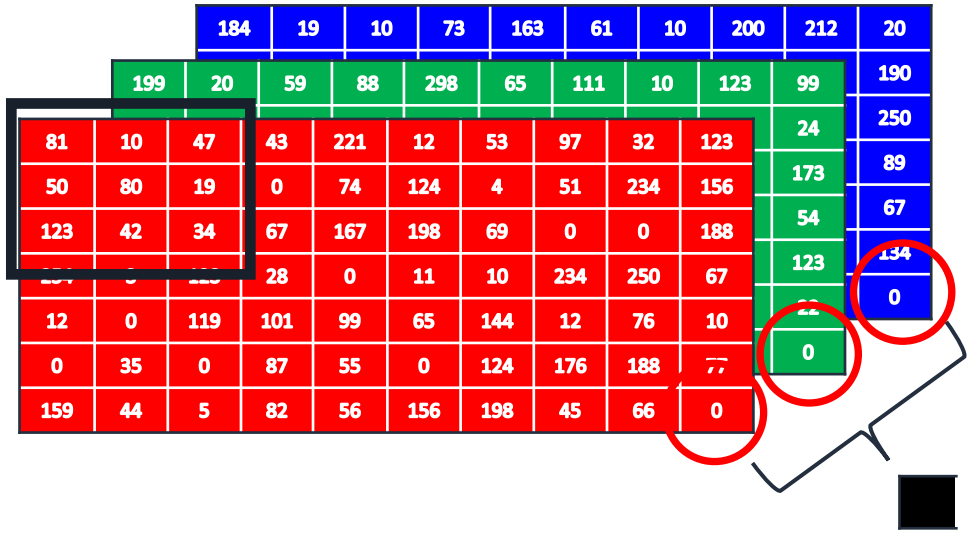
Replace the center pixel with the average of each neighborhood

Images in Python



Replace the center pixel with the average of each neighborhood

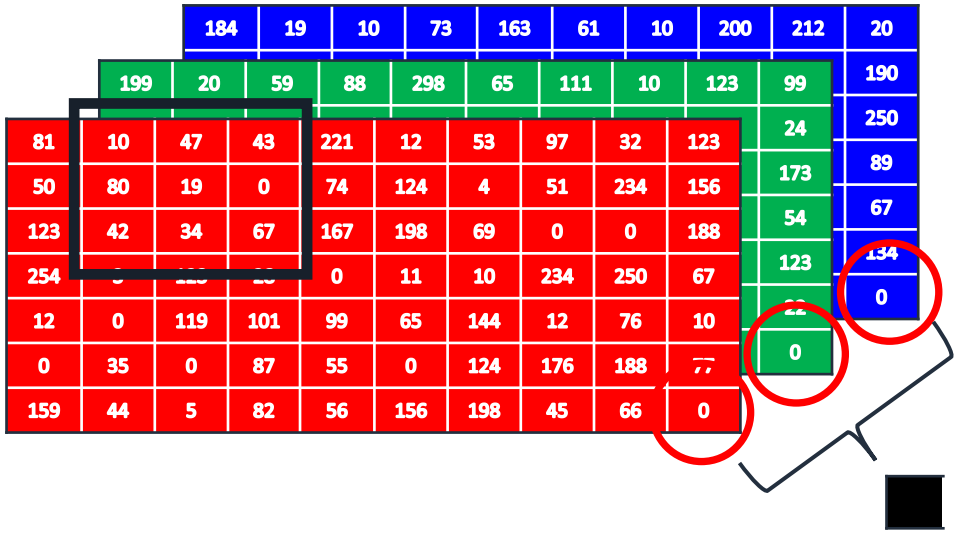
Images in Python



Replace the center pixel with the average of each neighborhood

(81+10+47+50+80+19+123+42+34)/9=54

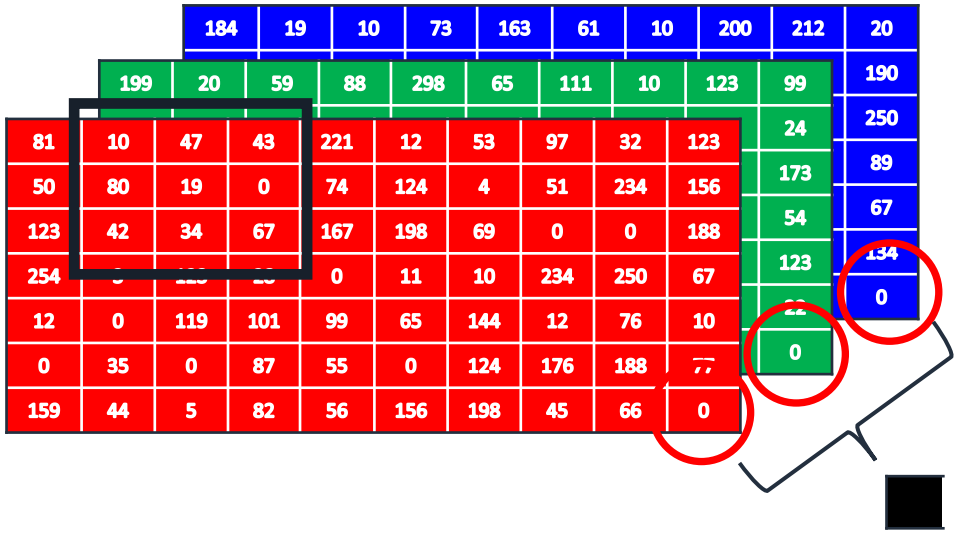
Images in Python



Replace the center pixel with the average of each neighborhood

(81+10+47+50+80+19+123+42+34)/9=54

Images in Python

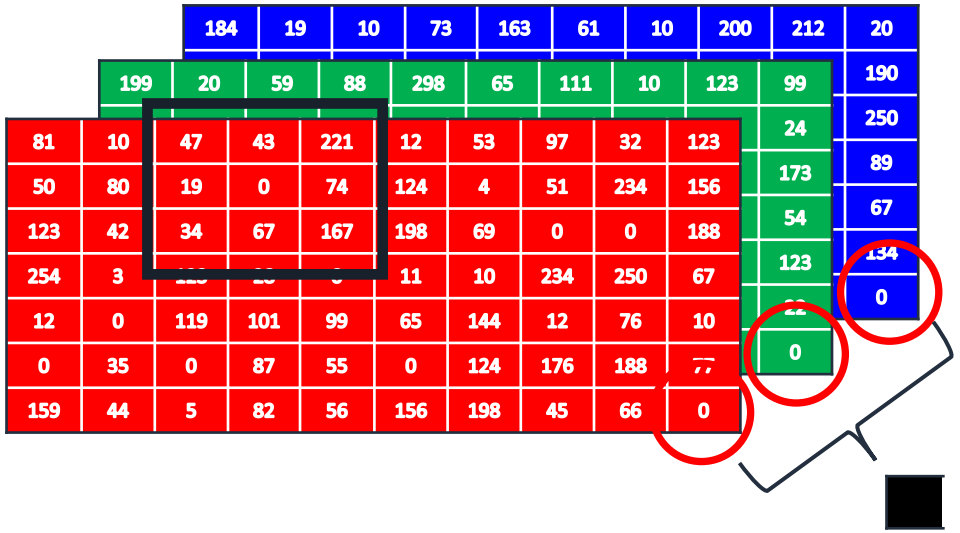


Replace the center pixel with the average of each neighborhood

(81+10+47+50+80+19+123+42+34)/9=54

(10+47+43+80+19+0+42+34+67)/9=38

Images in Python



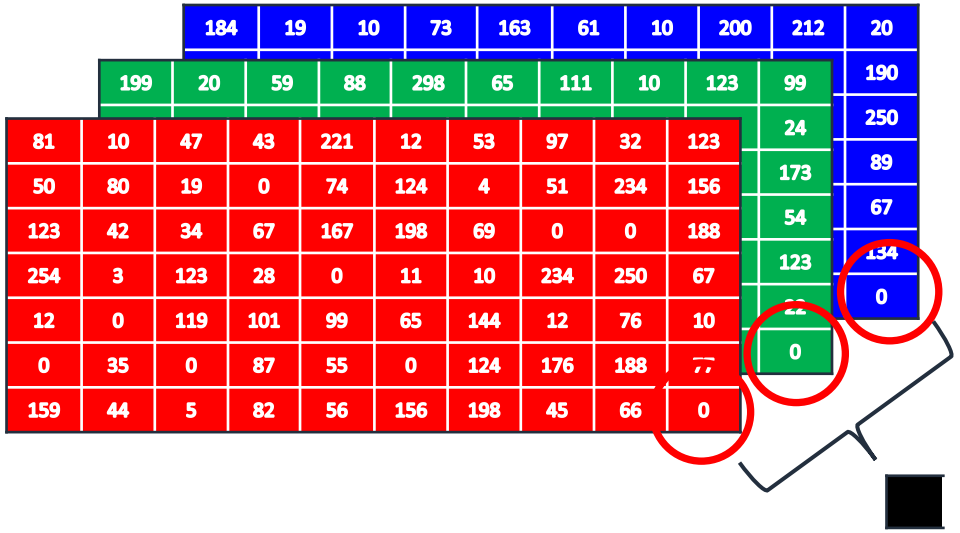
Replace the center pixel with the average of each neighborhood

(81+10+47+50+80+19+123+42+34)/9=54

(10+47+43+80+19+0+42+34+67)/9=38

…

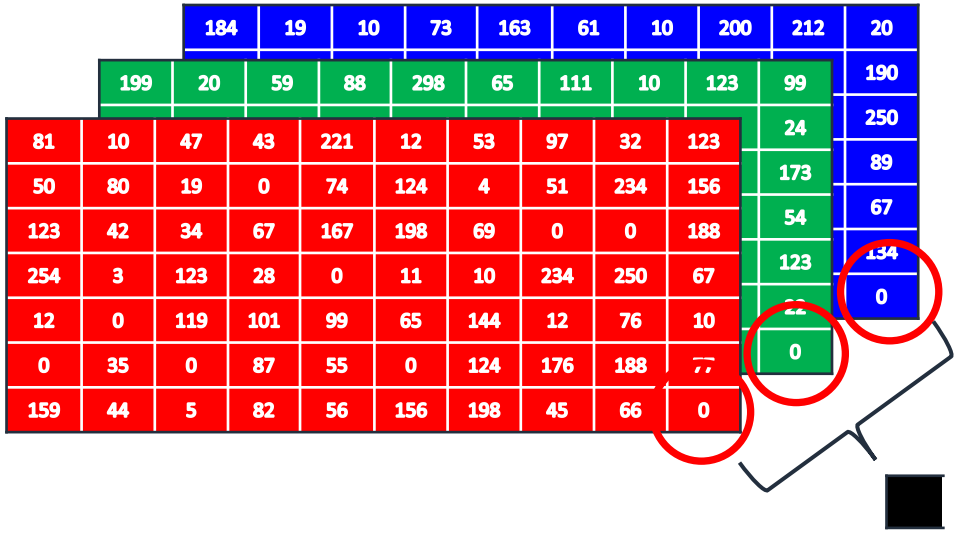
Images in Python



Replace the center pixel with the average of each neighborhood

Source: https://datacarpentry.org/image-processing/06-blurring.html

Images in Python

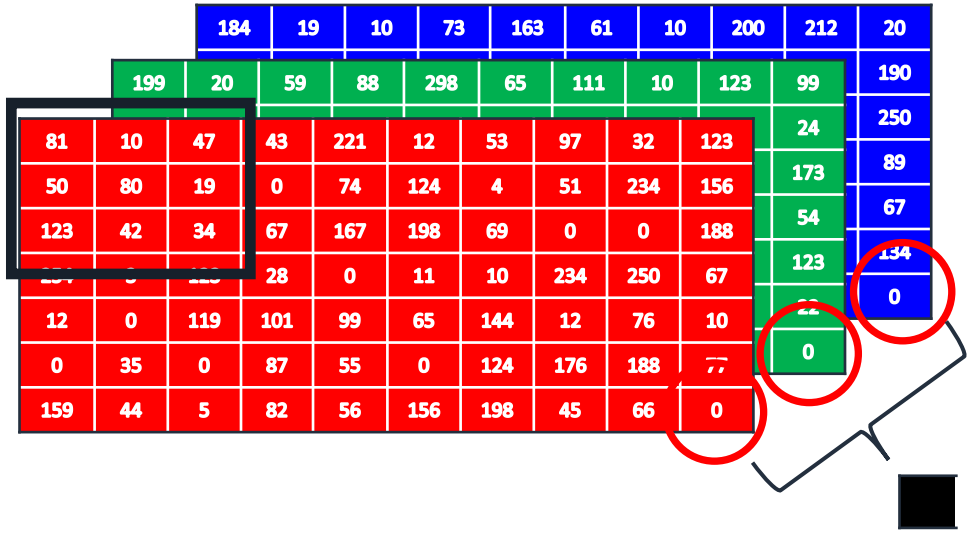


Replace the center pixel with the average of each neighborhood

tip *Every filter applies a numerical transformation to the image*

Source: https://datacarpentry.org/image-processing/06-blurring.html

Images in Python

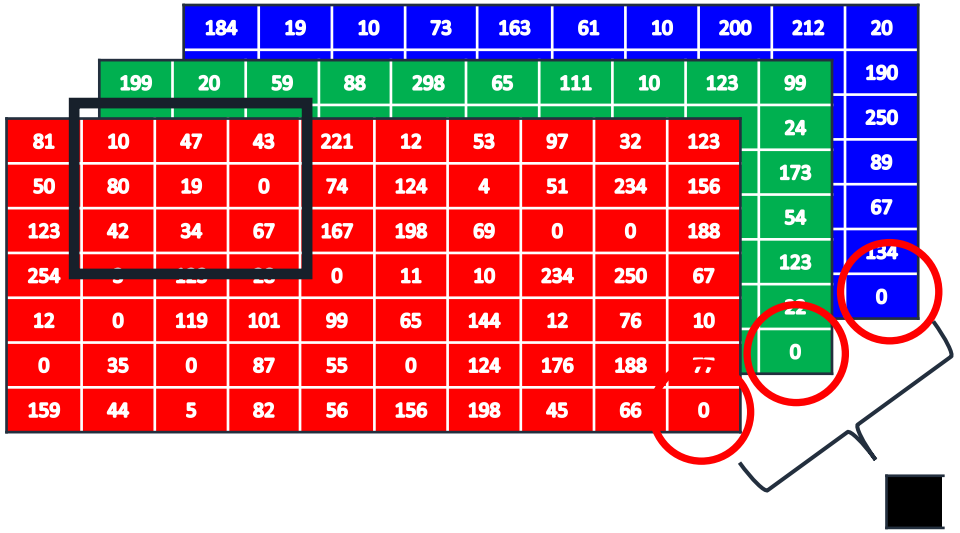


Replace the center pixel with the average of each neighborhood

tip *Every filter applies a numerical transformation to the image*

Source: https://datacarpentry.org/image-processing/06-blurring.html

Images in Python

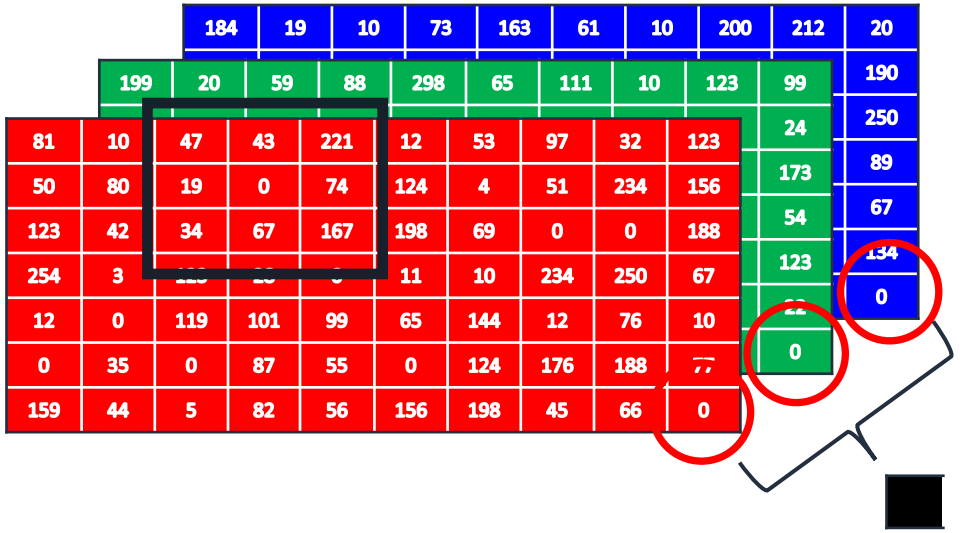


Replace the center pixel with the average of each neighborhood

tip *Every filter applies a numerical transformation to the image*

Source: https://datacarpentry.org/image-processing/06-blurring.html

Images in Python

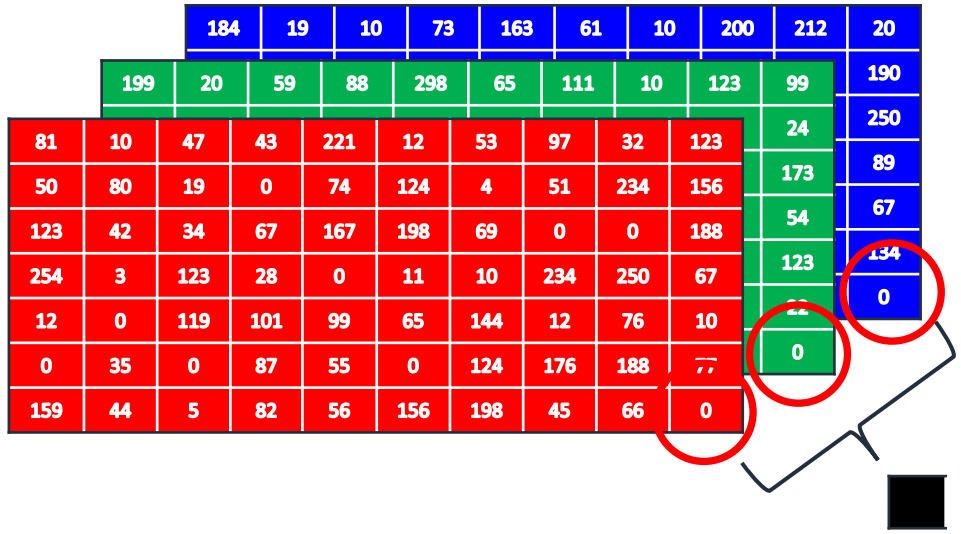


Replace the center pixel with the average of each neighborhood

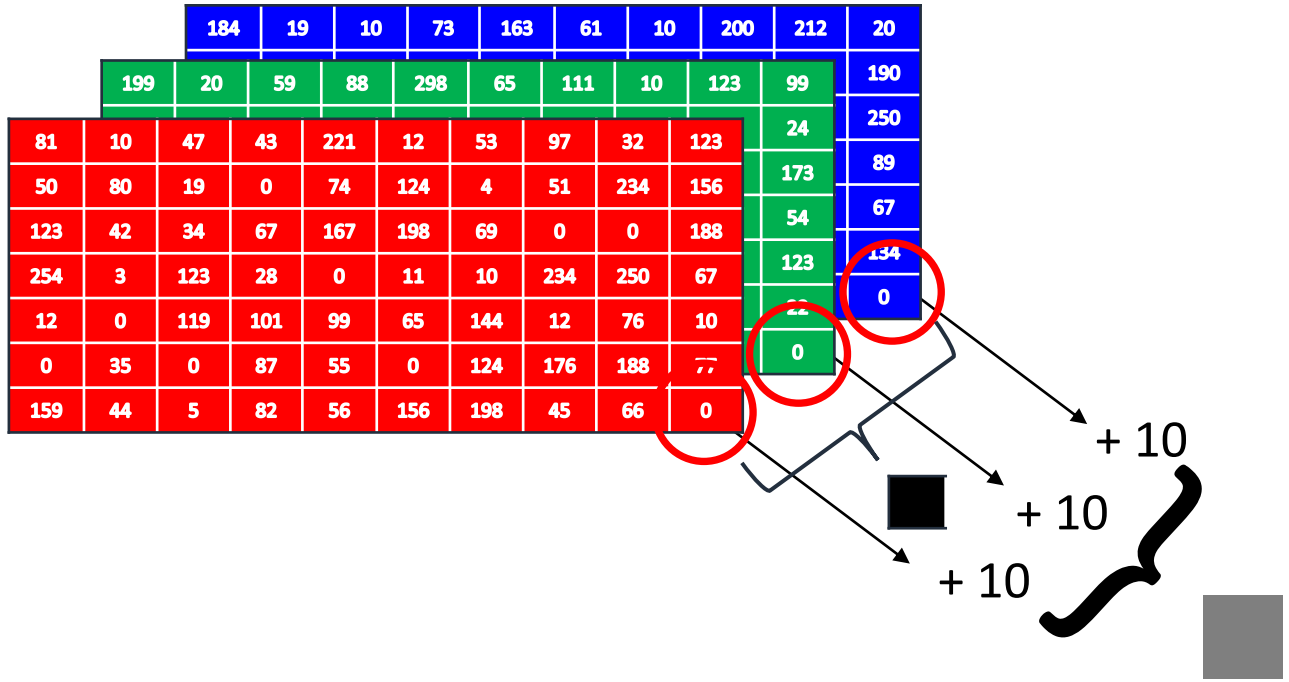
tip *Every filter applies a numerical transformation to the image*

Source: https://datacarpentry.org/image-processing/06-blurring.html

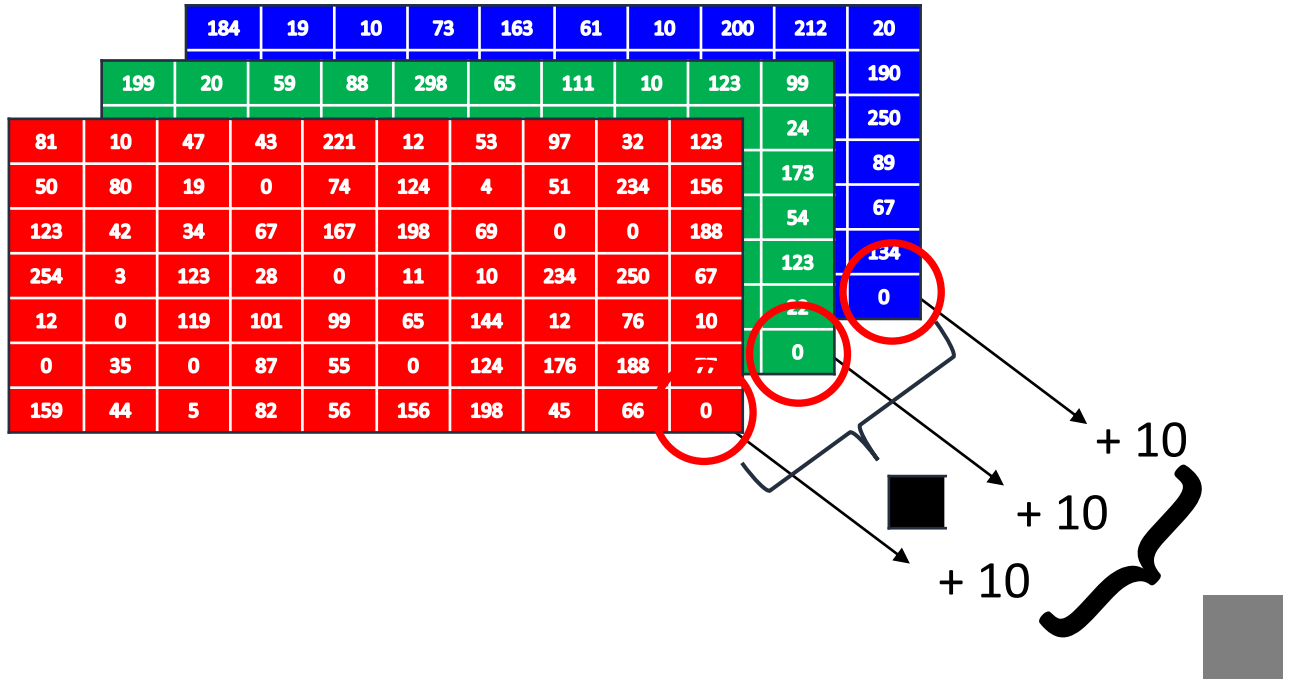
How can we make the image brighter?



How can we make the image brighter?

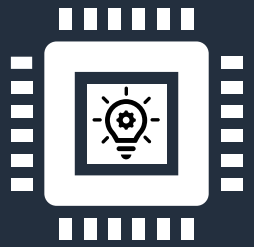


How can we make the image brighter?



Increase contrast?

Let’s practice!



Tasks

- Basic Python
- NumPy

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/python-crash-couse/numpy.ipynb

Machine Learning Techniques for Text

Section 5: The pandas library

Overview

- ***Pandas*** is a newer package built on top of NumPy
- Adds data structures and tools designed to work with table-like data
- Provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc.
- Provides an efficient implementation of a ***DataFrame***:
  - multidimensional arrays with attached row and column labels
  - often with heterogeneous types
  - and/or missing data
- Implements a number of powerful data operations familiar to users of both database frameworks and spreadsheet programs

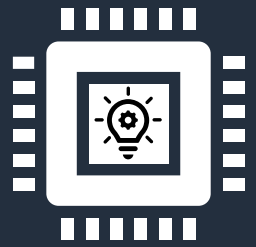
Pandas data structures

- ***Series***
  - A 1-dimensional labeled array
  - Supports many data types
  - Get and set values by index label
- ***DataFrame***
  - A 2-dimensional labeled data structure
  - A dictionary of Series objects
    - Columns can be of potentially different types
    - Get and set values by row and column label

Data ingestion

- Pandas supports data ingestion from various sources, including CSV files, Excel files, SQL databases, JSON, HTML, and more
  - Use ***pd.read_csv()*** to read data from a CSV (Comma-Separated Values) file
  - Use ***pd.read_excel()*** to read data from an Excel file
  - Use ***pd.read_sql()*** to read data from a SQL database query
  - Use ***pd.read_json()*** to read data from a JSON file or a JSON string
  - Use ***pd.read_html()*** to perform web scraping and read HTML tables

Let’s practice!



Tasks

- Basic Python
- NumPy
- pandas

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/python-crash-couse/pandas.ipynb

Machine Learning Techniques for Text

Questions?