A **roadmap to learning Python with Data Science and SQL** involves gaining proficiency in several key areas that will allow you to analyze data, work with databases, and apply machine learning techniques. This roadmap is structured for a gradual learning path, starting from the basics of Python, progressing through data science libraries, and then integrating SQL for working with databases.

1. This a topic from python official.

- 1. Whetting Your Appetite
- 2. Using the Python Interpreter
    - 2.1. Invoking the Interpreter
        - 2.1.1. Argument Passing
        - 2.1.2. Interactive Mode
    - 2.2. The Interpreter and Its Environment
        - 2.2.1. Source Code Encoding
- 3. An Informal Introduction to Python
    - 3.1. Using Python as a Calculator
        - 3.1.1. Numbers
        - 3.1.2. Text
        - 3.1.3. Lists
    - 3.2. First Steps Towards Programming
- 4. More Control Flow Tools
    - 4.1. `if` Statements
    - 4.2. `for` Statements
    - 4.3. The `range()` Function
    - 4.4. `break` and `continue` Statements
    - 4.5. `else` Clauses on Loops
    - 4.6. `pass` Statements
    - 4.7. `match` Statements
    - 4.8. Defining Functions
    - 4.9. More on Defining Functions
        - 4.9.1. Default Argument Values
        - 4.9.2. Keyword Arguments
        - 4.9.3. Special parameters
            - 4.9.3.1. Positional-or-Keyword Arguments

            - 4.9.3.2. Positional-Only Parameters
            - 4.9.3.3. Keyword-Only Arguments
            - 4.9.3.4. Function Examples
            - 4.9.3.5. Recap
        - 4.9.4. Arbitrary Argument Lists
        - 4.9.5. Unpacking Argument Lists
        - 4.9.6. Lambda Expressions
        - 4.9.7. Documentation Strings
        - 4.9.8. Function Annotations
    - 4.10. Intermezzo: Coding Style
- 5. Data Structures
    - 5.1. More on Lists
        - 5.1.1. Using Lists as Stacks
        - 5.1.2. Using Lists as Queues
        - 5.1.3. List Comprehensions
        - 5.1.4. Nested List Comprehensions
    - 5.2. The `del` statement
    - 5.3. Tuples and Sequences
    - 5.4. Sets
    - 5.5. Dictionaries
    - 5.6. Looping Techniques
    - 5.7. More on Conditions
    - 5.8. Comparing Sequences and Other Types
- 6. Modules
    - 6.1. More on Modules
        - 6.1.1. Executing modules as scripts
        - 6.1.2. The Module Search Path
        - 6.1.3. “Compiled” Python files
    - 6.2. Standard Modules
    - 6.3. The `dir()` Function
    - 6.4. Packages
        - 6.4.1. Importing * From a Package
        - 6.4.2. Intra-package References
        - 6.4.3. Packages in Multiple Directories
- 7. Input and Output

  - 7.1. Fancier Output Formatting
    - 7.1.1. Formatted String Literals
    - 7.1.2. The String format() Method
    - 7.1.3. Manual String Formatting
    - 7.1.4. Old string formatting
  - 7.2. Reading and Writing Files
    - 7.2.1. Methods of File Objects
    - 7.2.2. Saving structured data with `json`
- 8. Errors and Exceptions
  - 8.1. Syntax Errors
  - 8.2. Exceptions
  - 8.3. Handling Exceptions
  - 8.4. Raising Exceptions
  - 8.5. Exception Chaining
  - 8.6. User-defined Exceptions
  - 8.7. Defining Clean-up Actions
  - 8.8. Predefined Clean-up Actions
  - 8.9. Raising and Handling Multiple Unrelated Exceptions
  - 8.10. Enriching Exceptions with Notes
- 9. Classes
  - 9.1. A Word About Names and Objects
  - 9.2. Python Scopes and Namespaces
    - 9.2.1. Scopes and Namespaces Example
  - 9.3. A First Look at Classes
    - 9.3.1. Class Definition Syntax
    - 9.3.2. Class Objects
    - 9.3.3. Instance Objects
    - 9.3.4. Method Objects
    - 9.3.5. Class and Instance Variables
  - 9.4. Random Remarks
  - 9.5. Inheritance
    - 9.5.1. Multiple Inheritance
  - 9.6. Private Variables
  - 9.7. Odds and Ends
  - 9.8. Iterators

    - 9.9. Generators
    - 9.10. Generator Expressions
- 10. Brief Tour of the Standard Library
    - 10.1. Operating System Interface
    - 10.2. File Wildcards
    - 10.3. Command Line Arguments
    - 10.4. Error Output Redirection and Program Termination
    - 10.5. String Pattern Matching
    - 10.6. Mathematics
    - 10.7. Internet Access
    - 10.8. Dates and Times
    - 10.9. Data Compression
    - 10.10. Performance Measurement
    - 10.11. Quality Control
    - 10.12. Batteries Included
- 11. Brief Tour of the Standard Library — Part II
    - 11.1. Output Formatting
    - 11.2. Templating
    - 11.3. Working with Binary Data Record Layouts
    - 11.4. Multi-threading
    - 11.5. Logging
    - 11.6. Weak References
    - 11.7. Tools for Working with Lists
    - 11.8. Decimal Floating-Point Arithmetic
- 12. Virtual Environments and Packages
    - 12.1. Introduction
    - 12.2. Creating Virtual Environments
    - 12.3. Managing Packages with pip
- 13. What Now?
- 14. Interactive Input Editing and History Substitution
    - 14.1. Tab Completion and History Editing
    - 14.2. Alternatives to the Interactive Interpreter
- 15. Floating-Point Arithmetic: Issues and Limitations
    - 15.1. Representation Error
- 16. Appendix

- 16.1. Interactive Mode
    - 16.1.1. Error Handling
    - 16.1.2. Executable Python Scripts
    - 16.1.3. The Interactive Startup File
    - 16.1.4. The Customization Modules

2. Data Science Fundamentals

2.1. Python for Data Science Libraries

- **NumPy**: Efficient arrays and numerical operations (vectorized operations, array slicing, broadcasting).
- **Pandas**: Data manipulation and analysis (DataFrames, Series, handling missing data, filtering, merging datasets).
- **Matplotlib and Seaborn**: Basic plotting and data visualization (bar charts, histograms, line plots, box plots, pair plots).
- **SciPy**: Scientific computing (optimization, statistics, linear algebra).

Resources:

- NumPy Documentation
- Pandas Documentation
- Matplotlib Documentation
- "Python for Data Analysis" by Wes McKinney (book)

2.2. Data Wrangling and Preprocessing

- **Cleaning Data**: Handling missing values, outliers, and duplicate data.
- **Data Transformation**: Scaling, normalization, encoding categorical variables, handling dates/times.
- **Feature Engineering**: Creating new features to improve machine learning models.

Resources:

- "Data Wrangling with Python" (book)
- Pandas for Data Science

3. Statistics and Probability for Data Science

3.1. Descriptive Statistics

- Mean, median, mode, variance, standard deviation.
- Data distribution and frequency tables.

3.2. Inferential Statistics

- Hypothesis testing (t-tests, chi-squared tests).
- Confidence intervals, p-values, and statistical significance.

3.3. Probability Theory

- Basic probability concepts.
- Bayes' Theorem, conditional probability.

Resources:

- Khan Academy - Statistics and Probability
- "Practical Statistics for Data Scientists" (book)

4. Introduction to SQL for Data Science

4.1. Basic SQL Concepts

- **SQL Syntax**: `SELECT`, `FROM`, `WHERE`, `ORDER BY`, `GROUP BY`, `HAVING`, `LIMIT`.
- **Basic Queries**: Retrieving data, filtering results, sorting.
- **Joins**: Inner joins, left/right joins, full outer joins.
- **Aggregations**: COUNT, SUM, AVG, MIN, MAX, GROUP BY.
- **Subqueries**: Nested queries for more complex data retrieval.

Resources:

- SQLBolt (Interactive lessons)
- W3Schools SQL
- "SQL for Data Science" (Coursera course)

4.2. Advanced SQL Concepts

- **Window Functions**: `ROW_NUMBER()`, `RANK()`, `PARTITION BY`.
- **Indexes**: Optimizing query performance.
- **Transactions**: COMMIT, ROLLBACK.
- **Normalization**: Database design (1NF, 2NF, 3NF).

Resources:

- "Learning SQL" by Alan Beaulieu (book)
- LeetCode SQL (for practice)

5. Machine Learning

5.1. Introduction to Machine Learning

- **Types of ML**: Supervised, unsupervised, reinforcement learning.
- **Training and Testing**: Splitting data, cross-validation.

5.2. Machine Learning Libraries

- **Scikit-learn**: Implementing basic algorithms (linear regression, classification models, clustering, decision trees, random forests, KNN).
- **TensorFlow or PyTorch**: For deep learning (if you want to dive into neural networks).

Resources:

- Scikit-learn Documentation
- "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron (book)

5.3. Model Evaluation and Selection

- **Metrics**: Accuracy, precision, recall, F1 score, ROC-AUC.
- **Cross-validation**: K-fold cross-validation, stratified sampling.

5.4. Advanced Topics

- **Ensemble Methods**: Random forests, gradient boosting (XGBoost).
- **Hyperparameter Tuning**: Grid search, random search.

6. Working with Databases and SQL in Data Science

6.1. SQL for Data Retrieval

- Querying databases to extract data for analysis.
- Writing complex queries to aggregate and summarize data.

6.2. Python Database Connectivity

- **SQLite**: Local databases for simple projects.
- **SQLAlchemy**: Python ORM for interacting with SQL databases.
- **PostgreSQL or MySQL**: Advanced database management systems (for handling larger datasets).

Resources:

- SQLite Python tutorial
- SQLAlchemy Documentation
- PostgreSQL Documentation

6.3. ETL Processes

- **Extract**: Pull data from various sources.
- **Transform**: Clean and preprocess the data.
- **Load**: Store data in a database or data warehouse.

7. Data Visualization and Reporting

7.1. Visualization with Matplotlib, Seaborn, Plotly

- **Matplotlib**: Line plots, bar charts, histograms.

- **Seaborn**: More advanced statistical visualizations (heatmaps, pairplots).
- **Plotly**: Interactive charts (for dashboards or web applications).

7.2. Dashboards and Reporting Tools

- **Dash** (by Plotly): Create interactive web-based dashboards.
- **Jupyter Notebooks**: For creating and sharing reports with code, visualizations, and text.

Resources:

- Plotly Documentation
- Matplotlib Documentation

8. Real-World Projects

8.1. Projects to Work On

- **Data Wrangling**: Work on cleaning and transforming raw data from CSVs, APIs, or databases.
- **Exploratory Data Analysis (EDA)**: Create visualizations and summarize datasets.
- **Predictive Modeling**: Build and evaluate regression/classification models.
- **SQL Databases**: Write complex queries to analyze large datasets stored in relational databases.

Project Ideas:

- **Analyzing Sales Data**: Using SQL for querying, Python for analysis, and visualization.
- **Customer Segmentation**: Use clustering techniques (K-means, DBSCAN) to group similar customers based on their behavior.
- **Prediction Models**: Build a model to predict house prices, stock prices, or customer churn.

Summary Roadmap

Phase	Skills/Technologies

Phase 1: Python Basics	Python syntax, data structures, functions, error handling
Phase 2: Data Science Basics	NumPy, Pandas, Matplotlib, Seaborn, SciPy
Phase 3: SQL Basics	SQL queries, joins, aggregations, subqueries, basic database operations
Phase 4: Statistics/ML	Descriptive stats, probability, hypothesis testing, machine learning algorithms (Scikit-learn)
Phase 5: SQL & Python	Database connection (SQLite, PostgreSQL), ETL processes, advanced SQL
Phase 6: Data Visualization	Matplotlib, Seaborn, Plotly, Dash, Jupyter Notebooks
Phase 7: Real-World Projects	Apply everything in real-world data science projects