Girls' Programming Network

Cryptography

Create a Caesar Cipher encryptor and decryptor!

This project was created by GPN Australia for GPN sites all around Australia!

This workbook and related materials were created by tutors at:

Sydney, Canberra and Perth

Girls' Programming Network

If you see any of the following tutors don't forget to thank them!!

Writers

Renee Noble
Sarah Mac
Alex McCulloch
Jess Austin
Bhavna Yadav
Lyndsey Thomas

Testers

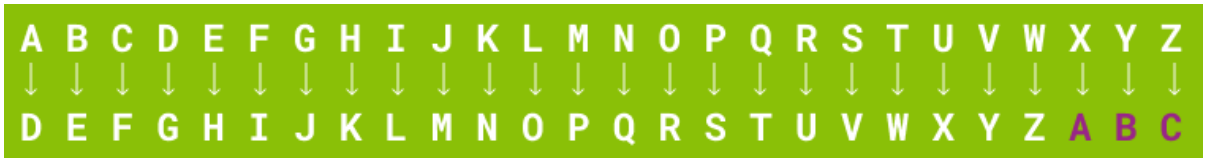
Courtney Ross
Maddie Jones
Jess Austin
Maggie Liuzzi

Part 0: Caesar Ciphers

Caesar Ciphers are a shift cipher. This means we are going to encrypt messages by shifting the alphabet along and replacing the letters in our secret message.

This is what the alphabet would look like if we shifted it by 3 letters:

We don't want some of our letters falling off the end so we wrap them around.



Another great way to represent this is in a circle! That does the wrapping for us!

To **encrypt** a message we replace a green letter with the matching purple letter:

GPN → JSQ

To **decrypt** a message we take a purple letter and replace it with the matching green letter:

FRGH → CODE

(reading purple to green is the same as rotating the wheel -3)

Task 0.1: Encrypting and decrypting messages

Using the rotated wheel above can you encrypt and decrypt these messages

Encrypt	Decrypt
SECRET → __ __ __ __ __ __	FUDFN → __ __ __ __ __
CIPHER → __ __ __ __ __ __	FDHVDU → __ __ __ __ __ __

Cipher Wheels

Sometimes we want to use **different keys**. So we want a different amount of rotation. We've given you a **cipher wheel** that lets you do this rotation!

Grab your cipher wheel! You can rotate the inner wheel to try different shifts.
Let's give it a try!

Task 0.2: Encrypt with a key of 10!

Let's try using a different key. **Let's try 10**. Rotate your cipher wheel so the **green A** lines up with the **purple K**. **Encrypt** this message:

MYSTERY → _ _ _ _ _ _ _

Task 0.3: Encrypt with a key of 24!

Let's try a **key of 24**. Rotate your wheel 24 spots and **encrypt** this message. (You can ignore spaces)

GPN IS GREAT → _ _ _ _ _ _ _ _ _ _ _ _

Task 0.4: Decrypt with a key of 7!

Let's try a **key of 7**. Rotate your wheel 7 spots and **decrypt** this message:

JYFWAVNYHWOF → _ _ _ _ _ _ _ _ _ _ _ _

Task 0.5: Decrypt with a key of 11!

Let's try a **key of 11**. Rotate your wheel 11 spots and **decrypt** this message:

DATY ESP HSPPW → _ _ _ _ _ _ _ _ _ _ _ _

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 1:

☐ You used your cipher wheel!

☐ You encrypted all the message!

☐ You decrypted all the secret messages!

Part 1: Setting up

Intro

Task 1.1: Making a python file

Open the start menu, and type 'IDLE'. Select IDLE Python 3.

1. Go to the file menu and select 'New File'. This opens a new window.
2. Go to the file menu, select 'Save As'
3. Choose Desktop and save the file as 'caesar_cipher.py'

Task 1.2: You've got a blank space, so write your name!

At the top of the file use a comment to write your name!
Any line starting with # is a comment.

```
# This is a comment
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 2:

☐ You should have a file called caesar_cipher.py

☐ Your file has your name at the top in a comment

☐ Run your file with F5 key and it does nothing!!

Part 2: Tell me your secrets

Task 2.1: Welcome message

1. Let's `print` out a message to welcome the user.

```
Welcome, this is the Caesar Cipher
```

Hint

If we wanted to print out "Hello, World!" we'd do it like this:

```
print("Hello, World!")
```

Task 2.2: What is your secret message?

1. Use `input` to ask for a secret message. Store the secret in a variable called `message` so we can use it in our code!

Hint

If you want to ask for somebody's name, you can do it like this:

```
name = input("What is your name? ")
```

Task 2.3: What is the key to your secret message?

1. Use `input` to ask for the key to your secret message. Store it in a variable called `key` so we can use it in our code!

For example, it might look like this:

```
How many letters to rotate by:
```

Task 2.4: Convert key to integer

1. Use `int()` to convert variable `key` to an integer.
2. Overwrite your current key with the new number version of the key

Hint

You can overwrite what's in your variable and use your variable at the same time!

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

Task 2.5: Print your secret message

1. Let's `print` our secret message that we got from the user earlier.

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 3:

☐ Print a welcome message
☐ Ask for secret message
☐ Ask for a key to encrypt the the secret message with
☐ Convert key to integer format
☐ Print your secret message
☐ Run your code

★ BONUS 2.6: What's the name of your secret language?

Waiting for the next lecture? Try adding this bonus feature!!

Let's come up with the name of our secret language and `print` the language of our secret message.

★ BONUS 2.7: Customize welcome message

Waiting for the next lecture? Try adding this bonus feature!!

Let's customise the welcome message for the user.

For example, if the user typed in their name was Lyndsey, then `print` the welcome message for Lyndsey.

```
Welcome to my amazing cipher, Lyndsey
```

String

Part 3: Getting a secret letter

Task 3.1 How do we get a secret letter?

Let's try doing this with pen and paper!

1. Get the first letter of the word and write it in the second column below.
2. Count how far through the alphabet that letter is. You may want to use your cipher wheels to count with. Put this in the third column. Remember that computers start counting at 0, so you should count like a computer does!
3. Using **5 as the key**, add 5 to the index of that first letter to get our secret letter index. Put that in the fourth column.
4. Now count through the alphabet to find what letter is at the secret letter index. Write it in the last column. Remember again that computers start counting from 0!
5. In the next steps, we'll write code to do this for us!

Message	First letter	Letter Index	Secret Letter Index	Secret Letter
apple	a	0	5	f
banana				
carrot				
grapes				

Hint

Counting the wheel can take a long time. You can use this table to look up the indexes of the letters:

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
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

Task 3.2: Let's store the alphabet!

1. Write a variable that stores the alphabet in a string like this:
   ```
   alphabet = 'abcdefghijklmnopqrstuvwxyz'
   ```

Task 3.3: Get a Letter!

1. Now **store the first letter** of the message in a variable called `current_letter`

2. To test your code print out the `current_letter`. Try these examples, write what is printed for `current_letter`! See if it is the first letter of the word!

Message	current_letter
gpn	
abracadabra	
pizza	
zoink	

Hint

Remember that you can get the first letter of a string like this:

```
name = "gpn"
first_letter = name[0]
```

Task 3.4: Find that letter!

Next we need to figure out where in the alphabet that letter is!

1. **Make a variable** called `current_index` which will store which position in the `alphabet` the `current_letter` is!

2. **Print** out the `current_index` and **write** what number is printed for each of these examples to make sure it's right! Remember that indexes start at 0.

Message	current_letter	current_index
gpn		
abracadabra		
pizza		
zoink		

Hint

You can use this to find the index of a letter in a string

```
word = "gpn"
p_index = word.index("p")
```

Task 3.5: Turn the key!

So we've got the position of the original letter, now we have to use the key to shift it to get a new index (later we'll use this to get the secret letter). To do that we need to work out the index of the secret letter which will be the **`current_index` plus the `key`.**

1. Make a variable called `new_index`, **set this using the formula above** to get the the location of the secret letter.

2. Test your code using 5 as the key, print out the `new_index` and write what number is printed for each of these examples to make sure it's right! Remember that it should be what you wrote down for the last task plus your key!

Message	current_letter	current_index	new_index
gpn			
abracadabra			
pizza			
zoink			

Hint

For example if we wanted to take the message "cipher" the table would look like this:

Message	current_letter	current_index	new_index
cipher	c	2	?

Since our key is 5 and our current_index is 2, we can work out that new_index should be the current_index plus the key, so it's 5+2 which is 7!

Task 3.6: Secret Letter!

Next let's use the new index to get the secret letter from the alphabet.

1. Make a variable called `new_letter` to store the letter in the alphabet that is at the `new_index` position.
2. Print the new letter!
3. Fill out the table below for each of the examples using 5 as the key.

Message	current_letter	current_index	new_index	new_letter
gpn				
abracadabra				
pizza				
zoink				

Did it work with zoink? Don't worry, we're going to fix it in the next task!

Hint

Remember that you can get a letter out of a string like this:

```
name = "Alex"
index = 2
letter = name[index]
```

Task 3.7: Zoink!

1. What happens when you put in "zoink" and a key of 5?

You get and error!
This is because when you try to go 5 letters past 'z' in the alphabet, there isn't a letter there! To fix this we need to make the index wrap around back to 0 when it gets to the end of the alphabet. We can use modulo to make our code do this!

2. Make the `new_index` be the `new_index` modulo 26
3. Overwrite `new_index` with the modulo version of itself

Hint

To modulo something by 5 you do this:

```
number = 21
number = number % 5
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 4:

☐ You have printed and checked the current_index of all the examples

☐ You have printed and checked the new_index of all the examples

☐ You have used your cipher wheel to check the new letter of all of the examples

★ BONUS 3.8: zoink, ZoInK and ZOINK

Waiting for the next lecture? Try adding this bonus feature!!

At the moment our code only works with letters that are lowercase. Try entering an ALL CAPS word as your message and see what happens.

You get an error!

To fix this we need to change the message to be all lowercase. We can use `word = word.lower()` to do that.

Update your code so we're always using the lowercase version of the message!

Loops

Part 4: What about words?

Task 4.1: Not just one letter

1. On your cipher wheel, try and encrypt ‘gpn’ with a **key of 4**, what letters do you get?

gpn = _____

2. Now try putting “gpn” with a key of 4 into your program and seeing if they match.

Did it match what you did by hand?

Right now your code only gets the first letter of the message variable. The computer doesn’t know that there are more letters in message, we need to tell it!

Task 4.2: Not just one letter

1. First we can remove the code that sets the `current_letter` variable to the first letter of your message. You can do this by making the whole line into a comment.

Hint

Remember:

```
# this is a comment
```

Task 4.3: Working letter by letter

Now we want to add a `for loop` so that the computer looks at every letter in the message and turns it into a secret letter.

1. Add the `for` loop right under the comment you just made.
2. Your for loop should loop through the `message`, this will give us the `current_letter` one at a time.

Hint

Here’s an example of a for loop. Don’t forget that when you code a for loop you need to use indents.

```
for letter in word:
    print(letter)
```

Task 4.4: Check the whole secret word

1. Use your cipher wheel to encrypt a 3 letter word, like ‘gpn’, rotated by 6.

   ___ ___ ___ → ___ ___ ___

2. Now type that word and key into your encryptor and see if they match.

3. If it’s working you can move onto the next step.

Task 4.5: Everybody get in line!

1. Try your program with a long word like ‘zooperdooper’.

   The secret word is way too
   L
   O
   N
   G
   !

   Let’s print it on one line so that it is easy to read.

Hint

Use the `print(variable, end='')` that you learned in lecture slides to print your secret word on one line.

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 4:

☐ Used a comment to remove a line of code
☐ Added a for loop
☐ Check your code with a 3 letter word
☐ Print the word on one line

★ BONUS 4.6: Take your time

Waiting for the next lecture? Try adding this bonus feature!!

Let's make it look like the computer is thinking very carefully about each letter before it encrypts it.

1. Add `import time` to the very top of your code.

2. At the end of your loop type `time.sleep(0.2)`

This will make the computer wait for 0.2 seconds before it gets the next letter.
Run your code and see what happens. Try other numbers like `0.4` or `1`

If

Part 5: Dealing With Spaces

Task 5.1: Testing With A Space!

1. Test your code with 2 words now instead on one, what happens?

You should get a ValueError, this is because the program doesn't know what a space is!

Task 5.2: What If?

To fix this issue, we are going to add some `if` statements to check if the character is in the alphabet variable.

1. In your loop add a line to check if the current letter is in the alphabet
2. If the character is in the alphabet, we want to make it secret with our code - indent your code into the `if` statement

Task 5.3: Ignore the spaces

1. Create an `else` statement to handle when characters are not in the alphabet.
2. Inside the else statement, `print` the character you are on, but don't encrypt it. Make sure they stay on the same line too.

Hint

An `else` always needs to be attached to an `if` statement

```
if raining == True:
      print ('oh no!')
else:
      print ('Yay!')
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 6:

- ☐ You have an `if` and `else` statement
- ☐ If your code can make more than one word secret
- ☐ Print out a secret sentence

Part 6: Let's Get Cracking!

Task 6.1: Make a Secret Code!

Before we can crack some codes, we need a message to decrypt.

Use your program to get a secret message that we can decrypt (write down your key!!) Fill in the table below so that we can decrypt it later!

Original Message	Key	Encrypted Message

Task 6.2: To encrypt or decrypt?

At the moment your program can make a secret code, but if we gave that code to someone else, they won't be able to read it!

We will need to add a mode where you can turn the secret message back into normal words using the same key that we used to make the secret message.

We can call the modes `'e'` for encrypt and `'d'` for decrypt.

1. Use an `input`() statement before the `for` loop to ask the user which mode they would like to use.

Task 6.3: Not So Secret Anymore!

If the user chooses mode `'d'` we will need to change the key value to become the opposite of the encryption key value

1. Go to just after where you get your integer version of your key.
2. On the next line add an if statement that checks what mode the user entered. We want to check if we are in decrypting mode.
3. If we are in decrypting mode we want to multiply the `key` by `-1`. Overwrite the current value with this new value.

For example: If the letters in the word were changed by 3 letters to encrypt the secret word, the letters need to be changed back by 3 letters.

***Hint** Operators in python*

Operators in python

*	Multiplication
/	Division
+	Add
-	Minus

☑ CHECKPOINT ☑

- ☐ Encrypts messages
- ☐ Decrypts messages with correct key
- ☐ Prints original message

7. Extension: Random Keys?

So far we have had to choose a number to shift the letters by. Now, we want to let the computer choose what number to use by typing ‘random’ when you ask how many letters to rotate by.

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 secret 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!

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 secret messages, we want to add a while loop near the top of the program.

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

Hint

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.

Remember while loops look like this:

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

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

Task 8.2: Add new line

Did you notice that `Do you want to encrypt or decrypt` doesn't appear on a new line when it runs in a loop? Look at the text in red in the below box.

```
Welcome to the Caesar Cipher!
Do you want to encrypt or decrypt (e/d): e
What is your message: Hello
How many letters to rotate by: 4
ippsoDo you want to encrypt or decrypt (e/d):
```

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

Hint

We want it to appear like this:

```
Welcome to the Caesar Cipher!
Do you want to encrypt or decrypt (e/d): e
What is your message: Hello
How many letters to rotate by: 4
ippso
Do you want to encrypt or decrypt (e/d):
```

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 a blank line 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`. We do this by writing `break` inside our `if` statement.
4. Don't forget to indent your `break`.

9. Extension: Reading Files

If you want to send and receive secret messages from your friends you'd want to use a file to store them. We are going to get your program to read the file and encrypt your message, without you having to type it!

Task 9.1: Save your file.

1. Download the file message.txt from http://bit.ly/gpn-crypto1
2. Make sure you save it in the same folder as your python file.

Task 9.1: 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 9.2: 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
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 message and see if it's what was in the file

Hint

To open and read a line from a file you can do this:

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

Here we also used strip to remove the new line from the end of the message.

Task 8.1: Encode your message!

You have a message in your variable again. Your code should just work when you run it now!

1. See if it decrypts the message from the file.

10. Extension: Writing Files

Now we want to send a secret message to a friend, we need to put our message in a file!

Task 10.1: Storing your secret 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 `""`.

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`.

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

Hint

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

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

Task 10.2: Putting it in a file

Now we need to make a file and put our message in!

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 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!

3. Run your code and see if a file is created with your encrypted message in it!