

IT1007 Introduction to Programming with Python and C

Lab Exercise 04

Submission instructions [Coursemology]:

1. There are two parts in this lab. For Part A, you have to submit within the same day of your lab session. For your Part B, you have **six days** to work on it. (E.g. if your TLab is on Monday, then your deadline is the coming Sunday midnight.)
2. Complete your code using the skeleton files provided, then **test your code on your computer first** before submitting to Coursemology.
3. To submit your code on Coursemology, click on “Labs” in the sidebar followed by the appropriate “Attempt” button.
4. **Copy ONLY the required function** from your completed skeleton file into the Coursemology code window.
5. Click “Run Code” to test that your function works on Coursemology.
6. Click “Finalise Submission” to submit your code for the **ENTIRE Part A/B**. **You will not be able to amend your code after you have finalised your submission.**
7. You must name your functions exactly as the questions state.

Failure to follow each of the instruction will result in 10% deduction of your marks.

Part A (Deadline: Same day of your TLab)

Question 1: Treasure Hunt – X marks the spot! (20 Marks)

You’re on a quest for treasure, and you’ve been given a 300 row x 300 column treasure map in the form of a CSV file. An example of such file is in `treasure.csv`.

The file looks something like this:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0

Within the treasure map file, there’s a particular cell which marks the place where the treasure is buried. The cell is marked with the string ‘x’.

Your task is to write a function `find_treasure` that:

- takes a string `file_path_map` (e.g. ‘`treasure.csv`’) and reads the CSV file into a **2D List**. (Hint: use the `csv` library!)
- returns a **tuple** (`row`, `column`) of the coordinates of the treasure (e.g. `(111, 222)`)**.

****No marks will be given if you just output the coordinate of the treasure via other means.**

Question 2: Decrypting the Map – Practice! (20 Marks)

Congratulations! You've found the spot and have dug up the spot only to find... another piece of paper! Rats! It seems to be encrypted in some form of gibberish language...

But wait! Not all hope is lost! You spot a **letter substitution translation guide** at the bottom right corner of the treasure map, which is in the form of a CSV file. An example of such file is in `translation.csv`.

The translation guide consists of two columns: the first column `cryptic_language` contains letters which corresponds to that in the second column `english`.

Your task is to write a function `read_translation_guide_into_dictionary` that:

- takes a string `file_path_translation` (e.g. `'translation.csv'`) and reads the CSV file.
- returns a **dictionary** with the *key* being `cryptic_language`, and the *value* being `english`.
- An example of the returned dictionary is on the right.

```
{ 'a': 'm',  
  'b': 'a',  
  'c': 'c',  
  'd': 'y',  
  'e': 't',  
  'f': 'v',  
  'g': 'o',  
  'h': 'u',  
  'i': 'x',  
  'j': 'e',  
  'k': 'j',  
  'l': 'w',  
  'm': 'f',  
  'n': 'z',  
  'o': 'd',  
  'p': 'l',  
  'q': 'i',  
  'r': 'k',  
  's': 'h',  
  't': 'n',  
  'u': 'g',  
  'v': 'b',  
  'w': 'q',  
  'x': 's',  
  'y': 'p',  
  'z': 'r' }
```

Part B (Deadline: Day of TLab + 6 days)

Question 3: Decrypt the Map! (30 Marks)

You have your trusty translation dictionary (pun intended?) at hand, now it's time to decrypt the message! An example of such file is in `message.txt`. (You have to write your own helper function to load it into memory like Part A Question 1, **but as a string**.)

Use your skills to write a function **decipher_message** that takes in the following:

- first argument: a translation **dictionary** (obtained from Part A Question 2), and
- second argument: a **string** (not a file path) containing the encrypted message.

Then decrypts the encrypted message, and returns the decrypted message as a **string**.

Characters in the encrypted message which are not found in the translation dictionary (keys) can be taken as-is without having to do any substitution.

Sample Run:

```
>>> sample_message = 'esqx qx b ejxe ajxxbuj!'
>>> decipher_message(translation_guide,sample_message)
'this is a test message!'
```

(Hint: How can you use what you've learned so far?)

Question 4: Find the Treasure...Map! (30 Marks)

You search the nearby area and manage to find another map! This is it! You're close to being rich beyond belief! But wait, it's encrypted again! An example of such file is in `encrypted_map.txt`.

Your task is to write a function **decrypt_map** that:

- takes a string containing the file path to an encrypted map (e.g. `'encrypted_map.txt'`).
- takes a string containing the file path to a translation guide (e.g. `'map_code.csv'`).
- decrypts this encrypted map using the translation guide.
- writes this decrypted map to a file called **decrypted_map.txt**.

An example of `decrypted_map.txt` (using the provided `encrypted_map.txt`) is on the right.

Note that there must not be any empty newline characters at the end of `decrypted_map.txt`.

Congratulations! You're rich!

```
WWWWWWWWWW,T,WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWW,WWWWWWWWWW,,,,WWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWW,,,,WWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWW,TT,WWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWW,,,,WWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWW,,WWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWW,,WWWWWWWWWWWWWWWWWW
```

Optional Question (Further Practice)

There is an optional question in Coursemology (Lab 04 Additional Practice) that carries no marks, but good for additional practice if you need it! [This one is harder than normal questions, so don't be too worried if you can't solve it!]