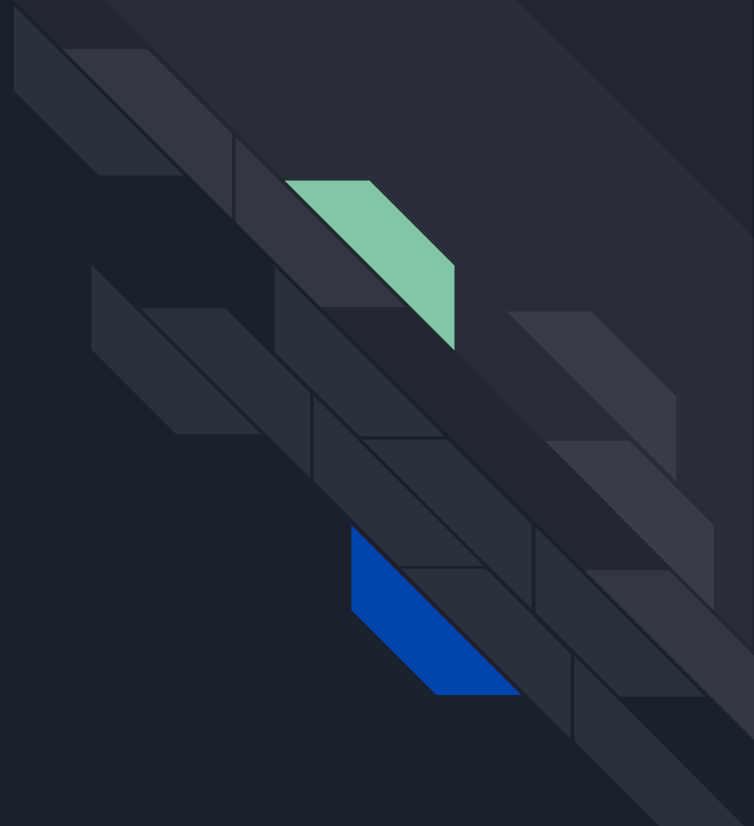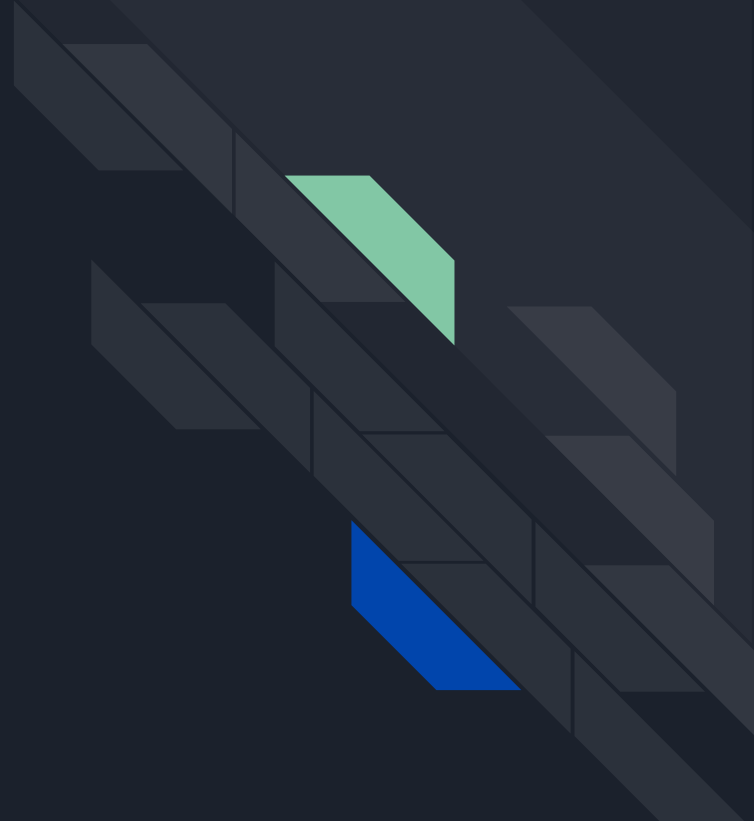# Introduction to Data Science with Python

Chapter 1

Fundamentals
Primitive Data Types

How is data stored and processed ?

- Values are stored in **variables**
- The four most important data types in Python:

```python
integer = 10

float = 2.8

string = "This is a string"

boolean = True
```

How is data stored and processed ?

- We can compute with these variables

```
a = 10
b = 5
c = a + b

print(c)
```

Output: 15

What kind of data type is this : `"27-03-2021"` ?
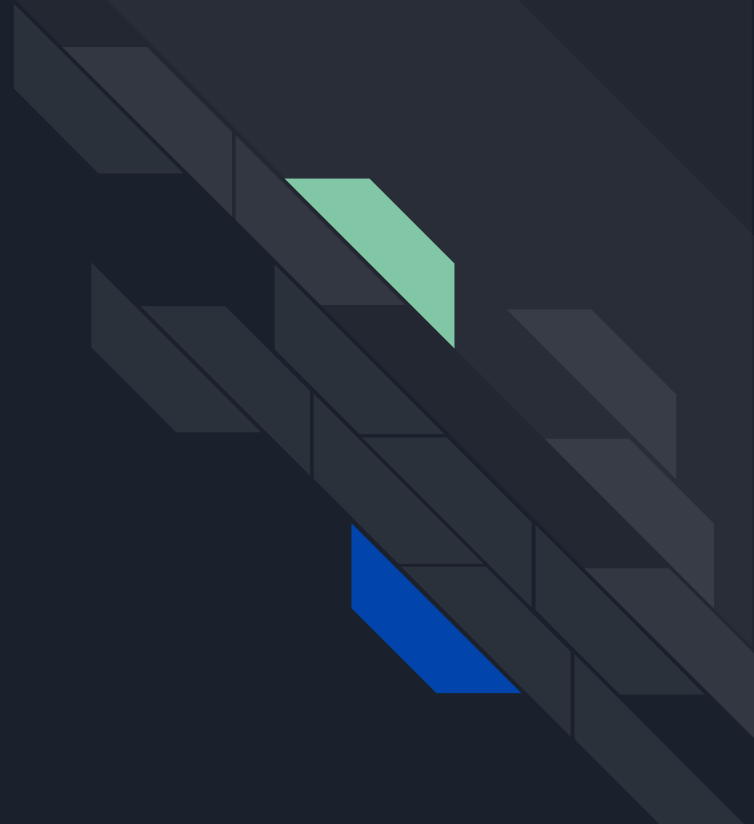
a) integer   b) float   c) string   d) date

QUIZ

What kind of data type is this : `"27-03-2021"` ?

a) integer   b) float   **c) string**   d) date

Fundamentals
Data Structures

Data Structures - Lists

We can combine values in lists

```python
a = [5, 3, 9, 7, 4, 10, 3]
b = ["Justus", "Peter", "Bob"]
```

Data Structures - Lists

Value	5	3	9	7	4	10	3
Index	0	1	2	3	4	5	6

Data Structures - Lists

Access the data with an index

```python
a = [5, 3, 9, 7, 4, 10, 3]
b = ["Justus", "Peter", "Bob"]

print(a[0])  # Output: 5
```

Data Structures - Lists

Access the data with an index

```python
a = [5, 3, 9, 7, 4, 10, 3]
b = ["Justus", "Peter", "Bob"]

a[3]   # Output: 7
b[1]   # Output: Peter
```

Data Structures - Lists

```
a[start:stop:step_size]
```

Data Structures - Lists

Value	5	3	9	7	4	10	3
Index	0	1	2	3	4	5	6

a[1 : 4]

Data Structures - Lists

Value	5	3	9	7	4	10	3
Index	0	1	2	3	4	5	6

a[1:4:2]

Data Structures - Dictionaries

```python
translate = {"Eins":"One",
             "Zwei":"Two",
             "Ja":"Yes"}
translate["Eins"]  # Output: "One"
```

Quick - Summary

Data types

integer   2

float   2.32

string   "Text"

boolean   True/False

Data structures

lists: a = [1,2,3]

dictionary: b={"a":1}

Exercise 1

```
Data structures - Hints
lists:
    create:  a = [1,2,3]
    access:     a[0]
dictionary:
    create: b={"a":1}
    access: b["a"]
```

Relational operators

- Compare variables

    a == b → is a equal to b?

    returns **True / False**

Relational operators

- Compare variables

  a == b   →   is a equal to b?

  returns **True / False**

==	is equal
<	smaller than
>	greater than
<=	smaller or equal than
>=	greater or equal than
!=	not equal to

Relational operators

- Compare variables

  a == b → is a equal to b?

  returns **True / False**

- Combine operators with "**and**" / "**or**"

  "and":   (a >= b) and (a<=c)

  "or":   (a >= b) or (a<=c)

==	is equal
<	smaller than
>	greater than
<=	smaller or equal than
>=	greater or equal than
!=	not equal to

QUIZ

```python
a = 1
b = 2
c = 2

(a>b) or (a<=c)
```

QUIZ

```python
a = 1
b = 2
c = 2
        (a>b) or (a<=c)
        False or True
```

QUIZ

```python
a = 1
b = 2
c = 2

      (a>b) or (a<=c)

      False or True →  True
```
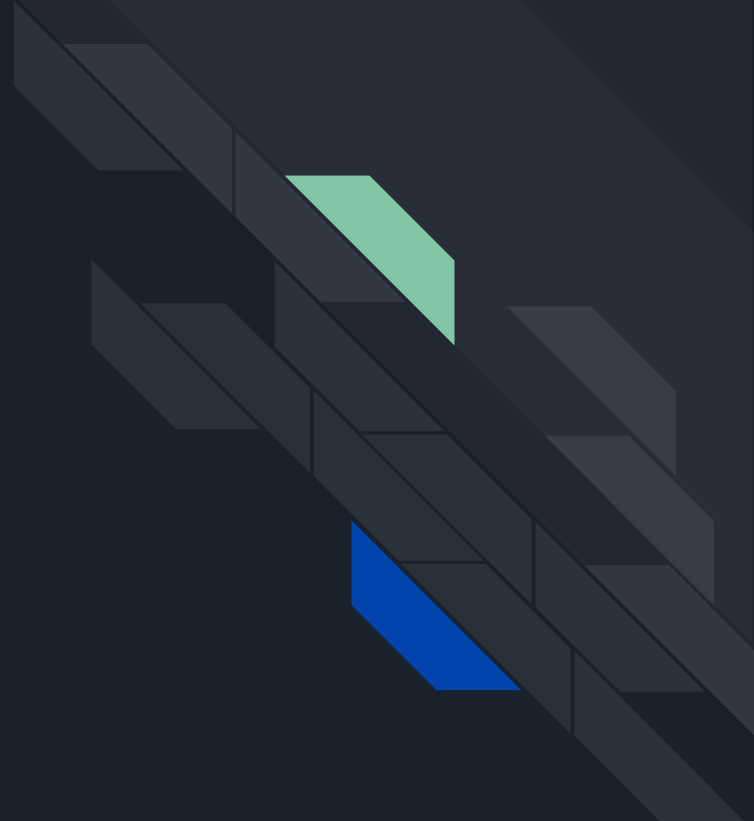
Very important for filtering

Name	Gender	Age
"Tim"	"M"	20
"Nina"	"F"	24
"John"	"M"	26

Select all Names with following condition:

(Gender == "F") **&** (Age > 20)

Fundamentals
Control Flow
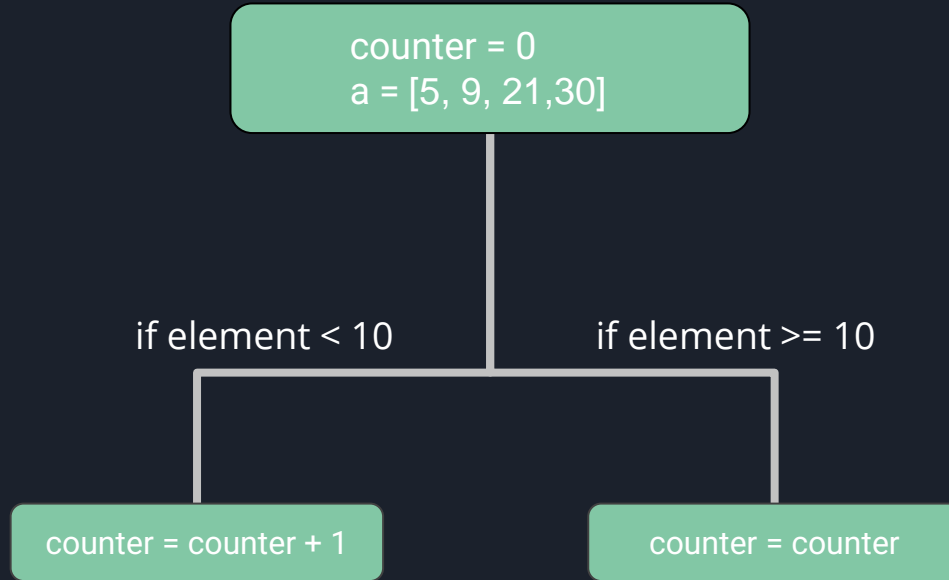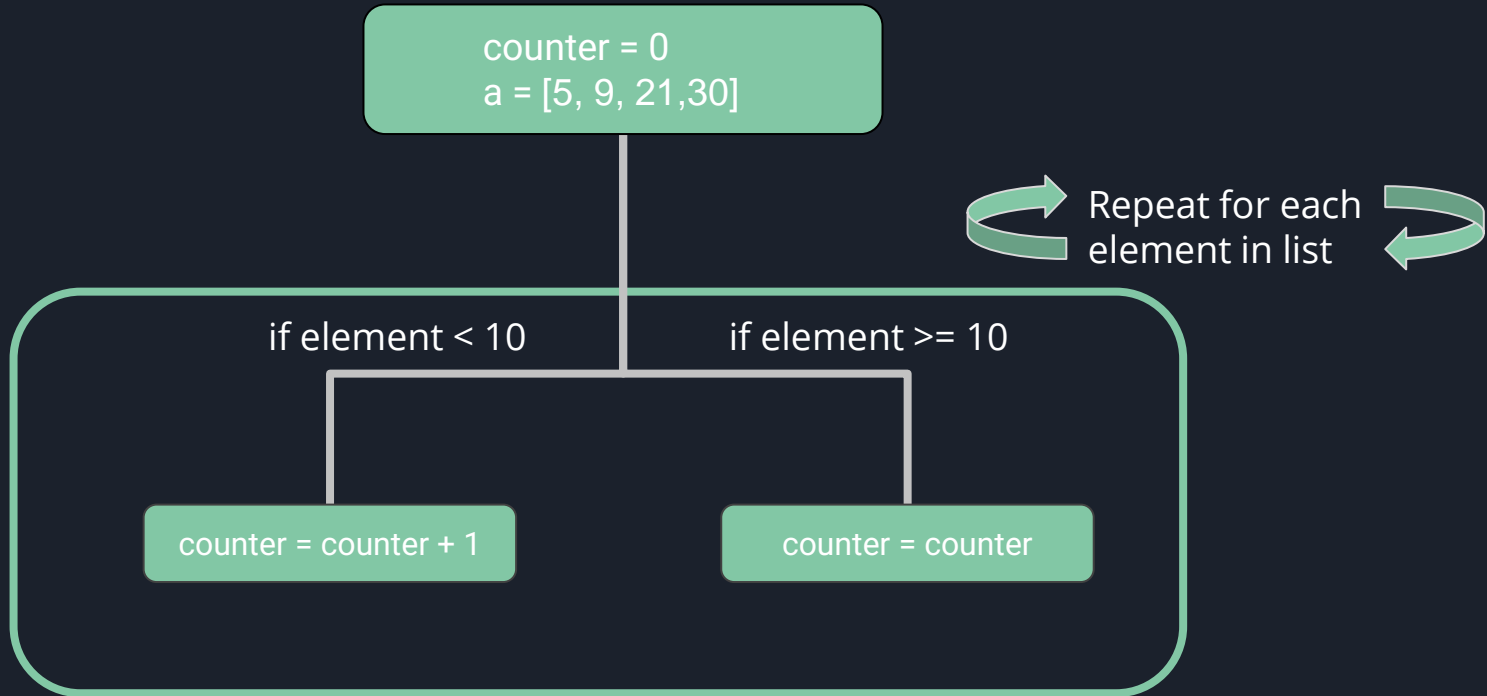
Count numbers smaller than 10 in a list

counter = 0
a = [5, 9, 21,30]

Count numbers smaller than 10 in a list

```
counter = 0
a = [5, 9, 21,30]
```

if element < 10   if element >= 10

```
counter = counter + 1
```

```
counter = counter
```

Count numbers smaller than 10 in a list

counter = 0
a = [5, 9, 21,30]

Repeat for each element in list

if element < 10

if element >= 10

counter = counter + 1

counter = counter

Control Flow - if / else

- Control which block of code will be executed
- Blocks defined by indentation

```python
if BOOLEAN-CONDITION:

    print("A")

else:

    print("B")
```

Control Flow - if / else

- Control which block of code will be executed
- Blocks defined by indentation

```python
if a>2:

    print("A")

else:

    print("B")
```

Control Flow - for-loop

- Repeat blocks of your code
- Use different values in each loop

```python
for element in [1,2,3,4]:

    print(element)
```

Exercise 2

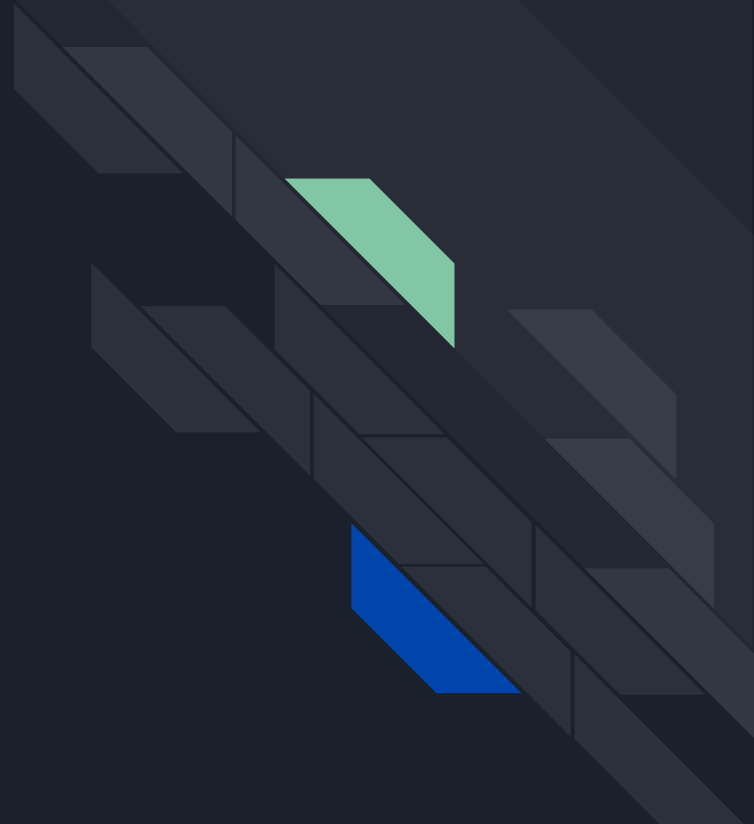Count amount of numbers in a list which are smaller than 5

```python
for element in [1,2,3,4]:
    print(element)
```

```python
if a>2:
    print("A")
else:
    print("B")
```

Fundamentals
Functions & Libraries

functions

```
a = doSomething(b)
```

functions

a = doSomething(b)

Name of function

functions

Argument of function
(can be a value, variable, list, dict,...)

a = doSomething(b)

Name of function

functions

Argument of function
(can be a value, variable, list, dict,...)

Variable for
return value

$$a = doSomething(b)$$

Name of function

functions - round function

```
b = 5.2

a = round(b)

→ a = 5.0
```

Built-in Functions

```
print()      sum()

round()      abs()

min()        range()

max()        sorted()
```

QUIZ

```python
round(2.34)   == ?
abs(-2)       == ?
a = [0,4,1,3,2]
max(a)        == ?
sum(a)        == ?
len(a)        == ?
sorted(a)     == ?
```

QUIZ

```python
round(2.34)   == 2.0
abs(-2)       == 2
a = [0,4,1,3,2]
max(a)        == 4
sum(a)        == 10
len(a)        == 5
sorted(a)     == [0,1,2,3,4]
```

Create own functions

- Define own functions for repeating tasks
- reduce amount of code lines

```python
def my_function(a,b):

    c = ...

    return c
```

Create own functions

- Define own functions for repeating tasks
- reduce amount of code lines

```python
def my_function(a,b):

    return a + b

my_function(1,2) # 3
```

Exercise 3

Convert your code which counts amount of numbers smaller 5 into a function

```python
def smaller_than(numbers, value):
    # your code here
    return counter
```

Libraries

- A collection of functions is bundled in a **library**
- we import these libraries and can use the defined functions
- Some libraries come with a Python installation, some need to be installed

 ... for plotting and visualization

 ... for working with tabular data (Excel-files, csv-files,...)

 ... creating machine learning models

Import Libraries

```python
import library


library.function_name()
```

Import Libraries

```python
import library as l


l.function_name()
```

Import Libraries

```python
from library import function

function()
```

NumPy - library

- Library for scientific computing
- Work with lists, matrices or higher dimensional structures
- NumPy lists have much more functionality than usual lists

```python
import numpy as np


a = np.array([1,2,3,4])
```

NumPy - library

- Library for scientific computing
- Work with lists, matrices or higher dimensional structures
- NumPy lists have much more functionality than usual lists

```python
import numpy as np


a = np.array([1,2,3,4])


a.sum() # 10

a.mean() # 2.5

a.std() # 1.118...
```

NumPy - Append

```python
import numpy as np


a = np.array([1,2,3,4])
np.append(a, [5,6,7]) # array([1, 2, 3, 4, 5, 6, 7])
```

NumPy - Append

```python
import numpy as np

a = np.array([1,2,3,4])
np.append(a, [5,6,7]) # array([1, 2, 3, 4, 5, 6, 7])
a = np.append(a, [5,6,7])
```

Compute with arrays - Broadcasting

```python
import numpy as np
a = np.array([1,2,3])
a * 2 # array([2, 4, 6])
a ** 2 # array([1, 4, 9])
a - 1 # array([0, 1, 2])
```

Compute with arrays

```python
import numpy as np
a = np.array([1,2,3])
b = np.array([1,1,1])
a + b # array([2, 3, 4])
```

Indexing

```python
import numpy as np

a = np.array([1,2,3,4])
a[0:2] # array([1,2])
```

Boolean Indexing

```python
import numpy as np


a = np.array([1,2,3,4])
a[[True,True,False,False]] # array([1,2])
```

Boolean Indexing

```python
import numpy as np

a = np.array([1,2,3,4])
a <= 2 # [True, True, False, False]
```

Boolean indexing

```python
import numpy as np

a = np.array([1,2,3,4])
a <= 2 # [True, True, False, False]
a[a<=2] # array([1,2])
```

Exercise 4

```python
import numpy as np

a = np.array([1,2,3,4])
a = np.append(a, [5,6,7])  # append
a[a<=2] # boolean indexing
```

Quick-Summary

```python
for element in [1,2,3,4]:
    print(element)
```

```python
if a >= 3:
    print("A")
else:
    print("B")
```

```python
import numpy as np

a = np.array([1,2,3,4])
a.sum()
```

```python
a = np.array([1,2,3,4])
a[a>2]
```

pandas - library

- data analysis and manipulation tool
- read in data of various types:
  - Excel files, CSV-Files, SQL-databases, tables on a website,....
- Within pandas data is stored in a **DataFrame**
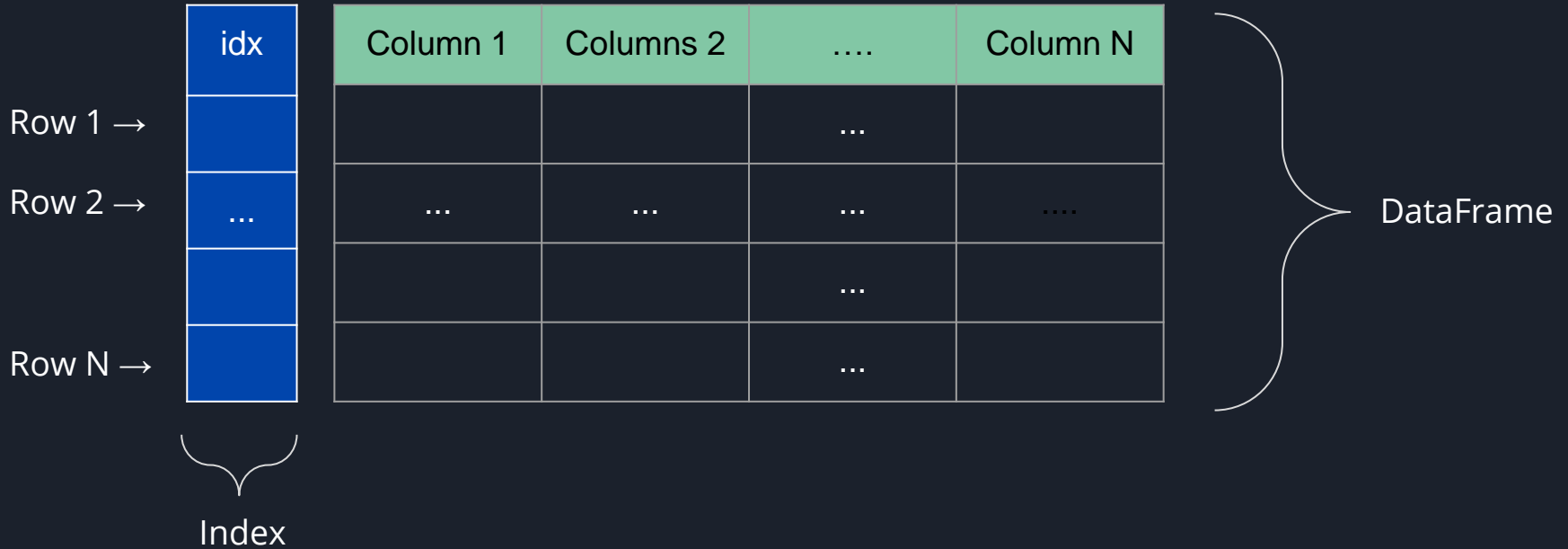- underlying data structure is a dictionary

```
data = { "Frankfurt":[10,12,15],
         "Berlin": [9,12,11],
         "Hamburg":[12,13,14] }
```

DataFrame:

	Frankfurt	Berlin	Hamburg
0	10	9	12
1	12	12	13
2	15	11	14

DataFrame - data structure

idx	Column 1	Columns 2	….	Column N
			...	
...	...	...	...	….
			...	
			...	

Row 1 →
Row 2 →
Row N →

Index

DataFrame

Operations on a DataFrame

- Sort
- filter, common filter examples
- group by
- merge
- apply computation