```
from google.colab import files
uploaded = files.upload()
```

Choose Files Elite Sports… in Data.csv

- **Elite Sports Cars in Data.csv**(text/csv) - 820452 bytes, last modified: 4/6/2025 - 100% done

```
Saving Elite Sports Cars in Data.csv to Elite Sports Cars in Data.csv
```

```
import numpy as np
import pandas as pd
df = pd.read_csv("Elite Sports Cars in Data.csv")
df.head()
```

	Brand	Model	Year	Country	Condition	Engine_Size	Horsepower	Torque	Weight	Top_Speed	...	Mileage	Popularity	Safety_Ra
0	Nissan	720S	2006	Asia	used	3.7	420	705	1785	238	...	96664	Low	
1	McLaren	911 Turbo S	2009	Europe	new	5.3	1104	766	992	386	...	159630	High	
2	Chevrolet	M4 Competition	2009	USA	new	5.5	153	1573	2022	397	...	111496	High	
3	Bugatti	Chiron	1982	Asia	used	5.4	544	1009	1091	151	...	217228	High	
4	Nissan	Chiron	2022	Europe	new	2.4	980	693	1232	385	...	150318	Low	

5 rows × 27 columns

```
import numpy as np

# Extracting useful columns as NumPy arrays
horsepower = np.array(df['Horsepower'], dtype=np.int32)
top_speed = np.array(df['Top_Speed'], dtype=np.int32)
weight = np.array(df['Weight'], dtype=np.int32)
```

```
#Fixed Type Arrays
print("Horsepower dtype:", horsepower.dtype)
```

```
Horsepower dtype: int32
```

```
#Creating Arrays
sample_array = np.array([100, 200, 300])
print("Sample Array:", sample_array)
```

```
Sample Array: [100 200 300]
```

```
#Array Indexing & Slicing
print("First 5 HP:", horsepower[:5])
print("HP at positions 0, 2, 4:", horsepower[[0, 2, 4]])
```

```
First 5 HP: [ 420 1104  153  544  980]
HP at positions 0, 2, 4: [420 153 980]
```

```
#Reshaping Arrays
reshaped = horsepower[:10].reshape(5, 2)
print("Reshaped:\n", reshaped)
```

```
Reshaped:
 [[ 420 1104]
 [ 153  544]
 [ 980 1091]
 [ 810 1189]
 [ 230  675]]
```

```
#Concatenation & Splitting
combined = np.concatenate((horsepower[:5], top_speed[:5]))
split = np.split(combined, 2)
print("Combined:", combined)
print("Split:", split)
```

```
Combined: [ 420 1104  153  544  980  238  386  397  151  385]
Split: [array([ 420, 1104,  153,  544,  980], dtype=int32), array([238, 386, 397, 151, 385], dtype=int32)]
```

```
#Universal Functions
print("Square Root of HP:", np.sqrt(horsepower[:5]))
print("Log of Speed:", np.log1p(top_speed[:5]))
```

```
Square Root of HP: [20.49390153 33.22649545 12.36931688 23.32380758 31.30495168]
Log of Speed: [5.47646355 5.95842469 5.98645201 5.02388052 5.95583737]
```

```
#Aggregations
print("Mean HP:", np.mean(horsepower))
print("Max HP:", np.max(horsepower))
print("Sum HP:", np.sum(horsepower))
```

```
Mean HP: 822.8916
Max HP: 1521
Sum HP: 4114458
```

```
# Broadcasting Rules
hp_to_weight = horsepower / weight
print("HP/Weight (first 5):", hp_to_weight[:5])
```

```
HP/Weight (first 5): [0.23529412 1.11290323 0.07566766 0.49862511 0.79545455]
```

```
#Comparisons & Boolean Arrays
fast_cars = top_speed > 300
print("Cars with speed > 300:\n", df[fast_cars])
```

```
Cars with speed > 300:
          Brand           Model  Year Country Condition  Engine_Size  \
1      McLaren     911 Turbo S  2009  Europe       new          5.3   
2    Chevrolet  M4 Competition  2009     USA       new          5.5   
4       Nissan          Chiron  2022  Europe       new          2.4   
5          BMW            GT-R  1986  Europe       new          5.4   
9      Bugatti             DBS  1992     USA       new          6.5   
...        ...             ...   ...     ...       ...          ...   
4982   Ferrari             DBS  1980     USA       new          3.3   
4987 Chevrolet  M4 Competition  1996    Asia      used          4.8   
4993    Nissan            720S  2024    Asia       new          5.1   
4994       BMW          Chiron  2013  Europe      used          7.7   
4999   McLaren          Chiron  2018  Europe       new          7.7   

      Horsepower  Torque  Weight  Top_Speed  ...  Mileage Popularity  \
1           1104     766     992        386  ...   159630       High   
2            153    1573    2022        397  ...   111496       High   
4            980     693    1232        385  ...   150318        Low   
5           1091     632    2135        341  ...   228779       High   
9            675    1209    1472        382  ...   101321        Low   
...          ...     ...     ...        ...  ...      ...        ...   
4982         775     271    1734        390  ...   138397        Low   
4987         930     531    1077        375  ...    24888       High   
4993         189     432    2370        323  ...    27472        Low   
4994        1258     983    1440        309  ...    53691       High   
4999        1075    1037    1312        381  ...    57970       High   

     Safety_Rating Number_of_Owners  Market_Demand  Insurance_Cost  \
1                2                2         Medium           10795   
2                1                2            Low            1716   
4                3                2         Medium           11324   
5                2                3         Medium            5274   
9                3                4         Medium           14850   
...            ...              ...            ...             ...   
4982             3                4            Low            2016   
4987             2                4            Low           13445   
4993             4                4         Medium           13986   
4994             3                4            Low           12474   
4999             2                3           High            1802   

      Production_Units  Log_Price Log_Mileage  Modification  
1                 1000  12.639334   11.980620           NaN  
2                20000  12.948902   11.621753           NaN  
4               100000  11.229289   11.920515           NaN  
5                  200  13.077665   12.340516           NaN  
9                 5000  13.020342   11.526059           NaN  
...                ...        ...         ...           ...  
4982            100000  13.031962   11.837889           NaN  
4987              5000  12.199476   10.122181           NaN  
4993              5000  12.084035   10.220959           NaN  
4994               200  11.918771   10.891019         Nismo  
4999             20000  11.267536   10.967698         Sport  
```

```
[1975 rows x 27 columns]
```

```
#Structured Arrays / Record Array
structured = np.array(
    list(zip(df['Brand'], df['Model'], horsepower)),
    dtype=[('Brand', 'U20'), ('Model', 'U30'), ('Horsepower', 'i4')]
)

print("Structured array sample:\n", structured[:5])

```

```
Structured array sample:
 [('Nissan', '720S',  420) ('McLaren', '911 Turbo S', 1104)
 ('Chevrolet', 'M4 Competition',  153) ('Bugatti', 'Chiron',  544)
 ('Nissan', 'Chiron',  980)]
```