Girls' Programming Network

Cryptography

TUTORS ONLY

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!!

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 to make our secret message.

This is what the alphabet would look like if we shifted it left by 3 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
↓ ↓
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

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

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

We call this a 'key' of 3 as we have shifted the alphabet by 3 letters.

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

Encrypting

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

Decrypting

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

GPN → JSQ

FRGH → CODE

Task 0.1: Encrypting and decrypting messages

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

Encrypt		Decrypt	
SECRET →	V H F U H W	FUDFN →	C R A C K
CIPHER →	F L S K H U	FDHVDU →	C A E S A R

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 try it!**

Task 0.2: Encrypt with a key of 10!

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

MYSTERY → W I C D O B I

Hint : remember you are **encrypting** this message so start with the green letter on the outside wheel and replace it with the matching purple letter on the inside wheel.

Task 0.3: Encrypt with a key of 24!

Let's try a **key of 24**. Rotate your inner cipher wheel 24 spots to the left. **Encrypt** this message (you can ignore spaces):

GPN IS GREAT → E N L G Q E P C Y R

Task 0.4: Decrypt with a key of 7!

Try a **key of 7**. Rotate your inner cypher wheel 7 spots to the left. **Decrypt** this message:

JYFWAVNYHWOF → C R Y P T O G R A P H Y

Hint : remember you are **decrypting** this message so start with the purple letter on the inner wheel and replace it with the matching green letter on the outer wheel.

Task 0.5: Decrypt with a key of 11!

Try a **key of 11**. Rotate your inner cipher wheel 11 spots to the left. **Decrypt** this message:

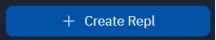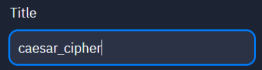DATY ESP HSPPW → S P I N T H E W H E E L

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 1:
- ☐ You encrypted all the messages!
- ☐ You decrypted all the secret messages!

Part 1: Setting up

Task 1.1: Making a python file

1. Go to https://replit.com/
2. Sign up or log in (we recommend signing up with Google if you have a Google account)
3. Click **+ Create Repl** in the top left hand corner to create a new Project.
4. Use the Python template
   Templates
   Python replit
5. Name your Project 'caesar_cipher'
   Title
   caesar_cipher

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

A main.py file will have been created for you!

At the top of the file use a comment to write your name!
Any line starting with # is a comment.
Comments are ignored by the computer but they help humans understand the code!

```
# This is a comment
```

Task 1.3: Run your code!

Run your code using the ▶ Run button at the top. It won't do anything yet! You can also use CTRL & ENTER to run.

TUTOR TIPS

- Make sure they used the Python template
- Make sure they are using main.py file

☑ CHECKPOINT ☑

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

☐ You should have a Project called caesar_cipher and a program file called main.py

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

☐ Run your file and it does nothing!!

Part 2: Tell me your message

Task 2.1: Welcome

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

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

2. Run your code to check it prints out a welcome to the user.
   In Replit, your code prints to the Console window.

Hint

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

Task 2.2: What is your message?

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

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 for your message?

1. Use `input` to ask for the key number to use for your message. Store it in a variable called `key`.
2. Use `int` to turn `key` from a string into an integer so we can use it in our code!
3. Run your code to check you can input a key number.

For example, it might look like this:

```
What is the key number?
```

Hint

If you wanted to ask someone their age and turn it into an integer it would look like this:
```
age = input("How old are you? ")
age = int(age)
```
OR you can do it in one line like this
```
age = int(input("How old are you? "))
```

Using this same pattern you could ask for the key number!
Remember we have to use `int` to convert the input to a number to use in our code!

Task 2.4: Print your message

1. Let's `print` our message that we got from the user earlier.
2. Run your code to check that the message that prints is what the user input.
3. Remove the print line - we don't need it now that we know what is in the variable `message` is correct.

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 3:
☐ Print a welcome
☐ Ask for message
☐ Ask for a key to use for your message
☐ Convert key to integer format
☐ Print your message
☐ Run your code
☐ Remove print message

The code should look like this (no bonuses):

```
# <the student's name>
print("Welcome, this is the Caesar cipher")
message = input("What is the message? ")
key = input("What is the key number? ")
key = int(key)
# remove print line below when they've verified message variable correct
print(message)
```

★ **BONUS 2.5: What's the name of your cipher language?**

Waiting for the next lecture? Try adding this bonus feature!!
Invent a fantastic name for your cipher language and `print` it.

★ **BONUS 2.6: Customize welcome**

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

Let's customise the welcome for the user.
For example, ask the user their name and then `print` a personalised welcome for them.

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

TUTOR TIPS

The code should look like this (with bonuses):

```
# <the student's name>
print("Welcome, this is the DAZZLING DONNA Caesar cipher")
name = input("What is your name? ")
print("Welcome to my amazing cipher, " + name)
message = input("What is the message? ")
key = input("What is the key number? ")
key = int(key)
```

Part 3: Getting a secret letter

Task 3.1 How do we get a secret letter?

How do we encrypt the letters in our message?

Let's try doing this with pen and paper! Use your Workbook Activity Sheet to write on

1. Choose a word you want to encrypt (eg 'ape'). We will use **5 as the key number.**

2. Write the first letter of the word in the first column of the table.

3. Count how far through the alphabet that first letter is. Computers start counting at 0, so you should count like a computer does and start at 0 for A.You can use your cipher wheels to help count or see *Hint* below. Write this number in the second column.

4. Put 5 in the third column as we are using 5 as our key number.

5. Add the numbers in columns 2 and 3 and write this in column 4.

6. Now count through the alphabet to find what letter is at the number in column 4. Write it in the last column. Remember again that computers start counting from 0 so you should too! Congratulations - you've encrypted the first letter in your word!

7. Try this again on the next line with the second letter of your word. Keep going with letters in your word until you've got the idea of how we are using the position (index) of letters in the alphabet and the key number to find out how to encrypt a letter.

8. You can see from the table below, that when the word 'ape' is encrypted using a key number of 5 it becomes 'fuj'.

9. In the next steps, we'll write code to do this for us!

Letter	Letter Index	Key Number	Secret Letter Index	Secret Letter
a	0	5	5	f
p	15	5	20	u
e	4	5	9	j

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

You might also like to write these numbers under the letters on the inner (purple) circle of your wheel. You can use them to quickly turn your purple wheel for a particular key. For example : if key = 6, look for 6 on the purple wheel and turn the purple wheel so 6 lines up with A on the green wheel. This saves you 'counting'. Try it and see that it gives you the same result as counting as you rotate the purple wheel.

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 the user entered in a variable called `current_letter`. *See the Hint below for help.*

2. To test your code, add a line to your code to print `current_letter`.

3. Run your code entering the Messages in the table below (you can use any key number) and check that the letter that prints for `current_letter` is the first letter of the Message entered.

Message	current_letter
gpn	g
pizza	p
zoink	z

Remember that you can get the first letter of a string like this:
```
name = "gpn"
first_letter = name[0]

print(first_letter) would print 'g'
```

Remember, you stored the message the user entered in a variable called `message`

Task 3.4: Find that letter!

Next we need to figure out where in the alphabet that letter is! This is a tricky step, so take time to read it through carefully.

1. **Make a variable** called `current_index` and set this to the position in the `alphabet` of `current_letter`. *See the Hint below for help.*

2. To test your code, add a line to your code to print `current_index`.

3. Run your code entering the Messages in the table below (you can use any key number) and check that the number that prints for `current_index` is what is listed in the table (this is the position of the first letter in the alphabet).

Message	current_letter	current_index
gpn	g	6
pizza	p	15
zoink	z	25

Hint

You can use this to find the position of the letter "p" (the index) in a string called word.
```
word = "gpn"
p_index = word.index("p")
```

Remember, you are trying to find position of whatever is stored in `current_letter` in a string called `alphabet`

Task 3.5: Turn the key!

So now we've got the position of the original letter, we use the shift key to get the position in the alphabet of the new secret letter (and later we'll use this to get the new secret letter itself).

The position (index) of the new secret letter will be `current_index` **plus `key`.**

1. Make a variable called `new_index` and **set this to `current_index + key`** to get the position in the alphabet of the secret letter.

2. To test your code, add a line to your code to print `new_index`.

3. Run your code entering the Messages in the table below and use 5 as the key. Check that the number that prints for `new_index` is what is listed in the table.

Message	current_letter	current_index	key	new_index
gpn	g	6	5	11
pizza	p	15	5	20
zoink	z	25	5	30

Can you see that we are writing code to encrypt the first letter of the input message like we encrypted the letters on paper in Task 3.1?

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

1. Make a variable called `new_letter` and set it to the letter in the alphabet that is at the `new_index` position in the alphabet. *See the Hint below for help.*

2. To test your code, add a line to your code to print `new_letter`.

3. Run your code entering the Messages in the table below and use 5 as the key. Check that the letter that prints for `new_letter` is what is listed in the table.

Message	current_letter	current_index	key	new_index	new_letter
gpn	g	6	5	11	l
pizza	p	15	5	20	u
zoink	z	25	5	30	oops!

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]
```

`print(letter)` would display 'e' - remember we start counting at 0

Remember, you are trying to find the letter in the `alphabet` string at position `new_index`

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

    You get an error!

    This is because when you try to go 5 letters past 'z' in the alphabet, there isn't a letter!

    To fix this we need to make the `new_index` variable wrap around back to 0 when it gets to the end of the alphabet.

    We can use the **modulo operator (%)** in our code to do this! Modulo is a maths operation that gives the remainder of a division.

    For example
    - 10 % 8 = 2 (10 divided by 8 is 1 with remainder 2)
    - 20 % 7 = 6 (20 divided by 7 is 2 with remainder 6)

    Think about what happens when we use modulo 26 (number of letters in alphabet)
    - 22 % 26 = 22 (22 divided by 26 is 0 with remainder 22)
    - 30 % 26 = 4 (30 divided by 26 is 1 with remainder 4)

    Using **modulo 26** in our code will wrap any `new_index` value > 25 back to the start of the alphabet.

2. Add a line to your code after you calculate `new_index` so that `new_index` becomes the modulo 26 version of itself. *See the Hint below for help.*

3. Run your code using 5 as the key. Check that the letter that prints for `new_letter` is correct when you enter each Message below.

Message	current_letter	current_index	key	new_index	new_letter
gpn	g	6	5	11	l
pizza	p	15	5	20	u
zoink	z	25	5	4	e

4. Use your cipher wheel to check the `current_letter` is getting converted to the correct `new_letter` in the above table. Remember a key of 5 means you rotate your inner cipher wheel 5 to the left.

To modulo something by 5 you do this:
```
number = 21
number = number % 5
```

Task 3.8: Remove print lines

1. In previous steps we added print lines to test our code. We can remove this code now as we know it's working properly and it's making our output look messy! Using print lines like we did is a good way to test your code and find errors, but make sure you remove them from your final code.

2. Remove the code lines that you added in 3.3, 3.4 and 3.5 that print `current_letter, current_index and new_index`. You can do this by making the whole line into a comment - *See Hint below.*

3. **DO NOT REMOVE the code line you added in 3.6 that prints `new_letter`**

Hint

Remember:
`# this is a comment`

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 4:
☐ You have printed and checked `current_letter, current_index, new_index and new_letter` in the examples
☐ You have used the cipher wheel to check `new_letter` in the examples
☐ You have removed the print lines for `current_letter, current_index, new_index`

This is a tricky piece of code.
Make sure that they have done Part 3.1 with their own word on the Workbook Activity
Sheet - this will help them understand what's happening.
The code should look like this (no bonuses):

```python
# <the student's name>
alphabet = 'abcdefghijklmnopqrstuvwxyz'
print("Welcome, this is the Caesar cipher")
message = input("What is the message? ")
key = input("What is the key number? ")
key = int(key)

current_letter = message[0]
current_index = alphabet.index(current_letter)
new_index = current_index + key
new_index = new_index % 26
new_letter = alphabet[new_index]

# comment out following 3 lines when they have verified correct
# print(current_letter)
# print(current_index)
# print(new_index)

print(new_letter)
```

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!

TUTOR TIPS

End Part 3 - The code should look like this (with bonuses):

```python
# <the student's name>
alphabet = 'abcdefghijklmnopqrstuvwxyz'
print("Welcome, this is the DAZZLING DONNA Caesar cipher")
name = input("What is your name? ")
print("Welcome to my amazing cipher, " + name)
message = input("What is the message? ")
message = message.lower()
key = input("What is the key number? ")
key = int(key)

current_letter = message[0]
current_index = alphabet.index(current_letter)
new_index = current_index + key
new_index = new_index % 26
new_letter = alphabet[new_index]

# print(current_letter)
# print(current_index)
# print(new_index)

print(new_letter)
```

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 = _k t r_

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? NO!

Right now your code is only encoding the first letter of the message variable. The computer doesn't know that there are more letters in the 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 a **for** loop right under the comment you just made.
   Your **for** loop should loop through each letter in the variable called `message`.
   Use the `current_letter` variable to hold each letter as you loop through.

2. Indent everything below your for loop so it's repeated for each letter. *Hint : you can do this by selecting all the lines below the for loop and hitting TAB ->*

Hint

Here's an example of a **for** loop that loops through each letter in the variable called 'word' and uses the variable called 'letter' to hold each letter as we loop through.

```
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', using a key of 6. Remember a key of 6 means you rotate your inner cipher wheel 6 to the left.

<p style="text-align:center;">g p n → m v t</p>

2. Run your code and enter that word and key = 6 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.

2. Find the line in your code where you print `new_letter`. Change this line so that the program will always print `new_letter` on the same line.

Hint

Use the `print(variable, end='')` to tell the program that you don't want the next print on a new line.

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 4:
☐ Used a comment to remove line of code
☐ Added a for loop
☐ Check your code with a 3 letter word
☐ Print the word on one line

The code should look like this (no bonuses):

```python
# <the student's name>
alphabet = 'abcdefghijklmnopqrstuvwxyz'
print("Welcome, this is the Caesar cipher")
message = input("What is the message? ")
key = input("What is the key number? ")
key = int(key)

# current_letter = message[0]
for current_letter in message:
    current_index = alphabet.index(current_letter)
    new_index = current_index + key
    new_index = new_index % 26
    new_letter = alphabet[new_index]
    # print(current_letter)
    # print(current_index)
    # print(new_index)
    print(new_letter, end='')
```

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 add `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`

BEGINNER TUTOR TIPS

End Part 4 - The code should look like this (with bonuses):

```python
# <the student's name>
import time

alphabet = 'abcdefghijklmnopqrstuvwxyz'
print("Welcome, this is the DAZZLING DONNA Caesar cipher")
name = input("What is your name? ")
print("Welcome to my amazing cipher, " + name)
message = input("What is the message? ")
message = message.lower()
key = input("What is the key number? ")
key = int(key)
# current_letter = message[0]
for current_letter in message:
    current_index = alphabet.index(current_letter)
    new_index = current_index + key
    new_index = new_index % 26
    new_letter = alphabet[new_index]
    # print(current_letter)
    # print(current_index)
    # print(new_index)
    print(new_letter, end='')
    time.sleep(0.2)
```

Part 5: Dealing With Spaces

Task 5.1: Testing With A Space!

1. Test your code with 2 words separated by a space. What happens?

You get a ValueError, this is because the program doesn't know how to encrypt a space!

Task 5.2: What If?

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

1. At the start of your loop, add a line to check if `current_letter` is in the alphabet

2. If the character is in the alphabet, encrypt the character with the existing code (make sure you indent your existing code under 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 everything still prints out on one line.

3. Try running your code with a message of more than one word separated by spaces

4. Try running your code with more than one word separated by any character that isn't a letter. Does it still work? It should encrypt the letters and keep the non-letter characters! This makes your encoded message look even more strange!

Hint

An `else` always needs an `if` statement beforehand

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

TUTOR TIPS

Some students may forget the colon after the else as well.
Make sure they also print in the else, otherwise those characters will be skipped.

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

☐ You have an `if` and `else` statement

☐ Your code can make more than one word secret

☐ Print out a secret sentence

TUTOR TIPS

The code should look like this (no bonuses):

```python
# <the student's name>
alphabet = 'abcdefghijklmnopqrstuvwxyz'
print("Welcome, this is the Caesar cipher")
message = input("What is the message? ")
key = input("What is the key number? ")
key = int(key)

# current_letter = message[0]
for current_letter in message:
    if current_letter in alphabet:
        current_index = alphabet.index(current_letter)
        new_index = current_index + key
        new_index = new_index % 26
        new_letter = alphabet[new_index]
        # print(current_letter)
        # print(current_index)
        # print(new_index)
        print(new_letter, end='')
    else:
        print(current_letter, end='')
```

Part 6: Let's Get Cracking!

Task 6.1: Make a Secret Code!

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

Use your program to get an encrypted message that we can decrypt (don't forget to write down the key you used!!)
Fill in the table below on your Workbook Activity Sheet so that we can decrypt it later!

Original Message	Key	Encrypted Message
example	10	o h k w z v o
test case	15	i t h i r p h t
Get the kids to do their own test. These are just some examples		

Task 6.2: To encrypt or decrypt?

At the moment your program can make an encrypted message, but if we gave that message and the key to someone else, they wouldn't be able to decrypt it with the code they've written.

We will need to add a mode where you can either encrypt or decrypt.

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

1. Use an `input`() statement after your welcome message to ask the user what 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 (negative) of the encryption key value.

This is because to decrypt a message we need to shift the alphabet the opposite way to what we did to encrypt the message.

1. Go to the input statement where the user enters the 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 of `key` with this new value.
    * is the multiplication operator in Python. For example x = y * 2

    If the letters in the word were changed by 3 letters to encrypt, the letters need to be changed back by 3 letters to decrypt.

4. Run your program choosing decryption mode to decrypt the encrypted messages you wrote down in 6.1 above. Does your program decrypt them to the correct original message?

5. Find a friend that has got to this step. Trade encrypted messages and keys to decrypt each others' messages!

<div style="text-align center">☑ CHECKPOINT ☑</div>

☐ Your code encrypts messages correctly when you choose 'e' mode

☐ Your code decrypts messages correctly when you choose 'd' mode and have the correct key

The code should look like this (no bonuses):

```
# <the student's name>
alphabet = 'abcdefghijklmnopqrstuvwxyz'
print("Welcome, this is the Caesar cipher")
message = input("What is the message? ")
key = input("What is the key number? ")
key = int(key)
mode = input("Do you want to encrypt or decrypt? (e or d) ")
if mode == 'd':
      key = key * -1

# current_letter = message[0]
for current_letter in message:
   if current_letter in alphabet:
      current_index = alphabet.index(current_letter)
      new_index = current_index + key
      new_index = new_index % 26
      new_letter = alphabet[new_index]
      # print(current_letter)
      # print(current_index)
      # print(new_index)
      print(new_letter, end='')
   else:
      print(current_letter, end='')
```