* Methods :- A class or types methods are functions that every instance of that class or type provides
- It's how you interact with the data in a object.
- Sorting is an eg of method that interacts with the data in the object.

* Creating your own types :-

Defining classes

class Circle

↓

Data Attributes :
radius, color

class Rectangle

↓

Data Attributes :-
colors height & width

Name of class

↑

class Circle(object):

class def^n

↓

class parent

constructor parameters

```
def __init__(self, radius, color):
    self.radius = radius;
    self.color = color;
```

The self parameter refers to the newly created instance of the class.

How to create an object of class Circle :

```
RedCircle = Circle(10, "red")
```

RedCircle : Name of Object

Circle(10, "red") : object constructor

Circle : Name of Class

Methods are functions that interact and change the data attributes

```
def add_radius (self, r):
    self.radius = self.radius + r
```

} method used to add r to radius

dir (Nameofobject) → The dir function is useful for obtaining the list of data attributes and methods associated with a class.

↓

The return value is a list of the objects data attributes

⇒ Working with Data in Python

* Reading files with Open

```
File1 = open ("/resources/data/Ex2.txt", "w")
```

File1 — file object; open — open function; "/resources/data/Ex2.txt" — file path; "w" — mode

common mode :- 'r' for reading
'w' for writing
'a' for appending

File1.name → to obtain name of the file

File1.mode → (data attribute mode) what mode the object is in using

File1.close() → close the file object using the method close

using with → because it automatically closes the file.

```
with open ("Example1.txt", "r") as File1:
    file_stuff = File1.read()
    print (File1.close file_stuff )
print (File1.closed)
print (File_stuff)
```

Indent block

→ stores the values of file in variable stuff as string

File_stuff = File1.readlines()

readline

↓

To read the first line of the file

Loop :-

```
with open ("Example1.txt","r") as File1:
    for line in File1:
        print (line)
```

Writing Files with open

Method :- .write ("This is line 1")

↓

to write data to that file

File1 = open ("/resources/data/Example2.txt", "w")

open function — File path

*

```
with open ("/resources/ data/ Eg 2.txt","w") as File1:
    File1.write ("This is Line A")
```

{ }

```
for line in Lines:
    File1.write (line)
```

a → append

* Copy one file to new file as follows :-

```
with open ("Example1.txt", "r") as readfile:
    with open ("Example3.txt", "w") as writeFile:
        for line in readfile
            writefile.write(line)
```

Loading Data with Pandas

csv is a typical file type used to store data

Importing Pandas :-

```
import pandas
csv_path = 'file1.csv'
df = pandas.read_csv (csv_path)
df.head()
```

This variable stores the path of the csv

To examine the first 5 rows of a dataframe

For Excel file :-

```
xlsx_path = 'file1.xlsx'
df = pd.read_excel (xlsx_path)
df.head()
```

key :- column labels

values :- corresponding to the rows

Working with and saving Data

unique :- determine the unique elements in a column of a dataframe

eg. df['Released'].unique()

df['Released'] >= 1980 → Result will be boolean True or false

df1 = df[df['Released'] >= 1980]

Save as CSV

df1.to_csv('new_songs.csv')

name of the file

One Dimensional Numpy :-

library for scientific computing

Numpy → basis for pandas

Basics and Array Creation

Numpy array → similar to list

each element is of same type and fixed length

import numpy as np

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

a[0] : 0

a[1] : 1

a : array([0, 1, 2, 3, 4])

type(a) : numpy.ndarray

a.dtype : dtype('int64') → to obtain data type of the array's element

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

1	2	3	4	5

a.size ⟶ The no. of elements in the array
→ 5

a.ndim : 1 ⟶ The no. of array dimensions or the rank of the array

a.shape : (5,) ⟶ shape is a tuple of

* Indexing and slicing :-

```
c = np.array([20,1,2,3,4])
c : array([20,1,2,3, 4])
c[0] = 100
c : array([100,1,2,3, 4])
```

slice :- `d = c[1:4]`

```
d : array([1,2,3])
c[3:5] = 300,400
c : array([100,1,2,300,400])
```

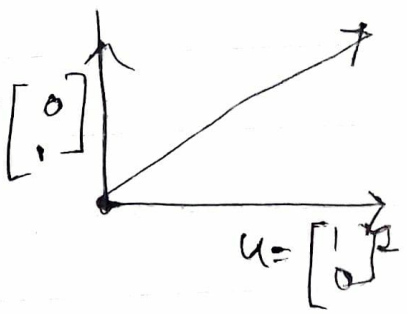
* Basic Operations

Vector addition and subtraction

$$u = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad v = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$z = u + v = \begin{bmatrix} 1+0 \\ 0+1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

```
u = [1,0]
v = [1,0]
z = [ ]
for n, m in zip(u, v):
    z.append(n+m)
```



```
u = np.array([1,0])
v = np.array([0,1])
z = u+v
z: array([1,1])
```

```
u = [1,0]
v = [0,1]
z = [0]
for n,m in zip(u,v):
    z.append(n+m)
```

scalar vector multiplication

```
y = np.array([1,2])
z = 2*y
z: array([2,4])
```

---X---

```
y = [1,2]
z = []
for n in y:
    z.append(2*n)
```

Product of two numpy arrays

→ Hadamard product

$$u = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad v = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \quad z = u \cdot v = \begin{bmatrix} 1*3 \\ 2*2 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

```
u = np.array([1,2])
v = np.array([3,2])
z = u*v
z: array([3,4])
```

```
u = [1,2]
v = [3,2]
z = []
for n,m in zip(u,v):
    z.append(n*m)
```

Dot product :-

$$u = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad v = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \quad u^T \cdot v = 1 \times 3 + 2 \times 1 = 5$$

```
u = np.array([1,2])
v = np.array([3,1])
result = np.dot(u,v)
result :5
```

Adding constant to an numpy Array

```
u = np.array([1,2,3,-1])
z = u+1
```

1, 2, 3, -1 ⇓ 1+1, 2+1, 3+1, -1+1

```
z: array([2,3,4,0])
```

→ broadcasting

* Universal Functions → that operates on ND arrays

```
a = np.array([1,-1,1,-1])
mean_a = a.mean()
mean_a: 0
b = np.array([1,2,3,4,5])
max_b = b.max()
max_b: 5
```

Q. np.pi → π

Q. x = np.array([0, np.pi/2, np.pi])

→ $x = [0, \frac{\pi}{2}, \pi]$

Q. y = np.sin(x)

→ $y = [\sin 0, \sin\frac{\pi}{2}, \sin(\pi)]$

```
np.linspace(-2, 2, num=5)
```

num: no. of samples to generate

Line space returns evenly spaced numbers over specified interval

* Plotting Mathematical Functions :-

```
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
import matplotlib.pyplot as plt
% matplotlib inline
plt.plot(x,y)
```

Two dimensional Numpy

a = [[11, 12, 13], [21, 22, 23], [31, 32, 33]]

A = np.array(a) → axis 1

A : $\begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix}$ ↓ axis 0

A.ndim : 2 ⟶ the no. of axes or dimensions

A.shape : (3,3) referred to as the rank

A.size : 9

```
X = np.array([[1,0],[0,1]])
Y = np.array([[2,1],[1,2]])
Z = X+Y;
Z : array([[3,1],[1,3]])
```

$X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $Y = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$

$X \cdot Y = \begin{bmatrix} (1)(2) & (0)1 \\ (0)1 & (1)2 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

```
X = np.array([[1,0],[0,1]])
Y = np.array([[2,1],[1,2]])
Z = X*Y;
```