# Course contents Applied Numerical Methods with Python and Python Libraries

**Module 1 Preamble, Learning to break fall and Foundations**
The goal of this module is to introduce a number of fundamental programming features and syntax and how they are supported in Python.

**Variables and Functions**
- Lexical Structure
- Python Strings
- Fundamental Data Types
- Functions
- Code layout rules

**Program Structure**
- Scope and Lifetime of Variables
- Control Flow Structure
- Looping and Iteration
- Iterables, Iterators and Tuples

**Fundamental data types**
- Numbers and Numeric Operations
- Testing for valid numbers
- Analysis of floating-point numbers

**Control flow statements**
- If and if-else statements
- While statement
- For statement
- Iterators

**Expressions and Operators**
- What is an expression?
- Operators and operator precedence
- Numeric operations and conversions
- Sequence operations

**Data Types**
- Vectors and lists
- Deque, queue and priority queue
- Sets
- Associative arrays and dictionaries

**Algorithm Analysis**
- Asymptotic behaviour of functions
- Asymptotic order; orders of complexity
- Hierarchy of orders
- Order relationships
- Hard Problems

**Types of Algorithms**
- Searching and sorting
- Merge
- Insertion, remove
- Linear and binary search

**Module 2 Object Oriented Programming in Python**
In this module we show how to write *correct* object-oriented code in Python by which we mean code that is maintainable and that bears some resemblance to real-life applications.

**Creating Classes in Python**
- Naming conventions
- My first class A-Z
- Constructors and initialisation
- Creating objects and class instantiation
- Access control issues

**Creating Larger Classes**
- Composition and Delegation
- Whole-part objects in Python
- Arrays and collections of objects
- Inheritance and subclassing
- Combining inheritance and composition

**Fundamental Arrays and Data Structures**
- Basic data types
- One-dimensional arrays; matrices
- n-dimensional arrays (`ndarray`)
- Data type objects (`dtype`)
- Tuples, dictionaries and lists

**Module 3 Functional Programming (FP) in Python**
In this module we discuss the extent to which Python supports programming in a functional style.

**Introduction to Higher-order Functions (HOFs)**
- What can we do with HOFs?
- Simplify HOFs by lambda forms and lambda expressions
- Lambdas and the lambda calculus
- Apply a function to a collection: `map ()`
- Pass/reject data with `filter ()`

**Advanced Functional Programming**
- Lazy and eager evaluation
- Applications of `filter ()`
- Generator expressions
- Recursion and reduction
- Folds

- Decorators
- Iterables and *itertools* module

**Exception Handling in Python Programs**
- Raising exceptions
- Handling exceptions
- Exception hierarchy and built-in exceptions
- User-defined exceptions

**Some Standard Library Modules**
- *sys* (state of Python interpreter)
- *functools* (functions and types supporting functional programming)
- *heapq* (keep list "nearly sorted")
- *copy* (deep and shallow copies of objects)

**Module 4 Essential Mathematical Structures**
Approximation of functions by polynomials is probably one of the most important activities in numerical analysis and its applications.

**Polynomials**
- 1d, 2d and 3d polynomials
- The algebra of polynomials
- Power series polynomials
- Operations on polynomials

**Special Polynomials and Functionality**
- Orthogonal polynomials: Chebychev, Legendre, Laguerre, Hermite
- 1d, 2d and 3d orthogonal polynomial grids
- Least Squares fitting
- Spline fitting

**Random Sampling**
- Simple random data
- Permuting and shuffling randomly
- Continuous and discrete distributions
- Drawing random samples
- Creating histograms

**Arrays**
- N-dimensional arrays `ndarray`
- Creating and manipulating arrays
- Iterating over arrays
- Applications

**Module 5 Fundamental Numerical Methods**
This module introduces several important libraries that are needed in many kinds of applications and that we use in later modules.

**Integration**
- General purpose integration schemes in one, two, three and n dimensions
- Gaussian and Romberg integration
- Trapezoid and Simpson's rules
- Gaussian quadrature
- roots of orthogonal polynomials

**Numerical Solution of Ordinary Differential Equations (ODE, `odeint`)**
- Real-valued and complex-valued ODEs
- First-order and higher-order ODEs
- Application areas

**Statistics**
- Random variables
- Probability and cumulative distribution functions
- T-test, Kolmogorov-Smirnov test
- Test for normality
- Comparing two samples
- Estimation
- Kernel density estimation (KDE)
- Univariate and multivariate estimation
- Applications

**An Introduction to Optimisation**
- Univariate minimisers and root finders
- Unconstrained and constrained multivariate optimization
- Least squares minimization and curve fitting
- Orthogonal distance regression (ODR)

**Module 6 Advanced Numerical Methods**
This module is central to all computationally-intensive applications because it discusses *numerical linear algebra* which consists of routines to solve matrix equations, eigenvalue and eigenvector computation as well as matrix decomposition methods based on LAPACK and Matlab. We also discuss interpolation algorithms in one and two dimensions.

**Mathematical Functions**
- Trigonometric and inverse trigonometric functions
- Rounding
- Sums, products and differences
- Exponential and logarithmic functions

**Linear Algebra: Overview**
- ATLAS LAPACK and BLAS libraries
- Basic routines
- Computing norms
- LU and Cholesky decomposition

**Advanced Linear Algebra**
- Eigenvalue and eigenvector computation
- Decomposition: QR, Schur, SVD (Singular Value Decomposition)
- Matrix functions (for example, the exponential of a matrix)
- Special matrices

**Matrix Library (`numpy.matlib`)**
- Matrix objects
- Creating and initializing matrices
- Using matrices in applications

**Interpolation**
- Overview of univariate and multivariate interpolation
- Interpolating a 1-d function
- Piecewise polynomial interpolation
- Piecewise linear interpolation in N dimensions
- Interpolation over a 2-d grid
- 2d splines

**Module 7 Numerical Solution of Ordinary and Partial Differential Equations (ODE/PDE)**
This module introduces modern finite difference (FDM) schemes that approximate the solution of time-dependent partial differential equations, in particular, parabolic PDEs.

**Automatic Differentiation (AD) Packages**
- What is AD?
- Using AD to compute gradient, Jacobian and Hessian
- Examples: Optimisation and ODE solvers
- Application to sensitivity analysis and Machine Learning (ML)

**Solving ODEs Numerically**
- Hand-crafted solutions versus `scipy.integrate.odeint`
- Scalar equations and systems of equations
- Stiff and non-stiff problems
- Using `scipy.integrate.odeint`

**Some Important Finite Difference Schemes**
- Explicit Euler, fully implicit
- Crank Nicolson
- Alternating Direction Explicit (ADE)
- Methods of Lines (MOL) using `scipy.integrate.odeint`

**Model PDE: the one-Dimensional Heat Equation**
- PDE formulation (initial boundary value problem)
- Finite difference methods for the heat equation
- Using Python libraries
- Creating a working program in Python

**Implementing Convection-Diffusion-Reaction (CDR) Equations**
- What is CDR?
- Numerical approximation
- Examples and applications

**Module 8 Auxiliary Libraries**
This module consists of several *utility* libraries for serialisation, multi-dimensional data, date time functions and producing machine code.

**Input and Output Essentials**
- Load and save MATLAB files
- Birds'-eye overview of HDF5
- Dictionary of `numpy` arrays
- Working with NetCDF files
- Examples and applications

**Python with HDF5**
- HDF5 tools
- Reading and writing data
- Working with datasets
- Chunking and compression

**Financial Functions**
- Future and (net) present values
- Computing payments
- Internal Rate of Return (IRR)
- Interest rate computation

**Datetime Support Functions**
- Business day functions
- Valid business days
- Rolling
- Number of days between two dates

**Advanced Statistical Functions**
- Overview of (extensive) functions and their applications
- Chi-square test
- Kruskal-Wallis
- Kolmogorov-Smirnov
- Calculating regression line
- Geometric and harmonics means

**JIT and fast Machine Code**
- Introduction to Numba
- Decorating Python code
- When (and when not) to use Numba

**Module 9 Advanced Data Access**
**Pandas Executive Overview**

- Datastructures for multi-dimensional heterogeneous data
- Pandas as client of NumPy
- Essential classes Series and DataFrame
- Data mungling / wrangling tasks
- Transforming and mapping data between formats

**Pandas (Data Analysis)**

- Data science tools in Python
- Data analysis workflow: DataFrame object
- Working with data operations
- Time series functionality
- Index objects
- Statistical reductions

**Python with HDF5 (Hierarchical Data Format )**

- Main features in HDF5 for storing large quantities of numerical data
- Two-way communication between arrays and disk
- Attributes and metadata
- Datasets
- Groups and directory-like structures
- Chunked storage

**Excel Automation**

- The Excel Object Model
- App, Workbook, Sheet, Range, Chart
- Working with DataFrames
- Plotting and Reporting

**Excel Functionality**

- Excel Add-in
- Read and write Excel data
- An Introduction to SQLite
- Excel and Relational Database Systems (sqlite3)
- Excel and Pandas

## Your Trainer

Daniel J. Duffy started the company Datasim in 1987 to promote C++ as a new object-oriented language for developing applications in the roles of developer, architect and requirements analyst to help clients design and analyse software systems for Computer Aided Design (CAD), process control and hardware-software systems, logistics, holography (optical technology) and computational finance.
He used a combination of top-down functional decomposition and bottomup object-oriented programming techniques to create stable and extendible applications.
Previous to Datasim he worked on engineering applications in oil and gas and semiconductor industries using a range of numerical methods (for example, the finite element method (FEM)) on mainframe and mini-computers.

Daniel Duffy has BA (Mod), MSc and PhD degrees in pure, numerical and applied mathematics and has been active in promoting partial differential equation (PDE) and finite difference methods (FDM) for applications in computational finance. He was responsible for the introduction of the Fractional Step (Soviet Splitting) method and the Alternating Direction Explicit (ADE) method in computational finance. He is also the originator of the *exponential fitting method* for time-dependent partial differential equations.

He is also the originator of two very popular C++ online courses (both C++98 and C++11/20) on www.quantnet.com in cooperation with Quantnet LLC and Baruch College (CUNY), NYC. He also trains developers and designers around the world. He can be contacted dduffy@datasim.nl for queries, information and course venues, in-company course and course dates