Unitedworld Institute of Technology

Data Science

By:

Dr. Shikha Deep

Unitedworld Institute of Technology

Contents

- Data Types
- Arrays
- Library : Numpy
- Pandas
- Data Frame

KARNAVATI UNIVERSITY Unitedworld Institute of Technology

Data Types

KARNAVATI UNIVERSITY Unitedworld Institute of Technology

Data Types

Data types are the classification or categorization of data items. It represents the kind of value a variable holds.

- Zelle, J. (2016). Python Programming: An Introduction to Computer Science (3rd ed.)

Data types tell Python what kind of data a variable holds - **numbers, text, boolean**, etc. Knowing the type helps Python decide what operations can be performed.

Unitedworld Institute of Technology

Basic Data Types

Type	Description	Example
int	Whole numbers	x = 10
float	Decimal numbers	y = 3.14
str	Text (string)	name = "Shikha"
bool	Boolean (True, False)	status = True

complex	Complex numbers (real + imag)	z = 2 + 3j

NOTE: type(value)

Code:

```
print(type(5))
print(type([1, 2, 3]))
print(type({"a": 1}))
```

Output:

```
<class 'int'>
<class 'list'>
```

<class 'dict’>

Array

KARNAVATI UNIVERSITY

Unitedworld Institute of Technology

Array

An array is a collection of elements, all of the same type, arranged in a contiguous block of memory and accessed by an index.

- R.L. Kruse, Data Structures and Program Design in C

(adapted for Python concept)

- In Python, an array is a data structure that stores multiple values of the same data type in a single variable.
- Like – integers, decimal, complex, strings, Boolean etc.

When to Use Array in Python?

- When you're dealing with many numbers of same type (e.g., integers or floats)
- When you want better performance and lower memory usage
- When doing scientific computing, we use NumPy

Python Array using array module

```
import array
# create an array of integers
marks = array.array('i', [80, 85, 90, 95])

# access elements
print(marks[0])
print(marks[2])
```

OUTPUT:

Format

array.array(typecode, [elements])

Typecode	Meaning
'i'	Integer
'f'	Float
'u'	Unicode char (rare)

Use NumPy Arrays (Instead of Lists):

Feature	Python List	NumPy Array (np.array)
Data type consistency	Can hold mixed types	All elements must be of same type
Performance	Slower for large data	Much faster for computation
Memory usage	Higher	Lower
Math operations	Not direct (for loops needed)	Vectorized (e.g., a + b works directly)
Used in libraries	Not used in ML/DS libraries	Used in pandas, sklearn, TensorFlow, etc.

How to Import NumPy

```
import numpy as np
```

KARNAVATI UNIVERSITY

Unitedworld Institute of Technology

EXAMPLES:

Example 1: Create numpy

```
import numpy as np
arr = np.array([85, 90, 78, 92])
print(arr)
```

Output:

[85 90 78 92]

Unitedworld Institute of Technology

Example 2: Add 5 marks to all students (Basic Operations)

```
print(arr + 5)
```

Output:

[90 95 83 97]

Example 3: Mean, Max, Min, Shape

```
print("Mean:", arr.mean())
print("Max:", arr.max())
print("Min:", arr.min())
print("Shape:", arr.shape)
```

OUTPUT:

Mean: 86.25

Max: 92

Min: 78

pe: (4,)

Array Methods

	Adds an element at the end of the list
	Removes all the elements from the list
	Returns a copy of the list
	Returns the number of elements with the specified value
	Add the elements of a list (or any iterable), to the end of the current list
	Returns the index of the first element with the specified value
	Adds an element at the specified position
	Removes the element at the specified position
	Removes the first item with the specified value

	Reverses the order of the list
	Sorts the list

Unitedworld Institute of Technology

Task to do

- Create a NumPy array of 5 subject marks.
- Add 10 bonus marks to all using + operator.
- Find average and max using mean() and max().
- Convert a Python list to NumPy array.

Unitedworld Institute of Technology

N-dimensional array

A dimension refers to the number of directions in which you can index or access elements in an array.

Dimension	Meaning	Example
1D	A single list of items (like a line)	[1, 2, 3]
2D	A list of lists (like rows and columns)	[[1, 2], [3, 4]]
3D+	An array with layers (like a cube)	[[[1, 2], [3, 4]], [[5, 6], [7, 8]]]

1 D array:

A 1D array is a linear collection of elements of the same data type, stored in contiguous memory locations.

2 D array:

A two-dimensional array (2D array) is a collection of elements arranged in rows and columns, forming a matrix-like structure.

3 D array:

A 3D array in Python is a data structure that organizes elements in three dimensions: rows, columns, and depth. It can be visualized as a cube or a stack of 2D matrices layered on top of each other.

Examples:

Ex 1:

```
import numpy as np
```

```
arr1d = np.array([10, 20, 30, 40])
print(arr1d)
print("Shape:", arr1d.shape)
print("Dimensions:", arr1d.ndim)
```

OUTPUT:

```
[10 20 30 40]
Shape: (4,)
Dimensions: 1
```

Ex 2:

Table of marks of 3 students in 3 subjects:

```
[ [85, 90, 88], ← Student 1
  [78, 80, 85], ← Student 2
```

[92, 94, 96]] ← Student 3

CODE:	OUTPUT
marks = np.array([[85, 90, 88], [78, 80, 85], [92, 94, 96]]) print("2D Array - Student Marks:") print(marks) print("Shape:", marks.shape)	2D Array - Student Marks: [[85 90 88] [78 80 85] [92 94 96]] Shape: (3, 3)

Ex 3: Seating Chart in a Classroom

Imagine 2 rows and 3 columns of seats labeled as seat numbers. seats = np.array([

[101, 102, 103], # Row 0 (1st row)

```
 [104, 105, 106] # Row 1 (2nd row)
])
 # Col 0, Col 1, Col 2
print("Seat:", seats[1][2])
```

OUTPUT:

?? 2nd Row, 3rd seat

Unitedworld Institute of Technology

General view of 3D

Shape = (layers, rows, columns)

= (depth, height, width)

= (semester, student, subject)


Unitedworld Institute of Technology

Ex 4 :

Student Marks in 2 Semesters (3 Students × 2 Subjects)

Semester	Student	Subjects (Math, English)
1	3 rows	2 columns
2	3 rows	2 columns


Unitedworld Institute of Technology import numpy as np

```
marks = np.array([
 [[85, 90], [78, 82], [92, 88]], # Semester 1
```

```
 [[86, 91], [80, 83], [94, 89]] # Semester 2
])

print("Shape:", marks.shape)
print("Marks of Student 2 in Semester 2, Subject 1:", marks[1][1][0])
```

OUTPUT:

```
Shape: (2, 3, 2)
Marks of Student 2 in Semester 2, Subject 1: 80
```

NumPy

- NumPy is the fundamental package for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

- Harris et al. (2020), Nature: Array Programming with NumPy.

- **NumPy** stands for **Numerical Python**.
- It gives us fast, powerful **arrays**, **math functions**, and **matrix operations** — all faster than regular Python lists.

Unitedworld Institute of Technology

Why Use NumPy?

Feature	Benefit
Fast array operations	Much faster than Python lists
Built-in math/statistics	mean(), sum(), std()
Works with ML/Data Science	Required by pandas, sklearn, TensorFlow, etc.
Handles matrices and vectors	Easily supports dot product, transpose, etc.
Less memory	More compact storage than lists

How to Use NumPy?

Step 1: Install

pip install numpy

Step 2: Import in Python

import numpy as np

Unitedworld Institute of Technology

Pandas

Unitedworld Institute of Technology

Pandas

- Pandas is a Python library used for data manipulation and analysis. It provides powerful, easy-to-use data structures like Series and DataFrame.

- McKinney, W. (2012). Python for Data Analysis (O'Reilly Media)

- A Pandas DataFrame is a two-dimensional table-like structure in Python where data is arranged in rows and columns. It's one of the most commonly used tools for handling data and makes it easy to organize, analyze and manipulate data.

Two Key Data Structures in Pandas:

1) **Series**

A 1D labeled array (like a column in Excel)

CODE:

```
import pandas as pd
s = pd.Series([10, 20, 30])
print(s)
```

OUTPUT:

0 10

1 20

2 30

dtype: int64

(2) **DataFrame**

• A 2D labeled table (like an entire Excel sheet)

• Rows and columns are labeled

CODE:

```
data = {
 "Name": ["Ravi", "Anu", "Riya"],
 "Marks": [85, 92, 78]
```

```
}
df = pd.DataFrame(data) #load data into a DataFrame object:
print(df)
```

OUTPUT:

```
  Name Marks
0 Ravi 85
1 Anu 92
2 Riya 78
```

Creating Data Frames

(1) From Dictionary

```
data = {
```

```
 "Student": ["Aman", "Zara", "Nikhil"],
 "CGPA": [8.2, 9.1, 7.5]
}
df = pd.DataFrame(data)
```

OUTPUT

```
 Student CGPA
0 Aman 8.2
1 Zara 9.1
2 Nikhil 7.5
```

(2) From Lists of Lists

```
data = [["Aman", 8.2], ["Zara", 9.1], ["Nikhil", 7.5]] df =
pd.DataFrame(data, columns=["Student", "CGPA"])
```

(3) From CSV

df = pd.read_csv("data.csv")

Unitedworld Institute of Technology

❖ Locate Row :

- Pandas use the loc attribute to return one or more specified row(s)

CODE:

```
print(df.loc[0]) #refer to the row index:
```

OUTPUT:

Student Aman

CGPA 8.2

Name: 0, dtype: int64

❖ Return row 0 and 1:

CODE:

```
#use a list of indexes:
print(df.loc[[0, 1]])
```

OUTPUT:

	Students	CGPA
0	Aman	8.2
1	Zara	9.1

❖ Named Indexes

With the index argument, you can name your own indexes.

CODE:

```
import pandas as pd
data = {
 “Students": [Aman, Zara, Nikhil],
 “CGPA": [8.2, 9.1, 7.5]
}
df = pd.DataFrame(data, index = ["day1", "day2",
"day3"]) print(df)
```

OUTPUT:

Student CGPA
day1 Aman 8.2
day2 Zara 9.1
day3 Nikhil 7.5

 ❖ Locate Named Indexes

Return "day2":

CODE:

```
#refer to the named index:
print(df.loc["day2"])
```

OUTPUT:

Students Zara

CGPA 9.1

Name: day2, dtype: int64

CODE: Descending Order

```
df.sort_values(by="CGPA", ascending=False)
```

OUTPUT:

Student College_ID CGPA Internship Extra_Curricular Placed

2 Zara CLG001 9.1 Yes 8 Yes 0 Riya CLG001 8.8 Yes 9 Yes 4

Tanu CLG001 8.2 Yes 7 Yes 5 Dev CLG002 7.9 Yes 6 Yes 1

Kunal CLG002 7.4 No 5 No 3 Aman CLG003 6.5 No 6 No

Unitedworld Institute of Technology