Girls' Programming Network

Cryptography!

Caesar Cipher Extensions

7. Extension: Random Keys?

So far we have had to enter a key number. Now, we want to let the computer choose a key number to use if the user types ‘random’ when asked to enter the key number.

Task 7.0: Let’s get random!

First we need to `import random` at the top of our program!

Task 7.1: Letters or numbers?

Now, when we ask the user for a key they can either type a number (the key) or the word “random”.

1. Add an if statement to check if the key they entered is the word `"random"` before the rest of your program.
2. Also add an else so that if they put in a number, the key is set to that number!

Hint

Don’t forget to use `int()` when you set the key to be the number

Task 7.2: So random

If the input is ‘`"random"` the computer should pick a random number and store it in the key variable.

Write code that chooses a random number between 1 and 25 and stores it in the key variable.

Hint

To get a random number between 1 and 10 you would write it like this:

```
number = random.randrange(1, 11)
```

Task 7.1: Write it down!

We need to know what the random number is so that we can give it to our friends to decode the message later. Add a print statement that prints the key before the encrypted message.

8. Extension: While loop

What if we have heaps of messages to encrypt and decrypt? Let's loop our whole program so that we can use it for heaps of messages!

Before you start this task, take a copy of your code to use in Task 9 & 10. Copy your code in Replit by clicking on the dots next to main.py file on the left hand window and selecting Duplicate.

Task 8.1: Whiling away the while

To make the code run over and over and over again so that we can keep encrypting all of our messages, we want to add a **while loop** near the top of the program.

1. Add a `while True` loop to your program!

Hint

While loops look like this:

```
while <SOME CONDITION>:
    statement
```

For our loop to run continuously, we can use `while True:`

Remember that you only want to put things you want to do EVERY TIME inside the while loop, so printing out the welcome statement and making your alphabet variable should be before the loop.

Task 8.2: Add new line

Did you notice that `What is your message?` doesn't appear on a new line when it runs in a loop? Look at the text in red in the below box. This is because we are using the `print(variable, end='')` line to print the letters in our encrypted message all on one line.

```
Welcome, this is the Caesar cipher
What is the message? hello
What is the key number? 4
Do you want to encrypt or decrypt? (e or d) e
lippsWhat is the message?
```

1. Use an empty print statement to print a blank line at the end of each loop

Hint

We want it to appear like this:

```
Welcome, this is the Caesar cipher
What is the message? hello
What is the key number? 4
Do you want to encrypt or decrypt? (e or d) e
lipps
What is the message?
```

Task 8.3: How to stop the loop?

Our while loop is running over and over and over forever. Let's make it stop when we want.

If the user enters nothing (ie just hits enter) when they are asked for their message, we want to end the program.

1. After your input message line, create an `if` statement.
2. The `if` statement should check if the message entered is an empty string, `""`
3. If it is, then we want to `break` to stop the program. We do this by writing `break` inside our `if` statement.
4. Don't forget to indent your `break` in the `if` statement.

9. Extension: Writing Files

If you want to send an encrypted message to your friend, you need to store it in a file. We are going to write your encrypted message to a file.

Rename the code file you used in Task 8 so it's no longer main.py. Rename the copy of your code that you made before you started Task 8 to main.py (this should not include your `while True:` loop). Replit only runs code called main.py.

Task 9.1: Storing your encrypted message

Instead of just printing out our message as we go we want to build up the encrypted message in a variable so we can store it in a file at the end.

1. Before your for loop, create a variable called `encrypted_message`, set it to the empty string `""`. This is called 'initialising' a variable.

2. You currently print out the encrypted letter (or print out the regular character if it wasn't in the alphabet). We need to replace both of these lines. Instead you need to add the character you would have printed to the end of `encrypted_message`. *See Hint below for how to add a string to the end of another string.*

3. To test that your code still works, print out your `encrypted_message` after your for loop ends.

4. Comment out the empty print statement that prints a blank line at the end of each loop that you added in Task 8.2. It's no longer needed.

Hint

You can add to a string and overwrite what was there before like this:

```
name = "harry"
last_name = "potter"
name = name + last_name
```

Remember you want to add the new encrypted letter to the end of `encrypted_message` and overwrite what was in `encrypted_message`

Task 9.2: Putting it in a file

When you have encrypted the whole message, we need to make a file and put our encrypted message in it!

1. Go to where you tested printing your `encrypted_message`, we're going to put our file code here.

   Use this line to make a file (called output.txt) that we can write to (that's what `'w'` does):

   ```
   with open('output.txt', 'w') as f:
   ```

2. Inside the `with` statement you can write to the file with this line: `f.write(encrypted_message)`. Remember to indent this line under the `with`.

3. Run your code and see if a file is created with your encrypted message in it! The file will be in the same location as your code.

10. Extension: Reading Files

We are going to get your program to read a file with an encrypted message, and print out the decrypted message.

Task 10.1: Download an encrypted file

1. Download the file caesar1.txt from the GPN page
2. Upload caesar1.txt into Replit (click on the 3 dots next to Files and select Upload File).

Task 10.2: No need to ask

Since we are getting the message from a file, we don't need to ask the user for a message. Comment out the input statement that gets the message from the user.

Task 10.2: What's inside?

Now we need to access what is inside the file.

1. At the beginning of your code (but after any imports you've added) we want to `open` the file caesar1.txt
2. Read the 1 line that is there
3. Strip the extra space from the end
4. Make sure you store the message in a variable called `message`
5. Print out the encrypted message and check it's the same as what's in the input file

Hint

To open and read a line from a file and store it in variable called word you can do this:

```
with open('file.txt') as f:
    word = f.read()
    word = word.strip()
```

Strip removes extra space & new line character from the end of the message.

Task 10.3: Decrypt your message

1. Now you have a value in your message variable again.
2. Comment out the changes you made in Task 9 above to initialise, build and print the `'encrypted_message'` variable
3. Uncomment out the changes you made in Task 9 to not print `'new_letter'` or `'current_letter'`.
4. Comment out the code to open and write to the file that you added in Task 9.
5. Run you code
    a. Enter the key number on the GPN page to decrypt the text file.
    b. Choose mode “d”
6. Your code should now print out a decrypted message!
7. Use you Cipher wheel to check if the decryption is correct

Task 10.4: So many messages to decrypt

1. Download the other encrypted files from the GPN page (caesar2 - caesar6.txt)
2. Use your program to decrypt them - don’t forget to change the input file name in your code for each file.