NumPy Cheatsheet

NumPy (Numerical Python) is the fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

1

Creating Arrays

- import numpy as np – Import NumPy
- **np.array([1, 2, 3])** – Create a 1D array
- **np.zeros((2, 3))** – Create a 2x3 array of zeros
- **np.ones((3, 3))** – Create a 3x3 array of ones
- **np.eye(3)** – Create a 3x3 identity matrix

2

Array Generation

- **np.arange(0, 10, 2)** – Array from 0 to 10 with step 2
- **np.linspace(0, 1, 5)** – 5 evenly spaced values from 0 to 1

3

Statistical Operations

- **np.mean(array)** – Mean of array
- **np.var(array)** – Variance
- **np.std(array)** – Standard deviation

4

Array Manipulation

- **np.reshape(array, (2,3))** – Reshape to 2 rows, 3 columns
- **np.sum(array, axis=0)** – Sum over columns
- **np.sum(array, axis=1)** – Sum over rows

Working with NumPy Arrays

Creating Arrays: Examples

```
import numpy as np

# Create a simple 1D array
arr1 = np.array([1, 2, 3, 4, 5])

# Create a 2D array (matrix)
arr2 = np.array([[1, 2, 3], [4, 5, 6]])

# Create arrays with specific values
zeros = np.zeros((3, 4))  # 3x4 array of zeros
ones = np.ones((2, 2))    # 2x2 array of ones
identity = np.eye(3)      # 3x3 identity matrix
```

Statistical Operations: Examples

```
# Sample array
data = np.array([1, 2, 3, 4, 5])

# Basic statistics
mean_value = np.mean(data)      # 3.0
median_value = np.median(data)  # 3.0
std_dev = np.std(data)          # ~1.41
variance = np.var(data)         # 2.0
min_value = np.min(data)        # 1
max_value = np.max(data)        # 5
```

Pandas Cheatsheet

Pandas is a powerful data manipulation and analysis library built on top of NumPy. It provides data structures like DataFrames that make working with structured data intuitive and efficient.

Loading Data

- **import pandas as pd** – Import Pandas
- **pd.read_csv('file.csv')** – Load CSV file into DataFrame

Viewing Data

- **df.head()** – Show first 5 rows
- **df.tail()** – Show last 5 rows
- **df.columns** – Get column names
- **df.shape** – Get number of rows and columns
- **df.describe()** – Summary statistics

Selecting Data

- **df['Column']** – Access a column
- **df[['Col1', 'Col2']]** – Access multiple columns
- **df.iloc[0]** – Access first row
- **df.loc[0, 'Column']** – Access specific value

Analyzing Data

- **df[df['Col'] > 10]** – Filter rows
- **df.sort_values('Col')** – Sort by column
- **df['Col'].mean()** – Mean of a column
- **df.isnull().sum()** – Check for missing values

Common Pandas Task	Code Example
Create DataFrame	df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
Conditional Selection	filtered_df = df[(df['A'] > 1) & (df['B'] < 6)]
Group By Operation	grouped = df.groupby('Category').mean()
Handle Missing Values	df.fillna(0) or df.dropna()
Merge DataFrames	pd.merge(df1, df2, on='key_column')

Both NumPy and Pandas are optimised for performance, with many operations vectorised for speed. When working with large datasets, these libraries can be hundreds of times faster than equivalent Python code using lists and dictionaries.