Scripting for Cybersecurity and Networks

Lab Answers

Python 3.7/3.8

Bruce McClelland

Petra Leimich

Sean McKeown

12/01/2021

Table of Contents

1.0 Lab_01 – Nums Keyspace

1.1 – 5. Using Thonny to Develop Python Code.

1. Which colour is used to highlight the results of the print statement?
A. White

2. What colour is the rest of the line after you type the opening quote?
A. Green

3. When does this colour disappear?
A. After you close the quotes.

4. What happens to the colouring when you close the bracket?
A. Brackets go white from black.

1.2 – 6. Using the IDLE IDE to Develop Python Code

1. What colour is used to highlight the results of the print statement?
A. Blue

2. If you type pri and press <TAB> what happens?
A. Function is autocompleted to print as it’s the only option.

3. If you use <ALT+P> again what do you get?
A. 2 statements past.

4. If you use <ALT+P> twice, and then <ALT+N> what does this do?
A. 2 statements past and then 1 statement forward.

1.3 – 8. Running Python from the Windows CMD Command Line

1. Which version of Python does py default to?
A. Python 3.?.? – depending on version installed.

2. What result do you get with the longer command python -V?
A. You also get the version number.

1.4 – 9. Python Numbers – Use Python as a Calculator

1. What is the difference between the / and // operators?
A. / is for floating point division, // only whole numbers rounded down.

2. What does the operator ** do?
A. Exponentiation.

3. What is the tax for this product, and the price including tax?
A. Tax – 12.50, Price – 112.50

4. What is the prefix used to indicate a hex nunmber?
A. 0x

5. What is the prefix used to indicate a binary number?
A. 0b

6. In what notation are the results returned?
A. base 10

1.5 – 10. Python Variables

1. Which of the following variable names are valid?
A. str_global = ‘hello’

2. What is the outcome and why?
A. Price hasn’t been defined.

1.6 – 11. Introducing Strings

1. When printed out, are str_var1 and str_var2 similar?
A. Yes, identical

1.7 – 12. Slicing Strings

1. What is the index of the first character in a string?
A. 0

2. What is the value of str_var1[3]?
A. i

3. What is the value of str_var1[-7]?
A. P

4. What index returns last char of a string?
A. -1

5. How could the last letter be returned without negative indexing, using len() as part of the slice?
A. str_var1[(len(str_var1)-1)]

6. What slice statement returns the first 3 chars from the string?
A. str_var1[0:3]

7. What does str_var1[:-2] extract from the string?
A. Don’t Panic

8/9. What is the value of str_var1[::-1]?
A. It starts at index 0, goes through the variable in -1 which is basically reverse.
!!!cinaP tnoD

1.8 – 13. Iteration of Strings.

1. How many chars per line are prints?
A. 1

1.9 – 14. Python Scripts (files).

1. What is the output from the script?
A. Hello World

2. Is the Python Interpreter still running now?
A. No

1.10 – 15. Python Functions and Scripts.

1. Which two things let Python know when the function definition ends and the function body starts?
A. Colon and tab

2. How does python know when the function ends and the rest of the script starts?
A. Unindentation

1.11 – 16. Python Basic Data Types – Numbers and Boolean

1. What is the % (modulo) operator for?
A. Finds the remainder after dividing 2 numbers.

2. What is the difference between / and //?
A. / divides and creates a float. // floor division creates an int with the number rounded down to the nearest whole number.

3. How can you use ** to find the square root of a number?
A. 49**(1/2)

1.12 – 17. The Math Module.

1. How many combinations for a 32 bit key?
A. 2^32 – 4.29e+9

2. How many combinations for a 128 bit key?
A. 2^128 – 3.40e+38

1.13 – 18. Case Study: Script to Calculate Possible Password Combinations

1. How many combinations would a brute force attack need to try for the 'test' 4 char password?
A. 81,450,625

2. How many combinations would a brute force attack need to try for a 1 char password?
A. 95

3. How many combinations would a brute force attack need to try for a 3 char password?
A. 857,375

4. How many combinations would a brute force attack need to try for a 8 char password?
A. 6.63e+15

5. What is the average search space for a 1 character password?
A. 47.5

6. What is the average search space for a 5 character password?
A. 3.86e+9

7. How many hours to find a 1 character password?
A. 2.37e-7h

8. How many hours to find a 3 character password?
A. 0.00214h

9. How many hours to find the password ‘napier123’?
1.575e+9

1.14 – 19. Pause to think.

1. How does the commenting differ from Hello World and keyspace calculator
A. Keyspace has inline comments, as well as block comments above the line, also has docstring. Hello World has no commenting except docstring.

2.0 Lab_02 – Strings and Lists

2.1 – 1. String Membership Operators

1. What is the difference between .find() and .index()?
A. .find() returns -1 if it is not present. Index does not.

2.2 – 2. Case of Strings

1. What built in string method can be used to create a new string str_capital from str_var2 with the first character of every word capitalised?
A. string.title()

2. How to check if a string is a digit.
A. string.isdigit

2.3 – 5. Functions that Return Values

1. How many return statements does it contain?
A. 2

2. Why doesn’t the function itself do any printing?
A. There is no print functions within check_passwd. It is used to return a value to be used by main().

2.4 – 6. Module Boiler-plate

1. You have already run the script in the previous exercise using run>run current script or f5. What happened in this case?
A. It requested user input for the password. It then prints that the password is not strong regardless of the password.

2. What is the output when you import the module with >>> import passwd_checker?
A. Nothing is returned.

3. Is the output from these commands as expected?
A. Yes, passwd_checker.check_strength(‘a813xy67’), nothing is returned, passwd_checker.check_strength(‘a’), false.

2.5 – 7. Python Collection Data Types – Lists

1. What is the value of the following?
A. passwords[0] – password
passwords[5] – 123456

2. What is the result of print(len(passwords))
A. 6

3. What is the result of print (len(passwords[1]))?
A. 6, due to using len to find the length of the variable at index 1 which is qwerty.

2.6 – 8. List Object Methods

1. List a couple of the available methods of the list object, and consider what they might do for us?
A.

.append()	- adds to the end of the list
.pop()	- removes an item at a specific item and returns it.
.sort()	- sort a list alphabetically by default

2. There is no similar method to add a character to a string. Explain why?
A. Python strings are immutable so when you concatenate the string, you are creating a totally new string.

3. Which item (offset index) did pop() remove from the list object?
A. remotes the last index when one isn't declared

4. Which item did the pop() methods remove from the list object now? (1)
It removed qwerty from passwords as that was at location index 1.

5. How would you use the insert() method to add 'password123' to the list after password?
A. passwords.insert(1, 'password123')

6. How would you use the insert() method to add 'qwerty123' to the list after qwerty?
A. password.insert(2, 'qwerty123')

7. What does the extend() method take as input?
A. It takes one parameter in square brackets. It uses it to add to the list.

8. How would you use the extend() method to add 2 more numeric passwords, 12345678 and 123456789?
A. passwords.extend([12345678,123456789])

2.7 – 9. Accessing Objects in a List Object – Indexing/Slicing

1. Which BIF would you use to check the index upper limit, to prevent an error?
A. print(passwords.index(passwords[-1]))

2. What does the following return?
A. The first 5 elements at index 0, 1, 2, 3, 4

3. Is the original list changed?
A. Slicing just outputs the results. No change is made to the list. Unless you make the output of the slice change the sliced variable name. passwords = passwords[0:5]

4. Is this the same as how the slice operates on string objects?
A. Slicing words the same for variable strings and created strings

2.8 – 10. Iteration over Lists

1. Does the loop iterate and print each item in the list?
A. Yes, it takes each of the items in the list and performs the functions for them. In this case it is print.

2.9 – 11. List Comprehension

1. What does numbers contain?
A. Creates a list [0,1,2,3,4,5,6,7,8,9]

2. What number does numbers end with? Why?
A. 9, Range starts at 0 and excludes the last number. 0-9

3. What does numbers2 contain?
A. If n/2 has no remainders, then divide the number by 2. As we're dividing, it creates floats.

4. What is len(numbers2)?
A. 5, as we only iterated over even numbers?

5. How does this code work?
A. If the number / 2 has no remainder then it gets divided by 2 and saved to the variable nunmbers 2 as a list element.

6. How could you achieve the same result using a for loop?
A.

```
for i in range(10):
 If I % 2 == 0:
   numbers2.append(i/2)
```

2.10 – 12. Nested Lists

1. How is the nested list printed?
A.
1234
12345
default
[‘default’, ‘default’, ‘default123’]
password
qwerty

2. What does the following return? Why? >>> print(passwords[3][2])
A. default123, this is because the element at index 3 is a list. You can access this list the same way with indexes. The second index of the nested list is default123.

2.11 – 14. More String Formatting and Flattening Lists with .join()

1. What are the results of the 2 joins below? Can you explain what is happening here?
A.

>>> ‘???’.join(str_var1)	Each character has a ??? after it. As it is not a list it seperates after every character.
>>> str_var1.join(‘???’)	Str_var1 is the seperator so after the ??? and before the third it will put the string don’t panic!

2.12 – 15. Python Basic Data Types – Scientific Notation for Numbers

1. How can you use python to find out what 6.30249409726094e+17 is when written as a standard integer, without scientific notation?
A. i = int(i), int(f’{i:.0f}’)

2. The chances of randomly guessing a 3-char ascii password at the first attempt are p=1/(95**3). When python displays the result of this, in scientific notation, what is the e-value? How many zeroes would appear after the decimal point when it is written out in full?
A. After the e is -06. 5 x 0’s after the decimal point. 0.0000016635077.

3. Use a print statement with f-string notation to display these chances as a float with 7 decimal places.
A. p = 1/(95^3)

p
1.16635e-06
f'{p:07}'
0.0000012

4. What is greyed out?
A. out = 'The chances of randomly guessing a 4 character password at the first attempt are {0:.8f} or {0}'.format(1/95**4)

2.13 – 16. Challenge Exercise – Nested List Comprehension

biglist = [I for k in [[j,] if not isinstance(j,list) else j for j in passwords] for i in k]

2.14 – 17. Case Study Part 2: Hiding the Input Password

1. What is the name of the module that contains utilities to get a password
A. getpass

2. What does import getpass, dir(getpass), help(getpass.getpass) do?
A. Imports the module getpass, provides modules for getpass and then help for the function getpass.

3/4 How are dir('getpass') and help(getpass'getpass()) different?
A. lists all the different attributes and modules for the string 'getpass' and doesn't provide helpful information but lets you test what it would look like when you do run it.

5. Try to run the modified script via thonny?
A. Thonny can't echo and will just hang when you run the script.

6. Thonny > run > run current script in terminal
A. It now works properly.

2.15 – 18. Case Study Part 3: Keep Asking Until You Get a Strong Password

While result != True: then it will loop

2.16 – 19. PEP8

1. Why did pycodestyle not show up the first time?

A. It didn’t appear as an option because the description doesn’t contain the keyword pep8.

2. Examples of PEP8 issues
A.
- Use inline comments sparingly
- Same line as the statements they are referring to
- separate in line comments by 2 or more spaces from the statement
- start inline comments with a # and a single space, like block comments
- don’t use them to explain the obvious
- 79 chars per line max
- surround operators with a single space either side.

3.0 Lab_03 – Dict Tuple Files

3.1 – 1. Python Data Types – Dictionary Object

1. What is the object returned for dic[‘numbers’]?
A. [1,2,3]

2/3. What is the response to ‘dead’ in dic?
A. False, this is because you reference the key within the dict. Not the value. You’d need to use dic.values()

4. results of the following
A.
>>> dic.get(‘parrot’) – ‘dead’
>>> dic.get(‘dead’) – nothing, no error.
>>> dic.get(‘dead’, ‘dead not found’) – dead not found, so error - ‘dead not found’

5. What does get do?
A. .get returns the value for the key if present. It not, it will not return anything unless you includea message to return after the key you’re looking for.

6. In what data structure are the keys returned?
A. dict_keys

7. Which method outputs only the values, and in what data structure?
A. dict_values

8. With dic.items(), what type of data structure is output for each key:value pair?
A. type(dic.items())
dict_items
each key:value pair is a tuple.

9. How many items in dic?
A. There is only 3 keys. The values are only present when referenced by the key.

3.2 – 2. Iteration of Dictionaries

1. Is the outcome the same?
A. The output is the same

2. Is there a performance difference?
A. For much larger dictionaries will be a significant increase as you’re only accessing it once. Instead of twice.

3.3 – 3. zip() – Creating a Dictionary from Two Lists

1. Which list is used for the keys in dic1? Which list for the values?
A. When you use the zip function, the first argument is the key (english) and the second's the value (french).

2. What does the dict() function do? What happens if you forget it?
A. A dictionary is a collection which is unordered, mutable and indexed. The dict() function turns something into a dictionary. It creates tuples based on the lists if dict isn't present within the function.

3. What is the benefit of using the .get() function here?
A. Checks to see if the key is in the dictionary if it's not then it returns nothing.

4. Can we use dic1 to translate from French to English?
A. As the french is the value then it will not be able to point to the key. To create French > English, we can reverse the dictionary so the keys are values and vice versa.

5. Complete the highlighted 'for' loop in the code above. You need to replace the two sets of … with the appropriate code.
A.

```
dict3 = {}
for key, value in dict1.items():
    dict3[values]=key
```

3.4 – 4. Dictionary Comprehensions

1. Explain in your own words how this dictionary comprehension works.
A. It creates the dictionary dic4 with the values en, fr which is basically I, j but in reverse which is evident from the fr: en at the start. For key en and value french, we will turn that into key french and value en for the entire new dictionary.

3.5 – 5. External Data – User Input

1. What is the data type of the value returned by the input() function?
A. string.

2. Does the code work?
A. num1 = int(input())

```
num2 = int(input())
print(f'{num1 + num2}')
```

3. What happens if you input a float?
A. floats return an error.

3.6 – 6. Range() For and While

1. Why do we set reply = True before the start of the loop?
A. You can't create a variable after the variable you're about to create is called as it hasn't been created yet.

2. Why do we write while reply: rather than while reply == True:?
A. It will only be true when the number is 1. 1 = True, 0 = False. While = True – all nums > 0 are considered True.

3.7 – 7. External Data – Reading from Files

1. Which type of object is passwords?
A. '_io.TextIOWrapper'

2. What does r mean?
A. open in read mode

3. Which methods might be useful to read data from the file object?
A. .read(), .readline(), .readlines(), .seek()

4. Which method reads a single line from the file and returns a string?
A. passwords.readline()

3.8 – 9. Iterations of Lines of File

1. What is the result?
A. prints each line with a newline break

2. Why are the first two lines missing from the output?
A. /n after each line invisible and a blank line from the start of output. Not 100% sure on this one.

3/4. Did it print out all the lines in the file?
A. After a print statement there is a newline feed \n. To fix that we do the following:

```
for I in passwords:
    print(I, end='')
```

3.9 – 10. Tuple Data Type

1. What type of brackets do we use for tuples? For lists? For dictionaries?
A. (), [], {}

2/3. What happens if you run the code above?
A. Runtime error as the last line contains 3 values not 2 which is what the code is looking for.

4. Does the maxsplit argument help the code run without error?
A. It works but it is not ideal. Error handling will be best.

5. Did this work?
A. Value error expected 2 to unpack

6. What is the problem?
A. The tuple were trying to assign results to has a length of 2. It can only handle one variable on top of the password.

3.10 – 11. Creating a Dictionary from Tuples – setdefault() method

1. Describe the purpose of the .setdefault() method in your own words.
A. Checks a dictionary for the key ka. If it is present it returns the list so we can append v to it which allows multiple values to be allocated to each key. If the key is not present it inserts the key to the dict with [] which then gets appended with v to create the key:value pair.

2. Why are the dictionary values in this situation lists?
A. The values are in dictionary lists as .setdefaul returns the corresponding values in []. The value was set in the .setdefault() arguments. Means we can add multiple values to one key.

3.11 – 12. Hash Signatures – Python Cyber Security Applications

1. What is the hash signature of your own name?
A.
```
import hashlib
md5hash = hashlib.md5(‘Bruce’.encode(‘utf-8’))
print(md5hash.hexdigest())
4be90ed9c41356ec34247e49aec714a8       - 16 bytes, 32 char length, 128 bits
```

2. Is this different if you use lower case only?
A. yes a change in a single digit can totally change the outcome of the hash.

3. How is the hash signature being displayed? (What encoding)?

A. utf-8

4. What is the length of the md5 hash signature?
A. 16 bytes, 32 char length 128 bits.

5. What is the length of the hash signature generated?
A. 32 chars again

6. Compare this with the length of the hash of your name. What do you find?
A. Exact same

7. What would be the length of the md5 hash signature generated for this entire pdf lab sheet?
A. 32 chars

8. What dictates the length of the hash signature?
A. 128 bits, 4 bits per char = 32 chars

9. SHA1 of name
A. sha1obj = hashlib.sha1(‘Bruce’.encode(‘utf-8’))
print(sha1obj.hexdigest())
40 characters, hex, 160 bit, 4 bit per char, 20 bytes

3.12 – 13. Hash Password ‘Recovery’

1. What is the password for the 2 test cases in main?
A. ending 7a93 is 123123, ending 7a92 does not exist. This shows that the slightest change in a hash totally changes the values and it is unrecoverable. Putting it through an engine online shows that it is basically 123123, but not quite.

2. 5badcaf789d3d1d09794d8f021f40f0e
A. Starwars

3. 0d107d09f5bbe40cade3de5c71e9e9b7
A. letmein

4. 5c916794deca0f7c3eeaee426b88f8bd
A. can’t recover

5. 5c916794deca0f7c3eeaee426b88f8bd
A. MONTYPYTHON – code to add uppercase implementation

6. a67778b3dcc82bfaace0f8bc0061f20e
A. Cheese – Code to add title

4.0 Lab_04 – Exceptions OS Sys Socket

4.1 – 1. External Data – User Input Exceptions

1/2. What happens when you run add_2_ints.py?
A. ValueError as knight is not int.

3. Run the modified script with Hello and World as inputs. What happens? Why?
A. hello world, concatenates them together as strings

4. Use 6 and 7 as inputs. What happens? Why?
A. 67, this is due to input taking strings as their value and the conversion has been stripped.

5. Int vs Eval
A. Int() converts to integer and is a lot safer
eval() used to verify an expression. If it starts with 0 there is an error.

4.2 – 3. OS Module – Working With Files (CTD)

1. Which function in the OS module can change directory in the underlying file system?
A. chdir

2. Which type of object is returned by os.listdir()
A. list

4.3 – 4. Platform Independent Constants

1. What is the Windows line separating characters?
A. /r/n in Windows carriage return, line feed on printers

4.4 – 5. More Python Exception Handling: Try and Except

1. In the listing above, circle the code which may cause the exception, and also any dependent code.
A. The exception is caused by the last line of the common variants file having 3 elements

2. Did this stop the crash and Traceback error?
A. Yes.

4.5 – 6. File Handling Exceptions

1. What happens?
A. Error messaging

2. What time of exception is raised?
A. File not found error

3. Does your code run and only the intended error message is shown to the user?
A. Yes

4.6 – 7. Using Specific Exception Types

1. Does your code run and show the correct error message to the user?
A. Yes

4.7 – 8. Reading common_variants.txt Into a List

1. What does the passwds[] list contain?
A. Contains a nested list with each line being a list within the list which encompasses all lines

2. What does the ‘\n’ in the last element from each line mean?
A. new line break

4.8 – 9. Chomping Strings

1. chompable_str = ‘Pesky newline char str\n’ what do you get after printing?
A. After printing, there is a new line break.

2. Files.py create a line of code to chomp each_line string before appending it to the list.
A.
```
import os
with open('com.txt', 'r') as passwords:
    passwds = []
    for line in passwords:
        passwds.append(line.strip().split(',', line.count(',')))
    for i in passwds:
        print(i)
```

3, Where is the strip() method called in the example code above?
A. During the try statement, stripping each line and then splitting by comma.

4. Why is this preferable to adding the separate line:
each_line = each_line.strip()
before the passwds.append() line.
A. More efficient, will take less time.

4.9 – 10. Python Script Arguments

1. What python object type is sys.argv?
A. List

2. What is the content of sys.argv?
A. [‘’] or arguments given for a function

3. Do you understand how the boilerplate code is working for each type of test?
A. Checks if the script is running or being imported and performs functions depending. If running, runs main() else it imports the functions to use at will.

4. How many objects are in the sys.argv list now?
A. 3, args.py, hello, world

4.10 – 11. Adding Functionality to the Script

1. Did the script correctly print out the usage string, and exit?
A. Yes, displays error and exist from the script.

2. Did the script correctly print out only the ‘real’ arguments and not the name of the script?
A. print_args(sys.argv[1:])

4.11 – 12. Testing Scripts with Args from IDLE

1. If sys.argv is a list object how might we add our own test arguments from within the script?
A. .append to add to the back of the list.

2. How might we add multiple test arguments? What code might we use?
A. sys.argv.extend([‘arg’, ‘arg’])

4.12 – 13. Python System Scripting: the sys module

1. What type of object is sys.modules?
A. dict

4.13 – 14. File System Interactions – The os.path module

1. Does the output now show absolute paths?
A. Yes, os.path.join(os.path.abspath(r<path from root>, fname)
print(os.path.abspath(path))

2. Why might the absolute path be needed?
A. Much easier to navigate folders and great when using subprocesses. A lot of different os modules require full path to check existance .

3. Does the dir exist?
A. False >>> os.path.exists(r'c:\nonexistent')

4. Why does the lower case c: not throw an error?
A. Windows, for the most part, is not case sensitive. Compared to linux systems.

5. why will >>> os.path.exists(os.curdir) always return true?
A. Your current dir will always be your current dir.

6. Is os.curdir a file or directory?
A. current dir is a directory.

7. How many files are in your current directory?
A. print(f'{len(path)} files and directories')

8. If path is a file, prefix with -, dir prefix with d.
A.
```
if os.path.isfile(filename) == True:
   print(f' – {os.path.abspath(path)}')
elif os.path.isdir(filename) == True:
   print(f' d {os.path.abspath(path)}')
```

4.14 – 15. File Sizes and MAC times

1. What sizes are the two files?
A.
a-print(os.getsize(r'c:\users\Bruce\appdata\local\programs\python\python37\LICENSE.txt'))
b-print(os.getsize(r'c:\users\Bruce\appdata\local\programs\python\python37\python.exe'))

a-30195 bytes
b-99856 bytes

2. What unit does getsize use?

A. Getsize uses bytes. os.stat(filename).st_size is the return from getsize.

3. What is the creation timestamp for the file?
A. print(os.path.getctime(path)) 1553549036.0

4. In what unit the timestamp returned?
A. epoch

5. timestamp = os.path.getctime(path)
import datetime
print(datetime.datetime.fromtimestamp(timestamp))
2019-05-02 19:37:49.256894

4.15 – 16. Joining and Splitting Paths

1. What does the dirname() method return?
A. returns the path of the original path excluding the file.

2. What does the basename() method return?
A. Returns file or dir within path

3. What does the splitext() method return?
A. Returns, root + ext to create path. path = os.curdir\\ls', '.py' = ext

4. What does path1.split(os.sep) return?
A. ['os.curdir', 'ls.py']

4.16 – 17. External Data – Writing Data to Files

1. has the file been created in c:\temp? >>> outfile = open(r''c:\temp\outfile.txt', 'w')
A. The file 'outfile' was created in the temp dir

2. Which methods might be useful to write data to the file?
A. outfile.write('test line\n')

3. Check the contents of the file. Have the lines been written?
A. Only writes on closure of the file.

4. Check the contents of the file. have the lines been written now?
A. Yes

5. Check the contents of the file again after you create the 2 extra lines.

A. 4 lines

6. Check the contents of the file after creation of an extra 2 lines.
A. 6 lines in total.

4.17 – 18. Writing Directory Listing to a File

1. Does your ls.py script now write the file details listing to the log file in c:\temp?
A. Yes, it should work perfectly.

2. Why do we have a finally: block in the above example solution?
A. Will make sure that if the file is available within local it will be closed. This will make sure it does not remain open and will commit to any writes.

3. If your code doesn't already use "with open" re-write it so that it does. Why is it a good idea to use "with open"?
A. withopen is used to automatically close the files when you unindent.

4.18 – 20. Extending the password cracking script C and D

1. How many common passwords need to be hashed to crack the password ‘123’?
A. 1 as it’s the first in the list.

2. How many common passwords need to be hashed to crack the password ‘arsenal?
A. 14, depending on the hash functionality, if implementing capital, upper, etc

3. If the password you are trying to crack is not in the list of common passwords at all, how many hashes do you need to calculate until you know that it cannot be cracked?
A. Again, 19 words, 7 numbers, depending if lower / upper / capital. 3 x 19 + 7 potentially.

4. What would happen if you had a much longer list of common passwords, that contains 10,000 words?
A. Would have to hash them all if the password is not there. Potentially multiple times each if using upper / lower case hashing methods

5. Change your script to try and crack two passwords in a row, i.e. with two test cases and two function calls. What happens to the hashes you calculated during the first function call?
A. They get overwritten. Currently not the most efficient as it will mean you will need to go over the entire hashes again, which is perfectly fine given a small list. But, when dealing with a massive library of potential passwords. Could cause quite a burden.

4.19 – 22. Simple Server using Sockets – Receiving Data

1. What does localhost represent? Why do we use that for the initial test?
A. Loopback address used to test the network of the host machine 127.0.0.1

2. What IP address does the localhost resolve to?
A. 127.0.0.1

3. When creating the socket object, what do the following terms mean?
A.
socket.AF_INET – used for ipv4 addressing. IPv4 addresses are made up of 4 octal values separated by dots.
socket.SOCK_STREAM – means it is a tcp stream.

4. What is the ip address and port that the server is listening on?
A. 127.0.0.1 – 5005 port tcp

5. What is the state and process ID of your server?
A. netstat -ano | findstr 5005
tcp 127.0.0.1:5005 0.0.0.0 LISTENING 12472
LISTENING – 12472

6. What is the state and process id of your server?
A. LISTENING 12472

7. What is the state and process ID of the telnet client? What port is it running on?
A. ESTAbLISHED 12960 port 57945

8. Explain in your own words, how the script works.
A. You start by establishing a socket. You then bind and listen to the server. Then from the client connect via telnet. It receives packets from the client at 1024 bytes. It decodes what character you send and prints it while at the same time it prints a confirmation email to the client. The try, except clause around it stops the error message closing the server.

4.20 – 23. Client & Server using Sockets

1. In which order do we need to run the two scripts? Why?
A. You need to load the server first and then the client as the client is trying to connect to the server. If you don't you get a connection refused error.

2. Why do we need 2 separate IDLE shells?
A. One for the server script and one for the client script.

3. Why do we use the servers name or address in both scripts? Why don't we need to know the clients address?
A. The server listens to what connects to it. From there it has all of the required information needed.

4. What happens to the message received by the server?
A. Gets just 10 characters due to the buffer size being at 10.

5. Why do we use a while loop and not a for loop? How exactly does this while loop work?
A. While there is data being sent to the server the loop is run.

6. What ip address is your server running on?
A. 192.168.56.1 : 5005

7. What is your IP?
A. IP config – 192.168.0.13

8. Accept ip and port as cli arguments
A.

```
import sys
if len(sys.argv) == 3:
    sock.connect((sys.argv[]), int(sys.argv[2]))
```

5.0 Lab_05 – Algorithm Complexity

5.1 – 1. Functions (In the mathematical sense) (Partially complete)

1. Which of the graphs are functions?
A. You can tell if there is a function, by using a ruler, if the rule whilst holding it vertically makes contact with multiple points of the functions, then it is not a function.
a, b, e are functions.a – o(x2), b – o(xy), d – o(x3), e – o(y), f – o(x)

5.2 – 2. Python Functions and Methods

1. Fill in the table
A. domain is the allowed input type, co-domain is the possible output type
d

Function / method	domain	co-domain	Result when a = ‘abc’	Result when a = 4
a+a	numbers,strings, lists,tuples	numbers, strings, lists, tuples	‘abcabc’	8
len(a)	strings, list, dict, tuple, int	int	3	type error
bool(a)	string, list, dict, tuple, int	bool	true	true
max(a)	string, list, dict, tuple	int, str	‘c’	type error
a.capitalize()	string	str	‘Abc’	attribute error
a.endswith(‘c’)	string	str	true	attribute error
a.find(‘b’)	string	str	1	attribute error
a.isalpha()	string	bool	true	attribute error
a.isdigit()	string	bool	false	attribute error

5.3 – 3. Complexity of Algorithms (Not 100% sure, sorry, Partially Complete)

1. What is the “problem size”, i.e. the factor that affects the number of questions asked?
A.1 = O(l), 2 = O(n), 3 = O(n2)

2. Determine the worst case and best case big-theta time complexity of each of the three algorithms in terms of the number of questions asked.
A.

3. If we change the problem to be “find out how many others in the same room have the same birthday as you”, how does this affect the complexity of the algorithms?
A.

5.4 – 4. Exponential Growth

1. When will the whole surface be covered if growth is exponential, doubling every day?
A. day 0 = 10m2, day 1 = 10m2 x 2 = 20m2, day 2 = 20m2 x 2 = 40m2 after 2 days.

2. How long 50% increase + 10% increase a day
A.

```
upper_limit = 40
x = 10
r = float (input(‘Enter the growth as a dec’)     # either 10 or 50
t = 0

while   x < upper_limit:
        x = x + (x*r)
        t + = 1

print(f’{t} no. of days’)
```

50% = 4 days
10% = 15 days

5.5 – 5. Algorithmic Complexity of the Password Cracker

1. List containing n passwords
A. O(n/2) Average

2. First in list
A. O(I) Best

3. Not in list
A. O(n) worst

5.6 – 6. Extending the Password Cracker

1. How many passwords does rockyou.txt contain (that is our n)?
A. n = 14575229 - num of passwords

2. How many hashes need to be calculated on average to crack a password that is in the list?
A. 7287815 on average - O(n/2)

3. If the password you are trying to crack happens to be the first in the list, how many hashes need to be calculated to crack it?
A. 1. O(i)

4. If the password you are trying to crack is not in the list, how many hashes do you need to calculate until you know that it cannot be cracked?
A. O(n) or 14575229

5.7 – 7. Extending the Password Cracker: Using Rockyou.txt

1. Are the answers for the hashes from the first password cracker the same?
A. 7a93 – 123123 and 7a92 is still unrecoverable

2. Do you notice that it takes longer to crack the passwords with rockyou.txt than when using common3.txt?
A. Yes, this is due to having to hash and compare over 13 million hashes one at a time.

3. passwords
A. a - !!!abby!!!, rockyou, Petra999, easypeasy

4. Is there a correlation between the time taken and the position in the file?
A. Further line takes longer to find..

5. Describe the structure of the script.
A. Runs main, checks if the hash is an arg, if it is it uses that else uses the one written. Checks to make sure that the hash is 32 chars, 0-9 a-f A-f, if it passes it then checks the encoding type and hashes cap, upper, normal. Compare the hash with the inputted hash and then returns the pass.

5.8 – 8. Extending the Password Cracker: Using a Dictionary to Store the Hashes

1. Explain in your own words why a Python dictionary improves the efficiency of the password cracker.
A. After your first iteration, and you create the dict, it takes significantly less time as you're searching for the key and then you get the value instead of hashing the list every time.

5.9 – 9. Testing Search Algorithms

1. What do the scripts need in order to run without error?
A. Needs a file called words.txt, 2 arguments present when running the function or 3 if binary.

2. Do both scripts give the same results?
A. Yes, both of the scripts return “accommodation” and “zebede”

3. Do both scripts still give the same result?
A. No, both return zebede

4. Commend out the line that sorts the wordlist in the binary search. Rerun the script. Do the results change? What exactly is different?
A. As binary search traverses a sorted list, it won’t find the words as they are not sorted. Returns all 4 words.

6. Apart from the size of the text files, how is their structure different?
A. 100k has upper, lower, plurals, names, accents
20k has lower, unsorted, random

7. What might be the reason for 20k.txt being the structure it is?
A. Most commonly used English words

8. Why does this cause a problem for binary search?
A. Random, unsorted, so can’t search efficiently. Wont know if the word is in the first or the second half.

6.0 Lab_06 – Regex PCAP

6.1 – 2. Matching Text Characters

1. Has a match object been created?
A. Yes

2. What information does the “span” in the match object hold?
A. span = (35, 41), match = ‘parrot’)

3. What type of object is match.span()?
A. tuple

4. What does the match objects .group() method output?
A. prints the found element from the string if there is a match

5. What is the print statement used?
A.
```
if match:
   print(f”{match.group()}” between offsets {match.span()[0]} and {match.span()[1]}’
```

6. Has a match object been created?
A. No object has been created

6.2 – 3. RegEx: Matching Special Pattern Characters

1. What has matched against this pattern?
A. No object was created as the regex was wrong.

2. What is the correct patter?
A. match = re.search(r’p\w\w\w\wt’, str1) There was a w missing so it have matched p a r r ??? t

3. What has matched against the 2nd pattern?
A. 23

4. Why was the 1 not matched against?
A. it’s looking for 2 digits that are consecuritve.

5. What matched the 3rd pattern?
A. s3@rch3d

6. What matched the 4th pattern?
A. 445 3665

7, What matched the 5th pattern? \(\d\)\d+[\s]*\d+[-]\d+
A. '(0)131 445-3665

6.3 – 4. Repeating Pattern Characters

1. What has been matched against the pattern?
A. Matches e's until there is a character other than e.

2. What has been matched against the 2nd pattern?
A. Looks for 11 digits in a string.

3. What has been matched against the 3rd pattern?
A. Looks for undefined no. of digits.

4. What has been matched against the 4th pattern?
A. a number, any number of spaces (including 0) and then a number.

5. What was the working pattern?
A.
str = '(0)131 445-3665
match = re.search(r'\(\d\)\d+[\s]\d+[-]\d+\d", str1)

6.4 – 5. More special Characters

1. Did the pattern successfully match the IP Address?
A. Yes

2. What is the working pattern for the email address?
A. match = re.search(r'[\w._%-+]+@[\w._]+\.[a-zA-Z]{2,4}', str2)
\w word, ._%-+ any character, + any number, [a-zA-Z] character range, {2,4} 2 – 4 length.

3. Did the pattern match the host?
A. matched 146.166.55.2

4. Did the pattern successfully match the host at the start of the string?
A. Yes

5. Write a regext to match the ip address at the end of the string only. What is the regex used?
A. match = re.compile(r'[\d]+\.[\d]+\.[\d]+\.[\d]+')
match.findall(r'host: 146.166.55.2 net scanning 146.156.12.2') [-1]
'146.156.12.2'

6.5 – 6. Sets and ranges of characters with [] (Square Brackets)

1. What has been matched against the pattern?
A. 131 445

2. Did the pattern successfully match all the strings?
A. No

3. What was the working pattern?
A. match = re.search(r'[\d\(\)]+[-]*\d+[-]*\d+', str??) add the correct string here

4. What was the working pattern for the email?
A. re.compile(r'[\w.-]+@[\w]+\.[\w.]{3,6}')

6.6 – 7. Ranges of Characters using []

1. What has been matched against the pattern?
A. re.search('[a-z]*:[a-zA-Z0-9]+', str24)

2. Did the pattern successfully match all the strings?
A. re.search(r'\d+\.\d+\.\d+\.\d+)

3. What has been matched?
A. both ips were matched, ending with both 2 and 99.
re.search(r'146\.176\.123\.[0-9][0-9]?', str34)

4. Match the napier web servers.
A. re.search(r'146\.176\.123\.[0-9][0-9]?', str35) / str36

5. Can the pattern match all the servers in the range 0-99?
A. Yes

6. How would you change the pattern to match the full range 0-255?
A. match = re.search (r'[012][0-9]?[0-9]?\.[012][0-9]?[0-9]?\.[012][0-9]?[0-9]?\.[012][0-9]?[0-9]?',str4)

6.7 – 8. Options – Using the or | symbol

1. What has been matched against the pattern?
A. http

2. How would we fix the problem?

A. re.search(r‘(http[s]?|ftp)’, str44)

3. What does this do?
A. Looks for http, if it exists it looks for https. else, it looks for ftp.

6.8 – 9. Groups – With () Round Brackets

1. What has been matched against the entire pattern?
A. johncleese@montypython.com

2. What does group(1) match?
A. johncleese

3. What does group(2) match?
A. montypython.com

6.9 – 10 Finding All Matches in a String

1. What has been matched against the pattern?
A. johncleese@montypython.com

2. What kind of object is this?
A. str

3. What has been matched against the matches pattern?
A. [johncleese@montypython.com, rich@napier.com]

4. What kind of object is this?
A. list

5. Why don’t we need the if statement when using findlal?
A. re.findall returns a list which can be iterated over, or used as input to another task.

6.10 – 11 Findall() and Groups

1. What has been matched against the pattern?
A. all emails

2. What type of objects have been returned in the list?
A. tuple

6.11 – 12. Reading Content from Web Pages

1. Does the code successfully fetch the contents of the Napier web page?
A. Yes
webpage = urllib.request.urlopen(url)
pagecontents=webpage.read()

2. What specific type of string are the page contents returned as?
A. bytes

6.12 – 17. Familiarisation with Pcap Records? Using DPKT

1. What type of object is eth?
A.bytes

2. Can you identify the src and dst IP addresses?
A. import socket
socket.inet_ntoa(b'\x92\xb0\x4[']
src – 146.176.164.91
socket.inet_ntoa(b'\x17\x156\x83')
dst – 23.21.54.131

3. Can you identify the src and dst TCP ports?
A. sport – 53954, dport - 80

4. What sort of object is the TCP.data field?
A. GET request

6.13 – 18. Parsing PCAP Records Using DPKT

1. What data types are the ip addresses?
A. Bytes

6.14 – 20. Extract HTTP Requests From the PCAP File

1. What are the other two attributes in a http record?
A. version, method, uri, headers

2. How would you extract the 'method' values from the http record?
A. print(f'{repr(http.method)}')
'GET'

3. How would you extract the ‘referer’ values from http.headers?
A. print(f’{repr(http.headers[“referrer”])}’) – as referrer dict key

6.15 – 21. Challenge Question: HTTP Requests

1. The previous exercise extracts http requests, which use port 80. However, https requests are becoming much more common. They typically use port 443.
A.

```
if tcp.dport == 80 or tcp.dport == 443:
   http = dpkt.http.Request(tcp.data)
   print({tcp.dport} {repr(http.method)}{repr(http.headers[“referrer”])})
```

6.16 – 22 List Downloaded GIFs

1. In wireshark we used the filter. What are the equivalents of these two conditions used in Python?
A. The code gets an ethernet packet and ip packet of a packet capture from wireshark. IP translations of src and dest are saved in applicable variables.
http = dpkt.http.Request(tcp.data)
The get request is used to get the data attached to the site. Makes sure it is a get request, checks that the url contains ‘.gif’
Then displays the dsrc, uri and dst of each.

2. How to find .jpg?
A. change line 29 to if ‘.jpg’ in ur.
http.method == get and ‘.jpg’ in uri.lower()

6.17 – 25. Regex Challenge
1. Part 1 Question
A.
match = re.search(r’[A-Z]{1,2}[0-9]{1,2}\s[0-9][A-Z]{2}’, pk1) – pk2 and pk3 one at a time

2. Part 2 Question
A.
match = re.search(r’[A-Z]{1,2}[0-9A-Z]{1,2}\s[0-9][A-Z]{2}’, pk4)

7.0 Lab_07 – Pyplot, Probability & Statistics

7.1 – 1. Pip Install: Installing Non-Standand Packages

1. What is the reason for the error?
A. matplotlib hasn’t been installed.
py -m pip install matplotlib - pip3 install matplotlib

7.2 – 2. Plotting with Pyplot

1. Make graph
A.

```
import matplotlib.pyplot as plt
import numpy as np
t = np.arange(0.,10.,0.5)
t_sqrt = np.sqrt(t)
plt.plot(t, t_sqrt, ‘ms-‘, MarkerFaceColor = ‘yellow’, label=’sqrt(x)’)
plt.legend()
plt.grid()
plt.title(‘sqrt(x)’)
plt.show()
```

7.3 – 3. Students and Birthdays (Might need reworked?)

1. How many different possibilities are there for choosing two class reps?
A.
n = 42, r = 2
n! / r!(n-r)! 42! / 2!(42-2)! 42 x 41 / 2 1722 / 2 = 861 possible combinations

2. What are the chances that none of the students share a birthday with the teacher?
A.
n = 42, p = 1/365, r = 0
n! / r!(n-r)! = 1 P^(42 – 1/365) 0.997 ^ 41.997 1 * 0.88146 = 0.88146%

3. What are the chances that exactly one shares the teachers birthday?
A.
n = 42, p = 1/365, r = 1

n! / r!(n-r)! x p^r x (1-p)^(n-p)
42! / 1(42 – 1)! x 1/365^1 x (1 – 1/365)^(42 – 1/365)
42 x 0.00274 x 0.997

0.11507 x 0.88146
0.10143 chance

4. What is the probability that no pair share a birthday?
A.
p(match) = 1 – 365! / 365^n (365 – n)!
1 – 365! / 365^42(365 – 42)!
1 – 0.91403
noone – 0.08597%

7.4 – 4. Playing Cards

1. How many possibilities are there for getting the same card from each pack?
A. n – 104, 4 – 2
104! / 2! x (104-2)! 1/52

2. What is the probability of getting two clubs?
A. 13 / 52 x 13 / 52 1 / 4 x 1 / 4 1 / 16

3. What is the probability of getting two Queens?
A. 4 / 52 x 4 / 52

4. What is the probability of getting two Aces of spaces?
A. 1 / 52 x 1 / 52

7.5 – 5. 6-Letter Passwords

1. If you are allowed to use each letter no more than once?
A. 26! / (26-6)! 165765600

2. If each letter can be used multiple times?
A. 26^6 308915776

7.6 – 6. Desert Island

1. The first priority is to form a government of prime minister and chancellor. How many ways are there to do this?
A. 5! / 2! (5-2)! – order matters as pm + chancellor - 20

2. If they decide instead to form an executive committee with two “equal” members, how many ways are there to do this?

A. 5! / 2! (5-2)! = 20 / 2 = 10 as order doesn't matter.

7.7 – 7. Communication

1. You have an unreliable communication stream. The probability that one bit will get flipped is p. N bits are transmitted, what is the probability that 0 bits get flipped?
A.
n!/r!(n-r)! p^r 1-p(n-r)
=
1 – p ^ (n)

7.8 – 9. Script for Calculating Possible Combinations/Permutations, and Listing Them

1. In a football penalty shootout, 5 different members of the team of 11 players are selected to participate. Calculate how many possible ways are there of doing this?

A. C(8,2)	- ordering doesn't matter	= 8! / 2!(8-2)!
P(6,4)	– ordering matters	= 6! / (6-4)!

2. Assume the team of 5 has been selected, and consists of Jim, Alan, Gary, Sam and Owen. They now need to decide the order – who goes first etc.
   - Calculate how many different permutations there are
   - List all permutationts

A. 462 different combinations. C(11,5)
12- permutations

7.9 – 11. How many Spares

cumulative.py, 1000 machines, 1/1000, 1:1000000

7.10 – 12. The Secrets Module

1. Briefly summarise the rationale for introducing this module.
A. Secrets function is used to produce random numbers which are cryptographically secure. Used to generate random numbers, passwords, session keys.

7.11 – 13. Random Numbers in Python

1. How many executions do you get an answer that you had before?
A. 1 – 6, for me, 3

2. How many tries until you get the answer 1?
A. 1 – 6, for me, 2

3. What are the possible outcomes of this statement? Explain how this list comprehension works. Why is it using i+1 not i?
A. It is i+1 otherwise it will be range 0-5 as 6 won't be included.

7.12 – 16. Setting up and Implementing Hashtable Search

1. What is the purpose of target = target[:1000]
A. From target, it takes the first 1000 elements.

7.13 – 17. Comparing the Algorithms – Timing Your Code (Here Onwards Needs Finished)

1. Which is the fastest algorithm? Which is the slowest?
A. Hash fastest, Linear slowest.

Lab_08 – Graphs & Networks

8.1 – 1. Isomorphic Graphs

1. Which of the following graphs are isomorphic?
A. 1, 3 2, 6 4, 5

2. Which properties of the nodes can be used to determine that two graphs are not isomorphic?
A. Node connectivity, number of edges

8.2 – 2. Hamiltonian and Eulerian Paths

1. Eulerian Path / Cycle?
A. Path – 1, 3, 2, 6, 4, 5
Cycle – 1,3

2. Hamiltonian Path / Cycle?
A. Path – 1,3,2,6,4,5
Cycle – 1,3,4,5

3. Chromatic Numbers
A. 1/3 – 6 chromatic, 2/6 – 3 chromatic, 4/5 - 3

Lab_09 – Geolocations with Python

9.1 – 1. Check / Install geoip2 and a Geolocation Database

1. Name of your database file with path:
A. C:\Users\...\GeoLite2-City_20190129\GeoLite2-City_20190129.mmdb

9.2 – 2. A first look at geoip2

1. What data type is the reader object?
A. geoip2.database.Reader

2. What data type is the rec? What does it seem to contain?
A. geoip2.models.City

3. What is the name of the key that contains the country code in 2 letter ISO format, like 'GB'?
A. rec.country.iso_code

4. What are the names of the key and subkeys that contain the longitude and latitue?
A. rec.location.longitude, rec.location.latitude

5. What is the location in (longitude, latitude) format? How many decimal places?
A. N 55.95, W -3.2

6. Which key contains the postcode? Does it contain the full postcode?
A.rec.postal.code

7. Which keys contain the city name and the country name? Is there more than one name for each? Why? How do you get the city and country names in French?
A. rec.country.name['Fr']

8. Why should the reader object only be created once?
A. Creating it is very time consuming and you should only make one.

9. What are the city and postcode of this IP address?
A. Japan, AS, no postcode / city available

10. Which country is this IP address in? What is the location in? (long / lat)
A. Japan, 35.69 lat 139.69 long

11. Do you notice a difference in the accuracy of the geolocation info?
A. Outside of the UK it is not nearly as reliable.

12. Is ths information for this address complete?
A.

13. What is the result of rec3 = reader.city('192.168.255.255')
A. Not within the database

14. Why not?
A. Private IP address range

9.3 – 5. Create Your First KML File with Python

1. Look at the directory where your script is. Does it contain a new file called napier.kml?
A. Yes

2. How could you have changed the script to place the kml file in a different directory?
A. kml.save(r'<path>name')

Lab_10 – Timing and Tuning your Code

Lab_11 – APIs and Flask