Akilesh K

k.akilesh123@gmail.com

Data engineering - Batch 1

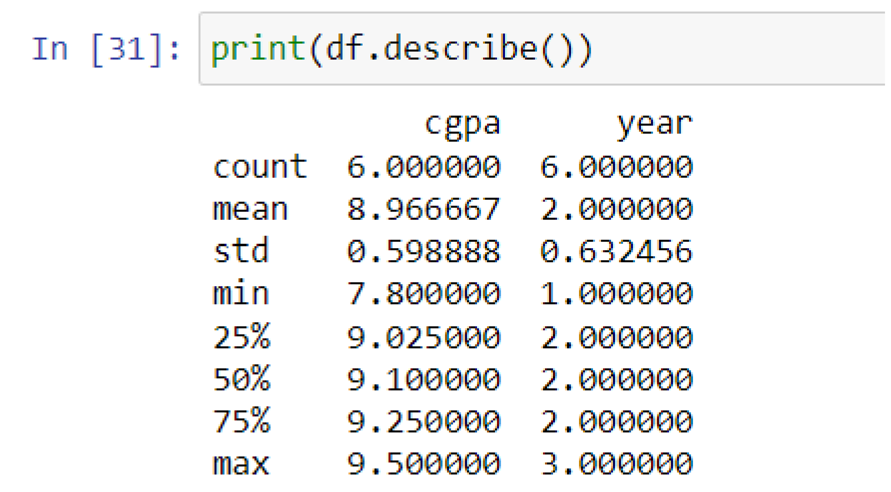
Date: 02-02-24

CODING ASSESSMENT - PYTHON

1)

Explain Pandas for Data Processing

Pandas is widely used in data science, machine learning, and analytics due to its simplicity and flexibility in handling and analyzing structured data.

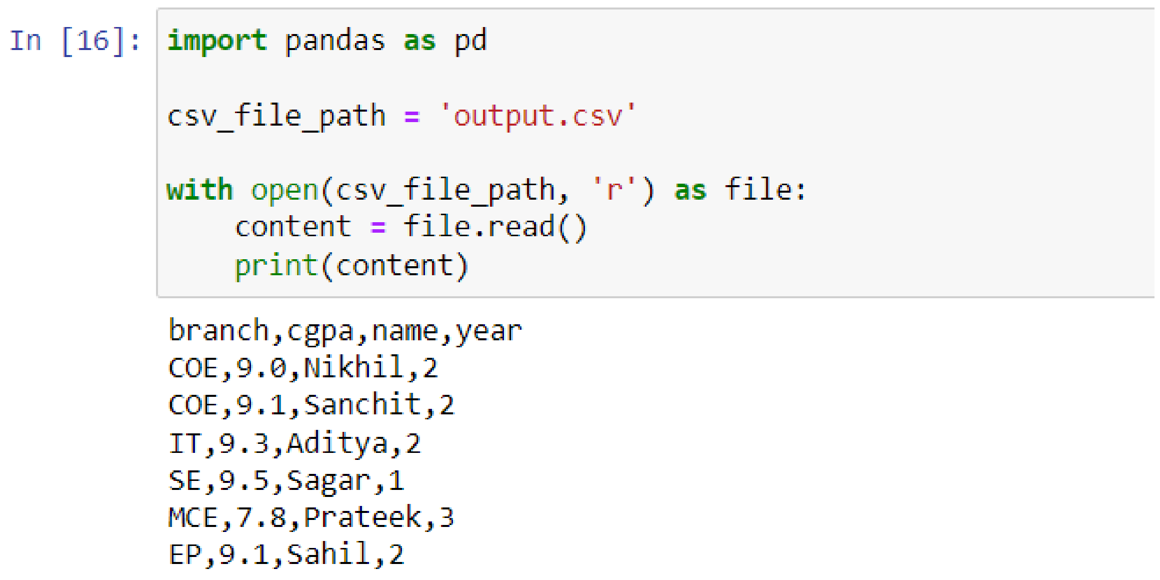


```
In [31]: print(df.describe())

           cgpa      year
count  6.000000  6.000000
mean   8.966667  2.000000
std    0.598888  0.632456
min    7.800000  1.000000
25%    9.025000  2.000000
50%    9.100000  2.000000
75%    9.250000  2.000000
max    9.500000  3.000000
```

- It provides functions and methods to handle missing data, and reshape datasets.
- It makes it easy to explore and understand your data.
- It allows you to filter and select data based on conditions.

Execute Reading CSV Data using Pandas

CSV file using standard Python file handling (open and read), and then it prints the content to the console. This approach treats the CSV content as a raw string

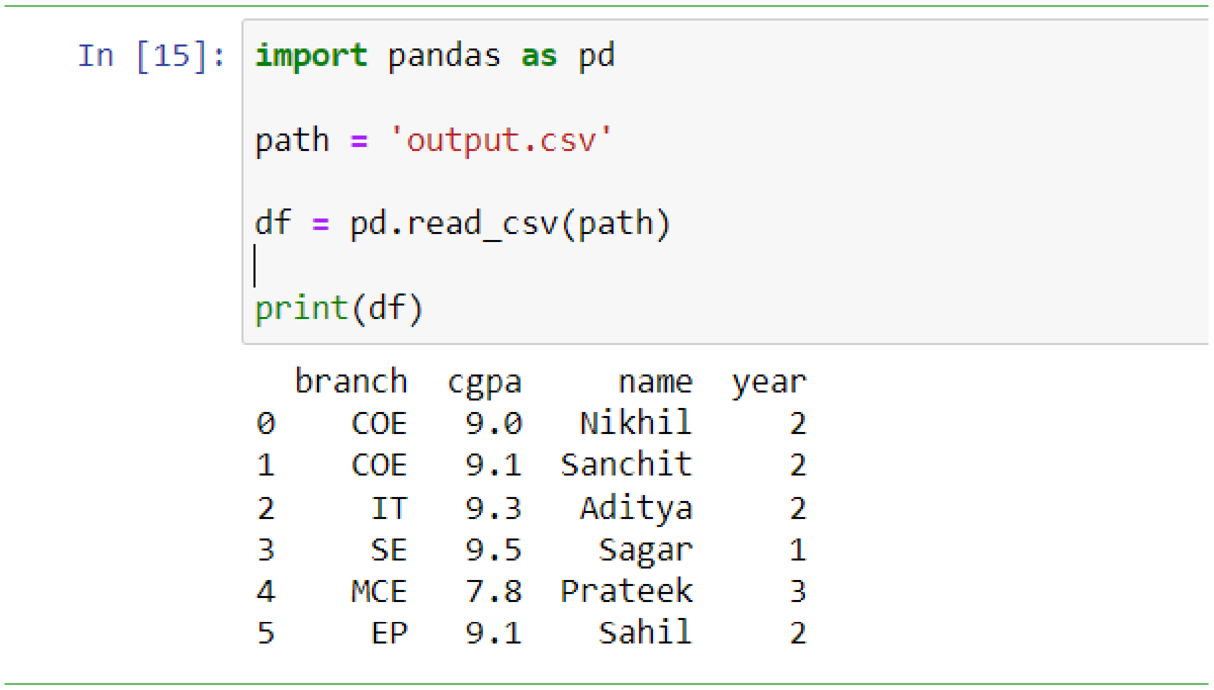


```
In [16]: import pandas as pd

         csv_file_path = 'output.csv'

         with open(csv_file_path, 'r') as file:
             content = file.read()
             print(content)
```

```
branch,cgpa,name,year
COE,9.0,Nikhil,2
COE,9.1,Sanchit,2
IT,9.3,Aditya,2
SE,9.5,Sagar,1
MCE,7.8,Prateek,3
EP,9.1,Sahil,2
```

Read Data from CSV Files to Pandas Dataframes

This uses Pandas to read the content of a CSV file into a Data Frame, and then it prints the Data Frame to the console. Pandas provides various functionalities for data manipulation, analysis, and exploration, making it a powerful tool for working with structured data in Python.



```
In [15]: import pandas as pd

         path = 'output.csv'

         df = pd.read_csv(path)

         print(df)
```

```
  branch  cgpa     name  year
0    COE   9.0   Nikhil     2
1    COE   9.1  Sanchit     2
2     IT   9.3   Aditya     2
3     SE   9.5    Sagar     1
4    MCE   7.8  Prateek     3
5     EP   9.1    Sahil     2
```

3)Filter Data in Pandas Dataframe using query.

Filters a Pandas DataFrame based on the conditions specified in the query method, creating a new DataFrame The resulting filtered DataFrame is then printed to the console, showing only the rows that satisfy the specified conditions.

```
In [17]: filtered_df = df.query('year > 1 and cgpa > 8.0')

         print("\nFiltered DataFrame:")
         print(filtered_df)
```

```
Filtered DataFrame:
  branch  cgpa     name  year
0    COE   9.0   Nikhil     2
1    COE   9.1  Sanchit     2
2     IT   9.3   Aditya     2
5     EP   9.1    Sahil     2
```

2)

LAMBDA FUNCTIONS

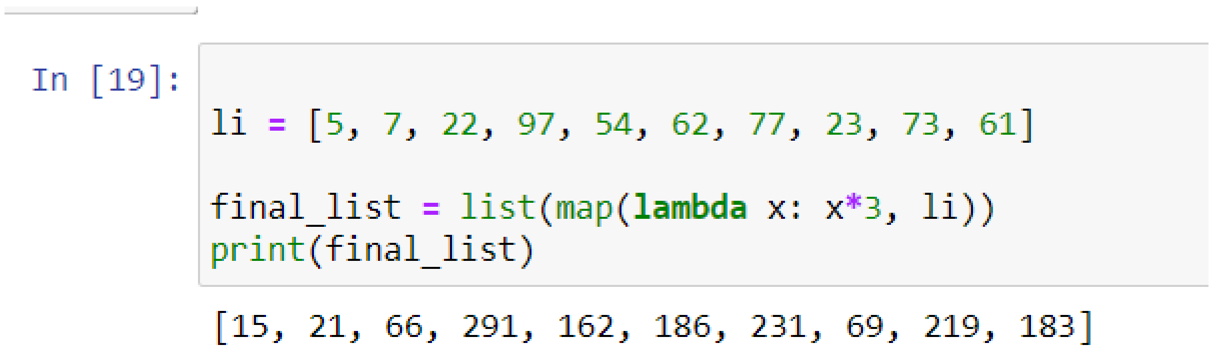
Lambda functions are commonly used in situations where a small, one-time function is needed. They are often used with functions like map(), filter(), and reduce() or in situations where a quick, inline function is required.

Uses filter and a lambda function to create a new list containing only the odd numbers from the original list. The lambda function serves as the filtering condition.

```
In [18]: li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

         final_list = list(filter(lambda x: (x % 2 != 0), li))
         print(final_list)
```

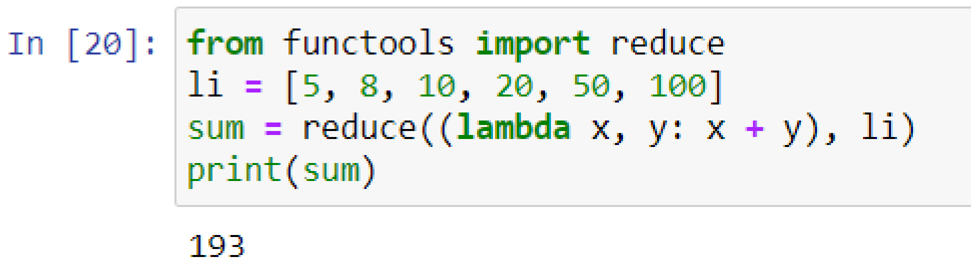
```
[5, 7, 97, 77, 23, 73, 61]
```

Uses map and a lambda function to create a new list where each element is obtained by multiplying the corresponding element from the original list by 3. The lambda function serves as the transformation applied to each element of the list.



```
In [19]:
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

final_list = list(map(lambda x: x*3, li))
print(final_list)

[15, 21, 66, 291, 162, 186, 231, 69, 219, 183]
```

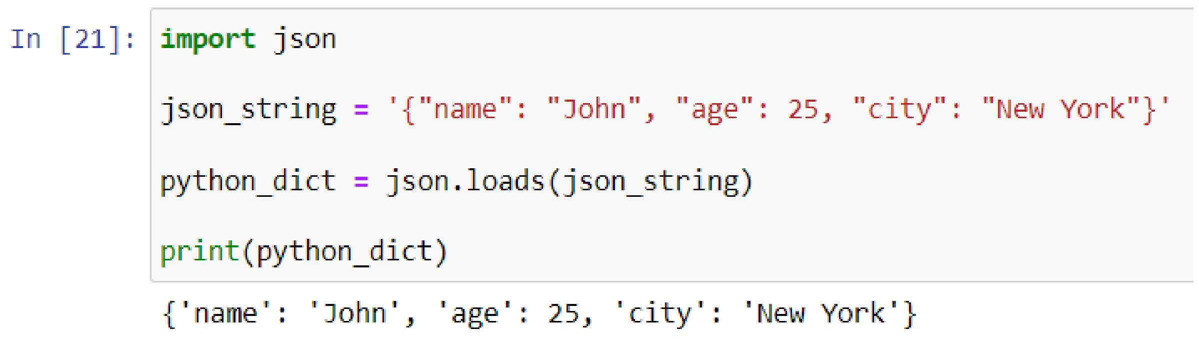
Reduce function along with a lambda function to calculate the sum of all elements in the list. The lambda function specifies the addition operation, and the reduce function applies this operation cumulatively to the elements of the list



```
In [20]: from functools import reduce
li = [5, 8, 10, 20, 50, 100]
sum = reduce((lambda x, y: x + y), li)
print(sum)

193
```

Read JSON Strings to Python dictionary

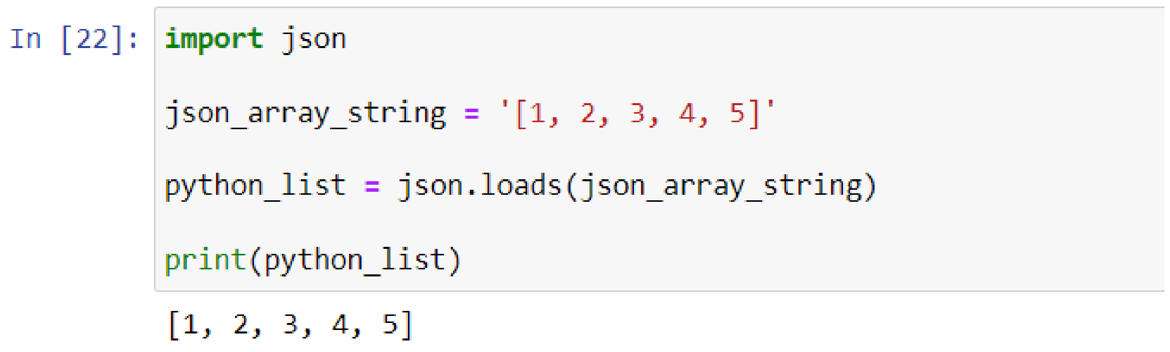
JSON-formatted string into a Python dictionary using the json.loads() function.



```
In [21]: import json

         json_string = '{"name": "John", "age": 25, "city": "New York"}'

         python_dict = json.loads(json_string)

         print(python_dict)

         {'name': 'John', 'age': 25, 'city': 'New York'}
```

Read JSON Strings to Python lists

conversion of a JSON-formatted string representing an array into a Python list using the json.loads() function.



```
In [22]: import json

         json_array_string = '[1, 2, 3, 4, 5]'

         python_list = json.loads(json_array_string)

         print(python_list)

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