Python Libraries and Their Uses with Key Functions

NumPy

Use: Numerical computations and array operations.

Key Functions:

- np.array(): Create arrays.
- np.arange(): Generate sequences.
- np.linspace(): Generate linearly spaced numbers.
- np.mean(), np.median(), np.std(): Basic statistics.
- np.dot(), np.matmul(): Matrix operations.

Pandas

Use: Data manipulation and analysis.

Key Functions:

- pd.DataFrame(): Create DataFrames.
- .head(), .tail(): View the beginning and end of DataFrames.
- .groupby(), .merge(): Group and merge data.
- .iloc[], .loc[]: Indexing and slicing.
- .apply(), .map(): Apply functions to data.

Matplotlib

Use: Data visualization.

Key Functions:

- plt.plot(), plt.scatter(): Line and scatter plots.
- plt.hist(), plt.bar(): Histogram and bar charts.
- plt.xlabel(), plt.ylabel(): Labeling axes.
- plt.legend(), plt.title(): Add legends and titles.

- plt.show(): Display plots.

Seaborn

Use: Statistical data visualization.

Key Functions:

- sns.scatterplot(), sns.lineplot(): Plotting data with styles.

- sns.heatmap(): Display correlation matrices.

- sns.boxplot(), sns.violinplot(): Distribution plots.

- sns.pairplot(): Plot pairwise relationships.

- sns.set(): Set theme styles.

Scikit-Learn

Use: Machine learning and predictive data analysis.

Key Functions:

- train_test_split(): Split data into training and testing sets.

- StandardScaler(), MinMaxScaler(): Scaling data.

- LinearRegression(), RandomForestClassifier(): ML models.

- cross_val_score(): Cross-validation.

- classification_report(), confusion_matrix(): Model evaluation.

TensorFlow

Use: Deep learning and neural networks.

Key Functions:

- tf.keras.layers: Build neural network layers.

- tf.data.Dataset: Work with datasets.

- tf.optimizers: Optimizers like Adam, SGD.

- tf.metrics: Track model performance.

- tf.train.Checkpoint: Save and restore models.

PyTorch

Use: Deep learning and neural networks.

Key Functions:

- torch.Tensor(): Multi-dimensional arrays.

- torch.nn: Neural network layers.

- torch.optim: Optimization algorithms.

- torch.utils.data: Dataset management.

- torch.autograd: Automatic differentiation.

Statsmodels

Use: Statistical analysis.

Key Functions:

- sm.OLS(): Ordinary least squares regression.

- sm.Logit(): Logistic regression.

- sm.tsa.ARIMA(): Time series analysis.

- sm.add_constant(): Add intercept to models.

- .summary(): View model summary.

SciPy

Use: Scientific computations.

Key Functions:

- scipy.optimize: Optimization functions.

- scipy.integrate: Integration routines.

- scipy.stats: Statistical tests.

- scipy.linalg: Linear algebra operations.

- scipy.spatial: Spatial algorithms and data structures.

NLTK (Natural Language Toolkit)

Use: Natural language processing.

Key Functions:

- nltk.word_tokenize(): Tokenize text.

- nltk.pos_tag(): Part-of-speech tagging.

- nltk.corpus: Access language corpora.

- nltk.Text(): Text processing.

- nltk.stem: Stemming words.

OpenCV

Use: Computer vision and image processing.

Key Functions:

- cv2.imread(), cv2.imshow(): Read and display images.

- cv2.resize(), cv2.rotate(): Transform images.

- cv2.Canny(): Edge detection.

- cv2.VideoCapture(): Handle video streams.

- cv2.cvtColor(): Change color spaces.

Requests

Use: HTTP requests.

Key Functions:

- requests.get(), requests.post(): Send HTTP GET and POST requests.

- .status_code: Check HTTP status codes.

- .json(): Convert response to JSON.

- .text: Get response text.

- requests.Session(): Manage sessions.

BeautifulSoup

Use: Web scraping and parsing HTML/XML.

Key Functions:

- bs4.BeautifulSoup(): Initialize BeautifulSoup object.
- .find(), .find_all(): Locate elements.
- .get_text(): Extract text.
- .attrs: Access element attributes.
- .select(): CSS selector.

Django

Use: Web development framework.

Key Functions:

- models.Model: Create database models.
- views.View: Handle HTTP requests.
- forms.Form: Handle forms.
- migrations: Manage database migrations.
- templates: Manage HTML templates.

Flask

Use: Lightweight web framework.

Key Functions:

- Flask(): Initialize a Flask app.
- @app.route(): Define routes.
- request: Access HTTP request data.
- render_template(): Render HTML templates.
- redirect(): Redirects users.