Basic Python Project: Caesar Cipher

NOTE: READ THIS DOCUMENT THOROUGHLY FOR SUCCESS

Background

Ciphers existed well before computers were formally conceptualized. However, they play a big part in the foundations of computer science. Check out the cracking of the enigma machine to find out more.

The concept of a cipher is quite simple yet powerful. We take a message that we would like to keep secret, apply some transformations to it, and send it to some second party that has the means to decipher it. This way, any third party that intercepts our messages is left clueless as to how to read this message.

An example would be pig latin. We first take our message: *"hello world*", encrypt it using the algorithm below, and send it out to our friend over in the other town via carrier pigeon.

1. Remove the first letter of each word.
2. Concatenate that letter to the end of each respective word.
3. Add “*ay*” to the end of each word.

From this, we transform "*hello world*" into "*ellohay orldway*."

Say that our message is intercepted by a malicious third party that lures our carrier pigeon via a delicious bowl of seeds. Without the algorithm we listed above, "*ellohay orldwa*y" is simply nonsense gibberish.

Caesar Cipher

According to legends, ciphers were even present in 50 BC when Julius Caesar was fighting the Gauls in the forests of modern-day France.

To communicate plans with his generals, Caesar sent a courier out into the wilderness with nothing but his sandals, sword, and blessings from Jupiter or whichever Pantheon was preferred during that day.

The risk of this courier being caught was high. Therefore, to ensure that any message in Latin was indecipherable to the Gauls, Caesar's armies encrypted their message using a method we know as Caesar Cipher.

The algorithm for this cipher is as follows:

1. Select some string value as your message. Label this string as “message”.
2. Select some numeric key value to encode your message. Label this value as `k`.
3. For each letter in “message”

a. Shift each letter in your message `k` letters forward.
b. If we go past the last letter in our alphabet (`z`), simply continue counting from the first letter (`a`).

For example, let's say you have the message “*hello world*”. Let's use the key value of 10.

We would take each letter of this word and shift it 10 letters forward. For example, the letter *h* would become *r*, *e* would become *o*, and so on until we get:

rovvy gybvn

Let's consider again what happens when we go past the letter *z*. For example, we have the letter *w*. Shifting this letter 10 letters forward, we would hit *z* on the 3rd shift.

Therefore, we just simply restart our count at *a* and count the remaining 7 letters forward until we get *g*.

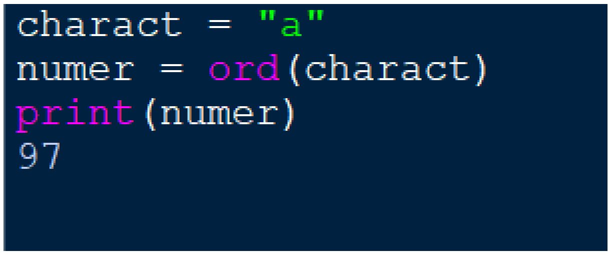
If you’d like more guidance on this, check out the following video.

ASCII

But how can we shift letters numerically when we know that they represent a completely different datatype? Well, as it turns out, computers have a numerical representation for almost all characters on our keyboard!

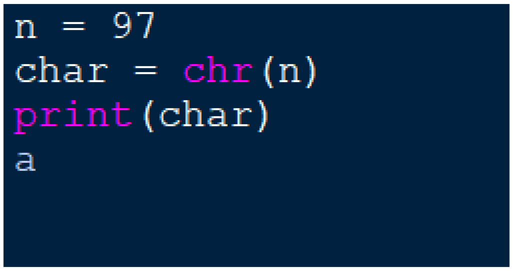
This is called American Standard Code for Information Interchange (ASCII). Check out the following link to see these numeric representations.

To get these numeric representations in Python, we simply use the **ord()** function. This function works by taking in a string argument that represents a character, and returns its ASCII representation. Using the following documentation, we can get the ASCII representation of *a* by doing the following:



```
charact = "a"
numer = ord(charact)
print(numer)
97
```

We can also go in the reverse! We can get the character representation of an ASCII value by using the chr() function. This function works by taking in a numerical value, and returning the character that corresponds to that ASCII value. For example, we can get the character representation of 97 by doing the following:



```
n = 97
char = chr(n)
print(char)
a
```

Instructions

Create a GitHub repository which you will clone to your computer.

Inside of this repository, create a Python file named `caesar.py`. From here, you can simply copy and paste the code provided for you in from the following replit.

This file takes in two arguments: the word you want to encrypt and the integer key. For example, you can encrypt `hello world` with the key value of 10 using:

python caesar.py helloworld 10

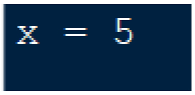
Note that you cannot include spaces in your first argument.

Submission

Submit a link to your repository. While a comprehensive writeup in your README will not be necessary, **you must still document your code where appropriate.**

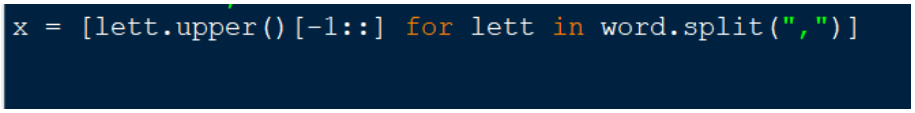
Ask yourself, is this code's purpose immediately obvious? If not, add a comment.

For example, this code is immediately obvious. We're just creating a variable, and this does not need any further explanation:



```
x = 5
```

This code, however, is not immediately obvious and therefore should require a comment:



```
x = [lett.upper()[-1::] for lett in word.split(",")]
```

GPT usage will be penalized