

# Data Mining - Lab - 2

## Numpy & Perform Data Exploration with Pandas

140 | Vishal Baraiya | 23010101014

---

### 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 [4]: a = np.arange(11)
print(a)
print(type(a))

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

```
In [6]: a = np.arange(2,9)
print(a)

[2 3 4 5 6 7 8]
```

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

```
In [15]: # error will occurs if it can not fit in 2-D array dimension
a = np.arange(12).reshape(3,4)
a
```

```
Out[15]: array([[ 0,  1,  2,  3],
   [ 4,  5,  6,  7],
   [ 8,  9, 10, 11]])
```

## Step 4. Create a Linspace array

```
In [18]: # ( 0 + 5 ) / 19 = 0.2631578947368421
np.linspace(0,5,20)
```

```
Out[18]: 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 [19]: np.random.rand(2)
```

```
Out[19]: array([0.46224826, 0.3958225 ])
```

```
In [20]: np.random.rand(2,4)
```

```
Out[20]: array([[0.9651745 , 0.57112977, 0.83058656, 0.21649437],
   [0.14483748, 0.25548204, 0.43413858, 0.04114541]])
```

## Step 6. Create a Random Integer Array

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

```
Out[21]: array([83, 99, 30, 66,  8, 41, 28,  6, 26, 60])
```

```
In [22]: np.random.randint(1,100,size=(2,4))
```

```
Out[22]: array([[67, 43, 75, 10],
   [22, 99, 54, 39]])
```

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

```
In [33]: arr = np.random.randint(1,100,size=10)
arr
```

```
Out[33]: array([12, 21, 30, 39, 14, 42, 53, 13, 36, 20])
```

```
In [34]: arr.max()
```

```
Out[34]: 53
```

```
In [35]: arr.min()
```

```
Out[35]: 12
```

```
In [36]: arr.argmax()
```

```
Out[36]: 6
```

```
In [37]: arr.argmin()
```

```
Out[37]: 0
```

```
In [38]: arr.mean()
```

```
Out[38]: 28.0
```

## Step 8. Indexing in 1D Array

```
In [40]: arr[8]
```

```
Out[40]: 36
```

```
In [39]: arr[1:5]
```

```
Out[39]: array([21, 30, 39, 14])
```

## Step 9. Indexing in 2D Array

```
In [46]: arr2d = np.array([
    [101, 102, 103, 104, 105],
    [201, 202, 203, 204, 205],
    [301, 302, 303, 304, 305],
    [401, 402, 403, 404, 405]])
arr2d
```

```
Out[46]: array([[101, 102, 103, 104, 105],
                [201, 202, 203, 204, 205],
                [301, 302, 303, 304, 305],
                [401, 402, 403, 404, 405]])
```

```
In [48]: arr2d
```

```
Out[48]: array([[101, 102, 103, 104, 105],
                [201, 202, 203, 204, 205],
                [301, 302, 303, 304, 305],
                [401, 402, 403, 404, 405]])
```

```
In [49]: arr2d[1::2]
```

```
Out[49]: array([[201, 202, 203, 204, 205],
                [401, 402, 403, 404, 405]])
```

```
In [50]: arr2d[::2,::2]
```

```
Out[50]: array([[101, 103, 105],  
                 [301, 303, 305]])
```

```
In [51]: arr2d[1:3:,1:4:]
```

```
Out[51]: array([[202, 203, 204],  
                 [302, 303, 304]])
```

## Step 10. Conditional Selection

```
In [57]: arr[arr>20]
```

```
Out[57]: array([21, 30, 39, 42, 53, 36])
```

```
In [59]: arr2d[arr2d>120]
```

```
Out[59]: array([201, 202, 203, 204, 205, 301, 302, 303, 304, 305, 401, 402, 403,  
                 404, 405])
```

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

## Pandas

### Step 1. Import the necessary libraries

```
In [60]: 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 [62]: df= pd.read_csv("https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.u  
df
```

Out[62]:

	<b>user_id</b>	<b>age</b>	<b>gender</b>	<b>occupation</b>	<b>zip_code</b>
<b>0</b>	1	24	M	technician	85711
<b>1</b>	2	53	F	other	94043
<b>2</b>	3	23	M	writer	32067
<b>3</b>	4	24	M	technician	43537
<b>4</b>	5	33	F	other	15213
...	...	...	...	...	...
<b>938</b>	939	26	F	student	33319
<b>939</b>	940	32	M	administrator	02215
<b>940</b>	941	20	M	student	97229
<b>941</b>	942	48	F	librarian	78209
<b>942</b>	943	22	M	student	77841

943 rows × 5 columns

## Step 4. See the first 25 entries

In [63]: `df.head(25)`

Out[63]:

	<b>user_id</b>	<b>age</b>	<b>gender</b>	<b>occupation</b>	<b>zip_code</b>
<b>0</b>	1	24	M	technician	85711
<b>1</b>	2	53	F	other	94043
<b>2</b>	3	23	M	writer	32067
<b>3</b>	4	24	M	technician	43537
<b>4</b>	5	33	F	other	15213
<b>5</b>	6	42	M	executive	98101
<b>6</b>	7	57	M	administrator	91344
<b>7</b>	8	36	M	administrator	05201
<b>8</b>	9	29	M	student	01002
<b>9</b>	10	53	M	lawyer	90703
<b>10</b>	11	39	F	other	30329
<b>11</b>	12	28	F	other	06405
<b>12</b>	13	47	M	educator	29206
<b>13</b>	14	45	M	scientist	55106
<b>14</b>	15	49	F	educator	97301
<b>15</b>	16	21	M	entertainment	10309
<b>16</b>	17	30	M	programmer	06355
<b>17</b>	18	35	F	other	37212
<b>18</b>	19	40	M	librarian	02138
<b>19</b>	20	42	F	homemaker	95660
<b>20</b>	21	26	M	writer	30068
<b>21</b>	22	25	M	writer	40206
<b>22</b>	23	30	F	artist	48197
<b>23</b>	24	21	F	artist	94533
<b>24</b>	25	39	M	engineer	55107

## Step 5. See the last 10 entries

In [64]: `df.tail(10)`

Out[64]:

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

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

In [66]: `df.shape[0]`

Out[66]: 943

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

In [67]: `df.shape[1]`

Out[67]: 5

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

In [71]: `df.columns`

Out[71]: `Index(['user_id', 'age', 'gender', 'occupation', 'zip_code'], dtype='object')`

## Step 9. How is the dataset indexed?

In [77]: `df.index`

Out[77]: `RangeIndex(start=0, stop=943, step=1)`

In [ ]:

```
Out[ ]: 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 [74]: df.dtypes
```

```
Out[74]: user_id      int64
          age         int64
          gender     object
          occupation  object
          zip_code    object
          dtype: object
```

## Step 11. Print only the occupation column

```
In [78]: df["occupation"]
```

```
Out[78]: 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 [86]: df.occupation.nunique()
```

```
Out[86]: 21
```

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

```
In [87]: df.occupation.value_counts().head(1)
```

```
Out[87]: occupation
          student    196
          Name: count, dtype: int64
```

## Step 14. Summarize the DataFrame.

```
In [94]: df.describe()
```

Out[94]:

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

## Step 15. Summarize all the columns

In [91]: `df.describe(include="all")`

Out[91]:

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

## Step 16. Summarize only the occupation column

In [93]: `df.occupation.describe()`

Out[93]:

```
count      943
unique     21
top       student
freq      196
Name: occupation, dtype: object
```

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

```
In [89]: df["age"].mean()
```

```
Out[89]: 34.05196182396607
```

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

```
In [92]: df.age.value_counts().tail()
```

```
Out[92]: 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! 

```
In [ ]:
```