Machine Learning - 2301CS621

635 || Vishal Baraiya || 23010101014

Lab - 1

NumPy Exercises

We'll start off with a few simple tasks

import numpy as np

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

In [2]:
```
arr = np.array([1,2,3,4,5])
```

In [3]:
```
print(arr)
```

```
[1 2 3 4 5]
```

In [4]:
```
type(arr)
```

Out[4]:
```
numpy.ndarray
```

In [5]:
```
type(arr[0])
```

Out[5]:
```
numpy.int32
```

In [6]:
```
arr.dtype
```

Out[6]:
```
dtype('int32')
```

In [7]:
```
arr1 = np.array(['1','2','3','4','5'])
arr1
```

Out[7]:
```
array(['1', '2', '3', '4', '5'], dtype='<U1')
```

In [8]:
```
arr1.dtype
```

Out[8]:
```
dtype('<U1')
```

Create an array of 10 zeros

In [9]:
```
np.zeros([10,10])
```

Out[9]:
```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

Create an array of 10 ones

In [10]:
```
np.ones(10)
```

Out[10]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

Create an array of 10 fives

In [11]:
```
np.ones(10)*5
```

Out[11]:
```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

Create an array of the integers from 10 to 50

In [12]:
```
np.arange(10,51)
```

Out[12]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
       44, 45, 46, 47, 48, 49, 50])
```

In [13]:
```
np.arange(10,51,2)
```

Out[13]:
```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48, 50])
```

Create a 3x3 matrix with values ranging from 0 to 8

In [14]:
```
np.arange(0,9).reshape(3,3)
```

Out[14]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [15]:
```
np.arange(0,9).reshape(1,9)
```

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

Create a 3x3 identity matrix

In [16]:
```
np.identity(3)
```

Out[16]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [17]:
```
np.eye(3)
```

Out[17]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Create following matrix =

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

In [18]:
```
m = np.arange(1,26).reshape(5,5)
m
```

Out[18]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

Output should be =

```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

In [19]:
```
m[2::,1::]
```

Out[19]:
```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

Output should be =

```
array([[ 2],
       [ 7],
       [12]])
```

In [20]:
```
m[0:3:,1:2:]
```

Out[20]:
```
array([[ 2],
       [ 7],
       [12]])
```

Find Odd Numbers

Extract all odd numbers from `arr`.

`arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])`

Use Condtional Selection

In [21]:
```
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
arr
```

Out[21]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [22]:
```
print(arr[1::2])
```

```
[1 3 5 7 9]
```

In [23]:
```
arr[arr%2!=0]
```

Out[23]:
```
array([1, 3, 5, 7, 9])
```

Stacking

Stack array `a` and `b` vertically.

`a = np.arange(10).reshape(2,-1)` `b = np.repeat(1, 10).reshape(2,-1)`

In [24]:
```
a = np.arange(10).reshape(2,-1)
b = np.repeat(1, 10).reshape(2,-1)
print(a)
print()
print(b)
```

```
[[0 1 2 3 4]
 [5 6 7 8 9]]

[[1 1 1 1 1]
 [1 1 1 1 1]]
```

In [25]:
```
np.vstack([a,b])
```

Out[25]:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]])
```

In [26]:
```
np.hstack([a,b])
```

Out[26]:
```
array([[0, 1, 2, 3, 4, 1, 1, 1, 1, 1],
       [5, 6, 7, 8, 9, 1, 1, 1, 1, 1]])
```

Range Filtering

Get all items between 5 and 10 from `a`.

```
a = np.array([2, 6, 1, 9, 10, 3, 27])
```

In [27]:
```
a = np.array([2, 6, 1, 9, 10, 3, 27])
a
```

Out[27]:
```
array([ 2,  6,  1,  9, 10,  3, 27])
```

In [28]:
```
output = a[(a>=5) & (a<=10)]
print(output)
```

```
[ 6  9 10]
```

Matrix Multiplication

Multiply a 5x3 matrix by a 3x2 matrix (dot product).

In [29]:
```
A = np.random.random((5,3))
B = np.random.random((3,2))
ans = np.dot(A,B)
print(ans)
```

```
[[0.48996822 0.74831816]
 [0.64072734 0.95360826]
 [0.5558653  0.98524962]
 [0.05997711 0.0972653 ]
 [0.71045136 1.46967314]]
```

Great Job! You're Halfway Through 🚀 | PANDAS Exercises

Context: You are a Data Analyst at "TechCorp". HR has asked you to analyze employee data to understand attrition, salary trends, and job satisfaction.

Instructions:

1. Write your code in the empty cells.

1. Load & Inspect

Load `employee_data.csv` into a DataFrame named `df`. Display the first 5 rows and check column data types.

In [30]:
```
import pandas as pd
df = pd.read_csv("employee_data.csv")
df
```

Out[30]:

	EmployeeID	Age	Department	JobRole	Attrition	MonthlyIncome	YearsAtCompan
0	1001	50	Research & Development	Data Scientist	No	8415	1
1	1002	36	Research & Development	Data Scientist	No	6391	1
2	1003	29	Sales	Data Scientist	No	14157	1
3	1004	42	Sales	Sales Exec	No	14835	1
4	1005	40	Sales	Software Engineer	No	8951	1
...	...	...	...	...	...	...	
195	1196	40	Research & Development	Sales Exec	Yes	9704	
196	1197	41	Sales	Lab Technician	No	7422	1
197	1198	53	Research & Development	Data Scientist	No	12984	
198	1199	28	Sales	Software Engineer	No	4759	1
199	1200	54	Research & Development	Sales Exec	Yes	14404	1

200 rows × 12 columns

2. Filter IT Department

Create a DataFrame `df_rd` containing only employees from the `'Research & Development'` department.

In [31]:
```
df_rd = df[df["Department"] == "Research & Development"]
df_rd.head()
```

Out[31]:

	EmployeeID	Age	Department	JobRole	Attrition	MonthlyIncome	YearsAtCompany	.
0	1001	50	Research & Development	Data Scientist	No	8415	19	
1	1002	36	Research & Development	Data Scientist	No	6391	19	
5	1006	44	Research & Development	HR Manager	Yes	8468	8	
6	1007	32	Research & Development	Data Scientist	No	11007	19	
8	1009	45	Research & Development	Data Scientist	No	12444	1	

3. Salary Check

Find the employee with the highest `MonthlyIncome`. Display their `JobRole` and `MonthlyIncome`.

In [32]:
```
highhest_paid = df.sort_values('MonthlyIncome',ascending=False).iloc[0]
highhest_paid[['JobRole','MonthlyIncome']]
```

Out[32]:
```
JobRole          HR Manager
MonthlyIncome         14954
Name: 28, dtype: object
```

4. Missing Values

Check if there are any missing values (`NaN`) in the dataset.

In [33]:
```
df.isnull().sum()
```

Out[33]:
```
EmployeeID          0
Age                 0
Department          0
JobRole             0
Attrition           0
MonthlyIncome       0
YearsAtCompany      0
JobSatisfaction     0
PerformanceRating   0
PercentSalaryHike   0
DistanceFromHome    0
MaritalStatus       0
dtype: int64
```

5. Average Age by Role

Calculate the average `Age` for each `JobRole`.

In [34]:
```
avg_age = df.groupby('JobRole')['Age'].mean()
avg_age
```

Out[34]:
```
JobRole
Data Scientist       38.771429
HR Manager           39.926829
Lab Technician       38.673469
Sales Exec           44.000000
Software Engineer    40.395349
Name: Age, dtype: float64
```

6. Attrition Count

How many employees have left the company? (Count values in the `Attrition` column).

In [35]:
```
df["Attrition"].value_counts()
```

Out[35]:
```
Attrition
No     167
Yes     33
Name: count, dtype: int64
```

7. High Performers

Filter for employees with `PercentSalaryHike` > 15 **AND** `PerformanceRating` == 4.

In [36]:
```
df_rd = df[(df["PercentSalaryHike"] > 15) & (df["PerformanceRating"] == 4)]
df_rd
```

Out[36]:

	EmployeeID	Age	Department	JobRole	Attrition	MonthlyIncome	YearsAtCompan
0	1001	50	Research & Development	Data Scientist	No	8415	1
1	1002	36	Research & Development	Data Scientist	No	6391	1
2	1003	29	Sales	Data Scientist	No	14157	1
3	1004	42	Sales	Sales Exec	No	14835	1
16	1017	59	Sales	Software Engineer	Yes	10577	
25	1026	37	Human Resources	Data Scientist	No	12510	1
29	1030	28	Research & Development	Software Engineer	No	10804	1
37	1038	47	Research & Development	Software Engineer	No	5192	
40	1041	49	Research & Development	Sales Exec	Yes	7033	1
45	1046	38	Research & Development	Software Engineer	No	14302	1
46	1047	57	Research & Development	Sales Exec	No	4970	1
51	1052	50	Research & Development	Sales Exec	No	11135	1
60	1061	29	Research & Development	Lab Technician	No	3922	
75	1076	45	Research & Development	HR Manager	No	14045	
77	1078	29	Research & Development	Data Scientist	No	10326	
103	1104	35	Research & Development	Lab Technician	No	13541	
108	1109	35	Sales	Sales Exec	No	10783	
110	1111	30	Research & Development	Lab Technician	No	14473	1
118	1119	51	Research & Development	Software Engineer	No	7451	1
120	1121	44	Research & Development	Sales Exec	No	6876	1

	EmployeeID	Age	Department	JobRole	Attrition	MonthlyIncome	YearsAtCompan
128	1129	49	Research & Development	Lab Technician	No	8109	1
151	1152	53	Sales	Sales Exec	No	14694	
157	1158	58	Research & Development	Data Scientist	No	7212	
174	1175	32	Research & Development	HR Manager	No	4205	1
181	1182	43	Sales	Data Scientist	Yes	7757	
185	1186	30	Research & Development	Data Scientist	No	14312	
187	1188	37	Sales	Data Scientist	No	7944	1

8. Satisfaction Analysis (Pivot Table)

Create a pivot table showing the **average** `MonthlyIncome` for each `Department` (rows) broken down by `JobSatisfaction` (columns).

In [37]:
```
df.pivot_table(
    values='MonthlyIncome',
    index='Department',
    columns='JobSatisfaction',
    aggfunc='mean'
)
```

Out[37]:

JobSatisfaction	1	2	3	4
Department				
Human Resources	8494.666667	8822.200000	8196.333333	7744.833333
Research & Development	10810.718750	8581.125000	8648.090909	8930.304348
Sales	9795.789474	9590.461538	8459.461538	9045.000000

9. Feature Engineering (Seniority)

Create a new column `Seniority`.

- If `YearsAtCompany` > 10, value is `'Senior'`
- Otherwise, value is `'Junior'`

Count how many Seniors vs Juniors there are.

In [38]:
```
df["Seniority"] = np.where(df['YearsAtCompany'] > 10,'Senior','Junior')
print(df["Seniority"].value_counts())
```

```
Seniority
Junior    104
Senior     96
Name: count, dtype: int64
```

10. Retention Risk (Complex Filter)

Identify 'High Risk' employees who meet **ALL** these criteria:

1. `MaritalStatus` is 'Single'
2. `JobSatisfaction` is less than 3
3. `DistanceFromHome` > 10

Display their `EmployeeID` and `JobRole`.

In [39]:
```
risk = df[(df['MaritalStatus'] == "Single") & (df['JobSatisfaction'] < 3) & (df['Di
risk[['EmployeeID','JobRole']]
```

Out[39]:

	EmployeeID	JobRole
20	1021	Software Engineer
30	1031	Lab Technician
42	1043	HR Manager
43	1044	HR Manager
45	1046	Software Engineer
78	1079	Data Scientist
93	1094	Lab Technician
108	1109	Sales Exec
112	1113	Data Scientist
119	1120	Software Engineer
124	1125	HR Manager
159	1160	HR Manager
161	1162	Data Scientist
168	1169	HR Manager
169	1170	Sales Exec
172	1173	Software Engineer
174	1175	HR Manager
183	1184	Lab Technician
185	1186	Data Scientist

In [ ]: