Open Awards

Quality Endorsed Unit

1 Unit Details

Unit Title:	Advanced Python Programming
Unit Code:	CK3/4/WR/001
Level:	4
Credit Value:	9

2 Learning Outcomes and Criteria

Learning Outcome (The Learner will):	Assessment Criterion (The Learner can):
1. Create and run more advanced Python programs.	1.1 Use the Windows Command Prompt to run programs.
	1.2 Use IDLE to input Python commands and statements.
	1.3 Create Python programs and run them in both IDLE and in a Command Prompt window.
	1.4 Install (using pip) and import Python packages.
	1.5 Create Python programs that incorporate decorators, iterators, and generators.
	1.6 Create Python programs that incorporate Python's latest features, including the handling of time zones and using the ChainMap class to merge dictionaries
2. Implement advanced Python strings and lists.	2.1 Convert, index, and slice strings, and apply a range of advanced string methods, including search and replace, strip, and split.
	2.2 Access list elements using positive and negative indexing and slicing, and apply a range of list methods, including count, pop, append, insert, and extend.
	2.3 Apply a range of list functions, including len, min, max, reversed, sorted, and sum.

		2.4	Create lambda anonymous functions and know when it is appropriate to use them.
		2.5	Create lists and dictionaries using comprehension, create multidimensional lists, and pass arguments to a function in a list.
3.	Use a range of Python's coding shortcuts.	3.1	Create intelligent for loops, perform list and string multiplication and advanced tuple assignment.
		3.2	Apply multiple assignment techniques and return multiple values from a function.
		3.3	Avoid unnecessary loops, use chained comparisons, use match and case and the "is" operator, and specifying a loop with a single line of code.
4.	Use advanced text, numbering, and regular expressions.	4.1	Format text using the % operator and format specifiers.
		4.2	Specify and implement variable length fields and utilise the format method.
		4.3	Specify "spec" formatting grammar to format and align text.
		4.4	Format numbers with leading zeros and thousand separators, percentages, and controlling precision.
		4.5	Create advanced regular expressions, including using the match object, refining matches, regex flags, meta characters, pattern quantifiers, and backtracking.
		4.6	Apply advanced regex grammar, including multiple patterns, negative look-ahead, and named groups.
5.	Deploy advanced Python file handling techniques.	5.1	Access the directory system, define file open exceptions, create binary and text files, use the "with" keyword, and using seek and tell.
		5.2	Apply a range of read/write operations, including reading text into a list, file seek operations, and use "pickling" methods.
		5.3	Use the "struct" package to read and write text and binary data, apply strict data formats, use the advanced feature of the pickle and shelve packages.
6.	Implement advanced classes and methods.	6.1	Apply inheritance and multiple inheritance and apply public and private variables and methods.
		6.2	Apply a range of magic methods, including representation methods, comparison methods, arithmetic methods, unary arithmetic methods, reflection methods, and collection class methods.

		6.3	Specify and apply decimal money classes, including the numeric class, the decimal class, fraction class, and complex class.
		6.4	Specify random functions and test random behaviour, create a random integer game, create a card deck object, and add pictograms to the deck object, contributing to a poker game.
		6.5	Specify a range of math data objects, trigonometric functions, and logarithmic functions.
7.	Use the Python numpy (numeric Python) package.	7.1	Create numpy arrays and apply a range of numpy data types, including bool, int8, int16, int32, int64, float, float32, float64, and complex numbers.
		7.2	Apply a range of numpy functions, including arange, linspace, empty, eyes, ones, zeros, full, and fromfunction.
		7.3	Implement large-scale numpy operations on very large numpy arrays or parts of a numpy array using slicing techniques.
		7.4.	Implement array to array operations and specify and process multidimensional numpy arrays, boolean arrays, and statistical functions, including mean, median, and standard deviation.
		7.5	Implement numpy's advanced math functions, including sine, cosine, tangent, power, radians, log, and sqrt.
		7.6	Plotting numeric data to produce line charts, pie charts, histograms, and three-dimensional plotting.
8.	Use multiple modules and implement advanced Object-Oriented Programming techniques.	8.1	Define multiple modules and import them into a main module, define global variables to access data across modules, and deal with problems associated with importing multiple modules.
		8.2	Implement constructors and destructors, define class, instance, and static methods, and implement data encapsulation.
		8.3.	Specify getters and setters using property decorators, implement polymorphism, method overloading, and method overriding.
		8.4	Utilise abstraction and composition as an alternative design approach, implement duck typing, and know when Object-Oriented Programming is not appropriate.

9.	Implement applications to download financial data over the Internet and chart the data.	9.1	Install and implement the pandas software library for data manipulation and analysis and write an app to download stock data from an authorised server by providing the ticker symbol.
		9.2	Produce a range of charts based on the stock data, including opening and closing prices, trading volume, highs, and lows.
		9.3	Create split charts showing two types of data, dual charts, plotting two stocks, and producing a moving average chart.
10.	Implement dataframes and multithreading and asynchronous programming techniques.	10.1	Create a dataframe from a Python dictionary, CSV file, spreadsheet, and numpy array, and apply a range of attributes and methods, including index, axes, columns, shape, and describe.
		10.2	Change the index of a dataframe from a list and setting one of the columns as an index.
		10.3	Navigate a dataframe, add a row or column, delete a row or column, rename indexes and columns, apply a function, query dataframe rows, and get statistics.
		10.4	Create a multithreaded module, including daemon threads, and synchronise and queue threads.
		10.5	Create asynchronous code using the asyncio module and distributing tasks.