

Contents

About	1
Chapter 1: Getting started with Python Language	2
Section 1.1: Getting Started	2
Section 1.2: Creating variables and assigning values	6
Section 1.3: Block Indentation	10
Section 1.4: Datatypes	11
Section 1.5: Collection Types	15
Section 1.6: IDLE - Python GUI	19
Section 1.7: User Input	21
Section 1.8: Built in Modules and Functions	21
Section 1.9: Creating a module	25
Section 1.10: Installation of Python 2.7.x and 3.x	26
Section 1.11: String function - str() and repr()	28
Section 1.12: Installing external modules using pip	29
Section 1.13: Help Utility	31
Chapter 2: Python Data Types	33
Section 2.1: String Data Type	33
Section 2.2: Set Data Types	33
Section 2.3: Numbers data type	33
Section 2.4: List Data Type	34
Section 2.5: Dictionary Data Type	34
Section 2.6: Tuple Data Type	34
Chapter 3: Indentation	35
Section 3.1: Simple example	35
Section 3.2: How Indentation is Parsed	35
Section 3.3: Indentation Errors	36
Chapter 4: Comments and Documentation	37
Section 4.1: Single line, inline and multiline comments	37
Section 4.2: Programmatically accessing docstrings	37
Section 4.3: Write documentation using docstrings	38
Chapter 5: Date and Time	41
Section 5.1: Parsing a string into a timezone aware datetime object	41
Section 5.2: Constructing timezone-aware datetimes	41
Section 5.3: Computing time differences	43
Section 5.4: Basic datetime objects usage	43
Section 5.5: Switching between time zones	44
Section 5.6: Simple date arithmetic	44
Section 5.7: Converting timestamp to datetime	45
Section 5.8: Subtracting months from a date accurately	45
Section 5.9: Parsing an arbitrary ISO 8601 timestamp with minimal libraries	45
Section 5.10: Get an ISO 8601 timestamp	46
Section 5.11: Parsing a string with a short time zone name into a timezone aware datetime object	46
Section 5.12: Fuzzy datetime parsing (extracting datetime out of a text)	47
Section 5.13: Iterate over dates	48
Chapter 6: Date Formatting	49
Section 6.1: Time between two date-times	49
Section 6.2: Outputting datetime object to string	49

Section 6.3: Parsing string to datetime object	49
Chapter 7: Enum	50
Section 7.1: Creating an enum (Python 2.4 through 3.3)	50
Section 7.2: Iteration	50
Chapter 8: Set	51
Section 8.1: Operations on sets	51
Section 8.2: Get the unique elements of a list	52
Section 8.3: Set of Sets	52
Section 8.4: Set Operations using Methods and Builtins	52
Section 8.5: Sets versus multisets	54
Chapter 9: Simple Mathematical Operators	56
Section 9.1: Division	56
Section 9.2: Addition	57
Section 9.3: Exponentiation	58
Section 9.4: Trigonometric Functions	59
Section 9.5: Inplace Operations	60
Section 9.6: Subtraction	60
Section 9.7: Multiplication	60
Section 9.8: Logarithms	61
Section 9.9: Modulus	61
Chapter 10: Bitwise Operators	63
Section 10.1: Bitwise NOT	63
Section 10.2: Bitwise XOR (Exclusive OR)	64
Section 10.3: Bitwise AND	65
Section 10.4: Bitwise OR	65
Section 10.5: Bitwise Left Shift	65
Section 10.6: Bitwise Right Shift	66
Section 10.7: Inplace Operations	66
Chapter 11: Boolean Operators	67
Section 11.1: `and` and `or` are not guaranteed to return a boolean	67
Section 11.2: A simple example	67
Section 11.3: Short-circuit evaluation	67
Section 11.4: and	68
Section 11.5: or	68
Section 11.6: not	69
Chapter 12: Operator Precedence	70
Section 12.1: Simple Operator Precedence Examples in python	70
Chapter 13: Variable Scope and Binding	71
Section 13.1: Nonlocal Variables	71
Section 13.2: Global Variables	71
Section 13.3: Local Variables	72
Section 13.4: The del command	73
Section 13.5: Functions skip class scope when looking up names	74
Section 13.6: Local vs Global Scope	75
Section 13.7: Binding Occurrence	77
Chapter 14: Conditionals	78
Section 14.1: Conditional Expression (or "The Ternary Operator")	78
Section 14.2: if, elif, and else	78
Section 14.3: Truth Values	78

Section 14.4: Boolean Logic Expressions	79
Section 14.5: Using the cmp function to get the comparison result of two objects	81
Section 14.6: Else statement	81
Section 14.7: Testing if an object is None and assigning it	82
Section 14.8: If statement	82
Chapter 15: Comparisons	83
Section 15.1: Chain Comparisons	83
Section 15.2: Comparison by `is` vs `==`	84
Section 15.3: Greater than or less than	85
Section 15.4: Not equal to	85
Section 15.5: Equal To	86
Section 15.6: Comparing Objects	86
Chapter 16: Loops	88
Section 16.1: Break and Continue in Loops	88
Section 16.2: For loops	90
Section 16.3: Iterating over lists	90
Section 16.4: Loops with an "else" clause	91
Section 16.5: The Pass Statement	93
Section 16.6: Iterating over dictionaries	94
Section 16.7: The "half loop" do-while	95
Section 16.8: Looping and Unpacking	95
Section 16.9: Iterating different portion of a list with different step size	96
Section 16.10: While Loop	97
Chapter 17: Arrays	99
Section 17.1: Access individual elements through indexes	99
Section 17.2: Basic Introduction to Arrays	99
Section 17.3: Append any value to the array using append() method	100
Section 17.4: Insert value in an array using insert() method	100
Section 17.5: Extend python array using extend() method	100
Section 17.6: Add items from list into array using fromlist() method	101
Section 17.7: Remove any array element using remove() method	101
Section 17.8: Remove last array element using pop() method	101
Section 17.9: Fetch any element through its index using index() method	101
Section 17.10: Reverse a python array using reverse() method	101
Section 17.11: Get array buffer information through buffer_info() method	102
Section 17.12: Check for number of occurrences of an element using count() method	102
Section 17.13: Convert array to string using tostring() method	102
Section 17.14: Convert array to a python list with same elements using tolist() method	102
Section 17.15: Append a string to char array using fromstring() method	102
Chapter 18: Multidimensional arrays	103
Section 18.1: Lists in lists	103
Section 18.2: Lists in lists in lists in..	103
Chapter 19: Dictionary	105
Section 19.1: Introduction to Dictionary	105
Section 19.2: Avoiding KeyError Exceptions	106
Section 19.3: Iterating Over a Dictionary	106
Section 19.4: Dictionary with default values	107
Section 19.5: Merging dictionaries	108
Section 19.6: Accessing keys and values	108
Section 19.7: Accessing values of a dictionary	109

Section 19.8: Creating a dictionary	109
Section 19.9: Creating an ordered dictionary	110
Section 19.10: Unpacking dictionaries using the ** operator	110
Section 19.11: The trailing comma	111
Section 19.12: The dict() constructor	111
Section 19.13: Dictionaries Example	111
Section 19.14: All combinations of dictionary values	112
Chapter 20: List	113
Section 20.1: List methods and supported operators	113
Section 20.2: Accessing list values	118
Section 20.3: Checking if list is empty	119
Section 20.4: Iterating over a list	119
Section 20.5: Checking whether an item is in a list	120
Section 20.6: Any and All	120
Section 20.7: Reversing list elements	121
Section 20.8: Concatenate and Merge lists	121
Section 20.9: Length of a list	122
Section 20.10: Remove duplicate values in list	122
Section 20.11: Comparison of lists	123
Section 20.12: Accessing values in nested list	123
Section 20.13: Initializing a List to a Fixed Number of Elements	124
Chapter 21: List comprehensions	126
Section 21.1: List Comprehensions	126
Section 21.2: Conditional List Comprehensions	128
Section 21.3: Avoid repetitive and expensive operations using conditional clause	130
Section 21.4: Dictionary Comprehensions	131
Section 21.5: List Comprehensions with Nested Loops	132
Section 21.6: Generator Expressions	134
Section 21.7: Set Comprehensions	136
Section 21.8: Refactoring filter and map to list comprehensions	136
Section 21.9: Comprehensions involving tuples	137
Section 21.10: Counting Occurrences Using Comprehension	138
Section 21.11: Changing Types in a List	138
Section 21.12: Nested List Comprehensions	138
Section 21.13: Iterate two or more list simultaneously within list comprehension	139
Chapter 22: List slicing (selecting parts of lists)	140
Section 22.1: Using the third "step" argument	140
Section 22.2: Selecting a sublist from a list	140
Section 22.3: Reversing a list with slicing	140
Section 22.4: Shifting a list using slicing	140
Chapter 23: groupby()	142
Section 23.1: Example 4	142
Section 23.2: Example 2	142
Section 23.3: Example 3	143
Chapter 24: Linked lists	145
Section 24.1: Single linked list example	145
Chapter 25: Linked List Node	149
Section 25.1: Write a simple Linked List Node in python	149
Chapter 26: Filter	150

Section 26.1: Basic use of filter	150
Section 26.2: Filter without function	150
Section 26.3: Filter as short-circuit check	151
Section 26.4: Complementary function: filterfalse, ifilterfalse	151
Chapter 27: Heapq	153
Section 27.1: Largest and smallest items in a collection	153
Section 27.2: Smallest item in a collection	153
Chapter 28: Tuple	155
Section 28.1: Tuple	155
Section 28.2: Tuples are immutable	156
Section 28.3: Packing and Unpacking Tuples	156
Section 28.4: Built-in Tuple Functions	157
Section 28.5: Tuple Are Element-wise Hashable and Equatable	158
Section 28.6: Indexing Tuples	159
Section 28.7: Reversing Elements	159
Chapter 29: Basic Input and Output	160
Section 29.1: Using the print function	160
Section 29.2: Input from a File	160
Section 29.3: Read from stdin	162
Section 29.4: Using input() and raw_input()	162
Section 29.5: Function to prompt user for a number	162
Section 29.6: Printing a string without a newline at the end	163
Chapter 30: Files & Folders I/O	165
Section 30.1: File modes	165
Section 30.2: Reading a file line-by-line	166
Section 30.3: Iterate files (recursively)	167
Section 30.4: Getting the full contents of a file	167
Section 30.5: Writing to a file	168
Section 30.6: Check whether a file or path exists	169
Section 30.7: Random File Access Using mmap	170
Section 30.8: Replacing text in a file	170
Section 30.9: Checking if a file is empty	170
Section 30.10: Read a file between a range of lines	171
Section 30.11: Copy a directory tree	171
Section 30.12: Copying contents of one file to a different file	171
Chapter 31: os.path	172
Section 31.1: Join Paths	172
Section 31.2: Path Component Manipulation	172
Section 31.3: Get the parent directory	172
Section 31.4: If the given path exists	172
Section 31.5: check if the given path is a directory, file, symbolic link, mount point etc	173
Section 31.6: Absolute Path from Relative Path	173
Chapter 32: Iterables and Iterators	174
Section 32.1: Iterator vs Iterable vs Generator	174
Section 32.2: Extract values one by one	175
Section 32.3: Iterating over entire iterable	175
Section 32.4: Verify only one element in iterable	175
Section 32.5: What can be iterable	176
Section 32.6: Iterator isn't reentrant!	176

Chapter 33: Functions	177
Section 33.1: Defining and calling simple functions	177
Section 33.2: Defining a function with an arbitrary number of arguments	178
Section 33.3: Lambda (Inline/Anonymous) Functions	181
Section 33.4: Defining a function with optional arguments	183
Section 33.5: Defining a function with optional mutable arguments	184
Section 33.6: Argument passing and mutability	185
Section 33.7: Returning values from functions	186
Section 33.8: Closure	186
Section 33.9: Forcing the use of named parameters	187
Section 33.10: Nested functions	188
Section 33.11: Recursion limit	188
Section 33.12: Recursive Lambda using assigned variable	189
Section 33.13: Recursive functions	189
Section 33.14: Defining a function with arguments	190
Section 33.15: Iterable and dictionary unpacking	190
Section 33.16: Defining a function with multiple arguments	192
Chapter 34: Defining functions with list arguments	193
Section 34.1: Function and Call	193
Chapter 35: Functional Programming in Python	194
Section 35.1: Lambda Function	194
Section 35.2: Map Function	194
Section 35.3: Reduce Function	194
Section 35.4: Filter Function	194
Chapter 36: Partial functions	195
Section 36.1: Raise the power	195
Chapter 37: Decorators	196
Section 37.1: Decorator function	196
Section 37.2: Decorator class	197
Section 37.3: Decorator with arguments (decorator factory)	198
Section 37.4: Making a decorator look like the decorated function	200
Section 37.5: Using a decorator to time a function	200
Section 37.6: Create singleton class with a decorator	201
Chapter 38: Classes	202
Section 38.1: Introduction to classes	202
Section 38.2: Bound, unbound, and static methods	203
Section 38.3: Basic inheritance	205
Section 38.4: Monkey Patching	207
Section 38.5: New-style vs. old-style classes	207
Section 38.6: Class methods: alternate initializers	208
Section 38.7: Multiple Inheritance	210
Section 38.8: Properties	212
Section 38.9: Default values for instance variables	213
Section 38.10: Class and instance variables	214
Section 38.11: Class composition	215
Section 38.12: Listing All Class Members	216
Section 38.13: Singleton class	217
Section 38.14: Descriptors and Dotted Lookups	218
Chapter 39: Metaclasses	219

Section 39.1: Basic Metaclasses	219
Section 39.2: Singletons using metaclasses	220
Section 39.3: Using a metaclass	220
Section 39.4: Introduction to Metaclasses	220
Section 39.5: Custom functionality with metaclasses	221
Section 39.6: The default metaclass	222
Chapter 40: String Formatting	224
Section 40.1: Basics of String Formatting	224
Section 40.2: Alignment and padding	225
Section 40.3: Format literals (f-string)	226
Section 40.4: Float formatting	226
Section 40.5: Named placeholders	227
Section 40.6: String formatting with datetime	228
Section 40.7: Formatting Numerical Values	228
Section 40.8: Nested formatting	229
Section 40.9: Format using Getitem and Getattr	229
Section 40.10: Padding and truncating strings combined	229
Section 40.11: Custom formatting for a class	230
Chapter 41: String Methods	232
Section 41.1: Changing the capitalization of a string	232
Section 41.2: str.translate: Translating characters in a string	233
Section 41.3: str.format and f-strings: Format values into a string	234
Section 41.4: String module's useful constants	235
Section 41.5: Stripping unwanted leading/trailing characters from a string	236
Section 41.6: Reversing a string	237
Section 41.7: Split a string based on a delimiter into a list of strings	237
Section 41.8: Replace all occurrences of one substring with another substring	238
Section 41.9: Testing what a string is composed of	239
Section 41.10: String Contains	241
Section 41.11: Join a list of strings into one string	241
Section 41.12: Counting number of times a substring appears in a string	242
Section 41.13: Case insensitive string comparisons	242
Section 41.14: Justify strings	243
Section 41.15: Test the starting and ending characters of a string	244
Section 41.16: Conversion between str or bytes data and unicode characters	245
Chapter 42: Using loops within functions	247
Section 42.1: Return statement inside loop in a function	247
Chapter 43: Importing modules	248
Section 43.1: Importing a module	248
Section 43.2: The <code>__all__</code> special variable	249
Section 43.3: Import modules from an arbitrary filesystem location	250
Section 43.4: Importing all names from a module	250
Section 43.5: Programmatic importing	251
Section 43.6: PEP8 rules for Imports	251
Section 43.7: Importing specific names from a module	252
Section 43.8: Importing submodules	252
Section 43.9: Re-importing a module	252
Section 43.10: <code>__import__()</code> function	253
Chapter 44: Difference between Module and Package	254
Section 44.1: Modules	254

Section 44.2: Packages	254
Chapter 45: Math Module	255
Section 45.1: Rounding: round, floor, ceil, trunc	255
Section 45.2: Trigonometry	256
Section 45.3: Pow for faster exponentiation	257
Section 45.4: Infinity and NaN ("not a number")	257
Section 45.5: Logarithms	260
Section 45.6: Constants	260
Section 45.7: Imaginary Numbers	261
Section 45.8: Copying signs	261
Section 45.9: Complex numbers and the cmath module	261
Chapter 46: Complex math	264
Section 46.1: Advanced complex arithmetic	264
Section 46.2: Basic complex arithmetic	265
Chapter 47: Collections module	266
Section 47.1: collections.Counter	266
Section 47.2: collections.OrderedDict	267
Section 47.3: collections.defaultdict	268
Section 47.4: collections.namedtuple	269
Section 47.5: collections.deque	270
Section 47.6: collections.ChainMap	271
Chapter 48: Operator module	273
Section 48.1: Itemgetter	273
Section 48.2: Operators as alternative to an infix operator	273
Section 48.3: Methodcaller	273
Chapter 49: JSON Module	275
Section 49.1: Storing data in a file	275
Section 49.2: Retrieving data from a file	275
Section 49.3: Formatting JSON output	275
Section 49.4: `load` vs `loads`, `dump` vs `dumps`	276
Section 49.5: Calling `json.tool` from the command line to pretty-print JSON output	277
Section 49.6: JSON encoding custom objects	277
Section 49.7: Creating JSON from Python dict	278
Section 49.8: Creating Python dict from JSON	278
Chapter 50: Sqlite3 Module	279
Section 50.1: Sqlite3 - Not require separate server process	279
Section 50.2: Getting the values from the database and Error handling	279
Chapter 51: The os Module	281
Section 51.1: makedirs - recursive directory creation	281
Section 51.2: Create a directory	282
Section 51.3: Get current directory	282
Section 51.4: Determine the name of the operating system	282
Section 51.5: Remove a directory	282
Section 51.6: Follow a symlink (POSIX)	282
Section 51.7: Change permissions on a file	282
Chapter 52: The locale Module	283
Section 52.1: Currency Formatting US Dollars Using the locale Module	283
Chapter 53: Itertools Module	284
Section 53.1: Combinations method in Itertools Module	284

Section 53.2: itertools.dropwhile	284
Section 53.3: Zipping two iterators until they are both exhausted	285
Section 53.4: Take a slice of a generator	285
Section 53.5: Grouping items from an iterable object using a function	286
Section 53.6: itertools.takewhile	287
Section 53.7: itertools.permutations	287
Section 53.8: itertools.repeat	288
Section 53.9: Get an accumulated sum of numbers in an iterable	288
Section 53.10: Cycle through elements in an iterator	288
Section 53.11: itertools.product	288
Section 53.12: itertools.count	289
Section 53.13: Chaining multiple iterators together	290
Chapter 54: Asyncio Module	291
Section 54.1: Coroutine and Delegation Syntax	291
Section 54.2: Asynchronous Executors	292
Section 54.3: Using UVLoop	293
Section 54.4: Synchronization Primitive: Event	293
Section 54.5: A Simple Websocket	294
Section 54.6: Common Misconception about asyncio	294
Chapter 55: Random module	296
Section 55.1: Creating a random user password	296
Section 55.2: Create cryptographically secure random numbers	296
Section 55.3: Random and sequences: shuffle, choice and sample	297
Section 55.4: Creating random integers and floats: randint, randrange, random, and uniform	298
Section 55.5: Reproducible random numbers: Seed and State	299
Section 55.6: Random Binary Decision	300
Chapter 56: Functools Module	301
Section 56.1: partial	301
Section 56.2: cmp_to_key	301
Section 56.3: lru_cache	301
Section 56.4: total_ordering	302
Section 56.5: reduce	303
Chapter 57: The dis module	304
Section 57.1: What is Python bytecode?	304
Section 57.2: Constants in the dis module	304
Section 57.3: Disassembling modules	304
Chapter 58: The base64 Module	306
Section 58.1: Encoding and Decoding Base64	307
Section 58.2: Encoding and Decoding Base32	308
Section 58.3: Encoding and Decoding Base16	309
Section 58.4: Encoding and Decoding ASCII85	309
Section 58.5: Encoding and Decoding Base85	310
Chapter 59: Queue Module	311
Section 59.1: Simple example	311
Chapter 60: Deque Module	312
Section 60.1: Basic deque using	312
Section 60.2: Available methods in deque	312
Section 60.3: limit deque size	313
Section 60.4: Breadth First Search	313

Chapter 61: Webbrowser Module	314
Section 61.1: Opening a URL with Default Browser	314
Section 61.2: Opening a URL with Different Browsers	315
Chapter 62: tkinter	316
Section 62.1: Geometry Managers	316
Section 62.2: A minimal tkinter Application	317
Chapter 63: pyautogui module	319
Section 63.1: Mouse Functions	319
Section 63.2: Keyboard Functions	319
Section 63.3: Screenshot And Image Recognition	319
Chapter 64: Indexing and Slicing	320
Section 64.1: Basic Slicing	320
Section 64.2: Reversing an object	321
Section 64.3: Slice assignment	321
Section 64.4: Making a shallow copy of an array	321
Section 64.5: Indexing custom classes: <code>getitem</code>, <code>setitem</code> and <code>delitem</code>	322
Section 64.6: Basic Indexing	323
Chapter 65: Plotting with Matplotlib	324
Section 65.1: Plots with Common X-axis but different Y-axis : Using <code>twinx()</code>	324
Section 65.2: Plots with common Y-axis and different X-axis using <code>twinx()</code>	325
Section 65.3: A Simple Plot in Matplotlib	327
Section 65.4: Adding more features to a simple plot : axis labels, title, axis ticks, grid, and legend	328
Section 65.5: Making multiple plots in the same figure by superimposition similar to MATLAB	329
Section 65.6: Making multiple Plots in the same figure using plot superimposition with separate plot commands	330
Chapter 66: graph-tool	332
Section 66.1: PyDotPlus	332
Section 66.2: PyGraphviz	332
Chapter 67: Generators	334
Section 67.1: Introduction	334
Section 67.2: Infinite sequences	336
Section 67.3: Sending objects to a generator	337
Section 67.4: Yielding all values from another iterable	338
Section 67.5: Iteration	338
Section 67.6: The <code>next()</code> function	338
Section 67.7: Coroutines	339
Section 67.8: Refactoring list-building code	339
Section 67.9: Yield with recursion: recursively listing all files in a directory	340
Section 67.10: Generator expressions	341
Section 67.11: Using a generator to find Fibonacci Numbers	341
Section 67.12: Searching	341
Section 67.13: Iterating over generators in parallel	342
Chapter 68: Reduce	343
Section 68.1: Overview	343
Section 68.2: Using reduce	343
Section 68.3: Cumulative product	344
Section 68.4: Non short-circuit variant of <code>any/all</code>	344
Chapter 69: Map Function	345
Section 69.1: Basic use of <code>map</code>, <code>itertools.imap</code> and <code>future_builtins.map</code>	345

Section 69.2: Mapping each value in an iterable	345
Section 69.3: Mapping values of different iterables	346
Section 69.4: Transposing with Map: Using "None" as function argument (python 2.x only)	348
Section 69.5: Series and Parallel Mapping	348
Chapter 70: Exponentiation	351
Section 70.1: Exponentiation using builtins: ** and pow()	351
Section 70.2: Square root: math.sqrt() and cmath.sqrt	351
Section 70.3: Modular exponentiation: pow() with 3 arguments	352
Section 70.4: Computing large integer roots	352
Section 70.5: Exponentiation using the math module: math.pow()	353
Section 70.6: Exponential function: math.exp() and cmath.exp()	354
Section 70.7: Exponential function minus 1: math.expm1()	354
Section 70.8: Magic methods and exponentiation: builtin, math and cmath	355
Section 70.9: Roots: nth-root with fractional exponents	356
Chapter 71: Searching	357
Section 71.1: Searching for an element	357
Section 71.2: Searching in custom classes: __contains__ and __iter__	357
Section 71.3: Getting the index for strings: str.index(), str.rindex() and str.find(), str.rfind()	358
Section 71.4: Getting the index list and tuples: list.index(), tuple.index()	359
Section 71.5: Searching key(s) for a value in dict	359
Section 71.6: Getting the index for sorted sequences: bisect.bisect_left()	360
Section 71.7: Searching nested sequences	360
Chapter 72: Sorting, Minimum and Maximum	362
Section 72.1: Make custom classes orderable	362
Section 72.2: Special case: dictionaries	364
Section 72.3: Using the key argument	365
Section 72.4: Default Argument to max, min	365
Section 72.5: Getting a sorted sequence	366
Section 72.6: Extracting N largest or N smallest items from an iterable	366
Section 72.7: Getting the minimum or maximum of several values	367
Section 72.8: Minimum and Maximum of a sequence	367
Chapter 73: Counting	368
Section 73.1: Counting all occurrence of all items in an iterable: collections.Counter	368
Section 73.2: Getting the most common value(-s): collections.Counter.most_common()	368
Section 73.3: Counting the occurrences of one item in a sequence: list.count() and tuple.count()	368
Section 73.4: Counting the occurrences of a substring in a string: str.count()	369
Section 73.5: Counting occurrences in numpy array	369
Chapter 74: The Print Function	370
Section 74.1: Print basics	370
Section 74.2: Print parameters	371
Chapter 75: Regular Expressions (Regex)	373
Section 75.1: Matching the beginning of a string	373
Section 75.2: Searching	374
Section 75.3: Precompiled patterns	374
Section 75.4: Flags	375
Section 75.5: Replacing	376
Section 75.6: Find All Non-Overlapping Matches	376
Section 75.7: Checking for allowed characters	377
Section 75.8: Splitting a string using regular expressions	377
Section 75.9: Grouping	377

Section 75.10: Escaping Special Characters	378
Section 75.11: Match an expression only in specific locations	379
Section 75.12: Iterating over matches using <code>`re.finditer`</code>	380
Chapter 76: Copying data	381
Section 76.1: Copy a dictionary	381
Section 76.2: Performing a shallow copy	381
Section 76.3: Performing a deep copy	381
Section 76.4: Performing a shallow copy of a list	381
Section 76.5: Copy a set	381
Chapter 77: Context Managers (“with” Statement)	383
Section 77.1: Introduction to context managers and the with statement	383
Section 77.2: Writing your own context manager	383
Section 77.3: Writing your own contextmanager using generator syntax	384
Section 77.4: Multiple context managers	385
Section 77.5: Assigning to a target	385
Section 77.6: Manage Resources	386
Chapter 78: The <code>__name__</code> special variable	387
Section 78.1: <code>__name__ == ‘__main__’</code>	387
Section 78.2: Use in logging	387
Section 78.3: <code>function class or module. __name__</code>	387
Chapter 79: Checking Path Existence and Permissions	389
Section 79.1: Perform checks using <code>os.access</code>	389
Chapter 80: Creating Python packages	390
Section 80.1: Introduction	390
Section 80.2: Uploading to PyPI	390
Section 80.3: Making package executable	392
Chapter 81: Usage of “pip” module: PyPI Package Manager	394
Section 81.1: Example use of commands	394
Section 81.2: Handling ImportError Exception	394
Section 81.3: Force install	395
Chapter 82: pip: PyPI Package Manager	396
Section 82.1: Install Packages	396
Section 82.2: To list all packages installed using <code>`pip`</code>	396
Section 82.3: Upgrade Packages	396
Section 82.4: Uninstall Packages	397
Section 82.5: Updating all outdated packages on Linux	397
Section 82.6: Updating all outdated packages on Windows	397
Section 82.7: Create a requirements.txt file of all packages on the system	397
Section 82.8: Using a certain Python version with pip	398
Section 82.9: Create a requirements.txt file of packages only in the current virtualenv	398
Section 82.10: Installing packages not yet on pip as wheels	399
Chapter 83: Parsing Command Line arguments	402
Section 83.1: Hello world in argparse	402
Section 83.2: Using command line arguments with argv	402
Section 83.3: Setting mutually exclusive arguments with argparse	403
Section 83.4: Basic example with docopt	404
Section 83.5: Custom parser error message with argparse	404
Section 83.6: Conceptual grouping of arguments with <code>argparse.add_argument_group()</code>	405
Section 83.7: Advanced example with docopt and <code>docopt_dispatch</code>	406

Chapter 84: Subprocess Library	408
Section 84.1: More flexibility with Popen	408
Section 84.2: Calling External Commands	409
Section 84.3: How to create the command list argument	409
Chapter 85: setup.py	410
Section 85.1: Purpose of setup.py	410
Section 85.2: Using source control metadata in setup.py	410
Section 85.3: Adding command line scripts to your python package	411
Section 85.4: Adding installation options	411
Chapter 86: Recursion	413
Section 86.1: The What, How, and When of Recursion	413
Section 86.2: Tree exploration with recursion	416
Section 86.3: Sum of numbers from 1 to n	417
Section 86.4: Increasing the Maximum Recursion Depth	417
Section 86.5: Tail Recursion - Bad Practice	418
Section 86.6: Tail Recursion Optimization Through Stack Introspection	418
Chapter 87: Type Hints	420
Section 87.1: Adding types to a function	420
Section 87.2: NamedTuple	421
Section 87.3: Generic Types	421
Section 87.4: Variables and Attributes	421
Section 87.5: Class Members and Methods	422
Section 87.6: Type hints for keyword arguments	422
Chapter 88: Exceptions	423
Section 88.1: Catching Exceptions	423
Section 88.2: Do not catch everything!	423
Section 88.3: Re-raising exceptions	424
Section 88.4: Catching multiple exceptions	424
Section 88.5: Exception Hierarchy	425
Section 88.6: Else	427
Section 88.7: Raising Exceptions	427
Section 88.8: Creating custom exception types	428
Section 88.9: Practical examples of exception handling	428
Section 88.10: Exceptions are Objects too	429
Section 88.11: Running clean-up code with finally	429
Section 88.12: Chain exceptions with raise from	430
Chapter 89: Raise Custom Errors / Exceptions	431
Section 89.1: Custom Exception	431
Section 89.2: Catch custom Exception	431
Chapter 90: Commonwealth Exceptions	432
Section 90.1: Other Errors	432
Section 90.2: NameError: name '???' is not defined	433
Section 90.3: TypeErrors	434
Section 90.4: Syntax Error on good code	435
Section 90.5: IndentationErrors (or indentation SyntaxErrors)	436
Chapter 91: urllib	438
Section 91.1: HTTP GET	438
Section 91.2: HTTP POST	438
Section 91.3: Decode received bytes according to content type encoding	439

Chapter 92: Web scraping with Python	440
Section 92.1: Scraping using the Scrapy framework	440
Section 92.2: Scraping using Selenium WebDriver	440
Section 92.3: Basic example of using requests and lxml to scrape some data	441
Section 92.4: Maintaining web-scraping session with requests	441
Section 92.5: Scraping using BeautifulSoup4	442
Section 92.6: Simple web content download with urllib.request	442
Section 92.7: Modify Scrapy user agent	442
Section 92.8: Scraping with curl	442
Chapter 93: HTML Parsing	444
Section 93.1: Using CSS selectors in BeautifulSoup	444
Section 93.2: PyQuery	444
Section 93.3: Locate a text after an element in BeautifulSoup	445
Chapter 94: Manipulating XML	446
Section 94.1: Opening and reading using an ElementTree	446
Section 94.2: Create and Build XML Documents	446
Section 94.3: Modifying an XML File	447
Section 94.4: Searching the XML with XPath	447
Section 94.5: Opening and reading large XML files using itersparse (incremental parsing)	448
Chapter 95: Python Requests Post	449
Section 95.1: Simple Post	449
Section 95.2: Form Encoded Data	450
Section 95.3: File Upload	450
Section 95.4: Responses	451
Section 95.5: Authentication	451
Section 95.6: Proxies	452
Chapter 96: Distribution	454
Section 96.1: py2app	454
Section 96.2: cx_Freeze	455
Chapter 97: Property Objects	456
Section 97.1: Using the @property decorator for read-write properties	456
Section 97.2: Using the @property decorator	456
Section 97.3: Overriding just a getter, setter or a deleter of a property object	457
Section 97.4: Using properties without decorators	457
Chapter 98: Overloading	460
Section 98.1: Operator overloading	460
Section 98.2: Magic/Dunder Methods	461
Section 98.3: Container and sequence types	462
Section 98.4: Callable types	463
Section 98.5: Handling unimplemented behaviour	463
Chapter 99: Polymorphism	465
Section 99.1: Duck Typing	465
Section 99.2: Basic Polymorphism	465
Chapter 100: Method Overriding	468
Section 100.1: Basic method overriding	468
Chapter 101: User-Defined Methods	469
Section 101.1: Creating user-defined method objects	469
Section 101.2: Turtle example	470
Chapter 102: String representations of class instances: <code>__str__</code> and <code>__repr__</code>	

methods	471
Section 102.1: Motivation	471
Section 102.2: Both methods implemented, eval-round-trip style repr()	475
Chapter 103: Debugging	476
Section 103.1: Via IPython and ipdb	476
Section 103.2: The Python Debugger: Step-through Debugging with pdb	476
Section 103.3: Remote debugger	478
Chapter 104: Reading and Writing CSV	479
Section 104.1: Using pandas	479
Section 104.2: Writing a TSV file	479
Chapter 105: Writing to CSV from String or List	480
Section 105.1: Basic Write Example	480
Section 105.2: Appending a String as a newline in a CSV file	480
Chapter 106: Dynamic code execution with `exec` and `eval`	481
Section 106.1: Executing code provided by untrusted user using exec, eval, or ast.literal_eval	481
Section 106.2: Evaluating a string containing a Python literal with ast.literal_eval	481
Section 106.3: Evaluating statements with exec	481
Section 106.4: Evaluating an expression with eval	482
Section 106.5: Precompiling an expression to evaluate it multiple times	482
Section 106.6: Evaluating an expression with eval using custom globals	482
Chapter 107: PyInstaller - Distributing Python Code	483
Section 107.1: Installation and Setup	483
Section 107.2: Using Pyinstaller	483
Section 107.3: Bundling to One Folder	484
Section 107.4: Bundling to a Single File	484
Chapter 108: Data Visualization with Python	485
Section 108.1: Seaborn	485
Section 108.2: Matplotlib	487
Section 108.3: Plotly	488
Section 108.4: MayaVI	490
Chapter 109: The Interpreter (Command Line Console)	492
Section 109.1: Getting general help	492
Section 109.2: Referring to the last expression	492
Section 109.3: Opening the Python console	493
Section 109.4: The PYTHONSTARTUP variable	493
Section 109.5: Command line arguments	493
Section 109.6: Getting help about an object	494
Chapter 110: *args and **kwargs	496
Section 110.1: Using **kwargs when writing functions	496
Section 110.2: Using *args when writing functions	496
Section 110.3: Populating kwarg values with a dictionary	497
Section 110.4: Keyword-only and Keyword-required arguments	497
Section 110.5: Using **kwargs when calling functions	497
Section 110.6: **kwargs and default values	497
Section 110.7: Using *args when calling functions	498
Chapter 111: Garbage Collection	499
Section 111.1: Reuse of primitive objects	499
Section 111.2: Effects of the del command	499
Section 111.3: Reference Counting	500

Section 111.4: Garbage Collector for Reference Cycles	500
Section 111.5: Forcefully deallocating objects	501
Section 111.6: Viewing the refcount of an object	502
Section 111.7: Do not wait for the garbage collection to clean up	502
Section 111.8: Managing garbage collection	502
Chapter 112: Pickle data serialisation	504
Section 112.1: Using Pickle to serialize and deserialize an object	504
Section 112.2: Customize Pickled Data	504
Chapter 113: Binary Data	506
Section 113.1: Format a list of values into a byte object	506
Section 113.2: Unpack a byte object according to a format string	506
Section 113.3: Packing a structure	506
Chapter 114: Idioms	508
Section 114.1: Dictionary key initializations	508
Section 114.2: Switching variables	508
Section 114.3: Use truth value testing	508
Section 114.4: Test for " __main__ " to avoid unexpected code execution	509
Chapter 115: Data Serialization	510
Section 115.1: Serialization using JSON	510
Section 115.2: Serialization using Pickle	510
Chapter 116: Multiprocessing	512
Section 116.1: Running Two Simple Processes	512
Section 116.2: Using Pool and Map	512
Chapter 117: Multithreading	514
Section 117.1: Basics of multithreading	514
Section 117.2: Communicating between threads	515
Section 117.3: Creating a worker pool	516
Section 117.4: Advanced use of multithreads	516
Section 117.5: Stoppable Thread with a while Loop	518
Chapter 118: Processes and Threads	519
Section 118.1: Global Interpreter Lock	519
Section 118.2: Running in Multiple Threads	520
Section 118.3: Running in Multiple Processes	521
Section 118.4: Sharing State Between Threads	521
Section 118.5: Sharing State Between Processes	522
Chapter 119: Python concurrency	523
Section 119.1: The multiprocessing module	523
Section 119.2: The threading module	524
Section 119.3: Passing data between multiprocessing processes	524
Chapter 120: Parallel computation	526
Section 120.1: Using the multiprocessing module to parallelise tasks	526
Section 120.2: Using a C-extension to parallelize tasks	526
Section 120.3: Using Parent and Children scripts to execute code in parallel	526
Section 120.4: Using PyPar module to parallelize	527
Chapter 121: Sockets	528
Section 121.1: Raw Sockets on Linux	528
Section 121.2: Sending data via UDP	528
Section 121.3: Receiving data via UDP	529
Section 121.4: Sending data via TCP	529

Section 121.5: Multi-threaded TCP Socket Server	529
Chapter 122: Websockets	532
Section 122.1: Simple Echo with aiohttp	532
Section 122.2: Wrapper Class with aiohttp	532
Section 122.3: Using Autobahn as a Websocket Factory	533
Chapter 123: Sockets And Message Encryption/Decryption Between Client and Server	535
Section 123.1: Server side Implementation	535
Section 123.2: Client side Implementation	537
Chapter 124: Python Networking	539
Section 124.1: Creating a Simple Http Server	539
Section 124.2: Creating a TCP server	539
Section 124.3: Creating a UDP Server	540
Section 124.4: Start Simple HttpServer in a thread and open the browser	540
Section 124.5: The simplest Python socket client-server example	541
Chapter 125: Python HTTP Server	542
Section 125.1: Running a simple HTTP server	542
Section 125.2: Serving files	542
Section 125.3: Basic handling of GET, POST, PUT using BaseHTTPRequestHandler	543
Section 125.4: Programmatic API of SimpleHTTPServer	544
Chapter 126: Flask	546
Section 126.1: Files and Templates	546
Section 126.2: The basics	546
Section 126.3: Routing URLs	547
Section 126.4: HTTP Methods	548
Section 126.5: Jinja Templating	548
Section 126.6: The Request Object	549
Chapter 127: Introduction to RabbitMQ using AMQPStorm	551
Section 127.1: How to consume messages from RabbitMQ	551
Section 127.2: How to publish messages to RabbitMQ	552
Section 127.3: How to create a delayed queue in RabbitMQ	552
Chapter 128: Descriptor	555
Section 128.1: Simple descriptor	555
Section 128.2: Two-way conversions	556
Chapter 129: tempfile NamedTemporaryFile	557
Section 129.1: Create (and write to a) known, persistent temporary file	557
Chapter 130: Input, Subset and Output External Data Files using Pandas	558
Section 130.1: Basic Code to Import, Subset and Write External Data Files Using Pandas	558
Chapter 131: Unzipping Files	560
Section 131.1: Using Python ZipFile.extractall() to decompress a ZIP file	560
Section 131.2: Using Python TarFile.extractall() to decompress a tarball	560
Chapter 132: Working with ZIP archives	561
Section 132.1: Examining Zipfile Contents	561
Section 132.2: Opening Zip Files	561
Section 132.3: Extracting zip file contents to a directory	562
Section 132.4: Creating new archives	562
Chapter 133: Getting start with GZip	563
Section 133.1: Read and write GNU zip files	563

Chapter 134: Stack	564
Section 134.1: Creating a Stack class with a List Object	564
Section 134.2: Parsing Parentheses	565
Chapter 135: Working around the Global Interpreter Lock (GIL)	566
Section 135.1: Multiprocessing.Pool	566
Section 135.2: Cython nogil:	567
Chapter 136: Deployment	568
Section 136.1: Uploading a Conda Package	568
Chapter 137: Logging	570
Section 137.1: Introduction to Python Logging	570
Section 137.2: Logging exceptions	571
Chapter 138: Web Server Gateway Interface (WSGI)	574
Section 138.1: Server Object (Method)	574
Chapter 139: Python Server Sent Events	575
Section 139.1: Flask SSE	575
Section 139.2: Asyncio SSE	575
Chapter 140: Alternatives to switch statement from other languages	576
Section 140.1: Use what the language offers: the if/else construct	576
Section 140.2: Use a dict of functions	576
Section 140.3: Use class introspection	577
Section 140.4: Using a context manager	578
Chapter 141: List destructuring (aka packing and unpacking)	579
Section 141.1: Destructuring assignment	579
Section 141.2: Packing function arguments	580
Section 141.3: Unpacking function arguments	582
Chapter 142: Accessing Python source code and bytecode	583
Section 142.1: Display the bytecode of a function	583
Section 142.2: Display the source code of an object	583
Section 142.3: Exploring the code object of a function	584
Chapter 143: Mixins	585
Section 143.1: Mixin	585
Section 143.2: Overriding Methods in Mixins	586
Chapter 144: Attribute Access	587
Section 144.1: Basic Attribute Access using the Dot Notation	587
Section 144.2: Setters, Getters & Properties	587
Chapter 145: ArcPy	589
Section 145.1: createDissolvedGDB to create a file gdb on the workspace	589
Section 145.2: Printing one field's value for all rows of feature class in file geodatabase using Search Cursor	589
Chapter 146: Abstract Base Classes (abc)	590
Section 146.1: Setting the ABCMeta metaclass	590
Section 146.2: Why/How to use ABCMeta and @abstractmethod	590
Chapter 147: Plugin and Extension Classes	592
Section 147.1: Mixins	592
Section 147.2: Plugins with Customized Classes	593
Chapter 148: Immutable datatypes(int, float, str, tuple and frozensets)	595
Section 148.1: Individual characters of strings are not assignable	595
Section 148.2: Tuple's individual members aren't assignable	595

Section 148.3: Frozenset's are immutable and not assignable	595
Chapter 149: Incompatibilities moving from Python 2 to Python 3	596
Section 149.1: Integer Division	596
Section 149.2: Unpacking Iterables	597
Section 149.3: Strings: Bytes versus Unicode	599
Section 149.4: Print statement vs. Print function	601
Section 149.5: Differences between range and xrange functions	602
Section 149.6: Raising and handling Exceptions	603
Section 149.7: Leaked variables in list comprehension	605
Section 149.8: True, False and None	606
Section 149.9: User Input	606
Section 149.10: Comparison of different types	607
Section 149.11: .next() method on iterators renamed	607
Section 149.12: filter(), map() and zip() return iterators instead of sequences	608
Section 149.13: Renamed modules	608
Section 149.14: Removed operators <> and `, synonymous with != and repr()	609
Section 149.15: long vs. int	609
Section 149.16: All classes are "new-style classes" in Python 3	610
Section 149.17: Reduce is no longer a built-in	611
Section 149.18: Absolute/Relative Imports	611
Section 149.19: map()	613
Section 149.20: The round() function tie-breaking and return type	614
Section 149.21: File I/O	615
Section 149.22: cmp function removed in Python 3	615
Section 149.23: Octal Constants	616
Section 149.24: Return value when writing to a file object	616
Section 149.25: exec statement is a function in Python 3	616
Section 149.26: encode/decode to hex no longer available	617
Section 149.27: Dictionary method changes	618
Section 149.28: Class Boolean Value	618
Section 149.29: hasattr function bug in Python 2	619
Chapter 150: 2to3 tool	620
Section 150.1: Basic Usage	620
Chapter 151: Non-official Python implementations	622
Section 151.1: IronPython	622
Section 151.2: Jython	622
Section 151.3: Transcrypt	623
Chapter 152: Abstract syntax tree	626
Section 152.1: Analyze functions in a python script	626
Chapter 153: Unicode and bytes	628
Section 153.1: Encoding/decoding error handling	628
Section 153.2: File I/O	628
Section 153.3: Basics	629
Chapter 154: Python Serial Communication (pyserial)	631
Section 154.1: Initialize serial device	631
Section 154.2: Read from serial port	631
Section 154.3: Check what serial ports are available on your machine	631
Chapter 155: Neo4j and Cypher using Py2Neo	633
Section 155.1: Adding Nodes to Neo4j Graph	633

Section 155.2: Importing and Authenticating	633
Section 155.3: Adding Relationships to Neo4j Graph	633
Section 155.4: Query 1 : Autocomplete on News Titles	633
Section 155.5: Query 2 : Get News Articles by Location on a particular date	634
Section 155.6: Cypher Query Samples	634
Chapter 156: Basic Curses with Python	635
Section 156.1: The wrapper() helper function	635
Section 156.2: Basic Invocation Example	635
Chapter 157: Templates in python	636
Section 157.1: Simple data output program using template	636
Section 157.2: Changing delimiter	636
Chapter 158: Pillow	637
Section 158.1: Read Image File	637
Section 158.2: Convert files to JPEG	637
Chapter 159: The pass statement	638
Section 159.1: Ignore an exception	638
Section 159.2: Create a new Exception that can be caught	638
Chapter 160: CLI subcommands with precise help output	639
Section 160.1: Native way (no libraries)	639
Section 160.2: argparse (default help formatter)	639
Section 160.3: argparse (custom help formatter)	640
Chapter 161: Database Access	642
Section 161.1: SQLite	642
Section 161.2: Accessing MySQL database using MySQLdb	647
Section 161.3: Connection	648
Section 161.4: PostgreSQL Database access using psycopg2	649
Section 161.5: Oracle database	650
Section 161.6: Using sqlalchemy	652
Chapter 162: Connecting Python to SQL Server	653
Section 162.1: Connect to Server, Create Table, Query Data	653
Chapter 163: PostgreSQL	654
Section 163.1: Getting Started	654
Chapter 164: Python and Excel	655
Section 164.1: Read the excel data using xlrd module	655
Section 164.2: Format Excel files with xlswriter	655
Section 164.3: Put list data into a Excel's file	656
Section 164.4: OpenPyXL	657
Section 164.5: Create excel charts with xlswriter	657
Chapter 165: Turtle Graphics	660
Section 165.1: Ninja Twist (Turtle Graphics)	660
Chapter 166: Python Persistence	661
Section 166.1: Python Persistence	661
Section 166.2: Function utility for save and load	662
Chapter 167: Design Patterns	663
Section 167.1: Introduction to design patterns and Singleton Pattern	663
Section 167.2: Strategy Pattern	665
Section 167.3: Proxy	666
Chapter 168: hashlib	668

Section 168.1: MD5 hash of a string	668
Section 168.2: algorithm provided by OpenSSL	669
Chapter 169: Creating a Windows service using Python	670
Section 169.1: A Python script that can be run as a service	670
Section 169.2: Running a Flask web application as a service	671
Chapter 170: Mutable vs Immutable (and Hashable) in Python	672
Section 170.1: Mutable vs Immutable	672
Section 170.2: Mutable and Immutable as Arguments	674
Chapter 171: configparser	676
Section 171.1: Creating configuration file programmatically	676
Section 171.2: Basic usage	676
Chapter 172: Optical Character Recognition	677
Section 172.1: PyTesseract	677
Section 172.2: PyOCR	677
Chapter 173: Virtual environments	679
Section 173.1: Creating and using a virtual environment	679
Section 173.2: Specifying specific python version to use in script on Unix/Linux	681
Section 173.3: Creating a virtual environment for a different version of python	681
Section 173.4: Making virtual environments using Anaconda	681
Section 173.5: Managing multiple virtual environments with virtualenvwrapper	682
Section 173.6: Installing packages in a virtual environment	683
Section 173.7: Discovering which virtual environment you are using	684
Section 173.8: Checking if running inside a virtual environment	685
Section 173.9: Using virtualenv with fish shell	685
Chapter 174: Python Virtual Environment - virtualenv	687
Section 174.1: Installation	687
Section 174.2: Usage	687
Section 174.3: Install a package in your Virtualenv	687
Section 174.4: Other useful virtualenv commands	688
Chapter 175: Virtual environment with virtualenvwrapper	689
Section 175.1: Create virtual environment with virtualenvwrapper	689
Chapter 176: Create virtual environment with virtualenvwrapper in windows	691
Section 176.1: Virtual environment with virtualenvwrapper for windows	691
Chapter 177: sys	692
Section 177.1: Command line arguments	692
Section 177.2: Script name	692
Section 177.3: Standard error stream	692
Section 177.4: Ending the process prematurely and returning an exit code	692
Chapter 178: ChemPy - python package	693
Section 178.1: Parsing formulae	693
Section 178.2: Balancing stoichiometry of a chemical reaction	693
Section 178.3: Balancing reactions	693
Section 178.4: Chemical equilibria	694
Section 178.5: Ionic strength	694
Section 178.6: Chemical kinetics (system of ordinary differential equations)	694
Chapter 179: pygame	696
Section 179.1: Pygame's mixer module	696
Section 179.2: Installing pygame	697

Chapter 180: Pyglet	698
Section 180.1: Installation of Pyglet	698
Section 180.2: Hello World in Pyglet	698
Section 180.3: Playing Sound in Pyglet	698
Section 180.4: Using Pyglet for OpenGL	698
Section 180.5: Drawing Points Using Pyglet and OpenGL	698
Chapter 181: Audio	700
Section 181.1: Working with WAV files	700
Section 181.2: Convert any soundfile with python and ffmpeg	700
Section 181.3: Playing Windows' beeps	700
Section 181.4: Audio With Pyglet	701
Chapter 182: pyaudio	702
Section 182.1: Callback Mode Audio I/O	702
Section 182.2: Blocking Mode Audio I/O	703
Chapter 183: shelve	705
Section 183.1: Creating a new Shelf	705
Section 183.2: Sample code for shelve	706
Section 183.3: To summarize the interface (key is a string, data is an arbitrary object):	706
Section 183.4: Write-back	706
Chapter 184: IoT Programming with Python and Raspberry PI	708
Section 184.1: Example - Temperature sensor	708
Chapter 185: kivy - Cross-platform Python Framework for NUI Development	711
Section 185.1: First App	711
Chapter 186: Pandas Transform: Preform operations on groups and concatenate the results	713
Section 186.1: Simple transform	713
Section 186.2: Multiple results per group	714
Chapter 187: Similarities in syntax, Differences in meaning: Python vs. JavaScript	715
Section 187.1: `in` with lists	715
Chapter 188: Call Python from C#	716
Section 188.1: Python script to be called by C# application	716
Section 188.2: C# code calling Python script	716
Chapter 189: ctypes	718
Section 189.1: ctypes arrays	718
Section 189.2: Wrapping functions for ctypes	718
Section 189.3: Basic usage	719
Section 189.4: Common pitfalls	719
Section 189.5: Basic ctypes object	720
Section 189.6: Complex usage	721
Chapter 190: Writing extensions	722
Section 190.1: Hello World with C Extension	722
Section 190.2: C Extension Using c++ and Boost	722
Section 190.3: Passing an open file to C Extensions	724
Chapter 191: Python Lex-Yacc	725
Section 191.1: Getting Started with PLY	725
Section 191.2: The "Hello, World!" of PLY - A Simple Calculator	725
Section 191.3: Part 1: Tokenizing Input with Lex	727
Section 191.4: Part 2: Parsing Tokenized Input with Yacc	730

Chapter 192: Unit Testing	734
Section 192.1: Test Setup and Teardown within a unittest.TestCase	734
Section 192.2: Asserting on Exceptions	734
Section 192.3: Testing Exceptions	735
Section 192.4: Choosing Assertions Within Unittests	736
Section 192.5: Unit tests with pytest	737
Section 192.6: Mocking functions with unittest.mock.create_autospec	740
Chapter 193: py.test	742
Section 193.1: Setting up py.test	742
Section 193.2: Intro to Test Fixtures	742
Section 193.3: Failing Tests	745
Chapter 194: Profiling	747
Section 194.1: %%timeit and %timeit in IPython	747
Section 194.2: Using cProfile (Preferred Profiler)	747
Section 194.3: timeit() function	747
Section 194.4: timeit command line	748
Section 194.5: line_profiler in command line	748
Chapter 195: Python speed of program	749
Section 195.1: Deque operations	749
Section 195.2: Algorithmic Notations	749
Section 195.3: Notation	750
Section 195.4: List operations	751
Section 195.5: Set operations	751
Chapter 196: Performance optimization	753
Section 196.1: Code profiling	753
Chapter 197: Security and Cryptography	755
Section 197.1: Secure Password Hashing	755
Section 197.2: Calculating a Message Digest	755
Section 197.3: Available Hashing Algorithms	755
Section 197.4: File Hashing	756
Section 197.5: Generating RSA signatures using pycrypto	756
Section 197.6: Asymmetric RSA encryption using pycrypto	757
Section 197.7: Symmetric encryption using pycrypto	758
Chapter 198: Secure Shell Connection in Python	759
Section 198.1: ssh connection	759
Chapter 199: Python Anti-Patterns	760
Section 199.1: Overzealous except clause	760
Section 199.2: Looking before you leap with processor-intensive function	760
Chapter 200: Common Pitfalls	762
Section 200.1: List multiplication and common references	762
Section 200.2: Mutable default argument	765
Section 200.3: Changing the sequence you are iterating over	766
Section 200.4: Integer and String identity	769
Section 200.5: Dictionaries are unordered	770
Section 200.6: Variable leaking in list comprehensions and for loops	771
Section 200.7: Chaining of or operator	771
Section 200.8: sys.argv[0] is the name of the file being executed	772
Section 200.9: Accessing int literals' attributes	772
Section 200.10: Global Interpreter Lock (GIL) and blocking threads	773

Section 200.11: Multiple return	774
Section 200.12: Pythonic JSON keys	774
Chapter 201: Hidden Features	776
Section 201.1: Operator Overloading	776
Credits	777
You may also like	791

About

Please feel free to share this PDF with anyone for free,
latest version of this book can be downloaded from:

<https://goalkicker.com/PythonBook>

This *Python® Notes for Professionals* book is compiled from [Stack Overflow Documentation](#), the content is written by the beautiful people at Stack Overflow. Text content is released under Creative Commons BY-SA, see credits at the end of this book whom contributed to the various chapters. Images may be copyright of their respective owners unless otherwise specified

This is an unofficial free book created for educational purposes and is not affiliated with official Python® group(s) or company(s) nor Stack Overflow. All trademarks and registered trademarks are the property of their respective company owners

The information presented in this book is not guaranteed to be correct nor accurate, use at your own risk

Please send feedback and corrections to web@petercv.com