Python Object-Oriented Programming(OOP)

Grow Data Skills

OOP is like building a car using modular parts. Here, code is broken down into smaller, reusable pieces called **objects** that represent real-world entities.

Classes

Class is a template/blue-print for real-world entities

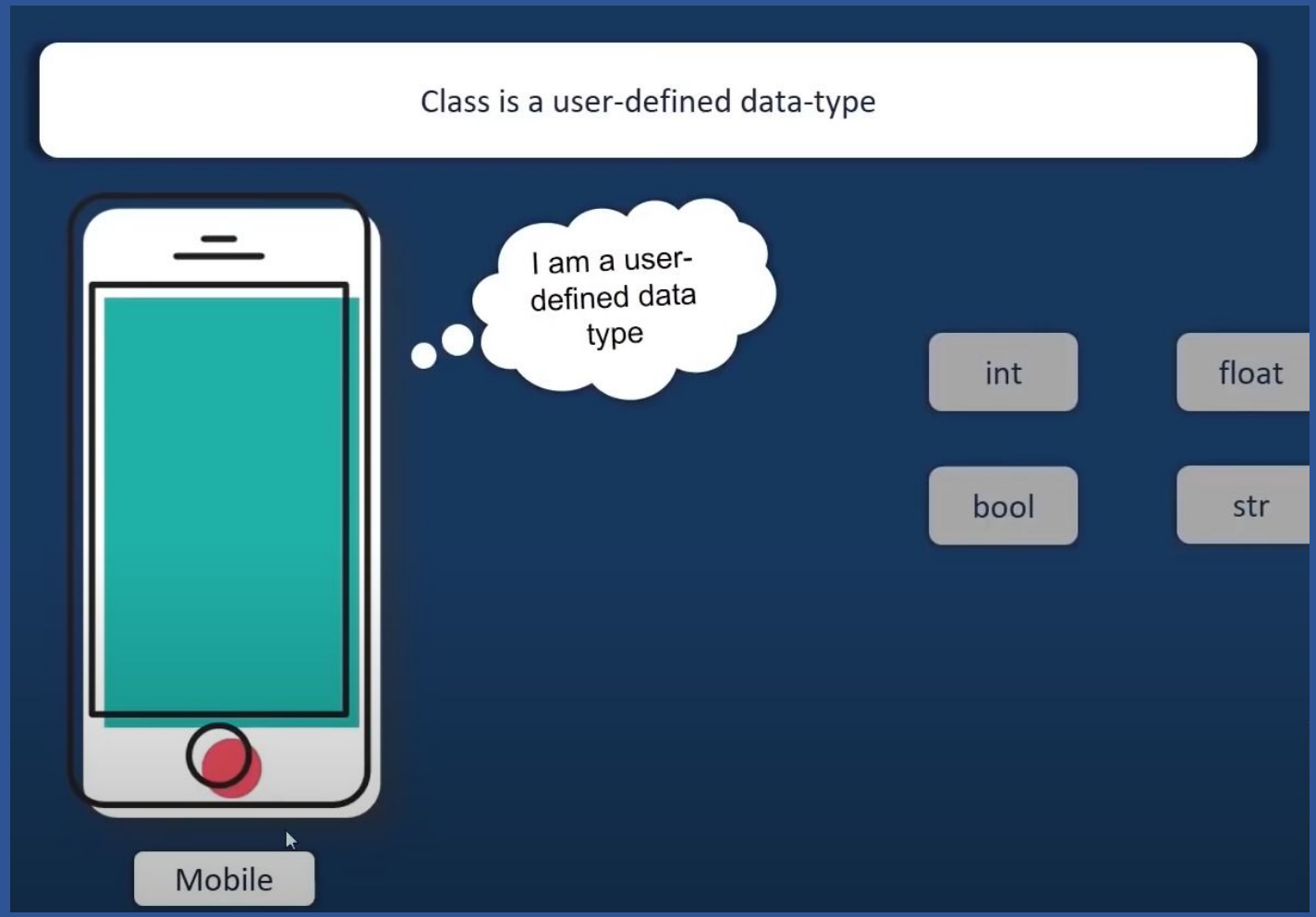
Properties

- Color
- Cost
- Battery Life

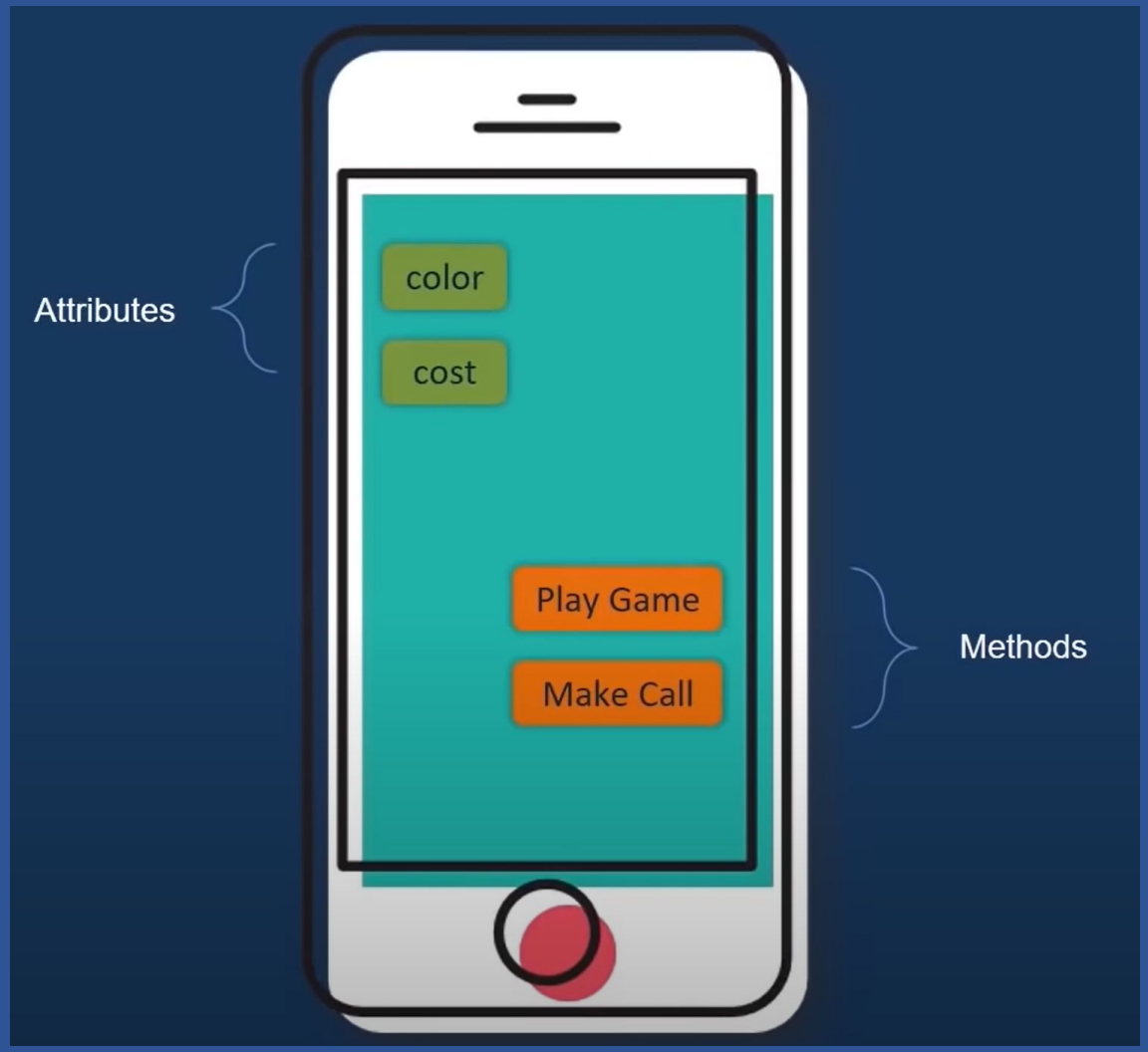
Behavior

- Make Calls
- Watch Videos
- Play Games

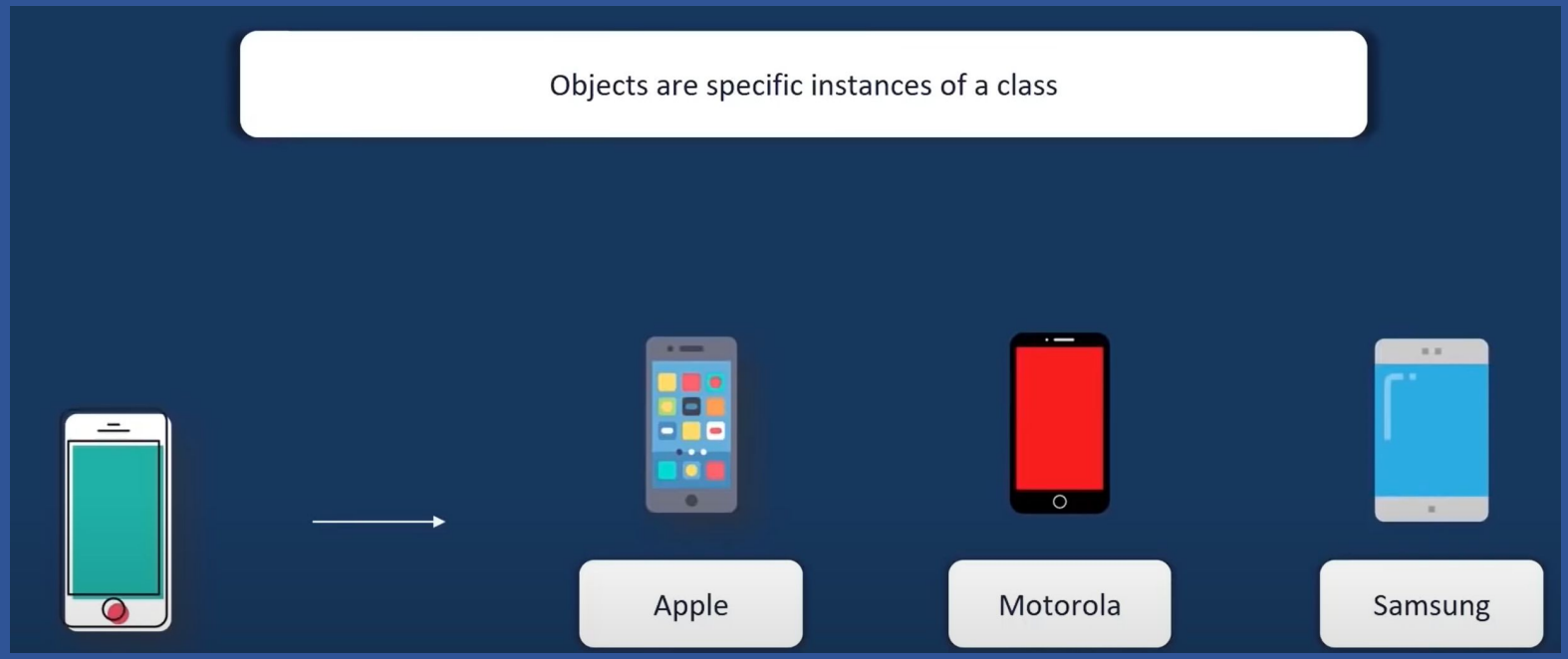
Class in Python



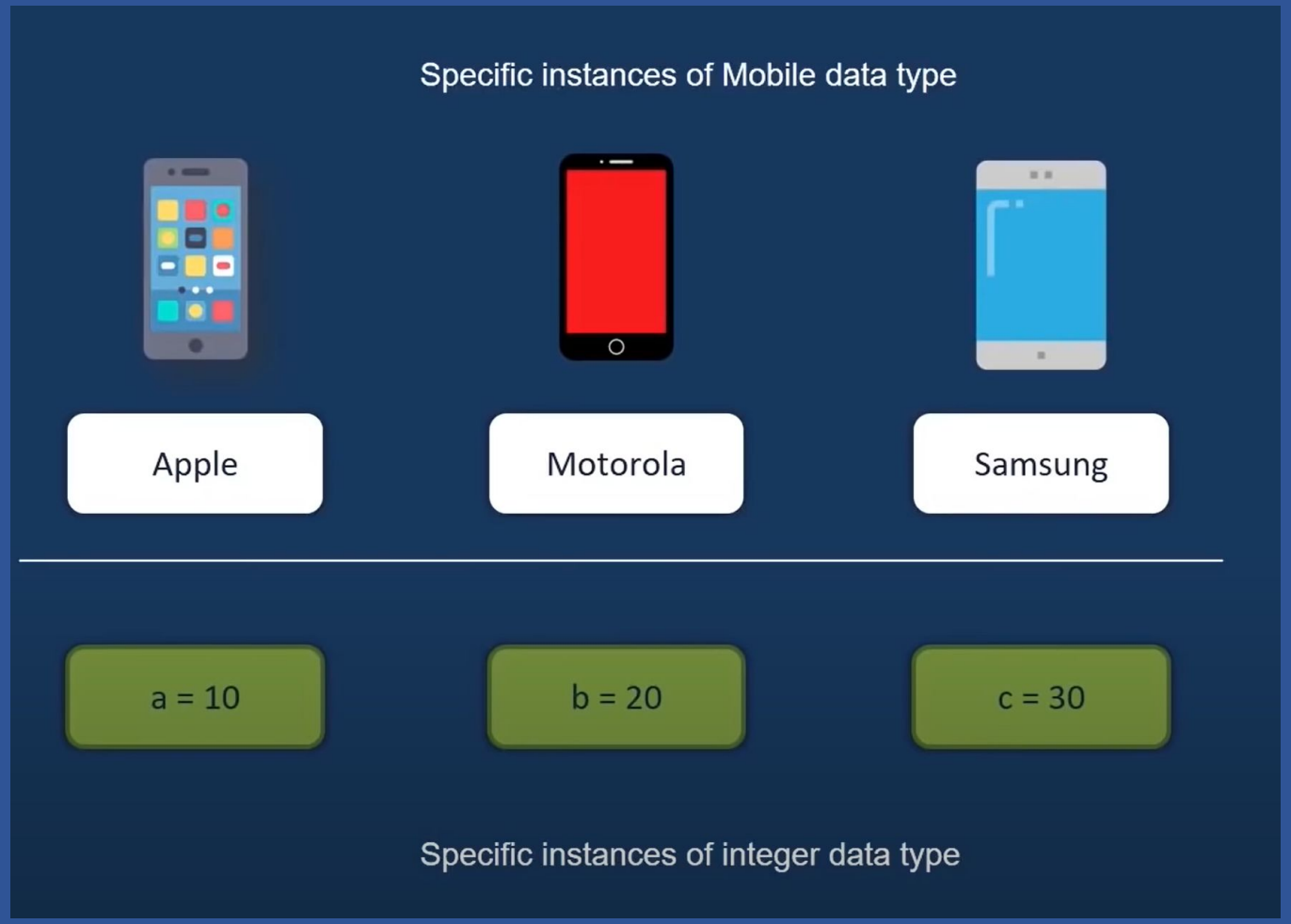
Attributes and Methods



Objects



Objects in Python



Creating first Class

Grow Data Skills

```
In [1]: class Phone:

            def make_call(self):
                print("Making phone call")

            def play_game(self):
                print("Playing Game")
```

Creating the 'Phone' class

```
In [38]: p1=Phone()
```

Instantiating the 'p1' object

```
In [39]: p1.make_call()
         Making phone call

In [40]: p1.play_game()
         Playing Game
```

Invoking methods through object

Adding parameters to the Class


```
In [42]: class Phone:

             def set_color(self,color):
                 self.color=color

             def set_cost(self,cost):
                 self.cost=cost

             def show_color(self):
                 return self.color

             def show_cost(self):
                 return self.cost

             def make_call(self):
                 print("Making phone call")

             def play_game(self):
                 print("Playing Game")
```

Setting and Returning the attribute values

Creating Class with Constructor

Grow Data Skills

```
In [4]: class Employee:
            def __init__(self,name,age, salary,gender):

                self.name = name
                self.age = age
                self.salary =  salary
                self.gender = gender

            def employee_details(self):
                print("Name of employee is ",self.name)
                print("Age of employee is ",self.age)
                print("Salary of employee is ",self.salary)
                print("Gender of employee is ",self.gender)
```

Init method acts as the constructor

Instantiating Object

Instantiating the ‘e1’ object

Invoking the ‘employee_details’ method

```
In [5]: e1 = Employee('Sam',32,85000,'Male')

In [6]: e1.employee_details()

        Name of employee is  Sam
        Age of employee is  32
        Salary of employee is  85000
        Gender of employee is  Male
```

Libraries in Python

Grow Data Skills

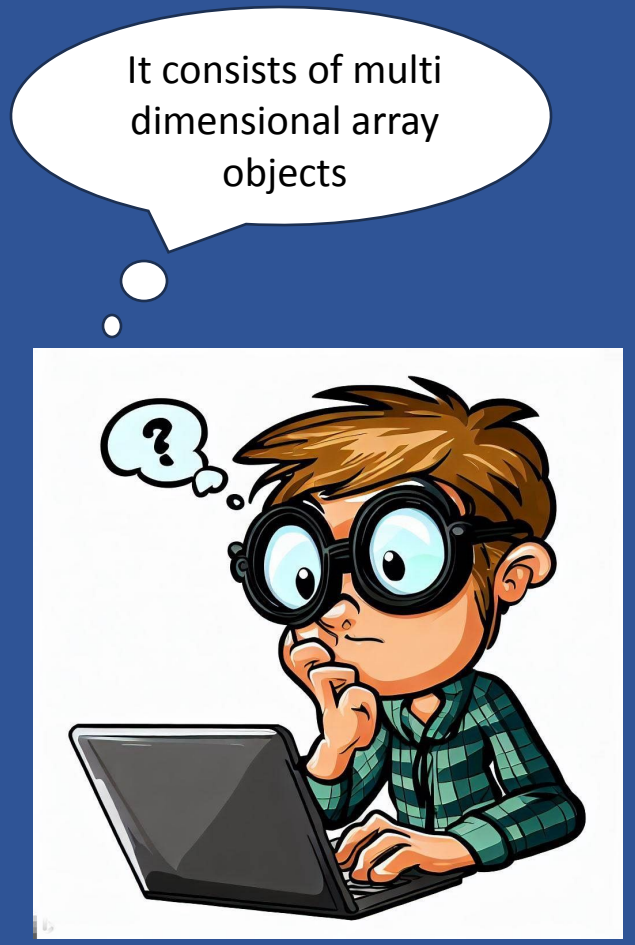
Python Library is a collection of pre-written code containing functions and modules that allows you to perform many actions without writing your code

Python NumPy

Grow Data Skills

NumPy stands for **Numerical Python** and is the core library for numeric and scientific computing

- Lists serve the purpose of arrays, but they are slow(NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.)
- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
- The array object in NumPy is called ndarray



Creating NumPy Array

Single-Dimensional Array

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

arr
```

```
array([1, 2, 3, 4, 5])
```

Multi- Dimensional Array

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

arr
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

Initializing NumPy Array

Initializing with 0

```
import numpy as np

n1=np.zeros((5,5))
n1
```

```
array([[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.]])
```

Initializing with same number

```
n1=np.full((2,2),10)
n1
```

```
array([[10, 10],
       [10, 10]])
```

Initializing NumPy Array

Initializing NumPy Array within a range

```
n1=np.arange(10,20)
n1
```

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

```
n1=np.arange(10,50,5)
n1
```

```
array([10, 15, 20, 25, 30, 35, 40, 45])
```

Joining NumPy Array

vstack()

```
n1=np.array([10,20,30])
n2=np.array([40,50,60])

np.vstack((n1,n2))
```

```
array([[10, 20, 30],
       [40, 50, 60]])
```

hstack()

```
n1=np.array([10,20,30])
n2=np.array([40,50,60])

np.hstack((n1,n2))
```

```
array([10, 20, 30, 40, 50, 60])
```

column_stack()

```
n1=np.array([10,20,30])
n2=np.array([40,50,60])

np.column_stack((n1,n2))
```

```
array([[10, 40],
       [20, 50],
       [30, 60]])
```

NumPy Intersection and Difference

Grow Data Skills

```
arr1 = np.array([10, 20, 30, 40, 50, 60])

arr2 = np.array([40, 50, 60, 70, 80, 90])
```

```
np.intersect1d(arr1,arr2)

array([40, 50, 60])
```

```
arr1 = np.array([10, 20, 30, 40, 50, 60])

arr2 = np.array([40, 50, 60, 70, 80, 90])
```

```
np.setdiff1d(arr1,arr2)

array([10, 20, 30])
```

```
arr1 = np.array([10, 20, 30, 40, 50, 60])

arr2 = np.array([40, 50, 60, 70, 80, 90])
```

```
np.setdiff1d(arr2,arr1)

array([70, 80, 90])
```

NumPy Array Mathematics

Basic Addition

```
n1=np.array([10,20,30])
n1=n1+1
print(n1)
```

```
[11 21 31]
```

Basic Multiplication

```
n1=np.array([10,20,30])
n1=n1*2
print(n1)
```

```
[20 40 60]
```

Basic Subtraction

```
n1=np.array([10,20,30])
n1=n1-1
print(n1)
```

```
[ 9 19 29]
```

Basic Division

```
n1=np.array([10,20,30])
n1=n1/2
print(n1)
```

```
[ 5. 10. 15.]
```

NumPy Math Functions

Grow Data Skills

Mean

```
n1=np.array([10,20,30,40,50,60])

np.mean(n1)
```

```
35.0
```

Median

```
n1=np.array([11,44,5,96,67,85])

np.median(n1)
```

```
55.5
```

Standard Deviation

```
n1=np.array([1,5,3,100,4,48])

np.std(n1)
```

```
36.59424666377065
```

Python Pandas

Grow Data Skills

Pandas stands for **Panel Data** and is the core library for **data manipulation and data analysis**

- **Data Manipulation and Transformation**: Pandas provides efficient data structures, such as DataFrames, that allow for easy manipulation, cleaning, and transformation of data.
- **Data Exploration and Analysis**: Pandas enables exploratory data analysis by providing powerful tools for slicing, indexing, and extracting insights from datasets.
- **Integration with other Libraries**: Pandas seamlessly integrates with other popular libraries in the data analysis and scientific computing ecosystem, such as NumPy, Matplotlib, and scikit-learn.

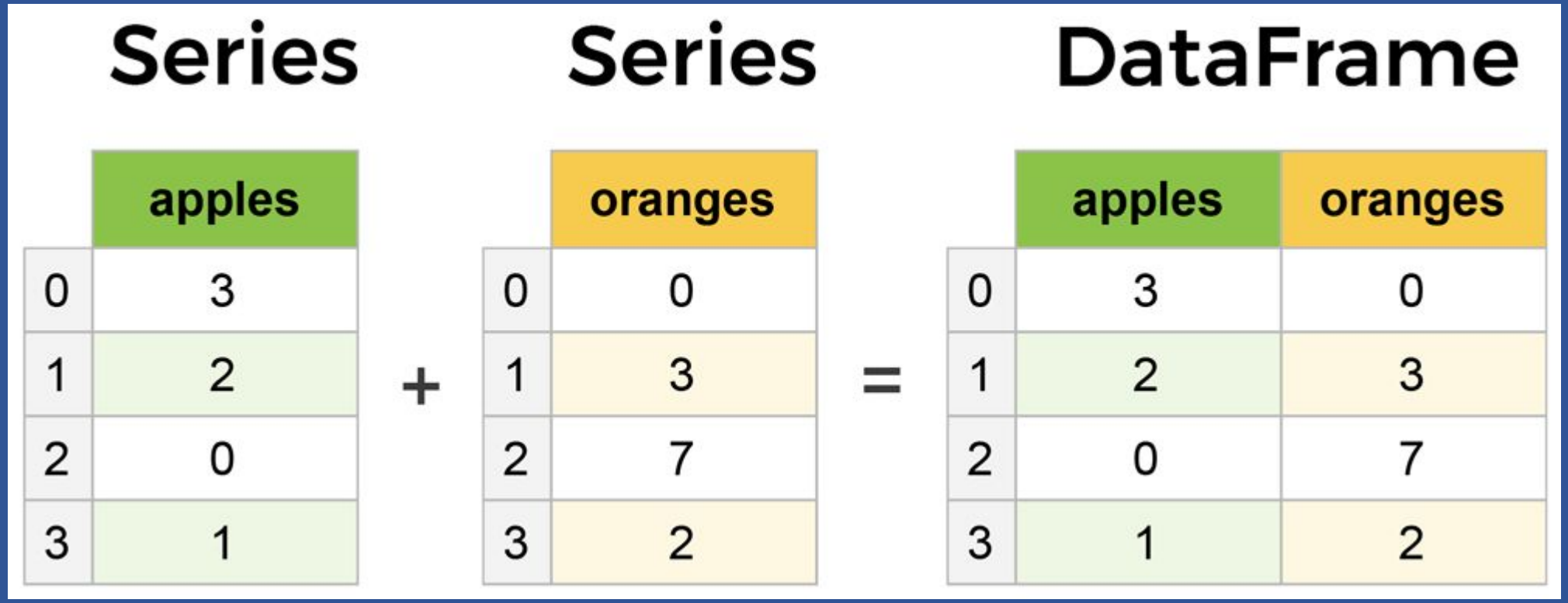
It consists of single and multi dimensional data structures for data manipulation

Pandas Data Structures

Grow Data Skills

Single Dimensional

Multi Dimensional



Pandas Series Object

Series Object is a One-Dimensional Labeled array

```
import pandas as pd

# Create a Series from a list
data = [10, 20, 30, 40, 50]
series = pd.Series(data)

series
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

```
type(series)
```

```
pandas.core.series.Series
```

Changing Index


```
import pandas as pd

s1=pd.Series([1,2,3,4,5])
s1
```

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

```
import pandas as pd

s1=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
s1
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

Series Object from Dictionary

Grow Data Skills

```
import pandas as pd


pd.Series({"a": 10, "b": 20, "c": 30})
```

```
a    10
b    20
c    30
dtype: int64
```

You can also create series object from a dictionary!!

Extracting Individual Elements

Grow Data Skills

single element

```
s1=pd.Series([1,2,3,4,5,6,7,8,9,10])

s1[9]
```

```
6
```

elements from back

```
s1=pd.Series([1,2,3,4,5,6,7,8,9,10])

s1[-4:]
```

```
6     7
7     8
8     9
9    10
dtype: int64
```

Sequence of elements

```
s1=pd.Series([1,2,3,4,5,6,7,8,9,10])

s1[:3]
```

```
0    1
1    2
2    3
dtype: int64
```

Basic Math Operations on Series

Adding scaler value to Series element

```
s1=pd.Series([1,2,3,4,5,6,7,8,9,10])

s1+10
```

```
0    11
1    12
2    13
3    14
4    15
5    16
6    17
7    18
8    19
9    20
dtype: int64
```

Adding two Series Object

```
s1=pd.Series([1,0,1,0,1,0,1,0,1])

s2=pd.Series([10,20,30,40,50,60,70,80,90])

s1+s2
```

```
0    11
1    20
2    31
3    40
4    51
5    60
6    71
7    80
8    91
dtype: int64
```

Pandas DataFrames

Grow Data Skills

Dataframe is a 2-Dimensional labelled data-structure

	Name	Age	City
0	John	25	New York
1	Emma	30	London
2	Peter	35	Paris
3	Olivia	28	Sydney

A Dataframe comprises of rows and columns

Creating a Dataframe

Grow Data Skills

Creating Dataframe from list

```
import pandas as pd
data = [1,2,3,4,5]
df = pd.DataFrame(data)
print(df)
```

```
   0
0  1
1  2
2  3
3  4
4  5
```

Creating Dataframe from Dictionary

```
# Create a DataFrame from a dictionary
data = {'Name': ['John', 'Emma', 'Peter', 'Olivia'],
        'Age': [25, 30, 35, 28],
        'City': ['New York', 'London', 'Paris', 'Sydney']}
df = pd.DataFrame(data)

# Print the DataFrame
print(df)
```

```
     Name  Age      City
0    John   25  New York
1    Emma   30    London
2   Peter   35     Paris
3  Olivia   28    Sydney
```

Dataframe In-Built functions

Grow Data Skills

len()

head()

describe()

tail()

dtypes()

shape()

loc[]


```python
import pandas as pd

data = {'Name': ['John', 'Emma', 'Peter', 'Olivia'],
        'Age': [25, 30, 35, 28],
        'City': ['New York', 'London', 'Paris', 'Sydney']}
df = pd.DataFrame(data)

print(df)
```

```
     Name  Age      City
0    John   25  New York
1    Emma   30    London
2   Peter   35     Paris
3  Olivia   28    Sydney
```

```python
print(df.loc[0:3,("Name","Age")])
```

```
     Name  Age
0    John   25
1    Emma   30
2   Peter   35
3  Olivia   28
```

iloc[]


```
import pandas as pd

data = {'Name': ['John', 'Emma', 'Peter', 'Olivia'],
        'Age': [25, 30, 35, 28],
        'City': ['New York', 'London', 'Paris', 'Sydney']}
df = pd.DataFrame(data)

print(df)
```

```
     Name  Age      City
0    John   25  New York
1    Emma   30    London
2   Peter   35     Paris
3  Olivia   28    Sydney
```

```
print(df.iloc[0:3,0:2])
```

```
    Name  Age
0   John   25
1   Emma   30
2  Peter   35
```