Girls’ Programming Network

Cryptography

1

TUTORS ONLY

Create a Caesar Cipher encryptor and decryptor!

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

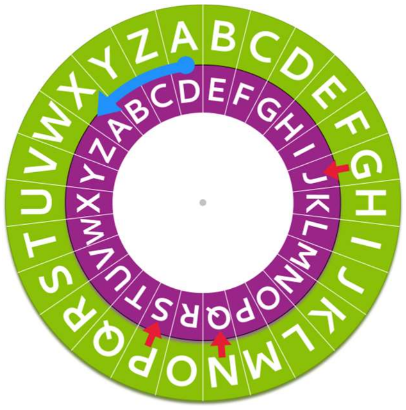
We call this a 'key' of 3 as we have shifted the alphabet by 3 letters.

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:

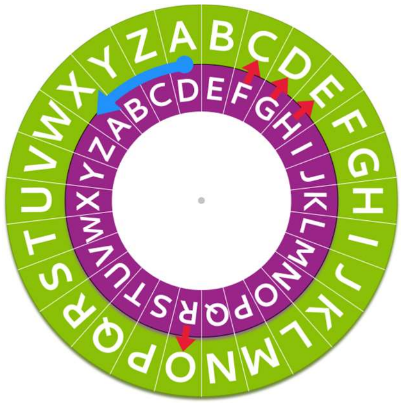
GPN → JSQ



Decrypting

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

FRGH → CODE



Task 0.1: Encrypting and decrypting messages

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

Encrypt

SECRET → V H F U H W

CIPHER → F L S K H U

Decrypt

FUDFN → C R A C K

FDHVDU → C A E S A R

Cipher Wheels

To encrypt a message, rotate the **inner** wheel anti-clockwise (to the left)
When encrypting read the wheel from outside in AKA the outer wheel contains our original letter, the inner wheel contains our encrypted letter

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 → WICDOBI

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 → ENL GQ EPCYR

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 → CRYPTOGRAPHY

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 → SPIN THE WHEEL

☑ 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 2: Tell me your message

Task 2.1: Welcome

1. Print: Welcome, this is the Caesar Cipher

Solution

```
# <the student's name>
print("Welcome, this is the Caesar cipher")
```

Task 2.2: What is your message?

1. Write an input statement asking for message to be encrypted

Solution

```
message = input("What is the message? ")
```

TUTOR TIPS

Some students may try and print "What’s your name", and use single quotes without escaping the apostrophe. Tell them to use double quotes instead.

Incorrect: print('What's your name?')

Correct: print("What's your name")

Task 2.3: What is the key for your message?

1. Write an input statement asking the user for the encryption ‘key’
2. Store the input in a variable called key
3. Set the variable to be type int

Answer

```
key = input("What is the key number? ")
key = int(key)
```

OR you can do it in one line like this

```
key = int(input("What is the key number? "))
```

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.

Solution

```
# remove print line below when they’ve verified message variable correct
print(message)
```

☑ 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

Full code Lesson 2

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)
```

Bonuses: Full code Lesson 2

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)
```

String s Ints

Part 3: Getting a secret letter

Task 3.1 How do we get a secret letter?

Students can practice encoding following the instruction in Lesson 3.1
Here is a worked example demonstrating the variables we will be thinking about

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

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

Students might 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 to 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.

Task 3.2: Let’s store the alphabet!

1. Store the alphabet in a variable as a string

Solution

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
```

Task 3.3: Get a Letter!

1. **store the first letter** of the message the user entered in a variable called `current_letter.`
2. print `current_letter` to test
3. Run the examples below to check it's working

Message	current_letter
gpn	g
pizza	p
zoink	z

Solution

```
current_letter = message[0]

Remember, index starts at 0
```

Task 3.4: Find that letter!

Find the index of each letter in your message

1. **Make a variable** called `current_index` and set this to the position in the `alphabet` of `current_letter.`

2. print `current_index` for testing

4. Run the examples below to check it's working

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

Solution

```
current_index = alphabet.index(current_letter)
```

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

Task 3.5: Turn the key!

Shift the index to get the encrypted letter
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. Print `new_index` for testing
5. Run the examples below to check it's working

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

Solution

```
new_index = current_index + key

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

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` and set it to the letter in the alphabet that is at the `new_index` position in the alphabet.
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!

You should get an error for Zoink

Solution

```
new_letter = alphabet[new_index]
```

Remember

- you are trying to find the letter in the `alphabet` string at position `new_index`
- Index starts at 0

Task 3.7: Zoink!

Zoink throws an error 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.

The **modulo operator (%)** returns 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.

Add a line to your code after you calculate `new_index` so that `new_index` becomes the modulo 26 version of itself

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

Solution

```
new_index = new_index % 26
```

Make sure this is placed **above** the line
`new_letter = alphabet[new_index]`

Task 3.8: Remove print lines

Cleanup code by commenting out or removing the extra print statements for `current_letter, current_index and new_index`

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

☑ 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`

Full Code Lesson 3

Make sure they have a go at 3.1 on paper if they are confused about the logic of encryption

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)

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)
```

★ BONUS 3.8: zoink, ZoInK and ZOINK

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.

Bonuses: Full Code Lesson 3

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

```
# <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)
```

For

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!

We will need to loop through our message to encode each letter

Task 4.2: Not just one letter

1. Comment out the line that sets the `current_letter` variable to the first letter of your message.

Solution

```
# current_letter = message[0]
```

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 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(new_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.

   gpn → mvt

2. Run your code and enter that word and key = 6 and see if they match.

Task 4.5: Everybody get in line!

1. Try your program with a long word like ‘zooperdooper’. Every letter prints on its own line, making it hard to read

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.

Solution

```
print(new_letter, end='').
```

☑ 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

Full Code Lesson 4

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)

# 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='')
```

★ BONUS 4.7: 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 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`

Bonuses: Full Code Lesson 4

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

```
# <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?

1. At the start of your loop, add an if statement to check that `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)

Solution

```
for current_letter in message:
   if current_letter in alphabet:
```

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.

Hint

```
else:
      print(current_letter, end='')
```

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.

☑ CHECKPOINT ☑

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

Full Code Lesson 5

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)

# 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!

Use the worksheet to write down some encrypted messages. We will use these later to test that our decryption is working

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.

Solution

```
mode = input("Do you want to encrypt or decrypt? (e or d) ")
if mode == 'd':
```

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.

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.

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?

☑ CHECKPOINT ☑

□ 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

Full Code Lesson 6

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='')
```

Extension 7 Full code

Random keys

```
# <the student's name>
import random

alphabet = 'abcdefghijklmnopqrstuvwxyz'
print("Welcome, this is the Caesar cipher")
message = input("What is the message? ")
key = input("What is the key number? ")
if key == "random":
    key = random.randrange(1,26)
    print("The key is: " + str(key))
else:
    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='')
```

Extension 8 Full Code

While loop

```
# <the student's name>
import random

alphabet = 'abcdefghijklmnopqrstuvwxyz'
print("Welcome, this is the Caesar cipher")

while True:

   message = input("What is the message? ")
   if message == "":
              break

   key = input("What is the key number? ")
   if key == "random":
       key = random.randrange(1,26)
       print("The key is: " + str(key))
   else:
       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='')
    print(“”)
```

Extension 9 Full Code

Writing files

```
# <the student's name>
import random

alphabet = 'abcdefghijklmnopqrstuvwxyz'
print("Welcome, this is the Caesar cipher")
message = input("What is the message? ")
key = input("What is the key number? ")
if key == "random":
    key = random.randrange(1,26)
    print("The key is: " + str(key))
else:
    key = int(key)

mode = input("Do you want to encrypt or decrypt? (e or d) ")
if mode == 'd':
      key = key * -1

encrypted_message = ""
# 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='')
      encrypted_message = encrypted_message + new_letter
   else:
      # print(current_letter, end='')
      encrypted_message = encrypted_message + current_letter

# print(“”)
print(encrypted_message)
with open('output.txt', 'w') as f:
     f.write(encrypted_message)
```

Extension 9 Full Code

Reading files

```
# <the student's name>
import random
with open('caesar1.txt') as f:
    message = f.read()
    message = message.strip()

alphabet = 'abcdefghijklmnopqrstuvwxyz'
print("Welcome, this is the Caesar cipher")
# message = input("What is the message? ")
key = input("What is the key number? ")
if key == "random":
    key = random.randrange(1,26)
    print("The key is: " + str(key))
else:
    key = int(key)

mode = input("Do you want to encrypt or decrypt? (e or d) ")
if mode == 'd':
      key = key * -1

# encrypted_message = ""
# 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='')
      # encrypted_message = encrypted_message + new_letter
   else:
      print(current_letter, end='')
      # encrypted_message = encrypted_message + current_letter


# print(“”)
# print(encrypted_message)
# with open('output.txt', 'w') as f:
#      f.write(encrypted_message)
```