The University of Texas
Computer Science

313E Programming Assignment 03
Cipher

Setup

This assignment may be completed individually or with a pair programming partner. Include your and your partner's name and EID in the Python file header. If you are working alone, you may delete one of them.

Complete these steps to prepare for the Assignment.

Check	Description
☐	Download cipher.py and input.txt. You will be working in the cipher.py file.
☐	Place both files in the same folder/directory.
☐	You may not change the file names. Otherwise, the grading script will not work.

Problem Description

Cryptography is an ancient study of secret writing. There is a wealth of literature in this field. An extremely readable book on this subject is *The Code Book* by Simon Singh. This is a field of study that is of particular relevance in Computer Science. Given the widespread use of computers, one of the things people are interested in is making transactions over the Internet more secure.

There are two classes of ciphers - transposition ciphers and substitution ciphers. Transposition ciphers scramble the characters in the string. The substitution ciphers on the other hand replace individual characters in a string. We will look at a cipher for each class. The transposition cipher that you will implement is called the Rail Fence Cipher. For the substitution cipher, you will implement the Vigenere Cipher.

Requirements

Write a program that can encode and decode the Rail Fence Cipher and Vigenere Cipher.

1. Your program should read using input() to parse an input file.
2. Your program should match the expected output, printing out the cipher, the encoding process, and the decoding process for **both** the Rail Fence Cipher and Vigenere Cipher.
3. **You must follow the template of the program given. You may not change the names of the functions listed. They must have the functionality as given in the specifications. You can always add more functions than those listed.**
4. **You may not import any additional external libraries in your solution.**

Rail Fence Cipher

The rail fence cipher is also known as the zigzag cipher and is a form of a transposition cipher that jumbles up the order of the letters. The rail fence cipher works by writing your plain text downwards and diagonally on successive rows. The number of rows is given by a key. Once you reach the bottom row, you traverse upwards and diagonally. After you reach the top row, the direction is changed again, leading to a zigzag manner in which the plain text is written. After the plain text is written in this form, the rows are combined to create the cipher text.

For example, let us consider the plaintext **helloworld** with key = **3**.

Plain Text: **`h e l l o w o r l d`**

Key: **3**

To encode this message we first write over three lines as follows:

```
h - - - o -  - - l -
- e - l - w  - r - d
- - l - - -  o - - -
```

The cipher text in this case is written by reading off the top row, then the second row, then the last row from left to right.

Cipher Text: **`h o l e l w r d l o`**

To decode a given cipher text, you would need to reconstruct the list. First, construct a 2D list with the number of rows equal to the given key and the number of columns equal to the length of the cipher text. Next, traverse the 2D list in the same diagonal manner as encoding and mark the cells in this 2D list that should contain a character. Then, traverse the list row by row and fill in the characters of your ciphertext. Lastly, read the matrix in the diagonal fashion to uncover your plain text.

In the transposition cipher, you are merely scrambling all the characters in the plain text. In your assignment keep **all** the characters of the plain text - lower case letters, upper case letters, digits, punctuation marks, and spaces. Your cipher text will have all the characters (unchanged) from the plain text.

Vigenere Cipher

One of the main downfalls of substitution ciphers is that they can be broken using frequency analysis. To make the cipher more secure you can use two or more letters to encode. The Vigenere cipher uses a passphrase to encode and decode based on the following table.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
b	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
c	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
d	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
e	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
f	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
g	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
h	h	j	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
i	i	k	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
j	j	l	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
k	k	m	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
l	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
m	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
n	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
o	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
p	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
q	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
r	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
s	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
t	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
u	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
v	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
w	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
x	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
y	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
z	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Let us say we want to encode **helloworld** and we use the passphrase **seal**. Here is how we would lay the phrases out. The passphrase is repeated over and over again in order with the text that needs to be encoded. The letter from the passphrase marks the row and the letter from the plain text marks the column.

```
  Pass Phrase: sealsealse
   Plain Text: helloworld
Encoded Text: zilwgaocdh
```

To decode, we would reverse the process. Use the same passphrase **seal** repeatedly with the encoded text. This time, the letter from the passphrase marks the column. Within that column search for the encoded letter. That row indicates the plain text.

```
  Pass Phrase: sealsealse
Encoded Text: zilwgaocdh
   Plain Text: helloworld
```

In your implementation of the Vigenere cipher, you will **not** use the 2-D list. You **must** come up with a formula that will allow you to do the transformation. You will find `chr()` and `ord()` to be helpful for this. In class, we discussed that to represent characters in numbers, we assigned a mapping. One way of doing this was ASCII, and in Python, `chr()` and `ord()` use another mapping called Unicode. Here is a simplified table for the lowercase letters and their Unicode values.

Unicode # (decimal)	97	98	99	100	101	102	103	104	105	106	107	108	109
character	a	b	c	d	e	f	g	h	i	j	k	l	m
	110	111	112	113	114	115	116	117	118	119	120	121	122
	n	o	p	q	r	s	t	u	v	w	x	y	z

If you would like a refresher, here is the documentation on what `chr()` and `ord()` do.

Also, you **must** write a function to filter the input string. This function will first convert the string to lowercase and then remove all characters that are not lowercase letters so that your Vigenere cipher can work.

Input

Your input will be from a file called input.txt. Do not hard code the name of the file in your program. You will read the file using input redirection. You will run the code using the following command:

```
python3 cipher.py < input.txt
```

Alternatively, if you are using a Windows computer and are having trouble with input redirection **<**, you can run:

```
cat input.txt | python3 cipher.py
```

Here is an example of an input file:

```
helloworld
3
holelwrdlo
3
helloworld
seal
zilwgaocdh
seal
```

The first set of inputs will be to encoded and decoded using the rail fence cipher. The second set of inputs will be encoded and decoded using the Vignere cipher. Here are the explanations of each line in the input file. Assume that there are no errors in the input file.

1. plain text to be encoded using rail fence cipher
2. key for the rail fence cipher
3. encoded text to be decoded using rail fence cipher
4. key for the rail fence cipher
5. plain text to be encoded using Vignere cipher
6. passphrase (no spaces) for the Vignere cipher
7. encoded text to be decoded using Vignere cipher
8. passphrase (no spaces) for the Vignere cipher

Output

Your TestCipher program will test the encoding and decoding methods of the two Cipher algorithms. Your output will have the following format and look something like this:

```
Rail Fence Cipher

Plain Text: helloworld
Key: 3
Encoded Text: holelwrdlo

Encoded Text: holelwrdlo
Enter Key: 3
Decoded Text: helloworld

Vigenere Cipher

Plain Text: helloworld
Pass Phrase: seal
Encoded Text: zilwgaocdh

Encoded Text: zilwgaocdh
Pass Phrase: seal
Decoded Text: helloworld
```

Grading

Visible Test Cases - 75/100 points

The program output exactly matches the sample solution’s output. To receive full credit, the program must produce the correct output based on all requirements in this document and pass all the test cases.

Hidden Test Cases - 15/100 points

The program output exactly matches the sample solution’s output. To receive full credit, the program must produce the correct output based on all requirements in this document and pass all the test cases.

Style Grading - 10/100 points

The wordle.py file must comply with the PEP 8 Style Guide.

If you are confused about how to comply with the style guide, paste the error or the error ID (e.g. C0116 for missing function or method docstring) in the search bar here in the Pylint Documentation, and you should find

The TAs will complete a manual code review for each assignment to confirm that you have followed the requirements and directions on this document. Deductions will occur on each test case that fails to follow requirements.

Submission

Follow these steps for submission.

Check	Description
☐	Verify that the code does not have a compile or runtime error. A compile or runtime error will result in a 0 for the assignment as a whole.
☐	Verify that you have no debugging statements left in your code.
☐	Submit **only** cipher.py (the starter code file with your updates) to the given assignment in Gradescope. This will cause the grading scripts to run.
☐	If you have a partner, make sure you are submitting it for you **and** your partner on Gradescope. See this Gradescope documentation for more details.
☐	When the grading scripts are complete, check the results. If there are errors, evaluate the script feedback, work on the fix, test the fix, and then re-submit the file to Gradescope. You can submit as many times as necessary before the due date. NOTE: By default, only the most recent submission will be considered for grading. If you want to use a previous submission for your final grade, you must activate it from your submission history before the due date.

Academic Integrity

Please review the Academic Integrity section of the syllabus. We will be using plagiarism checkers, and we will cross-reference your solution with AI-generated solutions to check for similarities. Remember the goal in this class is not to write the perfect solution; we already have many of those! The goal is to learn how to problem solve, so:

- don’t hesitate to ask for guidance from the instructional staff.
- discuss with peers about **only** bugs, their potential fixes, **high-level** design decisions, and project clarifying questions. Do not look at, verbalize, or copy another student’s code when doing so.

Attribution

Thanks to Dr. Shyamal Mitra for the assignment requirements and Dr. Carol Ramsey for the instruction template.