Top-Down Programming Example: Cæsar Cipher

Step #1: Problem Statement

Goal: Write a program to encipher input text using a Cæsar cipher

Specification: State the inputs, outputs, and assumptions

The user enters some input. We need to decide how to handle it.

- Non-letters: leave them alone
- Capital letters: encipher them as capital letters
- Lower-case letters: encipher them as lower-case letters
- Key: use "D" (3) as the key

The program should print both the input and the output.

Step #2: Design

High-level design: Describe how the program will work

```
set key to 3
read input string; quit on EOF (end of file)
loop
    get next character in string; fall out of loop if at end
    encipher it (see above)
    add it to output
endloop
print input string, output string, each surrounded by double quotes
```

Data: How do you represent the data?

Represent the input and the output as strings; we will *collect* the encrypted input into a string and output both at the end.
Store the key as an integer.
To encipher a letter, represent the letter as a number between 0 and 25 inclusive (0 being "a" or "A", . . . , 25 being "z" or "Z"), add the key value, reduce modulo 26, and translate back into a letter.

Functions: What functions do we need?

Enciphering occurs in 2 places: for capitals and for lower-case letters. Define a function to do this.
The function will do the actual encipherment:

- Parameter: integer p corresponding to a letter, as above
- Returns: integer c corresponding to the enciphered letter, as above
- Action: compute $c = (p+3) \bmod 26$

If we want to change the algorithm, we change it here

Refinement: Refine algorithm

```
 1. key = 3
 2. plain = input()
 3. on error or EOF, quit
 4. cipher = empty
 5. for next character in plain
 6.     if it's a capital
 7.         map it into 0...25
 8.         call encipherment function to encipher it
 9.         map result back into 'A'...'Z'
10.         append it to cipher
11.     else if it's lower case
12.         map it into 0...25
13.         call encipherment function to encipher it
14.         map result back into 'a'...'z'
15.         append it to cipher
16.     else
17.         append it to cipher
18. print plain --> cipher
```

Let's refine the input a bit, bearing the Python language in mind. We need to read the input using a `try`...`except` construct. We won't check for EOFError in the `except`, so any kind of error immediately causes the program to end. This means we need to exit the program in the body of the `except`. The cleanest way to do this is with a `return` statement, so we will write the program as a `main()` function.

Let's also make the key a variable. That way, we can change it very quickly. We normally put such variables at the beginning of the file, so we can find them easily.

Implementation

```
#
# this corresponds to a key of 'D' ('A' = 0, ... 'Z' = 25)
# it's near the top so we can find (and change) it easily
#
key = 3
#
# the encipherment function
#
def encipher(p, k):
        return (p + k) % 26
#
# now the main routine
#
def main():
        # ask user for input message (plaintext)
        try:
                plain = input("Enter your message here: ")
        except:
                return

        # initialize output (ciphertext)
        cipher = ""

        #
        # encipher each character and append it to current ciphertext
        #
        for i in plain:
                # leave non-letters alone
                if i in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
                        # map the letter into 0 . . 25
                        n = ord(i) - ord('A')
                        # shift it
                        c = encipher(n, key)
                        # map it back into a letter
                        lett = chr(ord('A') + c)
                elif i in "abcdefghijklmnopqrstuvwxyz":
                        # map the letter into 0 . . 25
                        n = ord(i) - ord('a')
                        # shift it
                        c = encipher(n, key)
                        # map it back into a letter
                        lett = chr(ord('a') + c)
                else:
                        lett = i
                # now append it
                cipher = cipher + lett

        # print it, surrounded by quotes
        print("'%s' --> '%s'" % (plain, cipher))
#
# run the program
#
main()
```