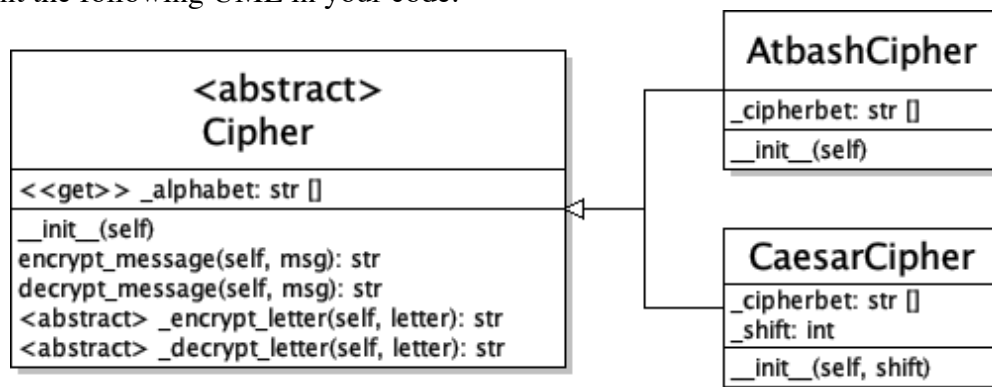


CECS 277 – Lab 7 – Abstract

Secret Decoder Ring

Write a program that allows the user to encrypt or decrypt messages using two different types of encryption methods. Encrypted messages will read from the console and then written to a file called 'message.txt', and decrypted messages will be read from the 'message.txt' file then displayed to the console.

Implement the following UML in your code:



Cipher (cipher.py) – is an abstract class that defines the interface for any cipher subclass.

1. Attributes: `_alphabet` is a list of letters A-Z (alternatively, this could just be one big string with the letters A-Z in it, since a string is effectively a list). Make a get property to allow the subclasses to access the alphabet.
2. `__init__(self)` – initializes the alphabet.
3. `encrypt_message(self, message)` – convert the message to upper case letters, then loop through the message string one character at a time, if it is a letter A-Z, then call the `encrypt_letter` method, otherwise ignore it. Build the encryption string using the encrypted letters and ignored characters, and return it (ie. leave all spaces, punctuation, and numbers the same, only letters in the string will be encrypted).
4. `decrypt_message(self, message)` – convert the encrypted message to upper case, then loop through each letter in the message if it is a letter A-Z, then call `deccrypt_letter`, otherwise ignore it. Build the encryption string using the encrypted letters and ignored characters, and then return it.
5. `_encrypt_letter(self, letter)` and `_decrypt_letter(self, letter)` – abstract methods (no code) – each class that inherits from **Cipher** will be forced to override these methods to encrypt and decrypt letters in its own way.

AtbashCipher (atbash.py) – extends from **Cipher** – Atbash is a substitution cipher where the encrypted message is obtained by looking up each letter and finding the corresponding letter in a reversed alphabet. The encoded letter can be found in one of two ways, either a parallel list look up (ex. letter to encode = 'B', location in alphabet = 1, encoded letter in cipherbet at location 1 is a 'Y'), or a calculated position in the list (ex. letter to encode = 'B', location in alphabet = 1, 25 – location = 24, encoded letter in alphabet at location 24, is a 'Y').

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
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
Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

1. `__init__(self)` – call super to initialize the alphabet, then create the cipherbet attribute as a list with the letters Z-A in it.
2. `_encrypt_letter(self, letter)` – look up the letter in the alphabet to find its location. Use that location to find and return the encrypted letter in the cipherbet.
3. `_decrypt_letter(self, letter)` – look up the letter in the cipherbet to find its location. Use that location to find and return the decrypted letter in the alphabet.
4. Note: if you're not using the parallel list look up and are calculating the location of the encrypted/decrypted letter instead, then you do not need to create the cipherbet attribute or have an `__init__` method.

CasesarCipher (caesar_cipher.py) – extends from Cipher – Caesar is another substitution cipher where the encrypted message is found by looking up each letter and finding the corresponding letter in a shifted alphabet (ex. letter to encode = 'B', location = 1, shift value = 3, location + shift value = 1 + 3 = 4, encoded letter location = 4, which is an 'E'). If the shift value causes the encoded letter to be past the end of the alphabet, then it should wrap around to the beginning (ex. letter to encode = 'X', location = 23, shift value = 3, location + shift value = 23 + 3 = 26, encoded letter location = 26, which is larger than 25, subtract the total number of letters in the alphabet to get the updated location, 26 – 26 = 0, which is an 'A').

Example alphabet with a shift value of 3:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
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
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

1. `__init__(self, shift)` – call super to initialize the alphabet, then set the shift value if it's between 0-25, otherwise raise an exception. Use the shift to create the cipherbet.
2. `_encrypt_letter(self, letter)` – look up the letter in the alphabet to find its location. Use that location to find and return the encrypted letter in the cipherbet.
3. `_decrypt_letter(self, letter)` – look up the letter in the cipherbet to find its location. Use that location to find and return the decrypted letter in the alphabet.
4. Note: if you're not using the parallel list look up and are calculating the location of the encrypted/decrypted letter instead, then you do not need to create the cipherbet attribute, but you will need the init method for setting the shift value.

Main (main.py): Have the user choose to encrypt or decrypt a message, then have them choose an encryption/decryption method (Atbash or Caesar cipher). If they chose to encrypt, then prompt them to enter a message to encrypt, then write the encrypted message to the file 'message.txt'. If they choose to decrypt a message, read the message from the file 'message.txt' and then display the decrypted message to the console. If they choose to use a Caesar Cipher for either encryption or decryption, then prompt the user to enter a shift value (0-25).

Example Output (user input is in italics):

Secret Decoder Ring:

1. Encrypt

2. Decrypt

1

Enter encryption type:

1. Atbash

2. Caesar

2

Enter message: *The soup is
poisoned!*

Enter shift value: *3*

Encrypted message saved to
"message.txt".

Secret Decoder Ring:

1. Encrypt

2. Decrypt

2

Enter decryption type:

1. Atbash

2. Caesar

2

Enter shift value: *3*

Reading encrypted message
from "message.txt".

Decrypted message: THE SOUP
IS POISONED!

Encrypted message written to file:

WKH VRXS LV SRLVRQHG!

Notes:

1. You may have 5 different files: cipher.py, atbash.py, caesar.py, main.py, and check_input.py (if used).
2. Check all user input (except for the message) for invalid values using `get_int_range`.
3. Do not create any extra methods or attributes in your classes, you may leave out the `cipherbet` attribute if you're calculating your positions.
4. Please do not create any global variables or use the attributes globally. Only access the attributes using the class's methods (note: it's usually ok for the subclass to access the attributes of the superclass directly, since they are also attributes of the subclass).
5. Use docstrings to document the classes, and each of their methods.
6. Place your names, date, and a brief description of the program in a comment block at the top of your main file. Place brief comments throughout your code.
7. Thoroughly test your program before submitting:
 - a. Make sure that the user input is validated.
 - b. Make sure that the encrypted messages are written to the file.
 - c. Make sure that decrypted messages are read from the file.
 - d. Make sure that the encrypted/decrypted text is correct given the type of encryption/decryption method and the shift value (when caesar is used).
 - e. Make sure that you test letters that are at the end of the alphabet so that when caesar's shift value is applied, it does not go out of bounds of the alphabet.
 - f. Make sure that any spaces, punctuation, or numbers are preserved in the encrypted/decrypted message (only letters need to be encrypted).

Secret Decoder Ring Rubric – Time estimate: 3 hours

Secret Decoder Ring 10 points	Correct. 2 points	A minor mistake. 1.5 points	A few mistakes. 1 point	Several mistakes. 0.5 points	No attempt. 0 points
Cipher class (in a separate file): 1. Inherits from ABC. 2. Has attribute for _alphabet. 3. Has methods: __init__, @property, encrypt_message, decrypt_message 4. Has abstract methods _encrypt_letter and _decrypt_letter. 5. encrypt_message loops through letters and calls _encrypt_letter. 6. decrypt_message loops through letters and calls _decrypt_letter.					
Atbash class (separate file): 1. Inherits from Cipher. 2. Has overridden _encrypt_letter and _decrypt_letter methods. 3. Optional: init and cipherbet. 4. Correctly converts letters using atbash cipher.					
Caesar class (separate file): 1. Inherits from Cipher. 2. Has overridden _encrypt_letter and _decrypt_letter methods. 3. init method sets shift attribute (optional: cipherbet attribute). 4. Correctly converts letters using caesar cipher.					
Main File: 1. Prompts user to encrypt or decrypt. 2. Prompts user atbash or caesar. 3. Prompts for shift if caesar. 4. Encrypting: prompt for message and write encrypted message to file. 5. Decrypting: read encrypted message from file and display decrypted msg. 6. Error checks user input.					
Code Formatting: 1. All code is in functions/methods. 2. Correct spacing and good naming. 3. No exceptions thrown. 4. No global variables. 5. Correct documentation. 6. No accessing attributes directly.					