DS Learning Objectives

Week 1: Beginning Programming (Tasks 1 - 7)
By the end of Week 1, students will be able to:

1. Set up and navigate the VS Code IDE, mastering an essential tool that enhances coding efficiency—a skill that is indispensable in the tech industry for both small-scale projects and enterprise-level software development.
2. Develop competence in Python fundamentals through the creation of interactive programs that perform data manipulation, focusing on string operations and indexing—essential skills for data science tasks such as data cleaning and preprocessing.
3. Design algorithms using loops and conditional statements, and build a financial calculator using the DS problem-solving methodology, thereby learning to translate mathematical models into code—a vital skill for analysing financial data and building predictive models.
4. Integrate the DRY principle to craft clean and maintainable code, alongside debugging skills for swift error resolution, which significantly reduces development time and is highly valued in technology roles.
5. Develop a targeted career plan linking programming competencies with data science roles, fostering a strategic approach to entering the job market and securing roles that require a strong foundation in coding.
6. Leverage the principles of Computational Thinking to deconstruct and solve complex problems, a critical skill set that underpins innovative solutions in data science and technology.
7. Translate computational problems into pseudo-code for structured planning before coding, enhancing the ability to write efficient code and effectively communicate algorithms—a skill that provides a competitive edge in technical interviews and collaborative development environments.
8. Understand the concept of an incremental, iterative approach to problem solving and the specific problem-solving methodology that will be followed throughout the bootcamp and apply it when solving problems.

Week 2: Iteration (Task 8)
By the end of Week 2, students will be able to:

1. Implement `for` loops to streamline data wrangling tasks, a fundamental skill for handling large datasets and preparing data for analysis.
2. Apply debugging techniques using trace tables and IDE tools to ensure code accuracy, a vital step in developing reliable data models and algorithms.
3. Differentiate `while` and `for` loops to select the most efficient iteration method, optimising code for data-intensive operations in predictive modelling.
4. Utilise 'break' and 'continue' within loops to refine data processing loops, essential for managing large data streams and maintaining performance in data applications.
5. Construct programs, following the DS problem-solving methodology, using loops and conditionals to automate data sorting and pattern recognition, key in extracting insights and trends from raw data.

6. Create an interactive speed typing game following the DS problem-solving methodology, which serves as a practical analogy for real-time data collection and user interaction handling in data-driven applications.

Week 3: Defensive Programming (Tasks 9 - 10)
By the end of Week 3, students will be able to:

1. Apply defensive programming strategies to write robust code that anticipates and handles errors, a practice critical for maintaining the integrity of data analysis projects.
2. Utilise conditional statements and exception handling to validate input and manage errors, skills essential for ensuring the accuracy and reliability of data outputs.
3. Design custom exceptions in Python to provide clear error reporting, enhancing the maintainability of code used in data pipelines and applications.
4. Analyse code for potential vulnerabilities and implement defensive measures, preparing students to produce secure and stable code for data science tasks.
5. Compare debugging methods to effectively resolve code issues, enabling the smooth execution of data processing and analysis workflows.
6. Optimise professional profiles and documents to showcase technical skills and data science projects, aligning with the expectations of potential employers and the industry standards.

Week 4-5: Sequences (Tasks 11 - 13)
By the end of Week 4, students will be able to:

1. Master string manipulation in Python through functions such as `upper()`, `lower()`, `replace()`, and `split()`, and apply these to alter and format string data, a fundamental skill for cleaning and preparing textual data for analysis.

By the end of Week 5, students will be able to:

2. Construct Python programs, following the DS problem-solving methodology, that use lists and dictionaries for data storage and retrieval, simulating real-world data structuring essential for developing data models and analytics tools.
3. Develop, using the DS problem-solving methodology, and execute a program that performs data calculations using Python's data structures, mirroring the analytical processes in managing and interpreting large datasets in a business context, such as inventory valuation.
4. Prepare for the data science job market by applying to at least 5 junior tech roles, constructing a well-represented professional profile, and articulating their skill set with a focus on data manipulation and analysis in various business sizes and sectors.

Week 7-8: Functions (Task 14)
By the end of Week 7, students will be able to:

1. Demonstrate proficiency in writing user-defined functions in Python, focusing on syntax and simple use cases such as calculating travel-related expenses, to establish a foundation for automating data analysis tasks.
2. Employ function design principles, including naming conventions and documentation, to create code that is readable and maintainable, setting the stage for professional coding practices.

By the end of Week 8, students will be able to:

1. Articulate the advantages of functions in programming for modularity and reusability, and identify their significance in building scalable data science applications.
2. Apply scope and parameter best practices in function creation to ensure clean, efficient, and error-free code that is fundamental for complex data manipulation and algorithm development.
3. Integrate control structures within functions to process user inputs dynamically, which is a key component in developing interactive data-driven applications.

Weeks 9-10: Data Visualization and Analysis (Tasks 15 - 16)

By the end of Week 9, students will be able to:

1. Illustrate the ability to select and create appropriate data visualisations, like bar graphs and line graphs, for varied data sets, enhancing interpretative skills for data storytelling.
2. Analyse and infer trends from graphical data representations, answering key questions to draw evidence-based conclusions that inform data-driven decisions.
3. Connect data visualisation outcomes with real-world events to provide a contextual narrative for observed data trends, improving the ability to communicate complex data insights.

By the end of Week 10, students will be able to:

1. Perform data selection and sorting using Pandas in Jupyter Notebook, a crucial skill for organising and preparing data for in-depth analysis.
2. Interpret multidimensional data visualisations, such as scatterplot matrices, to evaluate correlations and detect outliers, fostering a deeper understanding of multivariate data relationships.
3. Utilise statistical methods to analyse datasets, calculating key metrics like averages and totals, to distil actionable insights specific to demographic or attribute-focused inquiries.
4. Compile and document data analysis processes and outcomes within Jupyter Notebook, emphasising clarity in visualisations and accuracy in interpretations for professional presentation and review.

Week 11-13: Artificial Intelligence (Tasks 17 - 22)

By the end of Week 11, students will be able to:

1. Distinguish between supervised, unsupervised, and semi-supervised learning by identifying key characteristics and application scenarios, directly applying this knowledge to categorise real-world problems.

2. Select and justify machine learning algorithms for various case studies, with an emphasis on understanding the suitability of regression or classification approaches based on dataset analysis.

By the end of Week 12, students will be able to:

1. Construct and assess linear regression models using principles such as feature selection, model fitting, and loss function minimization to forecast outcomes and quantify model accuracy.
2. Employ NLP techniques using spaCy for tasks such as tokenization, named entity recognition, and semantic similarity assessments, culminating in the creation of a basic film recommendation engine using word vectors.

By the end of Week 13, students will be able to:

1. Implement sentiment analysis on textual data, detailing the full process from data preprocessing to result interpretation, and documenting findings in a structured report.
2. Refine personal marketing tools—CVs, job applications, pitches—and articulate AI and NLP skills, preparing for interview scenarios by practising responses based on the STAR method.
3. Demonstrate proficiency in AI and NLP concepts through a coding challenge, explaining the logic and solutions in a clear and comprehensible manner.

Week 14: Version Control (Tasks 23 - 24)

By the end of Week 14, students will be able to:

1. Install and configure Git, demonstrating fundamental version control operations with commands like “git init”, “git add”, and “git commit”, crucial for tracking changes and collaboration in data science projects.
2. Create and manage a professional GitHub account, showcasing the ability to maintain well-documented, structured repositories, vital for data science portfolio presentation and project accessibility.
3. Execute and manage branches with Git to ensure proper version control practices, facilitating simultaneous development streams in team-based data science projects.
4. Perform pull requests and merges on GitHub, reinforcing collaborative workflows and code integration skills, pivotal in data science teams for consolidating analyses and findings.
5. Enhance a GitHub profile and landing page to effectively display data science projects and technical skills, directly targeting recruitment and collaborative opportunities in the field.
6. Identify and assess alternative version control systems and their applicability to data science, providing insight into their management capabilities for large data sets and analytical pipelines.
7. Articulate the advantages of a Distributed Version Control System's structure for maintaining the integrity and traceability of data transformations in data science workflows.