Programming for Data Science Syllabus

Learn to use Python and SQL to solve problems with data

Before You Start

Prerequisites: There are no prerequisites for this program, aside from basic computer skills. You should feel comfortable performing basic operations on your computer (e.g., opening files, folders, and applications, copying and pasting).

Educational Objectives: Students will learn the programming fundamentals required for a career in data science. By the end of the program, students will be able to use Python, SQL, the terminal, and git.

Length of Program: 100 Hours*
Program Structure: Self-paced, 3 months of access
Textbooks Required: None
Student Services:

- Content: This includes video lectures, interactive quizzes, and other learning material.
- Project Reviews, technical forums, weekly office hours, mentorship, Slack community

*The length is an estimation of total hours the average student may take to complete all required coursework, including lesson and project time. Actual hours may vary.

Project 1: Investigate a Relational Database (45 hours)

In this project, you'll work with a relational database while working with PostgreSQL. You'll complete the entire data analysis process, starting by posing a question, running appropriate SQL queries to answer your questions and finishing by sharing your findings.

Supporting Lesson Content: Introduction to SQL

Lesson Title	Learning Outcomes
Basic SQL	➔ Write common SQL commands including SELECT, FROM, and WHERE ➔ Use logical operators like LIKE, AND, and OR
SQL Joins	➔ Write JOINs in SQL, as you are now able to combine data from multiple sources to answer more complex business questions ➔ Understand different types of JOINs and when to use each type
SQL Aggregations	➔ Write common aggregations in SQL including COUNT, SUM, MIN, and MAX ➔ Write CASE and DATE functions, as well as work with NULLs
Advanced SQL Queries	➔ Use subqueries, also called CTEs, in a number of different situations ➔ Use other window functions including RANK, NTILE, LAG, LEAD new functions along with partitions to complete complex tasks

Project 2: Explore US Bikeshare Data (45 hours)

You will use Python to answer interesting questions about bikeshare trip data collected from three US cities. You will write code to collect the data, compute descriptive statistics, and create an interactive experience in the terminal that presents the answers to your questions.

Supporting Lesson Content: Introduction to Python Programming

In this part, you'll learn to represent and store data using Python data types and variables, and use conditionals and loops to control the flow of your programs. You'll harness the power of complex data structures like lists, sets, dictionaries, and tuples to store collections of related data. You'll define and document your own custom functions, write scripts, and handle errors. You will also learn to use two powerful Python libraries - Numpy, a scientific computing package, and Pandas, a data manipulation package.

Lesson Title	Learning Outcomes
WHY PYTHON PROGRAMMING	➔ Gain an overview of what you'll be learning and doing in the course ➔ Understand why you should learn programming with Python
DATA TYPES AND OPERATORS	➔ Represent data using Python's data types: integers, floats, booleans, strings, lists, tuples, sets, dictionaries, compound data structures ➔ Perform computations and create logical statements using Python's operators: Arithmetic, Assignment, Comparison, Logical, Membership, Identity ➔ Declare, assign, and re-assign values using Python variables ➔ Modify values using built-in functions and methods ➔ Practice whitespace and style guidelines
CONTROL FLOW	➔ Write conditional expressions using if statements and boolean expressions to add decision making to your Python programs ➔ Use for and while loops along with useful built-in functions to iterate over and manipulate lists, sets, and dictionaries ➔ Skip iterations in loops using break and continue ➔ Condense for loops to create lists efficiently with list comprehensions
FUNCTIONS	➔ Define your own custom functions ➔ Create and reference variables using the appropriate scope ➔ Add documentation to functions using docstrings ➔ Define lambda expressions to quickly create anonymous functions ➔ Use iterators and generators to create streams of data
SCRIPTING	➔ Install Python 3 and Anaconda, and set up your programming environment

	➔ Run and edit python scripts ➔ Interact with raw input from users ➔ Identify and handle errors and exceptions in your code ➔ Open, read, and write to files ➔ Find and use modules in Python Standard Library and third-party libraries ➔ Experiment in the terminal using a Python Interpreter
NUMPY	➔ Create, access, modify, and sort multidimensional NumPy arrays (ndarrays) ➔ Load and save ndarrays ➔ Use slicing, boolean indexing, and set operations to select or change subsets of an ndarray ➔ Understand difference between a view and a copy of ndarray ➔ Perform element-wise operations on ndarrays ➔ Use broadcasting to perform operations on ndarrays of different sizes.
PANDAS	➔ Create, access, and modify the main objects in Pandas, Series and DataFrames ➔ Perform arithmetic operations on Series and DataFrames ➔ Load data into a DataFrame ➔ Deal with Not a Number (NaN) values

Project 3: Post your work on Github (12 hours)

In this project, you will learn important tools that all programmers use. First, you'll get an introduction to working in the terminal. Next, you'll learn to use git and Github to manage versions of a program and collaborate with others on programming projects. In this project you will add a completed project on GitHub, work with branches, edit a README file and project files, merge branches, stage and commit your changes to your project GitHub repository.

Supporting Lesson Content: Introduction to Shell and Version Control

Lesson Title	Learning Outcomes
SHELL WORKSHOP	➔ The Unix shell is a powerful tool for developers of all sorts. Get a quick introduction to the basics of using it on your computer.

PURPOSE & TERMINOLOGY	➔ Learn why developers use version control and discover ways you use version control in your daily life ➔ Get an overview of essential Git vocabulary ➔ Configure Git using the command line
CREATE A GIT REPO	➔ Create your first Git repository with git init ➔ Copy an existing Git repository with git clone ➔ Review the current state of a repository with the powerful git status
REVIEW A REPO'S HISTORY	➔ Review a repo's commit history git log ➔ Customize git log's output using command line flags in order to reveal more (or less) information about each commit ➔ Use the git show command to display just one commit
ADD COMMITS TO A REPO	➔ Master the Git workflow and make commits to an example project ➔ Use git diff to identify what parts of a file have been changed in a commit ➔ Learn how to mark files as "untracked" using .gitignore
TAGGING, BRANCHING, AND MERGING	➔ Tagging, Branching, and Merging ➔ Organize your commits with tags and branches ➔ Jump to particular tags and branches using git checkout ➔ Learn how to merge together changes on different branches and crush those pesky merge conflicts
UNDOING CHANGES	➔ Learn how and when to edit or delete an existing commit ➔ Use git commit's --amend flag to alter the last commit ➔ Use git reset and git revert to undo and erase commits
WORKING WITH REMOTES	➔ Create remote repositories on GitHub ➔ Learn how to pull and push changes to the remote repo
WORKING ON ANOTHER DEVELOP'S REPOSITORY	➔ Learn how to fork another developer's project ➔ Use GitHub to contribute to a public project
STAYING IN SYNC WITH A REMOTE REPOSITORY	➔ Learn how to sync new changes to a forked remote repository, retrieve and sync updates ➔ Create pull requests and squash commits with git rebase