Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Data Mining

Lab - 2

Raj Vekariya | 23010101298

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 [6]:
```
a=np.arange(11)
print(a)
print(type(a))
```

```
[ 0  1  2  3  4  5  6  7  8  9 10]
<class 'numpy.ndarray'>
```

In [9]:
```
a=np.arange(2,9)
a
```

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

Step 3. Reshape 1D to 2D Array

In [13]:
```
a=np.arange(12).reshape(3,4)
a
```

Out[13]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

Step 4. Create a Linspace array

In [14]:
```
np.linspace(0,5,20)
```

Out[14]:
```
array([0.        , 0.26315789, 0.52631579, 0.78947368, 1.05263158,
       1.31578947, 1.57894737, 1.84210526, 2.10526316, 2.36842105,
       2.63157895, 2.89473684, 3.15789474, 3.42105263, 3.68421053,
       3.94736842, 4.21052632, 4.47368421, 4.73684211, 5.        ])
```

Step 5. Create a Random Numbered Array

In [15]:
```
np.random.rand(2)
```

Out[15]:
```
array([0.7406834 , 0.56917623])
```

In [16]:
```
np.random.rand(2,4)
```

Out[16]:
```
array([[0.56916381, 0.52930315, 0.95050776, 0.39259408],
       [0.82080692, 0.18362901, 0.88810763, 0.47867718]])
```

Step 6. Create a Random Integer Array

In [21]:
```
np.random.randint(1,100,size=10)
```

Out[21]:
```
array([94,  4, 99, 55, 51,  3, 49, 89, 26, 14])
```

In [23]:
```
np.random.randint(1,100,size=(2,4))
```

Out[23]:
```
array([[39, 27, 96, 78],
       [19, 66, 22, 47]])
```

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

In [25]:
```
arr=np.random.randint(1,100,size=10)
arr
```

Out[25]:
```
array([ 2,  6, 14, 88, 39, 59, 46, 81, 16, 31])
```

In [26]:
```
arr.min()
```

Out[26]:
```
2
```

In [27]:
```
arr.max()
```

Out[27]:
```
88
```

In [28]:
```
arr.argmax()
```

Out[28]:
```
3
```

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

Out[29]:
```
0
```

Step 8. Indexing in 1D Array

In [30]:
```
arr[8]
```

Out[30]:
```
16
```

In [31]:
```
arr[1:5]
```

Out[31]:
```
array([ 6, 14, 88, 39])
```

Step 9. Indexing in 2D Array

In [33]:
```
arr2d=arr.reshape(2,5)
arr2d
```

Out[33]:
```
array([[ 2,  6, 14, 88, 39],
       [59, 46, 81, 16, 31]])
```

In [34]:
```
arr2d[0]
```

Out[34]:
```
array([ 2,  6, 14, 88, 39])
```

In [40]:
```
arr2d[0][1]
arr2d[0,1]
```

Out[40]:
```
6
```

In [41]:
```
arr2d[:1,2:]
```

Out[41]:
```
array([[14, 88, 39]])
```

In [53]:
```
n=np.arange(15).reshape(3,5)
n
```

Out[53]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

In [56]:
```
n[1:,2:]
```

Out[56]:
```
array([[ 7,  8,  9],
       [12, 13, 14]])
```

Step 10. Conditional Selection

In [43]:
```
arr[arr>4]
```

Out[43]:
```
array([ 6, 14, 88, 39, 59, 46, 81, 16, 31])
```

In [44]:
```
arr2d[arr2d>2]
```

Out[44]:
```
array([ 6, 14, 88, 39, 59, 46, 81, 16, 31])
```

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

Pandas

Step 1. Import the necessary libraries

In [2]:
```
import pandas as pd
```

Step 2. Import the dataset from this address.

In [3]:
```
df=pd.read_csv("user.csv",sep="|")
df
```

Out[3]:

	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
...	...	...	...	...	...
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

943 rows × 5 columns

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

In [4]:
```
df.set_index("user_id")
```

Out[4]:

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 [65]:
```
df.head(25)
```

Out[65]:

	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
10	11	39	F	other	30329
11	12	28	F	other	06405
12	13	47	M	educator	29206
13	14	45	M	scientist	55106
14	15	49	F	educator	97301
15	16	21	M	entertainment	10309
16	17	30	M	programmer	06355
17	18	35	F	other	37212
18	19	40	M	librarian	02138
19	20	42	F	homemaker	95660
20	21	26	M	writer	30068
21	22	25	M	writer	40206
22	23	30	F	artist	48197
23	24	21	F	artist	94533
24	25	39	M	engineer	55107

Step 5. See the last 10 entries

In [66]:
```
df.tail(10)
```

Out[66]:

	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 [76]:
```
df.shape[0]
```

Out[76]:
```
943
```

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

In [71]:
```
df.shape[1]
```

Out[71]:
```
5
```

Step 8. Print the name of all the columns.

In [90]:
```
df.columns
```

Out[90]:
```
Index(['user_id', 'age', 'gender', 'occupation', 'zip_code'], dtype='object')
```

Step 9. How is the dataset indexed?

In [8]:
```
df.index
```

Out[8]:
```
RangeIndex(start=0, stop=943, step=1)
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 943 entries, 0 to 942
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   user_id     943 non-null    int64 
 1   age         943 non-null    int64 
 2   gender      943 non-null    object
 3   occupation  943 non-null    object
 4   zip_code    943 non-null    object
dtypes: int64(2), object(3)
memory usage: 37.0+ KB
```

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

In [9]:
```
df.dtypes
```

Out[9]:
```
user_id        int64
age            int64
gender        object
occupation    object
zip_code      object
dtype: object
```

Step 11. Print only the occupation column

In [10]:
```
df['occupation']
```

Out[10]:
```
0         technician
1              other
2             writer
3         technician
4              other
            ...     
938          student
939    administrator
940          student
941        librarian
942          student
Name: occupation, Length: 943, dtype: object
```

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

In [15]:
```
len(set(df['occupation']))
```

Out[15]:
```
21
```

Step 13. What is the most frequent occupation?

In [16]:
```
df['occupation'].mode()
```

Out[16]:
```
0    student
Name: occupation, dtype: object
```

In [17]:
```
df['occupation'].value_counts().idxmax()
```

Out[17]:
```
'student'
```

Step 14. Summarize the DataFrame.

In [12]:
```
df.describe()
```

Out[12]:

	user_id	age
count	943.000000	943.000000
mean	472.000000	34.051962
std	272.364951	12.192740
min	1.000000	7.000000
25%	236.500000	25.000000
50%	472.000000	31.000000
75%	707.500000	43.000000
max	943.000000	73.000000

Step 15. Summarize all the columns

In [27]:
```
df.describe(include='all')
```

Out[27]:

	user_id	age	gender	occupation	zip_code
count	943.000000	943.000000	943	943	943
unique	NaN	NaN	2	21	795
top	NaN	NaN	M	student	55414
freq	NaN	NaN	670	196	9
mean	472.000000	34.051962	NaN	NaN	NaN
std	272.364951	12.192740	NaN	NaN	NaN
min	1.000000	7.000000	NaN	NaN	NaN
25%	236.500000	25.000000	NaN	NaN	NaN
50%	472.000000	31.000000	NaN	NaN	NaN
75%	707.500000	43.000000	NaN	NaN	NaN
max	943.000000	73.000000	NaN	NaN	NaN

Step 16. Summarize only the occupation column

In [19]:
```
df['occupation'].describe()
```

Out[19]:
```
count         943
unique         21
top       student
freq          196
Name: occupation, dtype: object
```

Step 17. What is the mean age of users?

In [20]:
```
df['age'].mean()
```

Out[20]:
```
34.05196182396607
```

Step 18. What is the age with least occurrence?

In [26]:
```
df['age'].value_counts().idxmin()
```

Out[26]:
```
7
```

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

In [ ]:
```

```