S.P Group of Institute

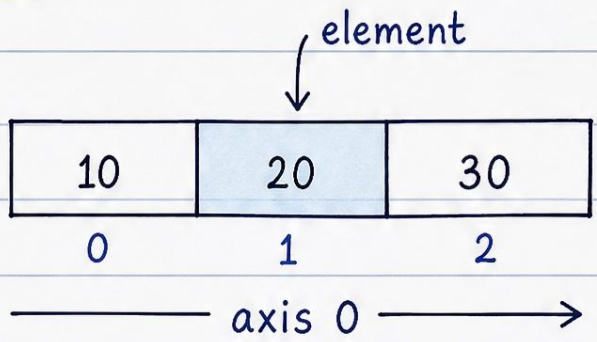
Programming and Problem Solving Through Python Language Notes

Unit Name: Unit 9: NumPy Basics

Introduction to NumPy ndarray

NumPy Python की numerical computing library है। ndarray इसका मुख्य N-dimensional array object है, जो समान प्रकार के elements को व्यवस्थित रूप से store करता है।

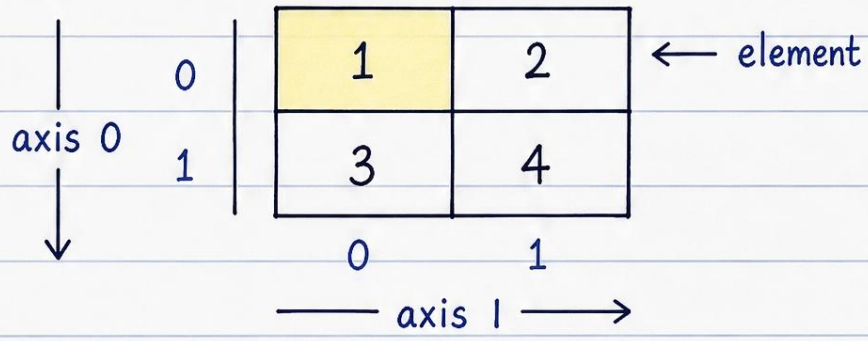
1-D array [10, 20, 30]



ndarray = homogeneous, fast, memory-efficient array

shape: (3,)

2-D array [[1, 2], [3, 4]]



shape: (2, 2)

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

Unit Name: Unit 9: NumPy Basics

NumPy datatypes

dtype array के elements का data type बताता है।
सभी elements सामान्यत: एक ही dtype में store होते है।

dtype	अर्थ
int32 (123)	पूर्णांक
float64 (3.14)	दशमलव संख्या
complex128 (a+bi)	complex number
bool_ (✓ ✗)	True / False
str_ ("abc")	text

उदाहरण (Example):

```
① np.array([1, 2, 3], dtype=np.int32)
② a.dtype
```

कैसे काम करता है? (Flow)

Python value	→	dtype	→	memory storage
10, 3.14, True, "hello" आदि		value का data type निर्धारित करता है		dtype के अनुसार memory में store होता है

★ dtype सही चुनने से memory और performance बेहतर होती है।

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

Unit Name: Unit 9: NumPy Basics

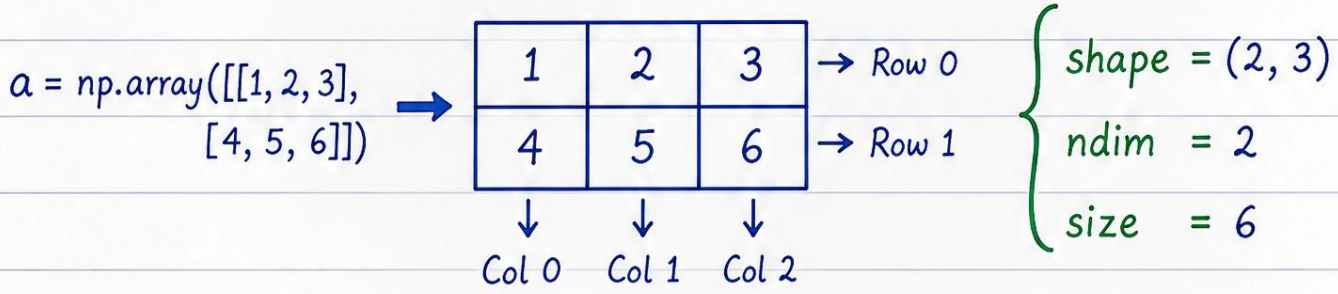
Array attributes

NumPy array एक समान data type के elements का संग्रह होता है, जिसमें हर array के साथ कुछ useful जानकारी (attributes) होती है। इन attributes की मदद से हम array की संरचना, आकार, data type और memory से संबंधित जानकारी प्राप्त कर सकते हैं।

उदाहरण:

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

1. ndim — dimensions की संख्या
2. shape — प्रत्येक dimension का size
3. size — कुल elements की संख्या
4. dtype — elements का data type
5. itemsize — एक element के bytes
6. nbytes — array की कुल memory



उदाहरण: (Python Code)

```
1. a.ndim      # array के dimensions की संख्या देता है
2. a.shape     # प्रत्येक dimension का size (tuple) देता है
3. a.size      # array में कुल elements की संख्या देता है
4. a.dtype     # array के elements का data type देता है
```

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

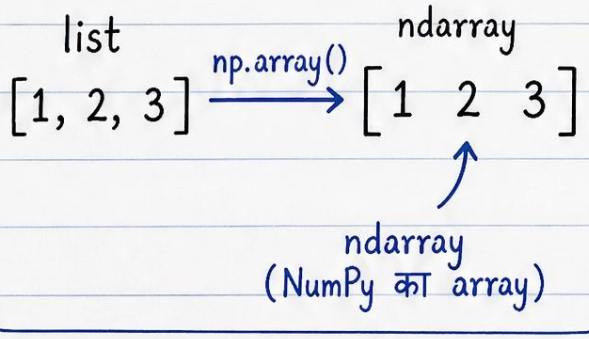
Unit Name: Unit 9: NumPy Basics

Array creation routines

NumPy में arrays बनाने के लिए कई ready-made routines उपलब्ध हैं।

`np.array([1, 2, 3])`	→ list से array
`np.zeros((2, 3))`	→ सभी values 0
`np.ones((2, 2))`	→ सभी values 1
`np.full((2, 3), 7)`	→ सभी values 7
`np.eye(3)`	→ identity matrix
`np.empty((2, 2))`	→ uninitialized array

list → np.array → ndarray



np.eye(3) – identity matrix

np.eye(3) का output:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

Unit Name: Unit 9: NumPy Basics

Array From Existing Data

पहले से उपलब्ध data जैसे list, tuple या दूसरे array से नया NumPy array बनाया जा सकता है।

उदाहरण (Examples):

- `np.array([1, 2, 3])`
- `np.asarray((4, 5, 6))`
- `np.frombuffer(buffer, dtype=np.int32)`
- `np.fromiter([1, 2, 3], dtype=int)`

list / tuple / buffer / iterator → NumPy array

⚠ `np.asarray()` अनावश्यक copy से बच सकता है।

★ `np.array()` सामान्यत: नया array बनाता है; `np.asarray()` input पहले से array हो तो view दे सकता है।

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

Unit Name: Unit 9: NumPy Basics

Array From Numerical Ranges

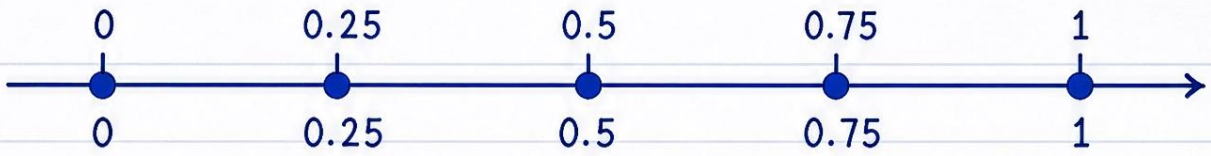
➡ Numerical range से लगातार numbers का array आसानी से बनाया जा सकता है।

★ Examples:

- `np.arange(0, 10, 2)` — 0 से 10 से पहले तक, step 2
- `np.linspace(0, 1, 5)` — 0 और 1 के बीच 5 equally spaced values
- `np.logspace(1, 3, 3)` — logarithmic scale पर values

`np.arange(0, 10, 2)` → `[0, 2, 4, 6, 8]`

`np.linspace(0, 1, 5)` → `[0. , 0.25, 0.5 , 0.75, 1. ]`



Note: ★ arange में stop value सामान्यतः शामिल नहीं होती; linspace में endpoint default रूप से शामिल होता है।

S.P Group of Institute

Programming and Problem Solving Through Python Language Notes

Unit Name: Unit 9: NumPy Basics

Indexing & Slicing

★ Indexing से किसी एक element को access करते हैं।
Python और NumPy में indexing 0 से शुरू होती हैं।

★ Slicing से array का एक भाग प्राप्त किया जाता हैं।

1-D Array (1-आयामी)

Index	0	1	2	3	4
Value	10	20	30	40	50

↑
Index 2 (Value = 30)

Slicing का स्वरूप

start : stop : step

↓ ↓ ↓

शुरूआत (inclusive) | समाप्ति (exclusive) | कदम (वैकल्पिक)

1-D Examples

- a = np.array([10, 20, 30, 40, 50])
- a[0] = 10
- a[-1] = 50
- a[1:4] = [20, 30, 40]
- a[:3] = [10, 20, 30]
- a[::2] = [10, 30, 50]

⚠ ध्यान दें!

stop index शामिल नहीं होता।

2-D Array (2-आयामी)

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

- b[1, 2] = 6
- b[:, 1] = [2, 5]

	0	1	2
0	1	2	3
1	4	5	6

(Column Index)

1 ↑ (Row Index)

↑ b[1, 2] = 6