Advanced Python

Course Title	Advanced Python
Course Duration	9.5 days of interactive training
Course Owner	Nikola Ignjatovic

Course Description

Brief Summary	The aim of the course is to introduce trainees to advanced topics in Python programming, including lambda functions & comprehension, exception handling, object-oriented programming, and data handling & visualization with popular open source packages: NumPy, Pandas, Matplotlib and Seaborn. Trainees learn how to apply object-oriented principles in Python, how to properly handle run time errors in their applications, and how to manipulate, filter, aggregate, group, analyze and display tabular data, as well as create new values from existing data through calculation, data manipulation and data cleaning.
Indicative Content	1. **Lambda functions & comprehension** Trainees learn when is beneficial to define lambda functions over ordinary functions, how to define and apply lambda functions. Trainees learn how to include an if-statement within lambda functions, are introduced to the concept of comprehension, and how list, dictionary & generator comprehension can be used within lambda functions. 2. **Exception handling** Trainees learn about run time errors and how to handle them in their application using exception handling statements. 3. **Object-oriented programming** Trainees learn how object-oriented programming (OOP) is done in Python. Principles of encapsulation and inheritance are explained, and a comparison is provided on how access to member attributes and methods is done in Python vs other object-oriented languages, like Java. Trainees learn how to define classes, implement a constructor, instance, class and static methods, when to use protected and private access modifiers and how to define protected and private attributes and methods. The differences between class and instance attributes are demonstrated, as well as how to create class and instance level constants. A detailed insight into how Python implements the getter, setter and deleter methods through property decorators is given. Both single and multiple inheritance is explained, but exercises cover single inheritance only. Trainees also learn the purpose of abstract classes and how to define them. Through writing client code, learners learn how to test functionalities of their classes. 4. **NumPy arrays** Trainees learn how create and use NumPy arrays and why they are faster and more efficient than built-in lists. Topics covered include different ways of creating NumPy arrays, indexing NumPy arrays, logical and arithmetic operations with NumPy arrays, manipulating NumPy arrays, applying aggregate functions on NumPy arrays, importing data from text files into NumPy arrays and saving data from NumPy arrays to text files. 5. **Data wrangling with Pandas** Trainees learn various ways of creating Pandas Series and Data Frames, load data from files into Data Frames, and how to manipulate, filter and aggregate data using library's API. Concatenating, merging and joining Data Frames is also covered, as well as how to apply data cleaning to Data Frames.

	6. **Data visualization** Trainees learn how to visualize data using matplotlib and seaborn. They learn how to produce bar, column, scatter, line plots, multiple plots in one figure, joint plots, and how to add plot elements, including plot title, axis labels, and data point labels. Learners are enabled to create advanced relational, distribution and categorical plots using Seaborn. Differences between axes-level and figure-level plots are explained as well as when to use each. Trainers are also introduced to different ways of changing plot style and colours through different colour palettes. 7. **External data sources** Trainees are introduced to SQLite, its features, advantages and disadvantages. They learn how to install and use SQLite to create & drop tables and to perform CRUD operations on database tables, save the result of an SQL query into a file, execute SQL statements from a file, use SQLite from Python to create/drop tables and perform CRUD operations on database tables with hard-coded values as well as through runtime binding, and use SQLite from Python to convert the result of an SELECT SQL statement into a Pandas DataFrame.

Learning Outcomes

Course Learning Outcomes	On successful completion of this course trainees will be able to: 1. Create lambda functions. 2. Control run time errors in an application using exception handling statements. 3. Apply object orient programming principles to build software that is robust, easy to maintain and extend. 4. Use NumPy arrays to produce code that is more efficient at data storage and manipulation than Python lists. 5. Wrangle data with Pandas. 6. Visualize data with Matplotlib and Seaborn to build bar charts, column charts, line charts, scatter plots and joint plots.

Delivery Approach	Trainer-led instruction, demonstrations, individual exercises, Q & A sessions

Assessment

Summative Assessment

Element	Type	Weighting	Learning outcomes assessed
1	Project		1,2,3,4,5,6

Description of each element of Assessment	The assessment is conducted through a project, consisting of 15 practical tasks. The first 5 tasks bear 15% each and are aimed at pass grade (75%). Task 1: lambda functions, task 2: OOP, task 3: NumPy, task 4: Pandas + Exception Handling, task 5: Matplotlib or Seaborn. Tasks 6 – 15 are weighted at 2-3% each. Tasks 6-10 are aimed at merit grade and tasks 11 – 15 are aimed at distinction. Tasks 6 – 10 and tasks 11 – 15 cover the same topics as the related pass tasks, where one of the tasks 5, 10 or 15 covers Matplotlib, and the remaining two Seaborn. Wherever

Advanced Python

	possible, the merit tasks are implemented as extension on related pass tasks, and the distinction tasks are implemented as extension on related merit tasks. Trainees can check the correctness of each task through unit testing, except from tasks 5, 10 and 15, covering visuals. For these tasks, the image of the required plot is included in the project brief. The assessor uses the same unit tests to check the outcome of all tasks, except for tasks 5, 10 and 15, which need to be assessed visually. Brief feedback on each task is given through DMS. Grading: a minimum of 75% is needed to achieve the pass grade, a minimum of 80% is required to achieve merit, and a minimum of 90% is required to achieve distinction. **Overall, each task is assessed against two criteria:** **1. does it work (determined by passing all unit tests, or in case of visuals – that the plot looks the same as illustrated in the project brief)** **2. clean code (incl. naming standards, use of comments, no obsolete or useless code, no input statements, no print statements unless required by the task)**
Minimum Pass Mark	75%

Administration

Prerequisite Learning Requirements	Python (Foundation)
Last Revision & Date	23 February 2022