Review what you learned

After completing this course, you learned many facts about data science methodology. Here are 14 key, high-level takeaway facts you’ll want to remember from this course.

* Foundational methodology, a cyclical, iterative data science methodology developed by John Rollins, consists of 10 stages, starting with Business Understanding and ending with Feedback.
* CRISP-DM, an open source data methodology, combines several data-related methodology stages into one stage and omits the Feedback stage resulting in a six-stage data methodology.
* The primary goal of the Business Understanding stage is to understand the business problem and determine the data needed to answer the core business question.
* During the Analytic Approach stage, you can choose from descriptive diagnostic, predictive, and prescriptive analytic approaches and whether to use machine learning techniques.
* During the Data Requirements stage, scientists identify the correct and necessary data content, formats, and sources needed for the specific analytical approach.
* During the Data Collection stage, expert data scientists revise data requirements and make critical decisions regarding the quantity and quality of data. Data scientists apply descriptive statistics and visualization techniques to thoroughly assess the content, quality, and initial insights gained from the collected data, identify gaps, and determine if new data is needed, or if they should substitute existing data.
* The Data Understanding stage encompasses all activities related to constructing the data set. This stage answers the question of whether the collected data represents the data needed to solve the business problem. Data scientists might use descriptive statistics, predictive statistics, or both.
* Data scientists commonly apply Hurst, univariates, and statistics such as mean, median, minimum, maximum, standard deviation, pairwise correlation, and histograms.
* During the Data Preparation stage, data scientists must address missing or invalid values, remove duplicates, and validate that the data is properly formatted. Feature engineering and text analysis are key techniques data scientists apply to validate and analyze data during the Data Preparation stage.
* The end goal of the Modeling stage is that the data model answers the business question. During the Modeling stage, data scientists use a training data set. Data scientists test multiple algorithms on the training set data to determine whether the variables are required and whether the data supports answering the business question. The outcome of those models is either descriptive or predictive.
* The Evaluation stage consists of two phases, the diagnostic measures phase, and the statistical significance phase. Data scientists and others assess the quality of the model and determine if the model answers the initial Business Understanding question or if the data model needs adjustment.
* During the Deployment stage, data scientists release the data model to a targeted group of stakeholders, including solution owners, marketing staff, application developers, and IT administration.,
* During the Feedback stage, stakeholders and users evaluate the model and contribute feedback to assess the model’s performance.
* The data model’s value depends on its ability to iterate; that is, how successfully the data model incorporates user feedback.

Module 2 Summary: Python Data Structures

Congratulations! You have completed this module. At this point, you know that:

* In Python, we often use tuples to group related data together.Tuples refer to ordered and immutable collections of elements.
* Tuples are usually written as comma-separated elements in parentheses “()".
* You can include strings, integers, and floats in tuples and access them using both positive and negative indices.
* You can perform operations such as combining, concatenating, and slicing on tuples.
* Tuples are immutable, so you need to create a new tuple to manipulate it.
* Tuples, termed nesting, can include other tuples of complex data types.
* You can access elements in a nested tuple through indexing.
* Lists in Python contain ordered collections of items that can hold elements of different types and are mutable, allowing for versatile data storage and manipulation.
* A list is an ordered sequence, represented with square brackets "[]".
* Lists possess mutability, rendering them akin to tuples.
* A list can contain strings, integers, and floats; you can nest lists within it.
* You can access each element in a list using both positive and negative indexing.
* Concatenating or appending a list will result in the modification of the same list.
* You can perform operations such as adding, deleting, splitting, and so forth on a list.
* You can separate elements in a list using delimiters.
* Aliasing occurs when multiple names refer to the same object.
* You can also clone a list to create another list.
* Dictionaries in Python are key-value pairs that provide a flexible way to store and retrieve data based on unique keys.
* Dictionaries consist of keys and values, both composed of string elements.
* You denote dictionaries using curly brackets.
* The keys necessitate immutability and uniqueness.
* The values may be either immutable or mutable, and they allow duplicates.
* You separate each key-value pair with a comma, and you can use color highlighting to make the key more visible.
* You can assign dictionaries to a variable.
* You use the key as an argument to retrieve the corresponding value.
* You can make additions and deletions to dictionaries.
* You can perform an operation on a dictionary to check the key, which results in a true or false output.
* You can apply methods to obtain a list of keys and values in a dictionary.
* Sets in Python are collections of unique elements, useful for tasks such as removing duplicates and performing set operations like union and intersection. Sets lack order.
* Curly brackets "{}" are helpful for defining elements of a set.
* Sets do not contain duplicate items.
* A list passed through the set function generates a set containing unique elements.
* You use “Set Operations” to perform actions such as adding, removing, and verifying elements in a set.
* You can combine sets using the ampersand "&" operator to obtain the common elements from both sets.

Module 3 Summary: Python Programming Fundamentals

Congratulations! You have completed this module. At this point, you know that:

* Python conditions use “if” statements to execute code based on true/false conditions created by comparisons and Boolean expressions.
* Comparison operations require using comparison operators such as == (equal to), > (greater than), and < (less than).
* Python uses the "!=" operator to determine whether two values are not equal.
* You can compare integers, strings, and floats.
* Python branching directs program flow by using conditional statements (for example, if, else, elif) to execute different code blocks based on conditions or tests.
* You can use the "if" statement with conditions to define actions if true.
* To perform actions when all previous conditions are false, you can use the "else" statement without a condition.
* The elif statement allows for additional checks only if the initial condition is false.
* To execute various operations on Boolean values, we use Boolean logic operators.
* Python loops are control structures that automate repetitive tasks and iterate over data structures like lists or dictionaries.
* The range() function generates a sequence of numbers with a specified start, stop, and step value for loops in Python.
* A for loop in Python iterates over a sequence, such as a list, tuple, or string, and executes a block of code for each item in the sequence.
* A while loop in Python executes a block of code as long as a specified condition remains true.
* Python functions are reusable code blocks that perform specific tasks, take input parameters, and often return results, enhancing code modularity and reusability.
* You may or may not have written the codes that are often included in functions.
* Python has a set of built-in functions such as "len" to find the length of a sequence or "sum" to find the total sum of a sequence.
* The "sorted" function creates a new sorted list, while "sort" sorts items in the original list.
* You can also create your own functions in Python.
* To ensure clarity and organization and facilitate understanding and maintenance of the code, developers must document functions using a documentation string enclosed in three quotes.
* The help command will return the documentation defined for a particular function.
* A function can have multiple parameters.
* If a function does not include a return statement, it returns None by default.
* You can use the "pass" keyword in a function to indicate that it does nothing (a placeholder for future code).
* A function will usually perform more than one task.
* In Python, the scope of a variable determines where you can access or modify that variable. Global scope allows access from anywhere, while local scope restricts it to a block or function.
* In Python, a programmer defines a local variable within a specific block or function, which can only be accessed or modified within that block or function.
* In Python, a global variable is a variable defined at the top level of a program that any part of the code can access or modify.
* Exception handling in Python is a mechanism for managing and responding to errors and exceptions that may occur during program execution, preventing them from crashing the program.
* In Python, you use the "try-except" statement to attempt a block of code and specify alternative actions to execute if an error occurs, allowing you to handle exceptions.
* In Python, you use the "try-except-else" statement to attempt a block of code, handle exceptions in the "except" block, and execute code in the "else" block when no exceptions occur.
* Python developers use the "try-except-else-finally" statement to attempt a block of code, catch exceptions in the "except" block, execute code in the "else" block when no exceptions occur, and ensure that the "finally" block always runs, regardless of whether exception was raised or not.
* In Python, objects are instances of classes that encapsulate data and behavior, serving as the foundation for creating and working with various data types and custom data structures.
* To determine the type of an object in Python, you can use the `type()` command.
* Methods may modify an object’s internal state, but the object’s type usually remains the same.
* Classes in Python are blueprints for creating objects, defining their attributes and methods, enabling code organization, and object-oriented programming.
* Function "init" is a special method used to initialize data attributes.
* We can create instances of a class in Python.
* Data attributes consist of the data defining the objects.
* Methods are functions that interact and change the data attributes.
* The method has a function that requires the self as well as other parameters.
* You can use the Union function to combine two sets, including both the common and unique elements from both sets.
* The sub-set method is used to determine if two or more sets are subsets.

Lesson Summary

Congratulations! You have completed this lesson. At this point in the course, you know:

* Each line in a dataset is a row, and commas separate the values.
* To understand the data, you must analyze the attributes for each column of data.
* Python libraries are collections of functions and methods that facilitate various functionalities without writing code from scratch and are categorized into Scientific Computing, Data Visualization, and Machine Learning Algorithms.
* Many data science libraries are interconnected; for instance, Scikit-learn is built on top of NumPy, SciPy, and Matplotlib.
* The data format and the file path are two key factors for reading data with Pandas.
* The **read\_CSV** method in Pandas can read files in CSV format into a Pandas DataFrame.
* Pandas has unique data types like object, float, Int, and datetime.
* Use the **dtype** method to check each column’s data type; misclassified data types might need manual correction.
* Knowing the correct data types helps apply appropriate Python functions to specific columns.
* Using **Statistical Summary** with **describe()** provides count, mean, standard deviation, min, max, and quartile ranges for numerical columns.
* You can also use **include='all'** as an argument to get summaries for object-type columns.
* The statistical summary helps identify potential issues like outliers needing further attention.
* Using the **info() Method** gives an overview of the top and bottom 30 rows of the DataFrame, useful for quick visual inspection.
* Some statistical metrics may return "NaN," indicating missing values, and the program can’t calculate statistics for that specific data type.
* Python can connect to databases through specialized code, often written in Jupyter notebooks.
* SQL Application Programming Interfaces (APIs) and Python DB APIs (most often used) facilitate the interaction between Python and the DBMS.
* **SQL APIs** connect to DBMS with one or more API calls, build SQL statements as a text string, and use API calls to send SQL statements to the DBMS and retrieve results and statuses.
* **DB-API**,Python's standard for interacting with relational databases,uses **connection objects** to establish and manage database connections and **cursor objects** to run queries and scroll through the results.
* Connection Object methods include the cursor(), commit(), rollback(), and close() commands.
* You can import the database module, use the **Connect API** to open a connection, and then create a cursor object to run queries and fetch results.
* Remember to close the database connection to free up resources.