Python Programming

DeepSphere.AI
Enterprise AI and IIoT for Analytics

USA | Singapore

What is Python?

Python is a popular programming language. It was created by Guido van Rossum and released in 1991.
It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.
- Data Science
- Machine Learning

Which Python is the latest version?

- **Python 3.11.8 - Feb. 6, 2024.**

DeepSphere.AI
Enterprise AI and IIoT for Analytics

What can Python do?

- Server to create web applications.
- Software to create workflows.
- Database systems. It can also read and modify files.
- Handle big data and perform complex mathematics.
- Rapid prototyping or production-ready software development.

Why Python?

•Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).

•Python has a simple syntax similar to the English language.

•Python has a syntax that allows developers to write programs with fewer lines than some other programming languages.

•Python runs on an interpreter system, which means that code can be executed as soon as it is written. This allows for very quick prototyping.

•Python can be treated as procedural, object-oriented, or functional.

Python Syntax, compared to other programming languages

- Python was designed for readability and has some similarities to **the English language** with influence from mathematics.

- Python uses new lines to complete a command, as opposed to other programming languages, which often use semicolons or parentheses.

- Python relies on indentation, using whitespace, to define scope, such as the scope of loops, functions, and classes. Other programming languages often use curly brackets for this purpose.

DeepSphere.AI
Enterprise AI and IIoT for Analytics

Creating a Comment

```
#This is a comment
print("Hello, World!") #This is a comment
#print("Hello, World!")
```

Data Types

• Text Type	:	str
• Numeric Types	:	int, float, complex
• Sequence Types	:	list, tuple, range
• Mapping Type	:	dict
• Set Types	:	set, frozenset
• Boolean Type	:	bool
• Binary Types	:	bytes, bytearray, memoryview
• None Type	:	NoneType

Python Programming

- 1.1 Variables
- 1.2 Data types
- 1.3 Data Structures
- 1.4 Operators
- 1.5 Control Structures
- 1.6 Functions and Modules
- 1.7 Error Handling
- 1.8 File I/O
- 1.9 Classes and Objects

Difference between LIST, SET, TUPLE, and DICTIONARY

		Indexed	Ordered	Changeable	Duplicate Members
List	[]	YES	YES	YES	YES
Set	{}	NO	NO	NO	NO
Tuple	()	YES	NO	NO	YES
Dictionary	{}	NO	YES	YES	NO

Google Colab:

https://colab.research.google.com/drive/1uwsWOpppTFWAmR1gG5LwF1daq-3q_k6y#scrollTo=2zOsUk4c-V74

Difference between LIST, SET, TUPLE, and DICTIONARY

- Python Programming on the Spot Quiz
- Duration: 10 Min
- Submission: Hard copy (paper)
- Course: Python Programming

		Indexed	Ordered	Changeable	Duplicate Members
List	[]				
Set	{}				
Tuple	()				
Dictionary	{}				

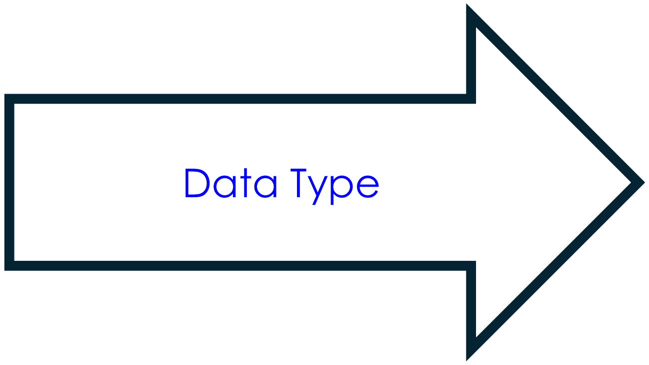
NUMPY Array

- Slicing
- Data Types
- Copy and View
- Shape
- Reshape
- Joining NumPy Arrays
- Splitting NumPy Arrays
- Searching Arrays
- Sorting Arrays
- Filter Array
- Broadcasting

https://colab.research.google.com/drive/1ys-z-hWaqxCb5yIyqbewSHGDClR9_ZwQ#scrollTo=2cHr9or6DanH&uniqifier=1

NUMPY Array

- i - integer ,
- b - boolean ,
- u - unsigned integer ,
- f - float
- c - complex float
- m - timedelta ,
- M - datetime ,
- O - object
- S - string,
- U - unicode string,
- V - fixed chunk of memory for other type (void)


Data Type

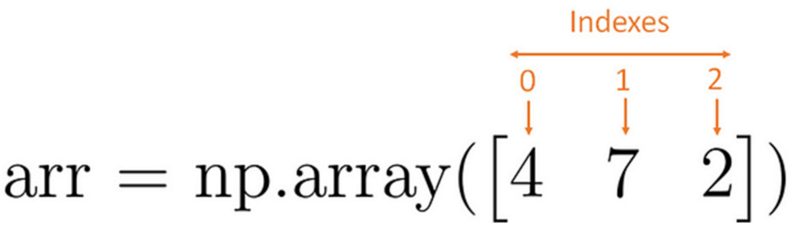
NUMPY Array

```
arr = np.array([111, 222, 333, 444, 555, 666, 777])
```

	111	222	333	444	555	666	777
Index	0	1	2	3	4	5	6
Negative Index	-7	-6	-5	-4	-3	-2	-1

NUMPY Array

One dimensional array

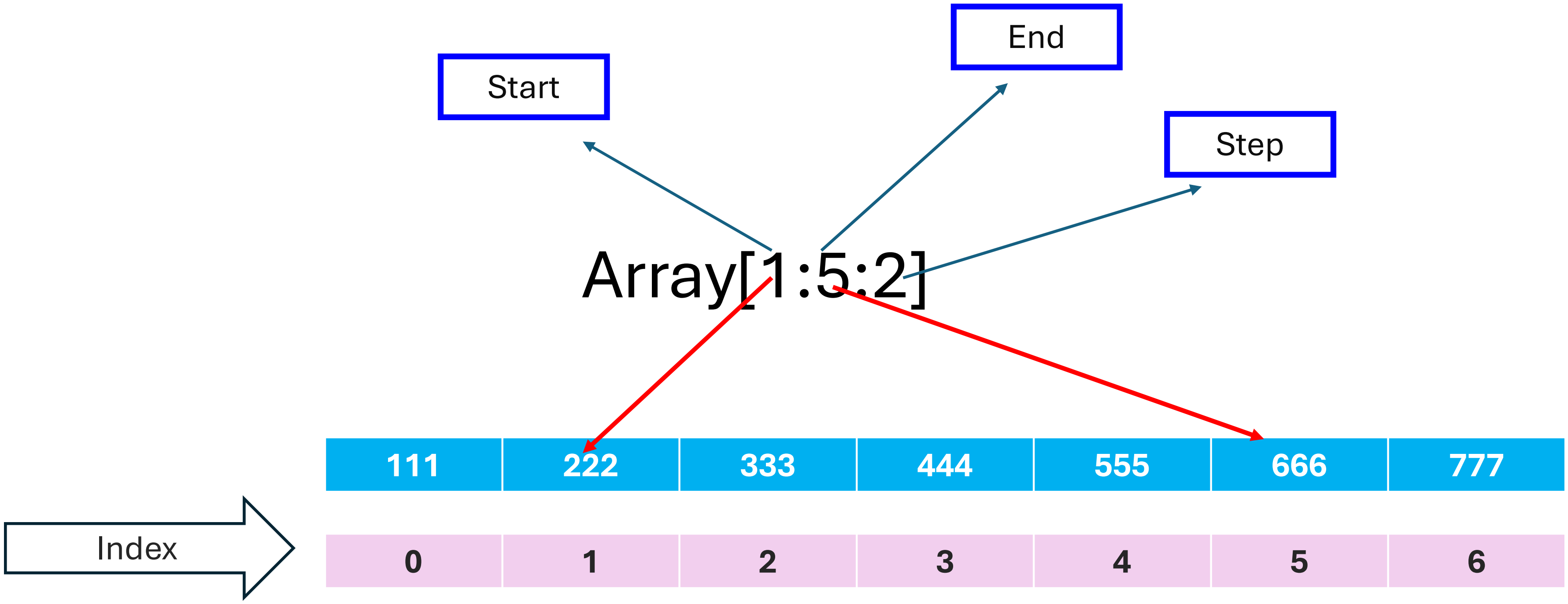


$$\text{arr} = \text{np.array}([4 \quad 7 \quad 2])$$

- ✓ Let’s say I want to print the number 7 (which is the second element). I get it by **indexing** the array “arr” with a 1 in square brackets.

- ✓ print(arr[1]) = 7

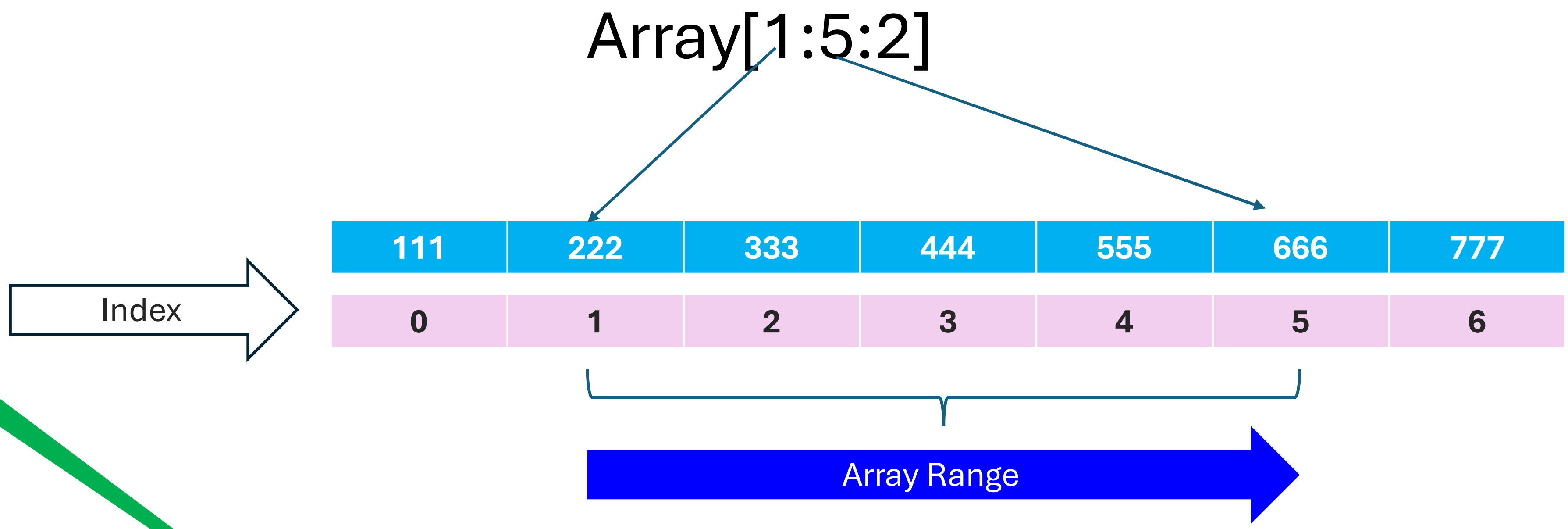
NUMPY Array



Array[1:5:2]

	111	222	333	444	555	666	777
Index	0	1	2	3	4	5	6

NUMPY Array



Array[1:5:2]

Index							
	111	222	333	444	555	666	777
Index	0	1	2	3	4	5	6

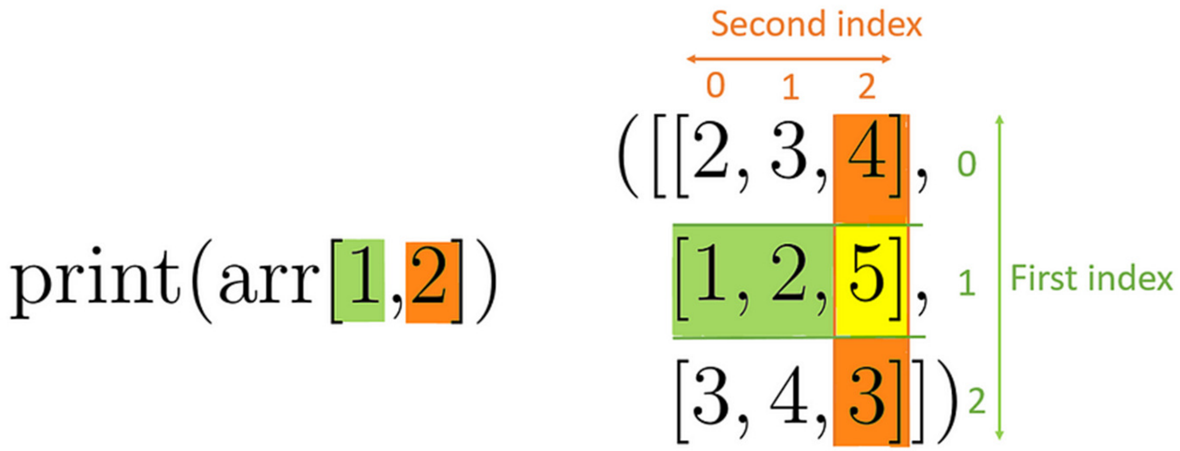
NUMPY Array

Two-dimensional arrays

$$
\begin{array}{rcccl}
 & \multicolumn{3}{c}{\text{Second index}} & \\
 & \multicolumn{3}{c}{\longleftrightarrow} & \\
 & 0 & 1 & 2 & \\
\text{arr} = \text{np.array}([& [2, & 3, & 4], & 0 \\
 & [1, & 2, & 5], & 1 \quad \text{First index} \\
 & [3, & 4, & 3]]) & 2
\end{array}
$$

NUMPY Array

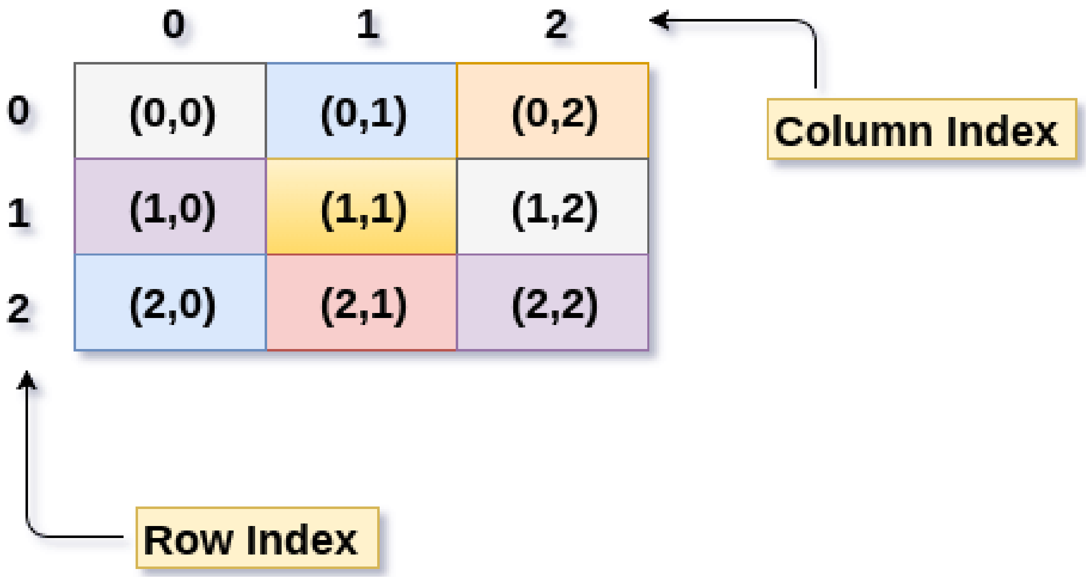
Two dimensional arrays



```
print(arr[1,2])
([[2, 3, 4],
  [1, 2, 5],
  [3, 4, 3]])
```

- ✓To get, for example, the number 5 from this array, you would have to index the array with the first and then the second index.
- ✓print(arr[1,5]) = 5

NUMPY Array



	0	1	2
0	(0,0)	(0,1)	(0,2)
1	(1,0)	(1,1)	(1,2)
2	(2,0)	(2,1)	(2,2)

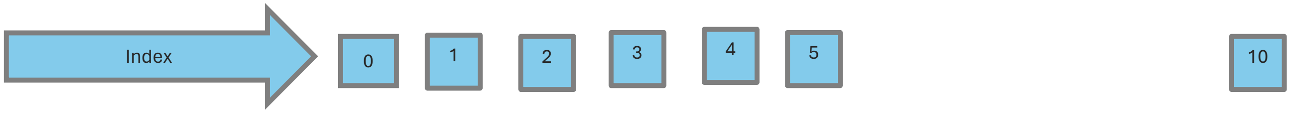
Column Index

Row Index

NUMPY Array

arr = np.arange(0,11)

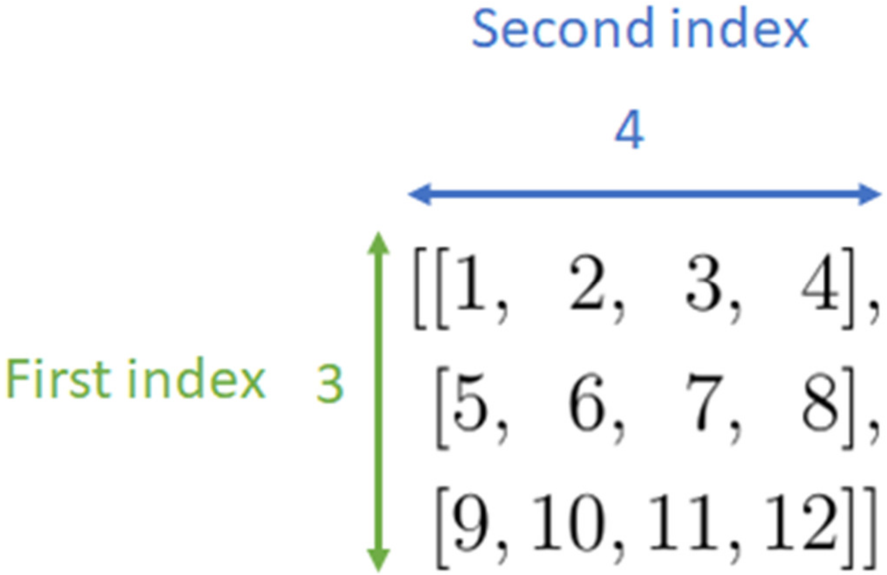
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])



Index 0 1 2 3 4 5 10

- ✓ arr[8] = 8
- ✓ arr[1:5] = [1,2,3,4]
- ✓ arr[0:5]= [0,1,2,3,4]
- ✓ arr[0:10]=[0,1,2,3,4,5,6,7,8,9,10]
- ✓ arr[:]=999 >>> [999 , 999 , 999 , 999 , 999 , 999 , 999 , 999 , 999 , 999 , 999]

NUMPY Array



Second index

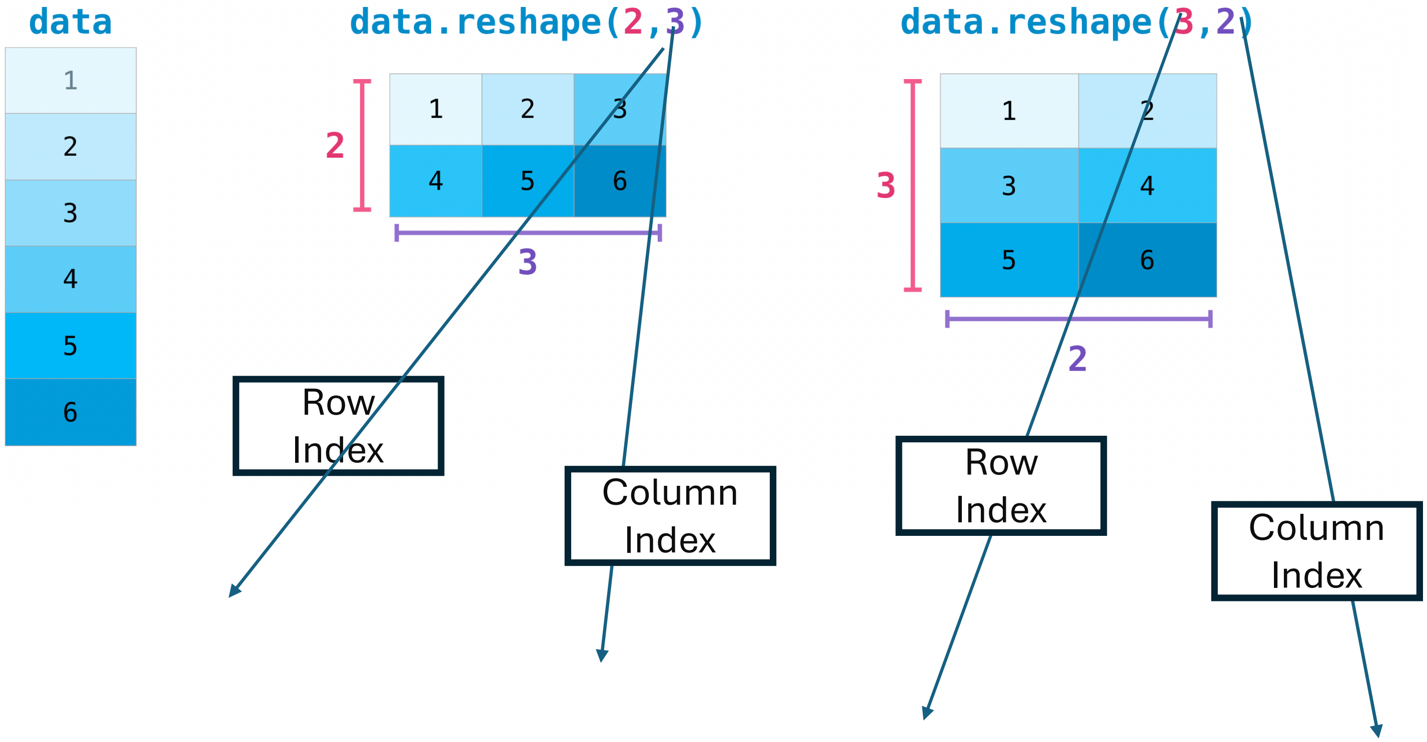
4

First index 3

$$\begin{matrix}[[1, & 2, & 3, & 4], \\ [5, & 6, & 7, & 8], \\ [9, & 10, & 11, & 12]]\end{matrix}$$

shape = (3,4)

NUMPY Array



`data`

1
2
3
4
5
6

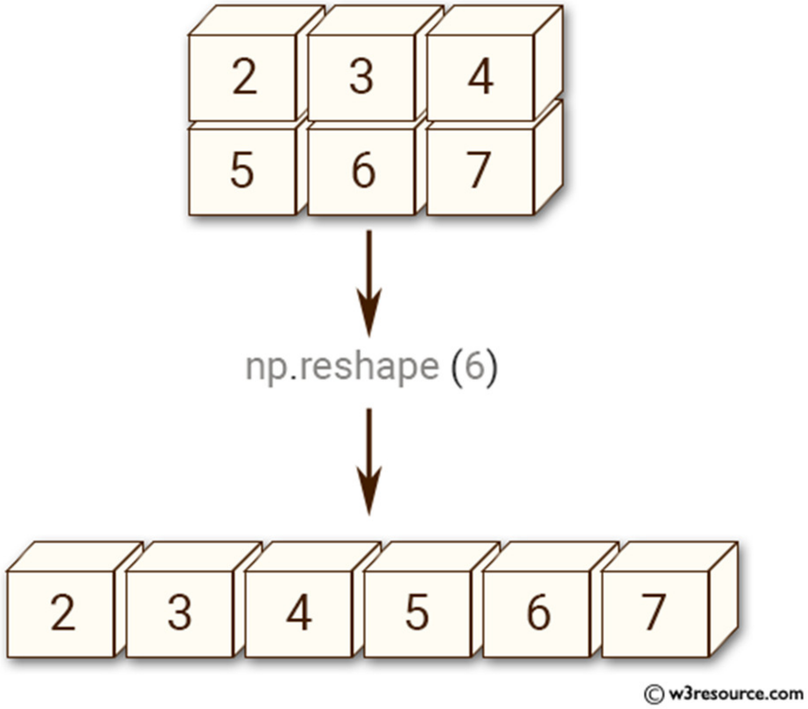
`data.reshape(2,3)`

1	2	3
4	5	6

`data.reshape(3,2)`

1	2
3	4
5	6

NUMPY Array



NUMPY Array

```
X=[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

Input

reshape(x, [2, 2, 5]))

```
[[[ 0  1  2  3  4]
  [ 5  6  7  8  9]]

[[10 11 12 13 14]
  [15 16 17 18 19]]]
```

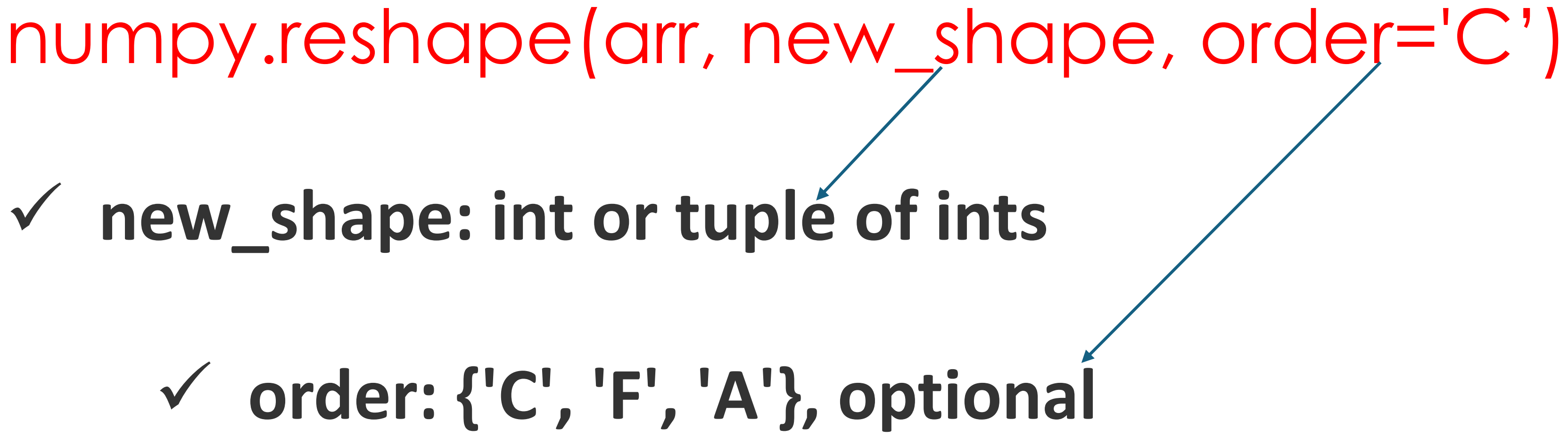
Output

- ✓ In the example above, we are converting the defined 1-D array with 20 elements into a 2-D array. The outermost dimension will have 2 arrays that contain 2 arrays, each with 5 elements**. >>> (2 * 2) * 5 = 20**

- **So an attempt like print(np. reshape(x, [5,6])) will run into a ValueError. This is because we cannot reshape an array of size 20 into shape (5,6) >> 5*6 = 30**

NUMPY Array

- We can **reshape** any array into any shape as long as the elements required for reshaping are equal in both shapes.



numpy.reshape(arr, new_shape, order='C’)

- ✓ **new_shape: int or tuple of ints**
- ✓ **order: {'C', 'F', 'A'}, optional**

Pandas

https://colab.research.google.com/drive/1dz-8VVKatCog4vmIDX3Q50_X1GLEeN_M#scrollTo=GmfiboPOzFsm

Pandas

- pandas Series
- pandas DataFrame
- pandas Index

DeepSphere.AI
Enterprise AI and IIoT for Analytics

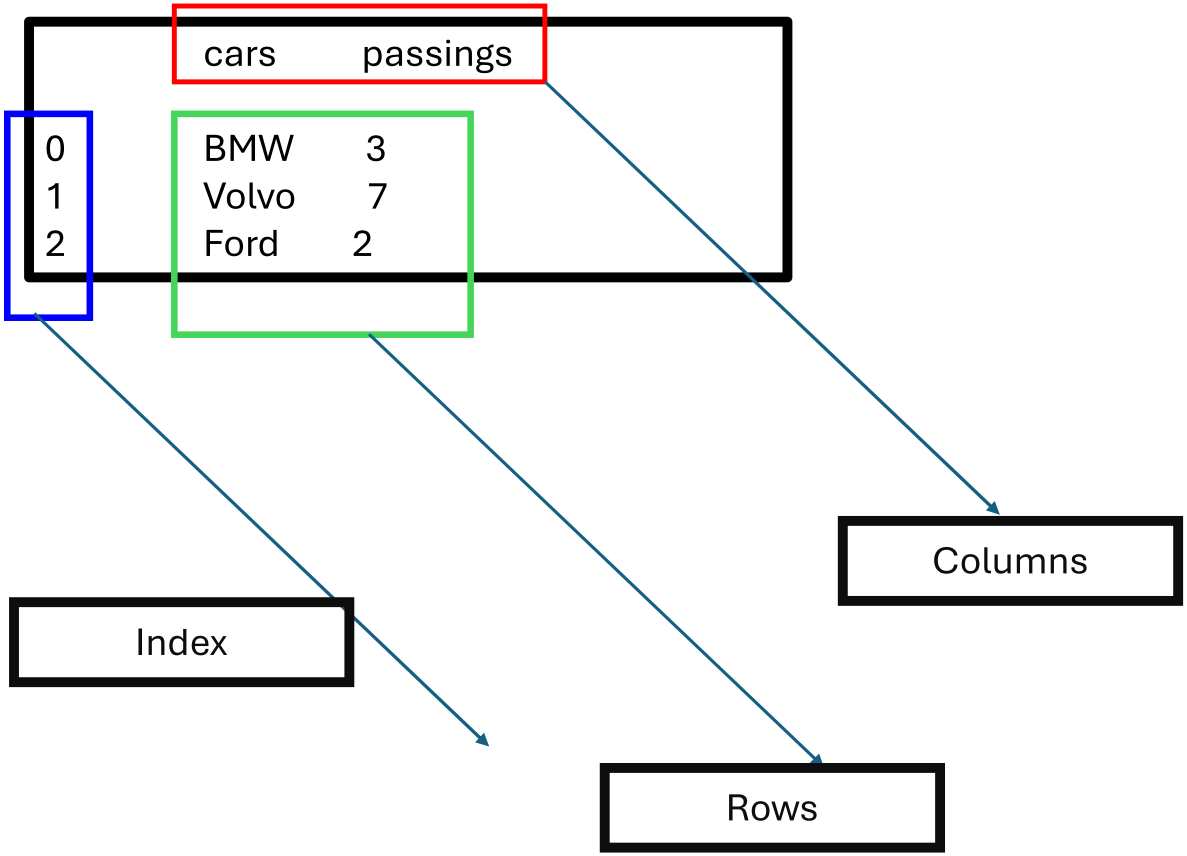
Pandas

The Pandas Series can be defined as a one-dimensional array capable of storing various data types. We can easily convert the list, tuple, and dictionary into series using the "series" method. The row labels of the series are called the index. A Series cannot contain multiple columns.

•**data** can be any list, dictionary, or scalar value.
•**index:** The index value should be unique, hashable, and the same length as the data. If we do not pass an index, the default **np. arrange(n)** will be used.
•**dtype:** It refers to the data type of the series.
•**copy:** It is used for copying the data.

Difference Between Pandas Series and Single Column DataFrame

	Pandas Series	**Single Column Data Frame**
Data Structure	1D Table	2D Table
Alignment	Not supported	Supported
Columns	None	1
Functionality	Less	More
Index	Required	Optional
Performance	Quick	Slow
Name	Optional	Optional



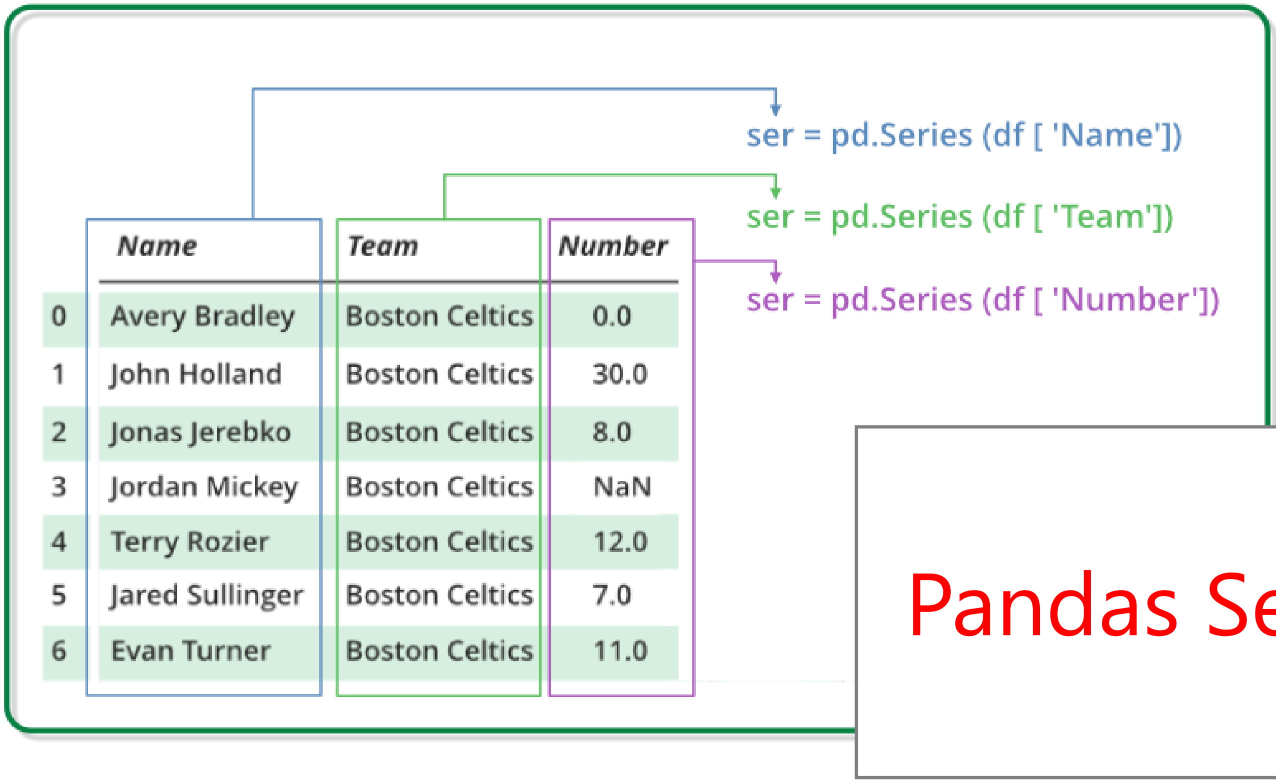
	cars	passings
0	BMW	3
1	Volvo	7
2	Ford	2

Index

Rows

Columns

What is a Series? A Pandas Series is like a column in a table. It is a one-dimensional array holding data of any type.

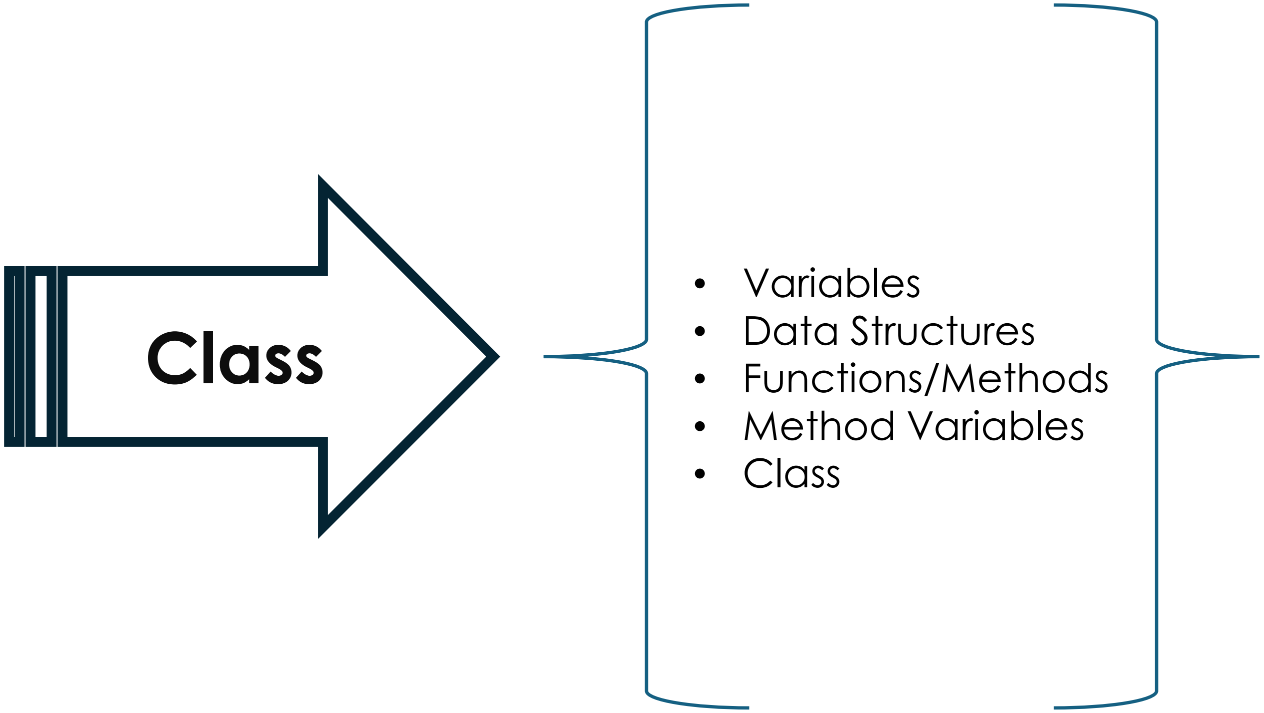


	Name	Team	Number
0	Avery Bradley	Boston Celtics	0.0
1	John Holland	Boston Celtics	30.0
2	Jonas Jerebko	Boston Celtics	8.0
3	Jordan Mickey	Boston Celtics	NaN
4	Terry Rozier	Boston Celtics	12.0
5	Jared Sullinger	Boston Celtics	7.0
6	Evan Turner	Boston Celtics	11.0

```
ser = pd.Series (df [ 'Name'])
ser = pd.Series (df [ 'Team'])
ser = pd.Series (df [ 'Number'])
```

Pandas Series

Class

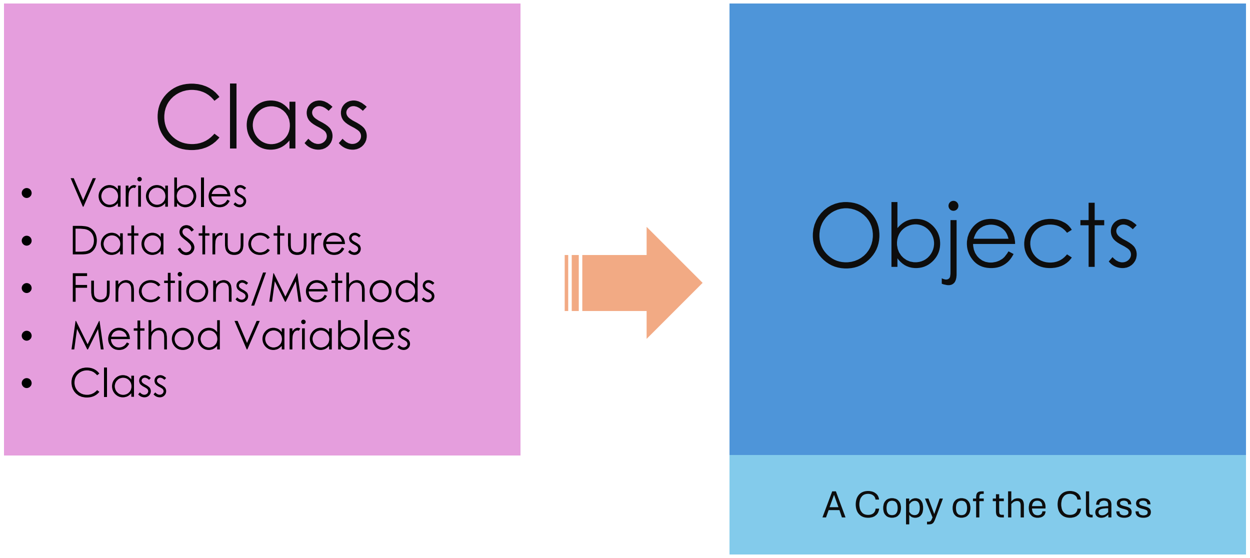


Class

- Variables
- Data Structures
- Functions/Methods
- Method Variables
- Class

- A class is a collection of variables, methods (functions), and other classes.
- We can access a class member through objects

Class



Class

- Variables
- Data Structures
- Functions/Methods
- Method Variables
- Class

Objects

A Copy of the Class

- We can access the class members via the Object.

- Object Name. Class Members

Class

```
def __init__(self,vAR_name,vAR_age):
```

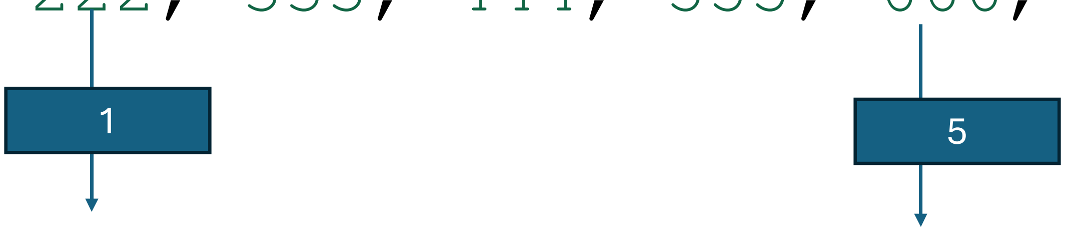
Every class has a __init__ method and it’s automatically called at the time of obeject initialization

```
Class HR:
        def __init___ (self, EID, ENAME, EDPT, ....._):
                self.EID= EID
                sef.ENAME=ENAME
                self.DEPT=DEPT
        def PAYROLL(self, PARGRD):
        def fun1():
          pass
        def fun2():
          pass
        def fun3():
          pass
```

```
HR_OBJ = HR(100,”JOTHI”, “10) # Object
```

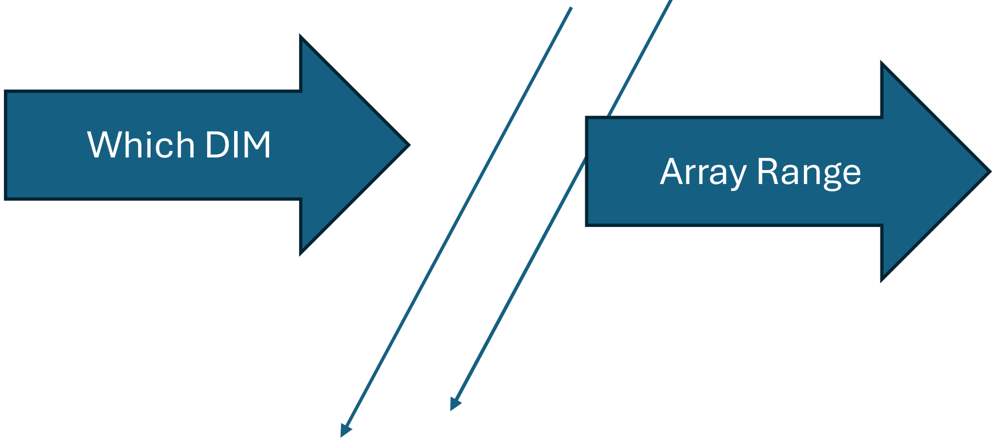
```
HR_OBJ. PAYROLL(B1)
```

```
arr = np.array([111, 222, 333, 444, 555, 666, 777])
```



```
print(arr[1:5:2]) #STEP - Return every other element from index 1 to index 5:
```

```
arr = np.array
(
[

[1, 2, 3, 4, 5],

[6, 7, 8, 9, 10]

]
)

    print(arr[1, 1:4])
```



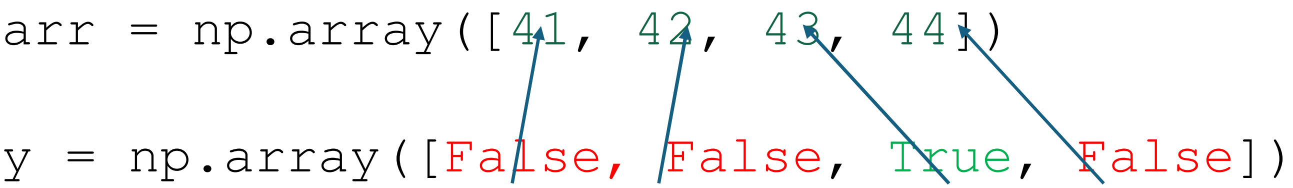
Which DIM

Array Range

```
arr = np.array([1, 2, 3, 4], dtype='f’)

newarr = arr.astype(bool)
```

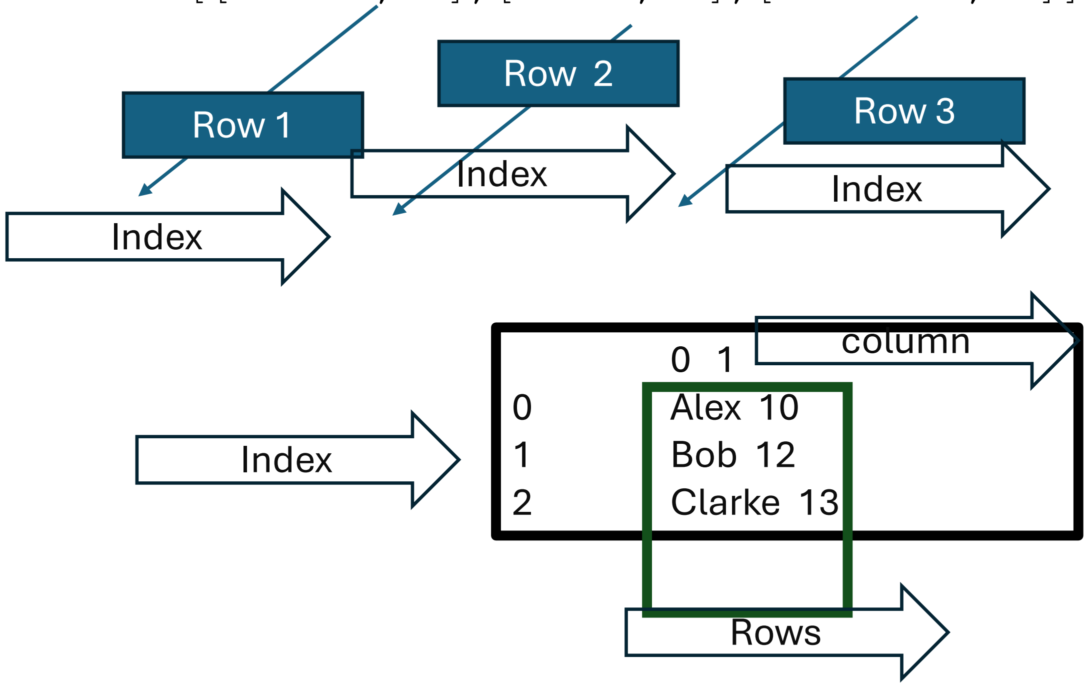
- # If the value at an index is True that element is contained in the filtered array,
- # if the value at that index is False that element is excluded from the filtered array.

```
arr = np.array([41, 42, 43, 44])

y = np.array([False, False, True, False])
```



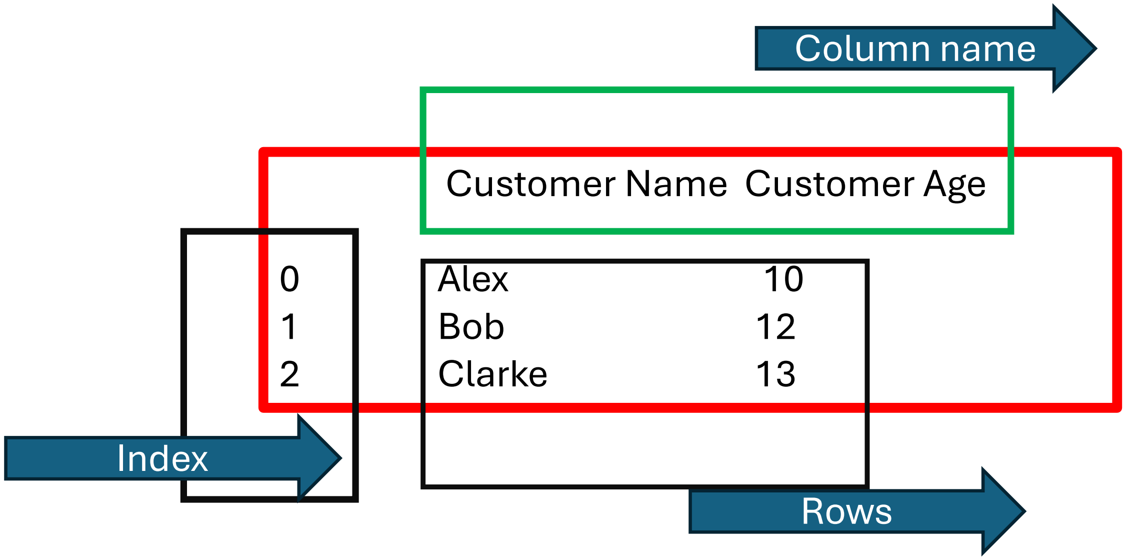
```
arr = np.array([41, 42, 43, 44])

y = np.array([False, False, True, False])

Newarr  = arr[y] # Here we are applying the filter to the ARR

Print(Newarr) =➔ 43
```

```
data = [['Alex',10],['Bob',12],['Clarke',13]]
```



```
        0  1
0       Alex  10
1       Bob  12
2       Clarke  13
```

```
     cars  passings
0    BMW      3
1    Volvo     7
2    Ford     2
```



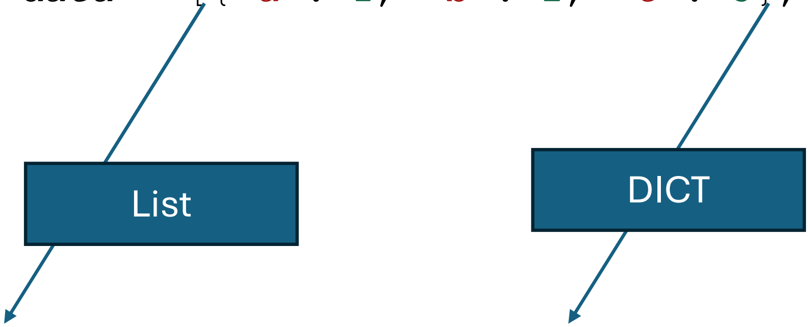
Column name

```
     Customer Name  Customer Age
0    Alex                10
1    Bob                 12
2    Clarke              13
```

Index

Rows

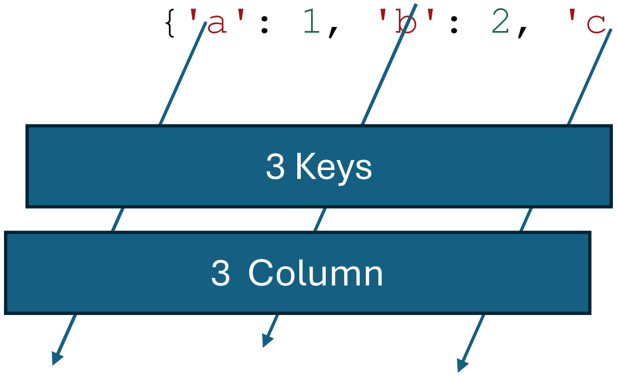
```
data = [{'a': 1, 'b': 2, 'c': 3},{'a': 5, 'b': 10, 'c': 20}]
```



List

DICT

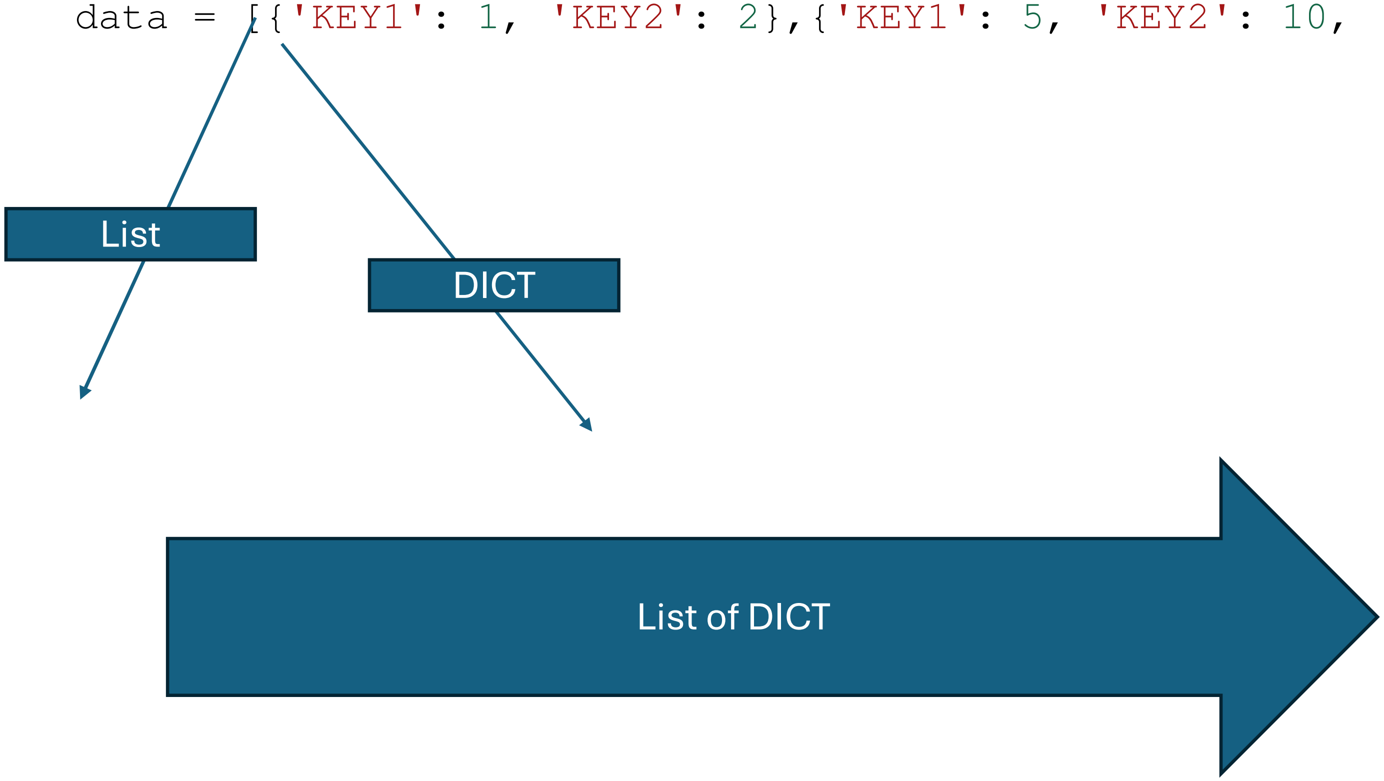
```
{'a': 1, 'b': 2, 'c': 3},{
```



3 Keys

3 Column

```
data = [{'KEY1': 1, 'KEY2': 2},{'KEY1': 5, 'KEY2': 10, 'KEY3': 20}]
```



List

DICT

List of DICT

```
data = [{'KEY1': 1, 'KEY2': 2},{'KEY1': 5, 'KEY2': 10, 'KEY3': 20}]
```

This is missing in DICT #1

Retail Analytics

Retail companies like Walmart India want to analyze product groups, products, and unit prices by city. The product and unit price data comes in key and value format in a CSV file. Create an appropriate data structure in Python to store products and unit prices. Print the product groups, products, and unit prices by the city as follows.

```
Product groups      | Product     | Unit Prices          | City
PG1                 | P1          | 100                  | Chennai
PG1                 | P2          | 200                  | Chennai
PG1                 | P3          | 300                  | Chennai
```