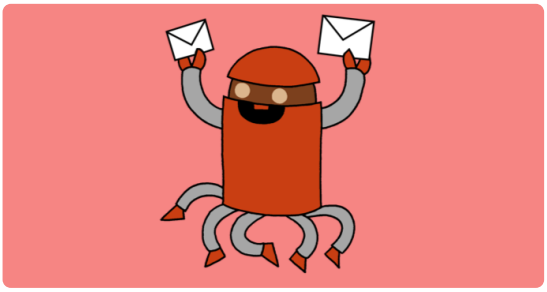
Secret Messages

Make your own encryption program, to send and receive secret messages.

Python



Step 1 Introduction:

In this project, you'll learn how to make your own encryption program, to send and receive secret messages with a friend. This project ties in with the "Earth to Principia" activity on page 16 of the Space Diary.

```
Please enter a message:  hello!
Enter a key (1-26):  5
Your new message is:  mjqqt!
```

```
Please enter a message:  mjqqt!
Enter a key (1-26):  -5
Your new message is:  hello!
```

Additional information for club leaders

If you need to print this project, please use the **Printer friendly version** (https://projects.raspberrypi.org/en/projects/secret-messages/print).

Club leader notes

Introduction:

In this project, children will learn how to make an encryption program, to send and receive secret messages with a friend. This project introduces iteration (looping) over a text string.

Online Resources

This project uses Python 3. We recommend using **trinket** (https://trinket.io/) to write Python online. This project contains the following Trinkets:

- **New (blank) Python Trinket – jumpto.cc/python-new** (http://jumpto.cc/python-new)

There is also a trinket containing the finished project:

- **'Secret Messages' Finished – trinket.io/python/402256078c** (https://trinket.io/python/402256078c)
- **'Friendship Calculator' Finished – trinket.io/python/2e852cd687** (https://trinket.io/python/2e852cd687)

Offline Resources

This project can be **completed offline** (https://www.codeclubprojects.org/en-GB/resources/python-working-offline/) if preferred.

You can find the completed project in the 'Volunteer Resources' section, which contains:

- messages-finished/messages.py
- messages-finished/friends.py

(All of the resources above are also downloadable as project and volunteer `.zip` files.)

Learning Objectives

- Iteration (looping) over a string variable;
- The `find()` method;
- The modulus operator (`%`).

This project covers elements from the following strands of the **Raspberry Pi Digital Making Curriculum** (http://rpf.io/curriculum):

- **Combine programming constructs to solve a problem.** (https://www.raspberrypi.org/curriculum/programming/builder)

Challenges

- Use a Caesar cipher - encrypy and decrypt letters and words manually;
- Variable keys - allowing the user to input a chosen key;
- Encrypting and decrypting messages - encrypting and decrypting whole messages;
- Friendship calculator - applying text iteration to a new problem.

Frequently Asked Questions

- When searching using `find()` or `if char in alphabet:`, note that searches are case-sensitive. Children can use:

  ```
  message = input("Please enter a message to encrypt: ").lower()
  ```

  to make the input lower case before searching.

Project materials

Project resources

- **.zip file containing all project resources** (https://projects-static.raspberrypi.org/projects/secret-messages/6d74c6a4d82a6f5550e8de16e0b1852311ebbd31/en/resources/secret-messages-project-resources.zip)
- **Online blank Python Trinket** (http://jumpto.cc/python-new)
- **Offline blank Python file** (https://projects-static.raspberrypi.org/projects/secret-messages/6d74c6a4d82a6f5550e8de16e0b1852311ebbd31/en/resources/new-new.py)

Club leader resources

- **.zip file containing all completed project resources** (https://projects-static.raspberrypi.org/projects/secret-messages/6d74c6a4d82a6f5550e8de16e0b1852311ebbd31/en/resources/secret-messages-volunteer-resources.zip)
- **Online completed Trinket project** (https://trinket.io/python/402256078c)
- **secret-messages-finished/messages.py** (https://projects-static.raspberrypi.org/projects/secret-messages/6d74c6a4d82a6f5550e8de16e0b1852311ebbd31/en/resources/secret-messages-finished-messages.py)
- **Online completed 'Friendship calculator' challenge** (https://trinket.io/python/2e852cd687)
- **offline complete 'Friendship calculator' challenge** (https://projects-static.raspberrypi.org/projects/secret-messages/6d74c6a4d82a6f5550e8de16e0b1852311ebbd31/en/resources/friendship-calculator-finished-friends.py)

Step 2 The Caesar cipher

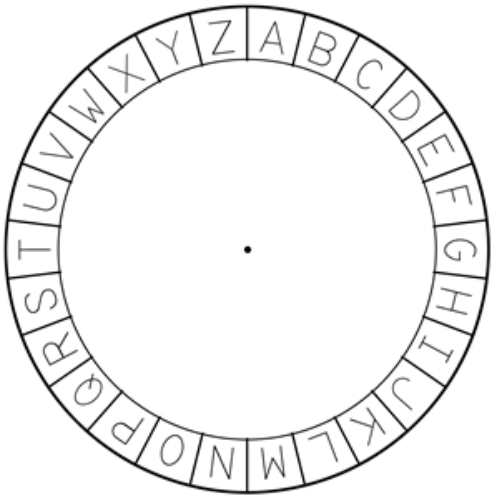
A cipher is a type of secret code, where you swap the letters around so that no-one can read your message.

You'll be using one of the oldest and most famous ciphers, the **Caesar cipher**, which is named after Julius Caesar.

Before we start coding, let's try using the Caesar cipher to hide a word.

- Hiding a word is called **encryption**.

  Let's start by encrypting the letter 'a'. To do this, we can draw the alphabet in a circle, like this:



- To make a secret encrypted letter from a normal one, you need to have a secret key. Let's use the number 3 as the key (but you can use any number you like).

  To **encrypt** the letter 'a', you just move 3 letters clockwise, which will give you the letter 'd':

- You can use what you've learnt to encrypt an entire word. For example, 'hello' encrypted is 'khoor'. Try it yourself.

  - h + 3 = **k**
  - e + 3 = **h**
  - l + 3 = **o**
  - l + 3 = **o**
  - o + 3 = **r**
- Getting text back to normal is called **decryption**. To decrypt a word, just subtract the key instead of adding it:

  - k - 3 = **h**
  - h - 3 = **e**
  - o - 3 = **l**
  - o - 3 = **l**
  - r - 3 = **o**

Step 3 Challenge: Use a Caesar cipher

Can you send a secret word to a friend? You'll both need to agree on a secret key before you start.

You could even send entire sentences to each other!

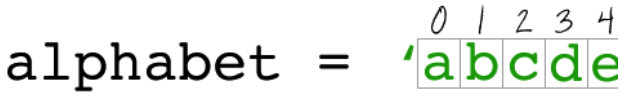
Step 4 Encrypting letters

Let's write a Python program to encrypt a single character.

- Open the blank Python template Trinket: **jumpto.cc/python-new** (http://jumpto.cc/python-new).
- Instead of drawing the alphabet in a circle, let's write it out as an `alphabet` variable.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
```

- Each letter of the alphabet has a position, starting at position 0. So the letter 'a' is at position 0 of the alphabet, and 'c' is at position 2.



- You can get a letter from your `alphabet` variable by writing the position in square brackets.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
print(alphabet[0])
print(alphabet[6])
print(alphabet[19])
```

```
a
g
t
```

You can delete the `print` statements once you've tried this out.

- Next, you'll need to store the secret `key` in a variable.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3
```

- Next, ask the user for a single letter (called a `character`) to encrypt.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3

character = input('Please enter a character: ')
```

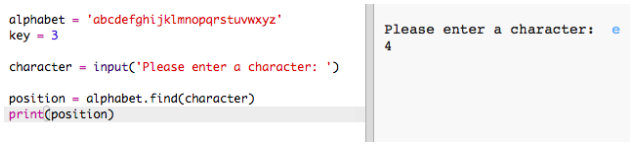
- Find the `position` of the `character`.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3

character = input('Please enter a character: ')

position = alphabet.find(character)
```

- You can test the stored `position` by printing it. For example, that character 'e' is at position 4 in the alphabet.



- To encrypt the `character`, you should add the `key` to the `position`. This is then stored in a `newPosition` variable.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3

character = input('Please enter a character: ')

position = alphabet.find(character)
print(position)

newPosition = position + key
```

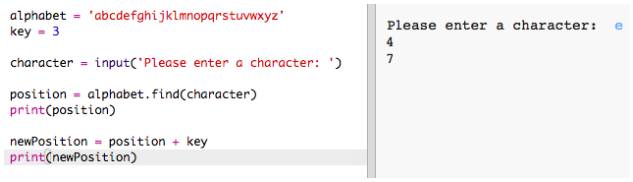
- Add code to print the new character position.

```
position = alphabet.find(character)
print(position)

newPosition = position + key
print(newPosition)
```

- Test out your new code. As your `key` is 3, it should add 3 to the `position` and store it in your `newPosition` variable.

  For example, letter 'e' is at position 4. To encrypt, you add the `key` (3), giving 7.



- What happens when you try and encrypt the letter 'y'?

```
Please enter a character: y
24
27
```

  Notice how the `newPosition` is 27, and there aren't 27 letters in the alphabet!

- You can use a `%` to tell the new position to go back to position 0 once it gets to position 26.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3

character = input('Please enter a character: ')

position = alphabet.find(character)
print(position)

newPosition = (position + key) % 26
print(newPosition)
```

```
Please enter a character:  y
24
1
```

- Finally, you want to print the letter at the new position.

  For example, adding the key to the letter 'e' gives 7, and the letter at position 7 of the alphabet is 'h'.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3

character = input('Please enter a character: ')

position = alphabet.find(character)
print(position)

newPosition = (position + key) % 26
print(newPosition)

newCharacter = alphabet[newPosition]
print(newCharacter)
```

```
Please enter a character:  e
4
7
h
```

- Try out your code. You can also remove some of your print statements, just printing the new character at the end.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3

character = input('Please enter a character: ')

position = alphabet.find(character)

newPosition = (position + key) % 26

newCharacter = alphabet[newPosition]
print('The new character is:', newCharacter)
```

```
Please enter a character:  e
The new character is: h
```

Step 5 Challenge: Variable keys

Modify your program, so that the user can enter their own key to use. You'll need to get the user's input, and store it in the `key` variable.

Remember to use the `int()` function to convert the input to a whole number.

You can then use a negative key to decrypt messages!

Step 6 Encrypting entire messages

Instead of just encrypting and decrypting messages one character at a time, let's change the program to encrypt entire messages!

- Firstly, check that your code looks like this:

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3

character = input('Please enter a character: ')

position = alphabet.find(character)
newPosition = (position + key) % 26
newCharacter = alphabet[newPosition]
print('The new character is:', newCharacter)
```

- Create a variable to store the new encrypted message.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3
newMessage = ''

character = input('Please enter a character: ')

position = alphabet.find(character)
newPosition = (position + key) % 26
newCharacter = alphabet[newPosition]
print('The new character is:', newCharacter)
```

- Change your code to store the user's message and not just one character.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3
newMessage = ''

message = input('Please enter a message: ')

position = alphabet.find(character)
newPosition = (position + key) % 26
newCharacter = alphabet[newPosition]
print('The new character is: ', newCharacter)
```

- Add a `for` loop to your code, and indent the rest of the code so that it is repeated for each character in the message.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3
newMessage = ''

message = input('Please enter a message: ')

for character in message:
  position = alphabet.find(character)
  newPosition = (position + key) % 26
  newCharacter = alphabet[newPosition]
  print('The new character is: ', newCharacter)
```

- Test your code. You should see that each character in the message is encrypted and printed one at a time.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3
newMessage = ''

message = input('Please enter a message: ')

for character in message:
  position = alphabet.find(character)
  newPosition = (position + key) % 26
  newCharacter = alphabet[newPosition]
  print('The new character is: ', newCharacter)
```

```
Please enter a message:  hello
The new character is: k
The new character is: h
The new character is: o
The new character is: o
The new character is: r
```

- Let's add each encrypted character to your `newMessage` variable.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3
newMessage = ''

message = input('Please enter a message: ')

for character in message:
  position = alphabet.find(character)
  newPosition = (position + key) % 26
  newCharacter = alphabet[newPosition]
  print('The new character is:', newCharacter)
  newMessage += newCharacter
```

- You can `print` the `newMessage` as it is being encrypted.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3
newMessage = ''

message = input('Please enter a message: ')

for character in message:
  position = alphabet.find(character)
  newPosition = (position + key) % 26
  newCharacter = alphabet[newPosition]
  print('The new character is:', newCharacter)
  newMessage += newCharacter
  print(newMessage)
```

```
Please enter a message:  hello
The new character is: k
k
The new character is: h
kh
The new character is: o
kho
The new character is: o
khoo
The new character is: r
khoor
```

- If you delete the spaces before the `print` statement, the encrypted message will only be displayed once at the end. You can also delete the code for printing the character positions.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key = 3
newMessage = ''

message = input('Please enter a message: ')

for character in message:
  position = alphabet.find(character)
  newPosition = (position + key) % 26
  newCharacter = alphabet[newPosition]
  #print('The new character is:', newCharacter)
  newMessage += newCharacter

print('Your new message is', newMessage)
```

```
Please enter a message:  hello
Your new message is khoor
```

Step 7 Extra characters

Some characters aren't in the alphabet, which causes an error.

- Test out your code with some characters that aren't in the alphabet.

  For example, you could use the message `hi there!!`.

```
Please enter a message:  hi there!!
Your new message is:  klcwkhuhcc
```

  Notice that the space and the `!` characters are all encrypted as the letter 'c'!

- To fix this, you only want to translate a character if it's in the alphabet. To do this, add an `if` statement to your code, and indent the rest of your code.

```
for character in message:
  if character in alphabet:
    position = alphabet.find(character)
    newPosition = (position + key) % 26
    newCharacter = alphabet[newPosition]
    newMessage += newCharacter
```

- Test your code with the same message. What happens this time?

```
Please enter your message:  hi there!!
Your new message is: klwkhuh
```

  Now, your code just skips any character if it's not in the alphabet.

- It would be better if your code didn't encrypt anything not in the alphabet, but just used the original character.

  Add an `else` statement to your code, which just adds the original character to the encrypted message.

```
for character in message:
  if character in alphabet:
    position = alphabet.find(character)
    newPosition = (position + key) % 26
    newCharacter = alphabet[newPosition]
    newMessage += newCharacter
  else:
    newMessage += character
```

- Test your code. You should see that any character in the alphabet is encrypted, but any other characters are left alone!

```
Please enter a message:  hi there!!
Your new message is:  kl wkhuh!!
```

Step 8 Challenge: Encrypting and decrypting messages

Encrypt some messages, and give them to a friend along with the secret key. See if they can decrypt them using their program!

You could also duplicate the project and create a separate program for decrypting messages.

Step 9 Challenge: Friendship calculator

Write a program to show how compatible 2 people are, by calculating a friendship score.

```
enter tha names of 2 people:  zenon & dave
your friendship score is : 70
```

The program could loop through each of the characters in the 2 names, and add points to a `score` variable each time certain letters are found.

You should decide on rules for awarding points. For example, you could award points for vowels, or characters that are found in the word “friend”:

```
if character in 'aeiou':
  score += 5
if character in 'friend':
  score += 10
```

You could also give the user a personalised message, based on their score:

```
if score > 100:
  print('best friends!')
```
