NumPy Basics

```
# Create arrays
np.array([1, 2, 3])                 # From list
np.zeros(5)                         # [0., 0., 0., 0., 0.]
np.ones((2, 3))                     # 2x3 array of 1s
np.arange(0, 10, 2)                 # [0, 2, 4, 6, 8]
np.linspace(0, 1, 5)                # [0., 0.25, 0.5, 0.75, 1.]
np.random.rand(3, 2)                # Random values 0-1

# Array Properties
arr = np.array([[1, 2, 3], [4, 5, 6]])
arr.shape                           # (2, 3) - dimensions
arr.ndim                            # 2 - number of dimensions
arr.size                            # 6 - total elements
arr.dtype                           # int64 - data type
len(arr)                            # 2 - length of first dimension

# Array Operations
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
a + b                               # [5, 7, 9] - element-wise
a * 2                               # [2, 4, 6] - scalar multiplication
a * b                               # [4, 10, 18] - element-wise
np.dot(a, b)                        # 32 - dot product
a ** 2                              # [1, 4, 9] - square each element
```

```
# Indexing and Slicing
arr = np.array([10, 20, 30, 40, 50])
arr[0]                              # 10 - first element
arr[1:4]                            # [20, 30, 40] - slice
arr[::2]                            # [10, 30, 50] - every 2nd
arr[-1]                             # 50 - last element

# 2D array indexing
arr2d = np.array([[1, 2, 3], [4, 5, 6]])
arr2d[0, 1]                         # 2 - row 0, column 1
arr2d[:, 1]                         # [2, 5] - all rows, column 1

# Array Reshaping
arr = np.arange(12)
arr.reshape(3, 4)                   # 3x4 array
arr.reshape(2, 3, 2)                # 3D array
arr.flatten()                       # Back to 1D
arr.T                               # Transpose (rows ↔ columns)
arr = np.array([[1, 2, 3], [4, 5, 6]])
arr.shape                           # (2, 3) - dimensions
arr.ndim                            # 2 - number of dimensions
arr.size                            # 6 - total elements
arr.dtype                           # int64 - data type
len(arr)                            # 2 - length of first dimension
```

NumPy Basics

```
# Useful function
arr = np.array([1, 2, 3, 4, 5])
np.sum(arr)                 # 15 - total sum
np.mean(arr)                # 3.0 - average
np.max(arr)                 # 5 - maximum value
np.min(arr)                 # 1 - minimum value
np.std(arr)                 # 1.41 - standard deviation
np.sort(arr)                # Sorted array
np.unique([1, 2, 2, 3])     # [1, 2, 3] - unique values
np.where(arr > 2)           # Indices where condition is true
np.array([1, 2, 3])         # From list
np.zeros(5)                 # [0., 0., 0., 0., 0.]
np.ones((2, 3))             # 2x3 array of 1s
np.arange(0, 10, 2)         # [0, 2, 4, 6, 8]
np.linspace(0, 1, 5)        # [0., 0.25, 0.5, 0.75, 1.]
np.random.rand(3, 2)        # Random values 0-1

# Boolean Operation
arr = np.array([1, 2, 3, 4, 5])
arr > 3                     # [False, False, False, True, True]
arr[arr > 3]                # [4, 5] - filter with condition
(arr > 2) & (arr < 5)       # [False, False, True, True, False]
```

```
# Save and load arrays
np.save('my_array.npy', arr)        # Save single array
np.load('my_array.npy')             # Load array
np.savetxt('data.txt', arr)         # Save as text
np.loadtxt('data.txt')              # Load from text
```

Data Cleaning in Pandas

```
# Handle missing and duplicate data
df.isnull().sum()                          # count nulls per column
df.dropna()                                # drop rows with missing values
df.fillna(method='ffill')                  # forward-fill missing values
df.drop_duplicates()                       # remove duplicate rows
df.replace({'old': 'new'})                 # replace values
# Inspect and understand your data
df.head()                                  # first rows
df.info()                                  # column dtypes and non-nulls
df.describe()                              # summary stats
#Rename, convert, and clean columns
df.rename(columns={'old': 'new'})          # rename columns
df.astype({'col': 'type'})                 # change dtype
df.drop(['col'], axis=1)                   # drop column(s)
df.reset_index(drop=True)                  # reset index
df.columns = df.columns.str.strip()        # strip whitespace in names
# Filter, slice, and select rows
df.loc[df['col'] > value]                  # filter by condition
df.iloc[0:5]                               # select by index
df['col'].isin(['val1', 'val2'])           # filter by values
df.query('col > 10 & col2 == "yes"')       # query with expressions
# Merge and group data
pd.concat([df1, df2], axis=0)              # stack rows
pd.merge(df1, df2, on='key')               # join on key
df.groupby('col').agg({'val': 'mean'})     # group and aggregate
df['col'].value_counts()                   # frequency of values
```