SQL & Python Package (180 hours)

Course List:

1. SQL Fundamentals [70 hours]
2. Introduction to Python [40 hours]
3. Working with Text in Python [20 hours]
4. Plotting and Drawing with Python [20 hours]
5. Working with Files in Python [10 hours]
6. Data Visualization with Python [20 hours]

Package Description:

You'll begin your Data Analyst training with an **SQL Fundamentals** course where you learn about data and databases, with emphasis on Relational Database Management Systems (RDBMSs), which are used in virtually all industries and organizations to store data about employees, products, services, inventory, financial transactions, etc. You learn how RDBMSs work, how to make basic queries, how to do extracts, how to use aggregate functions, how to create and manage tables, and how to use basic joins.

Your Python training then begins with **Introduction to Python**. This course provides a detailed and comprehensive overview of the Python programming language. You learn Python by solving programming problems of gradually increasing complexity, using simple calculations, loops, conditions, local and global variables, functions, exceptions, and recursion. You also become proficient in working with fundamental Python data structures, including tuples, lists, and dictionaries. Throughout the course, you are developing a good Python coding style and other good coding habits.

More than 80% of work computers do is processing text. Therefore, in the course **Working With Text in Python**, you learn how to process, analyze, and manipulate text strings with Python.

Python is known for its powerful graphic capabilities. In the course **Plotting and Drawing with Python**, you learn how to use the powerful Python library Matplotlib for plotting and drawing.

Most data is stored in files. Therefore, the course **Working with Files in Python** teaches you how to open files, read data from them, process the data, and write to files.

nclab
Learning for Industry 4.0

The world we live in is driven by data. Therefore, the course **Data Visualization with Python** teaches you how to visualize data in the form of simple graphs, bar charts, pie charts, color maps, surface plots, wireframe plots, and contour plots. You also learn how to visualize data on 2D Cartesian grids and unstructured triangulations.

Learning for Industry 4.0

Package Syllabi:

SQL Fundamentals

Unit 1: Introduction to Data, Databases and SQL

Section 1

- Define and explain the purpose of data and the most widely used data types.
- Understand the hierarchy of units used to calculate data size.
- Calculate the size of data.
- Define and explain the purpose of databases.
- Understand relational vs. non-relational databases.
- Use design and ethics principles to guide database use, including referential integrity and ACID.
- Know what Structured Query Language (SQL) is and its history.
- Understand the main differences between various SQL flavors: PostgreSQL, MySQL, and SQLite.
- Know further details about PostgreSQL, which will be used in this course.

Section 2

- Examine the structure of a SQL database.
- Use the SELECT statement to display all columns of a table.
- Know that SQL is case-insensitive.
- Know the purpose of the keyword NULL and SQL's three-valued logic.
- Order query results using ORDER BY.
- Limit the number of results using the keyword LIMIT.

Section 3

- Use the SELECT statement to only display one column of interest.
- Use the SELECT statement to display two or more columns.
- Change the order of the displayed columns if needed.
- Skip some rows in the output using the keyword OFFSET.
- Order multiple columns at the same time.

Section 4

- Filter the results of queries using the keyword WHERE.
- Access a table in a schema by typing schema.table.
- Remove duplicate results with SELECT DISTINCT.

Section 5

- Use aggregate functions:
    - Function COUNT() to count results.
    - Functions MIN(), MAX() to find minimum and maximum values.

    - Functions SUM(), AVG() to calculate the sum and average of values, respectively.
- Narrow down queries using WHERE … LIKE … and WHERE … BETWEEN ….
- Combine multiple conditions in the search using the keywords AND, OR, NOT.

Unit 2: SQL Queries

Section 6

- Practice basic SQL queries by exploring the Northwind database and in particular the table Employees.

Section 7

- Practice basic SQL queries by exploring the table Products of the Northwind database.

Section 8

- Practice basic SQL queries by exploring the table Products of the Northwind database.
- Use SQL for calculations.
- Use the WHERE … IN … clause.

Section 9

- Practice basic SQL queries by exploring the table Order Details of the Northwind database.
- Create new columns and name them using the keyword AS.
- Group results for better readability using the GROUP BY … and GROUP BY … HAVING … clauses.

Section 10

- Practice basic SQL queries by exploring the Customers table of the Northwind database.

Unit 3: Creating and Managing Tables

Section 11

- Explain the different data types in the SQL standard and in PostgreSQL.
- Create and drop schemata.
- Learn about the two schemata in NCLab where trainees can store their own data.
- Create tables.

Section 12

- Insert complete rows (without using column names).
- Insert incomplete rows (without using column names).
- Insert incomplete rows (using the names of columns).

Section 13

- Insert multiple rows at once.
- Create a new table by copying an existing table.
- Create a new empty table which has the same structure as an existing table.
- Create a new table by copying selected rows from an existing table.

Section 14

- Insert selected rows from a table into an existing table.
- Use the powerful statement ALTER TABLE to modify tables, such as:
    - renaming tables,
    - adding, renaming, dropping (= deleting) columns,
    - changing data types of columns, etc.
- Delete all rows from a table, selected rows, or entire tables.
- Delete selected rows from a table.
- Delete entire tables.
- Know that the result of the VALUES clause and of the SELECT statement is a TABLE.

Section 15

- Define constraints and specify default values.

Unit 4: Joining Tables

Section 16

- Combine data from two different tables using the inner join operation.
- Know that initially, the tables are matched based on a shared column of the same name.
- Write inner joins in three simple steps:
    - Use a basic SELECT query followed by the columns one wants to display,
    - Add INNER JOIN followed by the second table name,
    - Add USING followed by the name of the shared column in parentheses.
- Join three and more tables, and know that joining multiple tables is equally simple as joining just two.

Section 17

- Understand the structure of the tables in the schema World.
- Perform an inner join of tables based on columns of different names, using the keyword ON.
- Abbreviate table names using the keyword AS.
- Know that some keywords (for example, AS) can be omitted.
- Write inner joins using an alternative (implicit) syntax without the keywords INNER JOIN and USING/ON.

Section 18

- Practice inner joins by solving practical tasks related to the schema World.

Section 19

- Review the schema Northwind, and then perform calculations which require combining data from various tables in this schema.
- Combine inner joins with filtering, grouping and other basic SQL techniques.

Section 20

- Perform inner joins with calculations which combine data from various tables in the schema Northwind.
- Combine inner joins with filtering, grouping and other basic SQL techniques.

Introduction to Python

Unit 1

Section 1

- Know the history of Python. Know that Python can be used as a powerful scientific calculator
- Know the arithmetic operators +, -, *, /, **
- Know the priority (precedence) of arithmetic operators, parentheses
- Know Python libraries and the old and new ways of importing them
- Import the Fractions library and work with fractions
- Use the built-in function help()
- Define numerical (integer and real) variables and text strings
- Import Numpy and use its functionality
- Display results with the built-in function print()

Section 2

- Use the floor division operator //, the modulo operator %, and the power operator **
- Use the operator // with negative and real numbers
- Understand scientific notation
- Real numbers are not represented exactly in the computer
- Use the assignment operator = and the comparison operator ==
- Work with the Boolean values True and False
- Know that the result of the comparison operator == is either True or False
- Know that one should never use the operator == to compare real numbers
- Know the correct way to check if two real numbers are the same
- Use the built-in function abs() to calculate the absolute value of numbers
- Know that the result of the comparison operators <, >, <=, >=, != is either True or False
- Know how to reach the limit of finite computer arithmetic on any computer
- Use the arithmetic operators +=, -=, *=, /=, //=, %= and **=

- Know binary vs. unary operators
- Work with the most important units of data size including b, KB, MB and GB
- Know the difference between KB and kB

Section 3

- Define and call functions
- Know the importance of writing docstrings and commenting your code
- Know the difference between function parameters and arguments
- Know the difference between standard (positional) and named (keyword) arguments
- Know that functions can return multiple values
- Know global and local scopes and global and local variables
- Know the keyword 'global'
- Know that functions should never change the values of global variables
- Work with tuples, unpack them, and access individual items via indices
- Parse tuples one item at a time using the for loop
- Know the range() function and use nested for loops

Section 4

- Create empty and non-empty lists
- Obtain the length of lists, function len()
- Add items to lists, methods append() and insert()
- Remove items from lists, methods pop() and remove(), keyword del
- Add lists and multiplying them with integers
- Know the mutability of lists
- Parse lists with the for loop
- Access individual list items via their indices
- Use the while loop
- Slice lists, create copies, and create reversed copies of lists via slicing
- Reverse lists and sort them, and use the list methods reverse() and sort()
- Reverse lists and sort them, and use the built-in functions reversed() and sorted()
- Make list and tuple items unique

Section 5

- Work with Boolean expressions and variables
- The if, if-else and if-elif-else statements
- Generate random integers and real numbers
- Use the keyword 'in' to check if a given item is present in a tuple or list
- Use the method count() to count occurrences of given items in tuples and lists
- Use the method index() to obtain positions of given items in tuples and lists
- Work with the Boolean operators and, or, and not
- Chain arithmetic comparison operators
- Use the break and continue statements in loops
- Work with infinite while loops

- Use the command ‘pass’
- Use the else branch with for and while loops
- Use the math module, and know how it compares to Numpy

Unit 2

Section 6

- Understand tuples, and what kind of data they are intended to store
- Define empty and nonempty tuples
- Know whether tuples can be changed at runtime
- Unpack tuples, and obtain their length
- Access individual items in tuples via indices
- Go through tuples one item at a time using the for loop
- Use the built-in functions sum(), max() and min()
- Use for loops in combination with the built-in function range()
- Cast a range object to a tuple using the built-in function tuple()
- Slice tuples (extract their parts)
- Make copies and reversed copies of tuples via slicing
- Make a copy with the function copy(), and a reversed copy with the built-in function reversed()
- Use the built-in function zip() to go through two tuples simultaneously
- Cast a zip object to a tuple of pairs using the built-in function tuple()
- Use the keyword in to check for the presence of items in tuples
- Use the tuple method count() to count the occurrences of items in tuples
- Use the tuple method index() to obtain the position of items in tuples

Section 7

- Understand lists, and what kind of data they are intended to store
- Know the common functionality of lists and tuples:
    - Define empty and nonempty lists, unpack lists, obtain their length, and access individual list items via indices
    - Go through lists one item at a time using the for loop
    - Use the built-in functions sum(), max(), and min()
    - Cast a range object to a list using the built-in function list()
    - Slice lists, make copies and reversed copies of lists, and use the function reversed()
    - Use the zip() function to go through two lists simultaneously
    - Cast a zip object to a list of pairs using the function list()
    - Use the list methods count() and index(), and use the keyword in to check for items
- Know some new functionality of lists which is not available in tuples:
    - Cast lists (type list) to tuples (type tuple), and vice versa
    - Append items with the method append(), insert items with the method insert(), and modify list items
    - Add lists, multiply lists with integers, update lists, and use the method extend().

  - Reverse lists in place with the method reverse()
  - Make sorted copies with the built-in function sorted(), and sort lists in place with the method sort()
  - Empty lists both using L *= 0 and the method clear()

Section 8

- Remove and return items from lists with the method pop()
- Remove and destroy list items with the method remove()
- Remove and destroy list items and entire lists with the keyword del
- Know whether lists should be modified while going through them with a for loop
- Check the types of variables with the built-in function isinstance()
- Know the difference between mutable and immutable objects in Python
- Know whether lists (list) are mutable objects in Python
- Know whether tuples (tuple), integers (int), real numbers (float), text strings (str) and Booleans (bool) are immutable
- Use basic list comprehension
- Use comprehension with conditions
- Know ternary Boolean expressions, and how they differ from the if-else statement
- Use comprehension with ternary Boolean expressions
- Understand exceptions, and how to handle them with the basic try-except statement

Section 9

- Create empty and non-empty dictionaries
- Know that dictionary items are key:value pairs
- Know that keys are unique but values can be repeated
- Add/insert and overwrite items
- Remove and return items using the methods pop() and popitem()
- Empty a dictionary with the method clear()
- Use the keyword del to remove and destroy items and entire dictionaries
- Access values using keys
- Go through a dictionary using a for loop
- Extract from a dictionary the lists of keys, values, and items
- Remove repetitions by casting tuples and lists to sets
- Zip the lists of keys and values to create a dictionary
- Reverse a dictionary using comprehension
- Combine dictionaries and find keys which correspond to repeated values
- Know whether dict is a mutable type in Python
- Work with sets and frozen sets
- Include assertions and raise exceptions

Working with Text in Python

nclab
Learning for Industry 4.0

Section 1

- Define text strings, using single and double quotes
- Know problems associated with trailing spaces, and know the function repr()
- Compare text strings with the == operator
- Know the optional parameters ‘sep’ and ‘end’ for the built-in function print()
- Add text strings and multiply them with positive integers
- Update text string variables with the operators += and *=
- Know about the PEP8 — Style Guide for Python Code

Section 2

- Combine single and double quotes in text strings
- Obtain the length of text strings using the function len()
- Work with the special characters \n, \” and \’
- Cast numbers to text strings using the function str()
- Insert numbers into text strings
- Cast text strings to numbers using the functions int() and float()
- Use interactive keyboard input
- Display the type of variables using the function type()
- Check the type of variables at runtime, using the function isinstance()
- Use the text string methods lower(), upper(), and title()
- Know that text string methods never change the original text string
- Clean text strings with the methods rstrip(), lstrip(), and strip()
- Split a text string into a list of words using the method split()
- Check for substrings using the keyword ‘in’
- Make a text search case-insensitive
- Count the occurrences of substrings in text strings using the method count()

Section 3

- Work with the ASCII table using the functions ord() and chr()
- Search for and replace substrings in text strings using the method replace()
- Zip two lists and use the for loop to parse them at the same time
- Erase parts of text strings
- Clean text strings from unwanted characters
- Swap the contents of two text strings
- Swap two substrings in a text string
- Work with useful text string methods such as isalpha(), isalnum(), isdigit() etc.

Section 4

- Know that text strings are immutable objects in Python
- Obtain the memory address of Python objects using the function id()
- Access individual characters in text strings via their indices
- Slice text strings and reverse them

- Retrieve and work with system date and time
- Obtain the position of a substring in a given text string, method index()
- Count the occurrences of a substring in a given text string, method count()
- Translate decimal numbers into binary format, function bin()
- Understand how text strings are represented in computer memory
- Compare text strings using the operators <, <=, >, >=
- Create text characters which are not present on the keyboard

Section 5

- Know what regular expressions are and what are they useful for
- Know Python's regular expressions module 're'
- Use the functions search(), match(), and findall()
- Know greedy and non-greedy repeating patterns
- Use character classes and groups of characters
- Work with the most important metacharacters and special sequences
- Mine unknown file names and email addresses from text data

Plotting and Drawing with Python

Section 1

- Import the Matplotlib and Numpy libraries and abbreviate their names
- Define lines and polylines using X and Y arrays
- Create a plot using the function plot()
- Assign colors to objects
- Display the plot using the function show()
- Display two or more objects simultaneously
- Make both axes equally-scaled using axis("equal")
- Hide axes using axis("off")
- Fill closed areas with color using the function fill()
- Change the width of lines via the optional keyword argument linewidth or lw
- Interrupt lines with the keyword None
- Draw hollow objects using curves with opposite orientations

Section 2

- Use the Numpy function linspace() to create equidistant grids
- Use the arrays created by linspace() in calculations
- Plot graphs of functions using the linspace() array as the X variable
- Draw circles centered at (Cx, Cy) using the formula x = Cx + R*cos(t), y = Cy + R*sin(t)
- Draw regular polygons by reducing the number of edges of the circle

- Draw circular arcs
- Draw ellipses using the formula x = Cx + Rx*cos(t), y = Cy + Ry*sin(t)
- Draw spirals using the formula x = Cx + t*cos(t), y = Cy + t*sin(t)
- Cast the numpy.ndarray to a list and altering it as needed

Section 3

- Automate plotting with the for loop and nested for loops
- Work with indices in 1D and 2D arrays
- Set X and Y limits in Matplotlib plots via the axis() function
- Set titles in Matplotlib plots

Working with Files in Python

Section 1

- Open a text file for reading and parsing it line-by-line
- Know that the recommended way to work with files is using the with statement
- Know that the with statement automatically closes the file at the end
- Know that newline characters '\n' need to be removed from lines when parsing a text file line by line
- Know that f.tell() returns the current position of the file pointer
- Know that f.seek(x) sets the position of the file pointer to x
- Know that f.seek(0) rewinds the text file f back to the beginning
- Know that f.read(x) reads and returns the next x characters from the file f, and moves the file pointer by x positions forward
- Know that f.readline() will read a single line from an open text file f
- Know that f.readline() can be used to skip headings, comments, etc

Section 2

- Know that ASCII art gained a lot of popularity in the early era of the Internet
- Know that the first version of the Internet, created for military purposes, was named ARPANET
- Know that initially, the Internet was pure text - no graphics, sound or video
- Open a text file for writing and write text strings to it
- Use the file flags 'w+', 'r+' and 'a+'
- Know that writing to a text file can fail for various reasons
- Extract all lines from a text file at once as a list of text strings
- Write a list of text strings to a text file at once
- Read the whole text file into a text string
- Know that the backslash character '\' must be used in the form '\\'

---

nclab
Learning for Industry 4.0

Data Visualization with Python

Section 1

- Read CSV files manually as well as with Pandas
- Manipulate Pandas DataFrames
- Create simple and more complex bar charts
- Change the alignment of tics, and the color and transparency of bars
- Display data as pie charts
- Change colors and rotation of parts, calculating percentages
- Rotate pie charts, add shadow, and use a legend instead of labels

Section 2

- Read sequential data from files, and store them in 2D Numpy arrays
- Visualize data stored in 2D Numpy arrays with Matplotlib
- Create 2D Cartesian grids as products of 1D grids
- Visualize constant cell data on 2D Cartesian grids
- Visualize grid point data on 2D Cartesian grids
- Create 2D color maps and contour plots
- Create 3D surface plots, wireframe plots, and contour plots

Section 3

- What triangulations are, and how they differ from 2D Cartesian grids
- Create a triangulation manually
- Read a triangulation from a file and displaying it
- Display averaged vertex values
- Display linearly interpolated vertex values
- Display constant cell values