Darshan UNIVERSITY
योगः कर्मसु कौशलम्

# Roll No - 130

# Enrollment No - 24010101694

# Lab - 2

Karan Sonagara

---

## Numpy

1. NumPy (Numerical Python) is a powerful open-source library in Python used for numerical and scientific computing.
2. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on them efficiently.
3. NumPy is highly optimized and written in C, making it much faster than using regular Python lists for numerical operations.
4. It serves as the foundation for many other Python libraries in data science and machine learning, like pandas, TensorFlow, and scikit-learn.
5. With features like broadcasting, vectorization, and integration with C/C++ code, NumPy allows for cleaner and faster code in numerical computations.

### Step 1. Import the Numpy library

In [1]:
```
import numpy as np
```

### Step 2. Create a 1D array of numbers

In [19]:
```
arr = np.array([1, 2, 3])
arr
```

Out[19]:
```
array([1, 2, 3])
```

## Step 3. Reshape 1D to 2D Array

In [11]:
```
arr3 = np.arange(12).reshape(3, 4)
arr3
```

Out[11]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

## Step 4. Create a Linspace array

In [7]:
```
arr4 = np.linspace(1, 1, 5)
arr4
```

Out[7]:
```
array([1., 1., 1., 1., 1.])
```

## Step 5. Create a Random Numbered Array

In [9]:
```
arr5 = np.random.rand(3, 4)
arr5
```

Out[9]:
```
array([[0.17642223, 0.4339188 , 0.76623246, 0.24111004],
       [0.99867265, 0.65568381, 0.74620793, 0.42338742],
       [0.59718242, 0.44156543, 0.22621162, 0.85787908]])
```

## Step 6. Create a Random Integer Array

In [27]:
```
arr6 = np.random.randint(2, 10, 10)
arr6
```

Out[27]:
```
array([5, 4, 8, 7, 3, 3, 5, 9, 2, 7])
```

## Step 7. Create a 1D Array and get Max,Min,ArgMax,ArgMin

In [29]:
```
arr6.argmin()
```

Out[29]:
```
8
```

In [31]:
```
arr6.min()
```

Out[31]:
```
2
```

In [33]:
```
arr6.max()
```

Out[33]:
```
9
```

In [35]:
```
arr6.argmax()
```

Out[35]:
```
7
```

## Step 8. Indexing in 1D Array

In [53]:
```
arr8 = np.array([1, 2, 3, 4, 5, 6 ,7, 8, 9 , 10])
arr8[:7:2]
```

Out[53]:
```
array([1, 3, 5, 7])
```

## Step 9. Indexing in 2D Array

In [71]:
```
arr9 = np.arange(12).reshape(3, 4)
arr9
```

Out[71]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [79]:
```
arr9[::2]
```

Out[79]:
```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11]])
```

In [106…
```
arr9
```

Out[106…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [122…
```
arr9[:3:2]
```

Out[122…
```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11]])
```

## Step 10. Conditional Selection

In [130…
```
arr3 = np.arange(100)
arr3[arr3>50]
```

Out[130…
```
array([51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

## 🔥 You did it! 10 exercises down — you're on fire! 🔥

# Pandas

## Step 1. Import the necessary libraries

In [158…
```
import pandas as pd
```

## Step 2. Import the dataset from this address.

## Step 3. Assign it to a variable called users and use the 'user_id' as index

In [160…
```
users = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master/d
users
```

Out[160…

| user_id | age | gender | occupation | zip_code |
|---|---|---|---|---|
| **1** | 24 | M | technician | 85711 |
| **2** | 53 | F | other | 94043 |
| **3** | 23 | M | writer | 32067 |
| **4** | 24 | M | technician | 43537 |
| **5** | 33 | F | other | 15213 |
| **...** | ... | ... | ... | ... |
| **939** | 26 | F | student | 33319 |
| **940** | 32 | M | administrator | 02215 |
| **941** | 20 | M | student | 97229 |
| **942** | 48 | F | librarian | 78209 |
| **943** | 22 | M | student | 77841 |

943 rows × 4 columns

## Step 4. See the first 25 entries

In [166…
```
df = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master/data
df.head(10)
```

Out[166…

| | user_id | age | gender | occupation | zip_code |
|---|---|---|---|---|---|
| **0** | 1 | 24 | M | technician | 85711 |
| **1** | 2 | 53 | F | other | 94043 |
| **2** | 3 | 23 | M | writer | 32067 |
| **3** | 4 | 24 | M | technician | 43537 |
| **4** | 5 | 33 | F | other | 15213 |
| **5** | 6 | 42 | M | executive | 98101 |
| **6** | 7 | 57 | M | administrator | 91344 |
| **7** | 8 | 36 | M | administrator | 05201 |
| **8** | 9 | 29 | M | student | 01002 |
| **9** | 10 | 53 | M | lawyer | 90703 |

## Step 5. See the last 10 entries

In [170…

```
df = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master/data
df.tail(10)
```

Out[170…

| | user_id | age | gender | occupation | zip_code |
|---|---|---|---|---|---|
| **933** | 934 | 61 | M | engineer | 22902 |
| **934** | 935 | 42 | M | doctor | 66221 |
| **935** | 936 | 24 | M | other | 32789 |
| **936** | 937 | 48 | M | educator | 98072 |
| **937** | 938 | 38 | F | technician | 55038 |
| **938** | 939 | 26 | F | student | 33319 |
| **939** | 940 | 32 | M | administrator | 02215 |
| **940** | 941 | 20 | M | student | 97229 |
| **941** | 942 | 48 | F | librarian | 78209 |
| **942** | 943 | 22 | M | student | 77841 |

## Step 6. What is the number of observations in the dataset?

In [172…

```
users.shape
```

Out[172… (943, 4)

## Step 7. What is the number of columns in the dataset?

In [225…

```
users.count()
```

Out[225…
```
age           943
gender        943
occupation    943
zip_code      943
dtype: int64
```

## Step 8. Print the name of all the columns.

In [178…
```
df = pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master/data
df.columns
```

Out[178…
```
Index(['user_id', 'age', 'gender', 'occupation', 'zip_code'], dtype='object')
```

## Step 9. How is the dataset indexed?

In [182…
```
users.index
```

Out[182…
```
Index([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,
       ...
       934, 935, 936, 937, 938, 939, 940, 941, 942, 943],
      dtype='int64', name='user_id', length=943)
```

## Step 10. What is the data type of each column?

In [184…
```
users.dtypes
```

Out[184…
```
age            int64
gender        object
occupation    object
zip_code      object
dtype: object
```

## Step 11. Print only the occupation column

In [193…
```
users.occupation
users['occupation']
```

Out[193…
```
user_id
1         technician
2              other
3             writer
4         technician
5              other
           ...
939          student
940    administrator
941          student
942        librarian
943          student
Name: occupation, Length: 943, dtype: object
```

## Step 12. How many different occupations are in this dataset?

In [202…
```
users['occupation'].nunique()
```

```
users['occupation'].unique()
```

Out[202…
```
array(['technician', 'other', 'writer', 'executive', 'administrator',
       'student', 'lawyer', 'educator', 'scientist', 'entertainment',
       'programmer', 'librarian', 'homemaker', 'artist', 'engineer',
       'marketing', 'none', 'healthcare', 'retired', 'salesman', 'doctor'],
      dtype=object)
```

## Step 13. What is the most frequent occupation?

In [208…
```
users.occupation.value_counts().head(1)
```

Out[208…
```
occupation
student    196
Name: count, dtype: int64
```

## Step 14. Summarize the DataFrame.

In [210…
```
users.describe()
```

Out[210…

| | age |
|---|---|
| **count** | 943.000000 |
| **mean** | 34.051962 |
| **std** | 12.192740 |
| **min** | 7.000000 |
| **25%** | 25.000000 |
| **50%** | 31.000000 |
| **75%** | 43.000000 |
| **max** | 73.000000 |

## Step 15. Summarize all the columns

In [213…
```
users.describe(include='all')
```

Out[213...

| | age | gender | occupation | zip_code |
|---|---|---|---|---|
| **count** | 943.000000 | 943 | 943 | 943 |
| **unique** | NaN | 2 | 21 | 795 |
| **top** | NaN | M | student | 55414 |
| **freq** | NaN | 670 | 196 | 9 |
| **mean** | 34.051962 | NaN | NaN | NaN |
| **std** | 12.192740 | NaN | NaN | NaN |
| **min** | 7.000000 | NaN | NaN | NaN |
| **25%** | 25.000000 | NaN | NaN | NaN |
| **50%** | 31.000000 | NaN | NaN | NaN |
| **75%** | 43.000000 | NaN | NaN | NaN |
| **max** | 73.000000 | NaN | NaN | NaN |

## Step 16. Summarize only the occupation column

In [215...
```
users.occupation.describe()
```

Out[215...
```
count          943
unique          21
top        student
freq           196
Name: occupation, dtype: object
```

## Step 17. What is the mean age of users?

In [219...
```
users.age.mean()
```

Out[219...
```
34.05196182396607
```

## Step 18. What is the age with least occurrence?

In [221...
```
users.age.value_counts().tail()
```

Out[221...
```
age
7     1
66    1
11    1
10    1
73    1
Name: count, dtype: int64
```

## You're not just learning, you're mastering it. Keep aiming higher! 🚀