Masterclass in Python

Detailed Couse Contents

Part 1: The Python Programming Language: Basic

- **The programming fundamentals:**
  - A brief overview of computer and computer programming.
  - Introduction to computer languages – low-level languages: the machine language and assembly language – The high-level programming languages – Python programming languages.
  - What is computing? What is data? What is an algorithm? What is data type? What is IO? What is a system service? What is a library/package? What is a software development kit?
- **The Python as a Software Development Kit:**
  - The standard Python SDK.
  - Anaconda SDK for Machine Learning.
  - Flask package for web development.
  - Packages for automation.
- **The fundamentals:**
  - 'Hello, World' program.
  - Structure of Python program: Package/module, class statement, def statement.
  - Assignment statement – Fundamental ways to create data and manipulate data.
  - The internal format of the object in Python – Type, value, reference count.
  - Garbage collection.
  - Concept of a variable name. Concept of operators. Coverage of arithmetic and logic operators. Ary of an operator. Prefix, infix, and postfix operators. Left to right and right to left-associative operators. The relative precedence of operators.
- **Branching and looping statements:**
  - Control flow of Python program.
  - Block structure and indentation level of Python syntax.
  - What are conditions and how to write conditions in Python?
  - Conditional execution in Python.
    - Branching statements:
      - If statement.
      - If-else statement
      - If-elif-else statement
    - Looping statement.
      - For statement
      - While statement

        - Break statement
        - Continue statement
        - Pass statement.
    - Understanding the control flow of program using flow charts. Flow chart of branching and looping statements.
- **Data Types in Depth:**
    - Classification:
        - Atomic type vs container data types.
        - Sequential containers. Associative containers, neither sequential nor associative containers.
        - Mutable and immutable data type.
        - Understanding basics of bool, int, float, str, tuple, list, dictionary, set
    - Atomic data types in-depth:
        - Learning to create and process Boolean, int and float objects.
        - Logical not, logical and, logical or operators.
        - Arbitrary precision integer arithmetic in Python.
        - Integer division and floating-point division operator.
        - Exponential Operator.
    - Container data type operations in depth:
        - Str: concatenation, multiplication by scaler, membership testing, index, range, slice, index(), count(), strip(), lstrip(), rstrip(), split(), rsplit(), format(), isalpha(), isdigit(), isalnum(), isnumeric(), istitle(), isspace(), isprintable(), partition(), rpartition(), join()
        - List: concatenation, multiplication by scaler, index, range, slice, membership testing using 'in' operator, index assignment, range assignment, slice assignment, index deletion, range deletion, slice deletion, index(), count(), append(), extend(), insert(), remove(), pop(), clear(), copy(), sort(), reverse()
        - Tuple: concatenation, multiplication by scaler, index, range, slice, membership testing using 'in' operator, index(), count()
        - Dict: adding key-value pair in a dictionary, editing value of a given key, removing key-value pair in dict, keys(), values(), items(), pop(), popitem(), clear(), copy(), update(), fromkeys(), get(), sefdefault()
        - Set: Membership testing using 'in' operator, union(), intersection(), difference(), symmetric_difference(), update(), intersection_update(), difference_update(), symmetric_difference_update(), issubset(), issuperset(), isdisjoint(), add(), remove(), discard(), pop(), clear(), copy().

- **The Procedural Programming using Def statement:**
    - Writing a new algorithm. Input of an algorithm. Output of an algorithm.
    - Writing reusable code using the procedure. Formal parameters of a procedure. Actual parameters of a procedure. Return value of a procedure. Definition of a procedure. Call of a procedure.

- The def statement in Python as a means of writing new procedures.
- Internal processing of def statement: the function object, the code object, and Python Virtual Machine (PVM) code. What function definition is created in memory? How is the code object stored inside the function object? How is the code object stored inside the object executed?
- Calling a procedure. Pass by reference mechanism used by Python.
- Global and local variables. Observing global and local symbol tables by globals() and locals() functions.
- Nested def statement: Writing def statement inside def statement. Tracing control of a program having nested def statements. Looking at the memory allocation details of a nested procedure.
- Scope-visibility-lifetime model of Python:
  - Introducing four scopes in Python viz. local, enclosing, global and built-in.
  - Understanding LHS variable name processing.
  - Understanding LEGB scope rule of RHS variable lookup.
  - Unbound local error: cause and fix.
  - Free variable error: cause and fix.
  - Global statement
  - Nonlocal statement.
- Implicit state saving and function factory design pattern.
- Parameter passing:
  - Non-keyword and keyword syntax of sending specifying actual parameters.
  - Six types of formal parameters:
    - Positional arguments
    - Keyword arguments
    - Extra nonkeyword arguments
    - Default arguments
    - Keyword only arguments
    - Extra keyword arguments
    - *args, **kwargs technique
- Static namespace of function. Function.__code__, function.__name__, function.__dict__

- **Functional Programming:**
  - List comprehension:
    - Basic list comprehension of single variable
      - adding a filter condition to a comprehension
      - adding function to a comprehension, adding condition and
      - function simultaneously to list comprehension.
    - Multivariable list comprehension:
      - cartesian product,

        - computing simple two variable comprehension, generalizing to n variables,
        - adding condition to two variable and generalized comprehension,
        - adding function to two variable and generalized on comprehension
        - applying filter condition on the result of comprehension.
    - Lambda Expression:
        - Lambda expression to create an anonymous function object.
        - Formal parameters to lambda expression.
        - Writing a definition lambda expression using its formal parameters, already defined variables and constant expression
        - Using lambda expression.
    - Map Class: Learning to apply a common function on all elements in iterables.
    - Filter Class: Learning to apply a filter condition on all elements in iterables.
    - Reduce function: Reduce iterable to a single object by repeatedly operating on elements of the iterable.
    - The yield statement and generator object. Next method on generator object.
    - functools package in depth.

Part 2: Application Development – A

- **An overview of the standard Python package repository.**
    - The import statement and its variations. Import package, from package import attribute, from package import attribute as an alias.
    - Package names look up and the PYTHONPATH
    - Systematic handling of package hierarchy.
- **The 'os' and 'sys' packages.**
    - The high-level file handling: Learning to open, create, read, write, and close files using high-level file APIs. Developing file utilities: copy command, word count command, cat command.
    - File attributes: Obtaining stat information about the file.
    - Directory management:
        - Get the present working directory
        - change the present working directory.
        - Listing files in the current directory.
        - Recursively walking through the directory.
        - Removing files from a directory. Removing directory. Renaming file. Renaming directory.
    - Process management:
        - Creating child process. Os.fork()
        - Exchanging data between the child and the parent process.
        - Loading a new program in a newly created process. Os.exeve()
        - The parent-child synchronization. Os.wait()

        - Termination of a process. Os.exit()
    - Thread management:
        - Creating new threads.
        - Exchanging data between the child and the parent thread
        - Joining with the child thread
        - Detaching with the child thread
        - The thread synchronization mechanisms
        - Termination of a thread.
- **Data persistence and database connectivity:**
    - The pickle package. dump () and load () methods.
    - DBM files: Files, portability and close.
    - Shelve files: Storing built-in object types in Shelves, storing class instances in Shelve.
    - Sqlite3 connectivity.
- **GUI Application development with Tkinter:**
    - 'Hello, Win' Window – Widgets – Configuring widgets
    - Top-level windows – Top-level and Tk widgets – Top-level windows protocols.
    - Dialogs – Common dialogs – Custom dialogs
    - Binding events
    - Message and entry
    - Check buttons – Radio button and scale
    - Menus – List boxes – Text – Scroll bars – Grids

Part 3: Python Programming Language: Advanced

- **Object Oriented Programming in Python:**
    - The class statement:
        - Writing class statement. The root class 'type' in Python.
        - Writing constructor of a class __init__()
        - Creating an object from a class.
        - Adding a class method to a class.
        - Adding a static method to a class.
        - Implementing various types of practice class statements.
    - **Operator overloading:**
        - Overloading arithmetic and logic operators: __add__(), __sub__(), __mul__(), __truediv__(), __floordiv__(), __mod__(), __and__(), __or__(), __abs__(), __pow__
        - Overloading advanced operators:
            - Overloading [] operator with __getitem__(), and __setitem__()
            - Overloading () operator with __call__()
        - Operating overloading for built-in functions:
            - __str__, __repr__ for print()
            - __len__() for len()
            - Etc.
    - **Inheritance:**
        - Principle of extensibility and reusability in software engineering.

- Inheritance to achieve reusability.
- Deriving one class from another. The base class and the derived class.
- Inheriting a method of base class.
- Overriding a method of base class for extending purposes.
- Overriding a method of the base class for replacing purpose.
- The principle of information hiding and data encapsulation.
- The principle of polymorphism.
- Creating an abstract method with an abstracmethod decorator in abc package.
- Creating an interface with abc.ABCMeta and the abstract method decorator.
- The root base class, class object.
- Obtaining base classes of a given class with __bases__
- Method resolution order: __mro__
- Multi-level inheritance
- Multiple inheritance
- Multiple inheritance and the diamond problem.

- **Exception Handling:**
  - What are exceptions? Need for an exception.
  - Built-in exceptions: NameError, ValueError, TypeError, UnboundLocalError, LookupError, FileNotFoundError, PermissionError, ModuleNotFoundError etc.
  - Handling exceptions using try-except statements.
  - Handling multiple exceptions using single except block
  - Fetching exception information with the 'as' clause.
  - Writing generic exception handler block.
  - Try-except-else statement
  - Try-except-finally statement
  - Try-except-else-finally statement
  - Raise an exception using the raise statement. Three ways to write a raise statement.
  - Following most derived to most base order while handling multiple exceptions in a common inheritance hierarchy.
  - Defining an exception of one's own.
  - Fetching the last raised exception information with sys.exc_info()
  - Assert statement

- **Advanced Python Programming and Design patterns:**

  - **The Iterator Design Pattern:**
    - Understanding expansion of the for loop into the while loop.
    - Understanding __iter__() method.
    - Understanding Type_Iterator class.
    - Understanding __next__() method

- Iterator with read semantics.
- Iterator with consumer semantics.
- Writing iterators on our own.

- **The context management protocol:**
  - The with statement. The as clause.
  - __enter__() for post construction initialization.
  - __exit__() : a handler before destruction.
  - Achieving automatic acquire, use, and release cycle with context management protocol.
  - Writing __exit__() method properly to handle unhandled exceptions in the with block.
- **The decorator design pattern:**
  - Python as a language of hooks.
  - An overview of hooking techniques in Python.
  - Function decorating a function.
  - Function decorating a class.
  - Class decorating a function.
  - Class decorating a class.
- **Attribute Management:**
  - Attribute management using property class. Fget, fset, fdel attributes of property class. Getter(), setter(), and delete() methods on property objects. Creating a managed attribute using property class.
  - The descriptor protocol:
    - Creating a reusable managed attribute using a descriptor class.
    - __get__() method. __set__() method. __delete__() method
  - Manging attribute by using non-existent attribute: __getattr__(), __setattr__(), __delattr__()
  - Managing attributes method #4: __getattribute__(), __setattr__(), __delattr__()

- **Metaclass:**
  - Hooking class creation using metaclass.
  - Type.__call__ method.
  - Type.__new__ method.
  - Metaclass as a derived class of the root class type.
  - Overriding __new__() in metaclass to extend type.__new__()
  - Overriding __call__ in metaclass to extend type.__call__()
  - Implementing our own abstract base class meta.

Part 4: Fundamentals of Data Analysis and Machine Learning with numpy, pandas, matplotlib and scikitlearn

- **The Numpy Library: introduction**
  - ndarray: The heart of the library
    - Create an array
    - Types of data
    - The type option
    - Intrinsic creation of an Array
  - Basic Operations:
    - The arithmetic operators
    - The Matrix product
    - Increment and Decrement operators
    - Universal Functions (ufunc)
    - Aggregate functions
  - Indexing, slicing, and iterating
  - Conditions and Boolean Arrays
  - Shape manipulation
  - Array manipulation: Joining and splitting Arrays
  - Copies of views of an object
  - Vectorization
  - Broadcasting
  - Structured Arrays
  - Reading and Writing data on binary files
- **The Panda's library: introduction**
  - Introduction to Pandas data structure
    - The series
    - The data frame
    - The index objects
  - Other functionalities on indexes: Reindexing – Dropping – Arithmetic and data alignment
  - Operations between data structures: Flexible arithmetic methods, operations between DataFrame and Series
  - Function application by Mapping – Function by element – Functions by row or column, statistics function
  - Sorting and ranking
  - Correlation and Covariance
  - Non Number data: Assigning a NaN value – Filtering out NaN values – Filling in NaN occurrences.
- **Pandas: IO**
  - I/O API tools
  - CSV and textual files
    - Reading data in CSV or text files: Using RegExp to pars TXT files
    - Reading TXT files into Parts
    - Writing Data in CSV

- Reading and writing HTML files: Writing data in HTML, Reading data from HTML file
- Reading and writing data from XML files
- Reading and Writing data on Microsoft Excel files
- JSON data.

- **Data visualization with matplotlib**
  - The matplotlib architecture: Backend layer, Artist layer, Scripting Layer, pylab and pyplot
  - A simple interactive window with pyplot.
  - The plotting window: set properties of the plot. Matplotlib and Numpy
  - Using kwargs: Working with multiple figures and axes
  - Adding elements to the Chart: Adding text – Adding a grid – Adding a Legend
  - Saving charts: Saving a code – Converting session into an HTML file – Saving as an image
  - Handling Date values
  - Chart Typology
  - Lines chars
  - Histograms
  - Bar Charts – Horizontal bar charts – Multi serial bar charts – Multiseries bar with pandas DataFrame
  - Multiseries Stacked Bar Charts
  - Pie chart
  - Advanced charts: Contour plots, polar plots
  - The mplot3d toolkit: 3D surfaces, scatter plots in 3D, Bar charts in 3D
- **Machine Learning & Data Science -> DOMAIN KNOWLEDGE:**
  - **ESSENTIAL MATH:**
    - **BASICS of Matrices**: Vectors and Linear Algebra.
    - **BASICS of Statistics:** Describing a single set of data, central tendencies, dispersion, correlations,
    - **BASICS of Probability:** Dependence and Indepence, Conditional Probability, Baye's Theorem, Random variable,m Continuous distribution, Normal Distribution, Hypothesis and inference, p-values, Confidence Interval, p-Hacking.
  - **Essential Data Science**: Collect, explore, clean, munge, and manipulate data
  - The scikit-learn library
  - Categorization of Machine Learning Strategies:
    - Supervised, unsupervised, semi-supervised machine learning.
    - Batch mode ML vs Online ML
    - Instance-based ML vs. model-based ML.
    - Reinforcement learning
    - Deep learning
  - Machine learning application development cycle.
  - Where to get datasets for machine learning and data science?

- Selecting performance measure:
  - Mean Absolute Error
  - Root Mean Absolute Error.
- Creating test sets and training sets:
  - random sampling
  - binomial distribution
  - Stratified sampling
- Correlation matrices and plotting correlation matrices
- Structure of Scikit Learn Library: (a) Estimators (b) Transformers (c) Predictors
- Imputers: (a) Simple imputer (b) KNN imputer (c) Iterative Imputer
- Transformers: (a) Ordinal Encoder (b) OneHotEncoder (c) Min-Max Scaler (d) StandardScaler (e) RBF-kernel
- Hands-on Example on the following techniques:
  - KNN: Kth Nearest Neighbour
  - Naive Bayes
  - Simple Linear Regression
  - Multiple Regression
  - Logistic Regression
  - Decision Trees