MARKING CRITERIA for functions: Please colour/shade the relevant boxes below

	Poor	Limited	Fair	Good	Excellent	Outstanding	Exceptional
	0-39	40-49	50-59	60-69	70-79	80-89	90+
1. Code meets specification	Code does not fulfil required purpose and/or specification	Code almost fulfils required purpose and/or specification.	Code fulfils basic required purpose and specification, but may make extensive direct use of supplied code with only limited new development by the student.	A good and quite readable piece of code that clearly organizes material to solve problem set and includes some useful original elements. No globals* and at most limited unnecessary hardwired components.	Excellent, clear & readable code that includes extensive original materrial and some error trapping. Robust and mainly Pythonic. No globals* or unnecessary hardwired parts.	Elegant & robust Pythonic code. Extensive error trapping, outstanding treatment of options. Includes relevant Pythonic features beyond those in the core notes.	An exceptional quality piece of code that develops an elegant and efficient solution to the problem tasked and has exceptional demonstration of Python coding skills way showing extensive work beyond the core course notes.
2. Comments	No or very limited new comments, or comments simply copied from example codes.	Limited additional comments, but at least show which parts of the code come from different sources and which developed by the student.	Some useful new comments showing basic understanding of some elements of code functioning, including clear attribution of lines from example codes.	Solid comments through most of the code, indicating the broad flow of information and how the code purpose is fulfilled.	Extensive detailed comments throughout, making clear how the code operates and achieves its aim.	Outstanding detailed comments. Flow of information from arguments & inputs to outputs is explicit and clear.	Full, clear yet concise comments throughout, showing outstanding demonstration of understanding of how the code operates, limitations, options and complexities.
3. Function Argument(s), option(s), return value(s) and docstring (PEP257)	Argument(s)/options or return value(s) not given or not used. No docstring, or weak docstring that misses most PEP257 components.	Argument(s)/options and return value(s) given but poorly described or specified. Weak docstring with limited elements of PEP257.	Argument(s)/options and return value(s) given and sufficiently described, including at least expected data type. Basic multiline docstring given with some elements of PEP257.	Argument(s)/options used, given and well-described. Expected data type, range or other useful information. Some detail given for return value(s). Multiline docstring given with basic but useful help message.	Argument(s)/options used & given. Excellent clear description, including type, range etc.. Expected return value given in detail. Full docstring, including summary, clearly valuable as help. Description of side effects, exceptions raised, and restrictions, including file/URL requirements	Most relevant information about function inputs and outputs clearly used and specified, including return if errors encountered or other complexities. Outstanding clear, full, informative and very readable docstring.	All required function arguments, options and outputs as specified, with some additional features such as keyword options as appropriate. Exceptional presentation of docstring, including all inputs and outputs, any restrictions, behaviour of output(s) under errors, and some additional features as appropriate such as example of use.
4. Code layout (PEP8)	Poor code layout that is difficult to follow and read. Poor choice of variable names.	Basic code layout that has limited features of PEP8. Sometimes reasonable variable names.	Reasonably readable code, with some consistency, clear indentation taking account of main features of PEP8 style. Mostly reasonable variable names.	Clear, readable code that is neatly formatted and takes good account of many features of PEP8 style. Good choice of variable names.	Excellent readable code, with 'clean' features: 4-space indentation, clean treatment of long/continued, neat import statement, no extraneous whitespace etc. Solid variable names.	Elegant readable code layout that takes extensive note of PEP8 features. Very well-chosen variable names.	Beautiful code that uses all of the key ideas in the PEP8 style guide.

* There may be occasions when globals are required, but that should be rare in functions, and need to be fully justified if used in your code.