

# The Legend of the Python Pirate

*Following a recent discovery on-board the 'SS Great Britain' ship (located near the @Bristol science centre), various papers were found that indicate there is a hidden treasure belonging to the Python Pirate.*

*Can you help solve the mystery behind the Python Pirate?...*

## THE BRISTOL PAPER

[www.pihardware.com](http://www.pihardware.com)

BRISTOL'S FAVOURITE NEWSPAPER

- Since 2012

### DISCOVERY ON THE SS GREAT BRITAIN!

The **SS Great Britain**, designed by the great engineer Isambard Kingdom Brunel in 1843.



When it was built it was one of the largest passenger ships in the world, and was the first iron "steamer" ship with a submerged propeller.

The SS Great Britain saw short service as a luxury passenger ship from 1843 to 1846.

In 1937, the SS Great Britain was scrapped and sunk, later to be recovered in 1970.

Now located in Bristol Docks where it was first constructed, the ship has been lovingly restored to its former glory.

During some routine maintenance, a hidden panel was discovered in what would have been the captain's quarters.

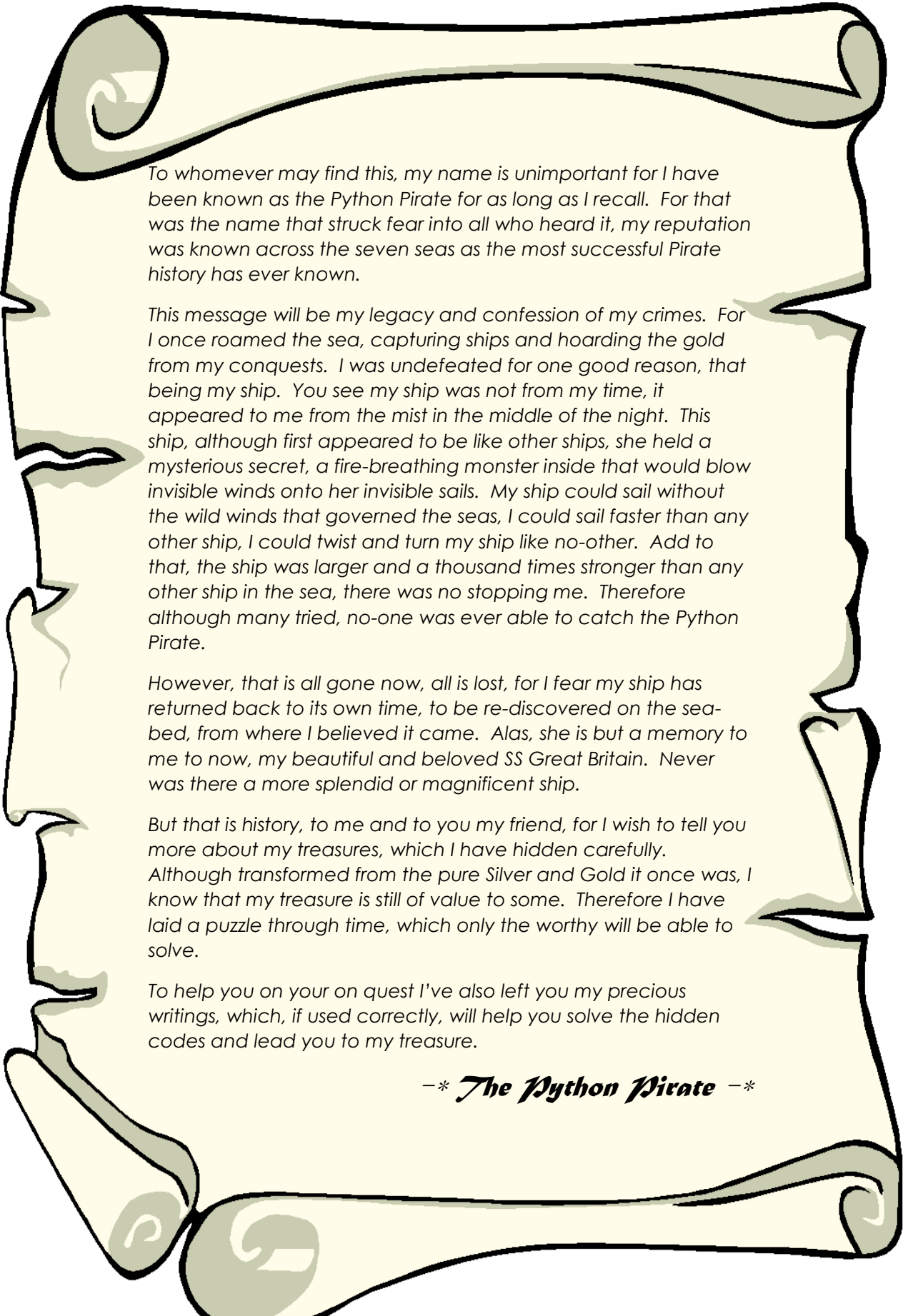
There inside lay some tightly wrapped parchments, their origin unknown. The papers have been dated to be from over a hundred years prior to the construction of the ship.

Although the documents have been studied in great

detail, their exact meaning have been unclear. However they have revived the old legend that talks of a notorious pirate simply known as the Python Pirate and his long lost treasure.

See page 2





To whomever may find this, my name is unimportant for I have been known as the Python Pirate for as long as I recall. For that was the name that struck fear into all who heard it, my reputation was known across the seven seas as the most successful Pirate history has ever known.

This message will be my legacy and confession of my crimes. For I once roamed the sea, capturing ships and hoarding the gold from my conquests. I was undefeated for one good reason, that being my ship. You see my ship was not from my time, it appeared to me from the mist in the middle of the night. This ship, although first appeared to be like other ships, she held a mysterious secret, a fire-breathing monster inside that would blow invisible winds onto her invisible sails. My ship could sail without the wild winds that governed the seas, I could sail faster than any other ship, I could twist and turn my ship like no-other. Add to that, the ship was larger and a thousand times stronger than any other ship in the sea, there was no stopping me. Therefore although many tried, no-one was ever able to catch the Python Pirate.

However, that is all gone now, all is lost, for I fear my ship has returned back to its own time, to be re-discovered on the sea-bed, from where I believed it came. Alas, she is but a memory to me to now, my beautiful and beloved SS Great Britain. Never was there a more splendid or magnificent ship.

But that is history, to me and to you my friend, for I wish to tell you more about my treasures, which I have hidden carefully. Although transformed from the pure Silver and Gold it once was, I know that my treasure is still of value to some. Therefore I have laid a puzzle through time, which only the worthy will be able to solve.

To help you on your on quest I've also left you my precious writings, which, if used correctly, will help you solve the hidden codes and lead you to my treasure.

—\* *The Python Pirate* —\*

# The Legend of the Python Pirate – Part 1

With the letter we found a number of strange notes, they appear to be written in some kind of language, which until recently had not been invented... Oddly it is only now we can start to make sense of these previously bizarre writings.

The first, it seems, allows us to translate to and from a special code, which we can only assume must be the codes which the Python Pirate mentioned in his letter.

---

## How to write your Python Script:

You can write your python scripts using the Raspbian desktop, using a program called Idle3. To start the Raspbian desktop from the command line interface type:

startx

Click on the IDLE 3 icon (since we are using Python 3):



Press **Ctrl+N**, to create a new file and ensure you save using the same filenames as used in the examples: i.e. `piratecode.py`

You can now write your programs in the IDLE 3 editor and run your scripts by pressing **F5** (if you haven't saved it by this point it will prompt you to enter a filename and save before it runs).

NOTE: You can also write the python scripts using just the command line interface using a program called `nano`. To start type:

```
cd ~ ← This ensure you are in your "home" directory
nano -c pythoncode.py
```

Save and exit by pressing **Ctrl + X, Y**, and **Enter**. To run your program, type:

```
sudo python3 piratecode.py ← You will need to use "sudo"
                              when interfacing with hardware
                              - which the later scripts do.
```

You can now create a small test program...just to check everything is working!

## hello.py

```
#!/usr/bin/python3
#piratecode.py
print("Ahoy there Matey!")
```

Remember press **Ctrl+N** to create a new file, when you have typed it in, press **F5** to run it.  
You can save the file as **hello.py**.  
You will see:  
Ahoy there Matey!

## piratecode.py

```
#!/usr/bin/python3
#piratecode.py
```

```
letter2code = {' ': '/',
               'N': '-*',
               'E': '* ',
               'S': '***',
               'W': '*--'}
```

Create a dictionary to convert letters into \* and - codes.  
\* is a short flash of the lighthouse  
- is a long flash

Reverse the dictionary so we can convert a code back into letters.

```
code2letter = dict((v,k) for k,v in letter2code.items())
```

```
ON,OFF=True,False
```

```
code2signal = {'*': [ON,OFF],
               '-': [ON,ON,ON,ON,OFF],
               ' ': [OFF],
               '/': [OFF]*3}
```

Create a dictionary to convert the code into logical ON/OFF signals.

```
def GetLetter(code):
    try:
        return code2letter[code]
    except KeyError:
        print("Error: Code not found (" ,code, ")")
        return "?"
```

Use the dictionary to convert a code into a letter or return ?

```
def GetCode(letter):
    """ """
    try:
        return letter2code[letter]
    except KeyError:
        print("Error: Letter not found (" ,letter, ")")
        return "?"
```

Use the dictionary to convert a letter into a code or return ?

```
def GetCodedMessage(message):
    codedmessage=""
    for letter in message.upper():
        codedmessage += (GetCode(letter))
        codedmessage += (" ")
    return codedmessage
```

Convert a message i.e. "NE" into a coded one "-\* \*".

```
def GetMessage(codedmessage):
    message=""
    codelist = codedmessage.split(" ")
    for code in codelist:
        message += (GetLetter(code))
    return message
```

Decode a coded message i.e. "-\* \*" -> "NE"

```
def GetSignal(codedmessage):
    """Generate a signal from a codedmessage"""
    signal=[]
    for code in codedmessage:
        try:
            signal += code2signal[code]
        except KeyError:
            print("Error: Code not found")
            return []
    signal += code2signal[" "]
    return signal
```

Generate a signal from a codedmessage.  
i.e. "-\*" -> "---- \_ \_"

## How to test your Python Script

Once we have written our script we can test it!

Add the following code to the bottom of your script and run it (**F5**).

```
if __name__ == "__main__":
```

```
    print(GetCode('N'))
    print(GetCode('E'))
    print(GetCode('S'))
    print(GetCode('W'))
    print(GetCode(' '))
```

The following code is only run when the file is run directly, so we can put any test code here.

First test GetCode() which uses the letter2code dictionary. You should see:

```
-*
 *
***
*--
/
```

If the above test works, continue to add more tests (otherwise - check your code for mistakes):

```
    print(GetLetter('*'))
    print(GetLetter('-*'))
    print(GetLetter('-**'))
```

```
E
N
Error: Code not found ( -* )
?
```

Note: When testing we should always test that errors are dealt with nicely.

```
    print(GetCodedMessage("NN EE SS WW"))
    print(GetCodedMessage("WEST"))
```

```
    print(GetMessage("*-- * ***"))
    print(GetMessage("*-- **-- -*"))
```

```
-* -* / * * / *** ** / *-- *--
Error: Letter not found ( T )
*-- * *** ?
```

Note: There isn't a code for T so it should report an error.

```
    print(GetSignal("*-- **-- -*"))
```

```
[True, False, True, True, True, True,
False, True, True, True, True, False,
False, True, False, True, False, True,
True, True, True, False, True, True,
True, True, False, False, True, True,
True, True, False, True, False, False]
```

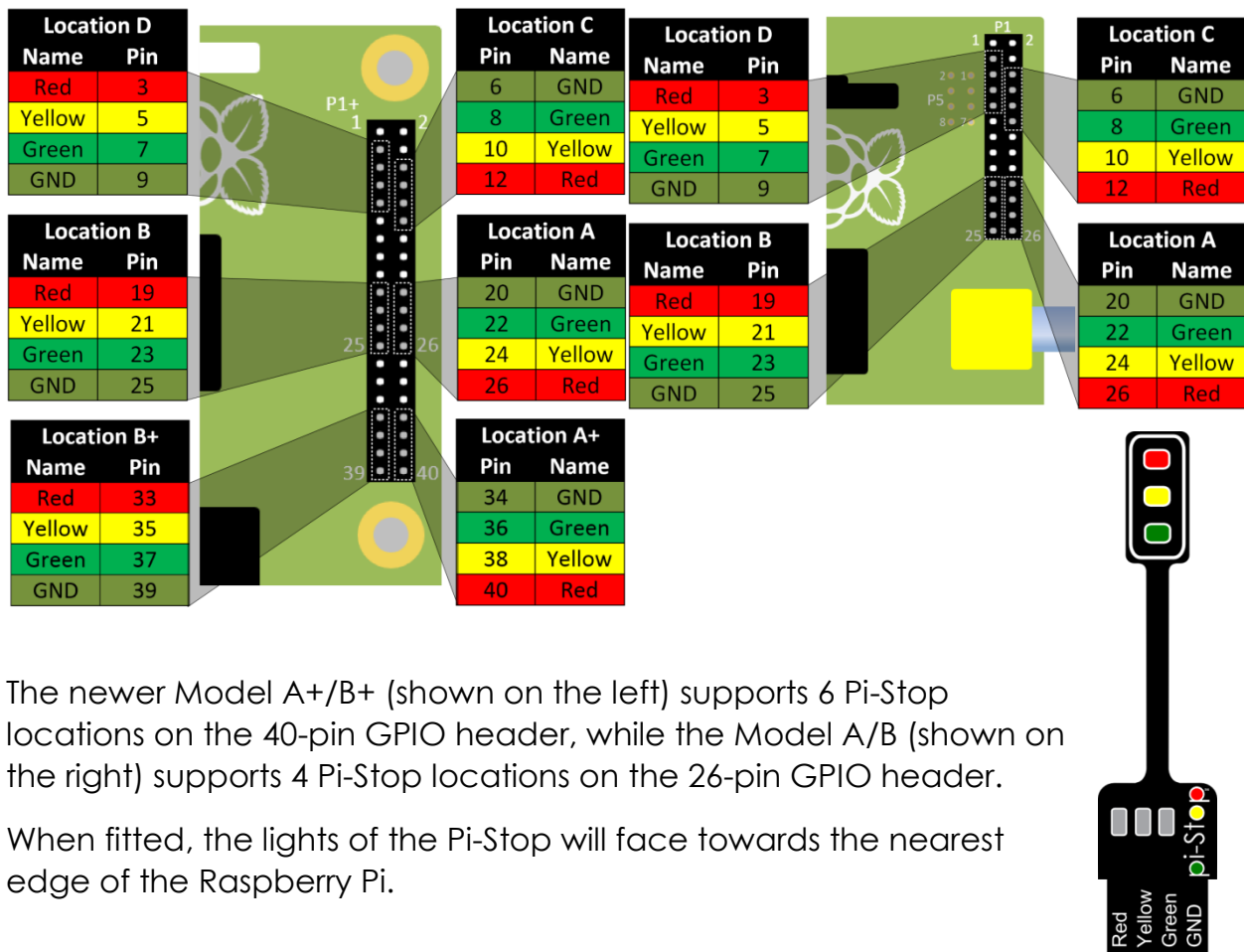
```
WES
Error: Code not found ( **-- )
W?N
```

## The Legend of the Python Pirate – Part 2

The next note, appears to control a lighthouse, producing a series of short and long flashes in some kind of code sequence. You can use a lighthouse (a Pi-Stop add-on) connected to the Raspberry Pi to find out what this script does.

### How to fit the Pi-Stop (the lighthouse) to your Raspberry Pi:

You will need to fit the Pi-Stop to the Raspberry Pi in “Location A”. You will need to know which Model of the Raspberry Pi you have (Model A+/B+ or Model A/B).



The newer Model A+/B+ (shown on the left) supports 6 Pi-Stop locations on the 40-pin GPIO header, while the Model A/B (shown on the right) supports 4 Pi-Stop locations on the 26-pin GPIO header.

When fitted, the lights of the Pi-Stop will face towards the nearest edge of the Raspberry Pi.

**Note:** If you have trouble finding Location A on the Model A+/B+, you can adjust the script to use Location A+.

Use `LED=[40,38,36]` this will set pins 40, 38 and 36 to control the lights.

## lighthouse.py

```
#!/usr/bin/python3
#lighthouse.py
import time
import RPi.GPIO as GPIO
```

Used to interface with hardware on the Raspberry Pi.

```
#HARDWARE SETUP
# Model P1
LED=[26,24,22] #RED, YELLOW, GREEN
# Model+ P1 (uncomment next line to use)
#LED=[40,38,36] #RED, YELLOW, GREEN
```

Sets the pins used for the Pi-Stop:  
HARDWARE SETUP  
Model P1  
2[=====XGYR]26  
1[=====]25  
Or:  
Model+ P1  
2[=====XGYR]40  
1[=====]39

```
FLASH_TIME=0.3
RED,YELLOW,GREEN=0,1,2
ON,OFF=True,False
```

Define some values to use later in the script.

```
def GPIOsetup():
    GPIO.setmode(GPIO.BOARD)
    for led in (RED,YELLOW,GREEN):
        GPIO.setup(LED[led],GPIO.OUT)
```

Use GPIO.BOARD so we can call the pins by their physical position in P1. Set each of the pins as outputs.

```
def ShowSignal(state):
    for led in (RED,YELLOW,GREEN):
        if (state[led]):
            print("# ",end="")
        else:
            print(" ",end="")
    print("")
```

We can use this function to display the state the Pi-Stop LEDs will be set to. i.e. " "=OFF "#=ON

```
def ControlLights(state):
    for led in (RED,YELLOW,GREEN):
        GPIO.output(LED[led],state[led])
    time.sleep(FLASH_TIME)
```

Control the Pi-Stop LEDs by setting them to the required states (and wait). i.e. TRUE=ON FALSE=OFF

```
def SendLighthouseSignal(signalRED,signalYELLOW,signalGREEN):
    numItems=[]
    allsignals=[signalRED,signalYELLOW,signalGREEN]
```

This function can display 3 different signals at the same time.

```
    for signal in allsignals:
        numItems.append(len(signal))
```

Find the length of each signal.

```
    for i in range(max(numItems)):
        state=[]
        for led in (RED,YELLOW,GREEN):
            try:
                state.append(allsignals[led][i])
            except IndexError:
                state.append(OFF)
        ShowSignal(state)
        ControlLights(state)
```

Take the first item of each signal and make into a list of states to set each LED:  
=> [ON, OFF, ON]  
Continue until we have processed all the items in the longest signal (using OFF for completed signals).

```
GPIOsetup()
```

Display the states on screen before controlling the Pi-Stop LEDs.

We always want to setup the pins, so we can do this here.



## Testing lighthouse.py

Add the following tests to the bottom of your script:

As before, these tests will only run when this file is run directly.

```
if __name__ == "__main__":
```

```
    import piratecode as CODE
```

This will use the previous script piratecode.py, so make sure your script is located within the same directory.

```
    codedmessage=CODE.GetCodedMessage("W W W W W W")
    print(codedmessage)
```

Displays the code for:

"W W W W W W"

You will see:

\*-- / \*-- / \*-- / \*-- / \*-- / \*--

```
    signal1=CODE.GetSignal(codedmessage)
    codedmessage=CODE.GetCodedMessage("W W W S S S")
    signal2=CODE.GetSignal(codedmessage)
    codedmessage=CODE.GetCodedMessage("w W W N N N N")
    signal3=CODE.GetSignal(codedmessage)
```

Create different signals for the Pi-Stop to display.

```
    SendLighthouseSignal(signal1,signal2,signal3)
```

Display the signals on the screen and on the Pi-Stop. You will see:

###

###

###

###

###

###

###

###

###

...etc...

The Pi-Stop lights will match the # characters. The lights will start the same "W" and then show different signals i.e. "S" and "N".

```
    codedmessage=CODE.GetCodedMessage("WWL")
    signal4=CODE.GetSignal(codedmessage)
    SendLighthouseSignal(signal4,signal4,signal4)
```

Check what happens if we use a letter not available?  
You will see:  
Error: Letter not found ( L )  
Error: Code not found



# The Legend of the Python Pirate – Part 3

## piratepirate.py

```
#!/usr/bin/python3
#pythonpirate.py

import piratecode as CODE
import lighthouse as LIGHT

def ReadCodedMessage():
    while(1):
        codedmessage = input("Enter Signal:")
        print(codedmessage)
        if not all(x in ["-", "*", " ", "/"] for x in codedmessage):
            print("Use: -=Long flash *=Short flash")
        else:
            return codedmessage

def PlayCodedMessage(codedmessage):
    signal2Play=CODE.GetSignal(codedmessage)
    LIGHT.SendLighthouseSignal(signal2Play,signal2Play,signal2Play)

def PlayAgain(codedmessage):
    while(1):
        answer = input("Play again? Y/N:")
        if (answer.upper().startswith("Y")):
            PlayCodedMessage(codedmessage)
        elif (answer.upper() == "N"):
            return

def PlayMessage(message):
    codedmessage = CODE.GetCodedMessage(message)
    print(codedmessage)
    PlayCodedMessage(codedmessage)

def DecodeCodedMessage(codedmessage):
    message = CODE.GetMessage(codedmessage)
    return message

def main():
    codedmessage = ReadCodedMessage()
    PlayCodedMessage(codedmessage)
    PlayAgain(codedmessage)
    message = DecodeCodedMessage(codedmessage)
    if (message=="?"):
        print("Unknown code - try again!")
    else:
        print ("Message: ", message)

if __name__ == "__main__":
    while(1):
        main()
```

This will use the previous scripts  
piratecode.py and lighthouse.py.

Prompt for a coded message:  
i.e. `-- *** */ --`  
Also check all characters are valid.

Convert a coded message  
into a signal and  
display on the Pi-Stop.

Prompt to play  
the signal again  
or not.

Transmit a message  
i.e. "N E S W"  
As a coded signal:  
`-- * *** --`  
Display it on the Pi-Stop.

Convert a coded signal  
into the message.  
i.e. `-- * = N`

The sequence for finding  
the treasure...  
>Enter the signal: `--*`  
>Playback on the Pi-Stop  
>Play again?  
>Show the message... N  
Go to the North Lighthouse,  
and enter the signal from  
the light there...

## Find the Pirate Treasure and the Code for the Lock

Now we have the Python Pirate's script, we are ready to find his treasure. We have located an island which matches the map the Python Pirate left with his papers, on the island are four ancient lighthouses which appear to flash random patterns!

*Perhaps they are the key to his treasure! But where should we start?...*

Look very carefully at his letter... can you spot anything next to his name at the bottom?

*This is our first clue!*

Run the `pythonpirate.py` script and enter the code you have found.

If entered correctly, this will re-create the signal on your Pi-Stop lighthouse and give you a message "N"=North, "E"=East, "S"=South or "W"=West.

Starting at this first lighthouse, watch the light signal carefully and enter this into your Python Pirate program (\*=short flash, -=long flash).

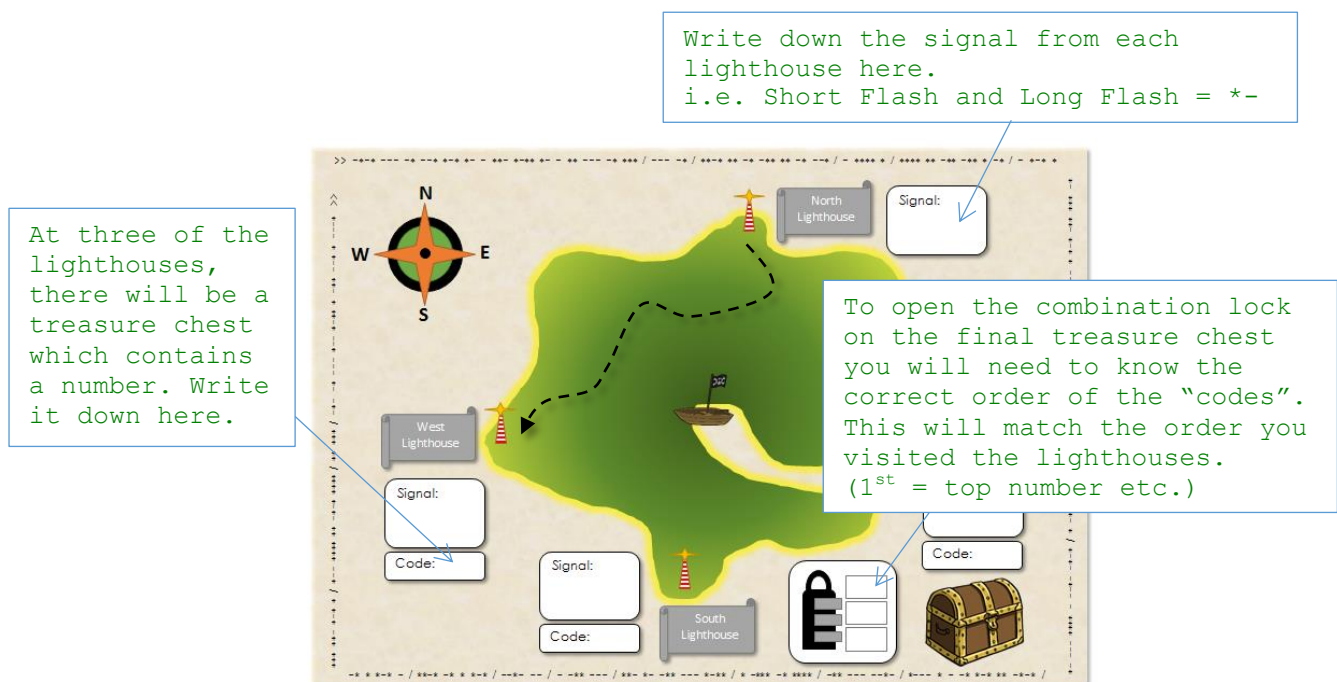
Check that your lighthouse signal matches the one you are watching.

If entered correctly, you will get another message to tell you which lighthouse you should visit next.

Proceed to the next lighthouse depending on the message discovered. Ensure you check the treasure chests at each of the three lighthouses and write down the number inside in the spaces provided on the map.

## The treasure map:

You will need a map to keep track of your journey and the codes you discover:



# The Legend of the Python Pirate – Part 4

## The Python Pirate's final secret!

Inside the last treasure chest we discovered the Python Pirate's treasure, and also one final puzzle. The last paper contains the full key to the Python Pirate's code.

```
letter2code = {' ': '/', '.': ' ', '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': '-***'}
```

This is in fact standard “Morse Code” which was developed in 1836, by Samuel Morse, Joseph Henry and Alfred Vail, as a means to transmit messages electronically. This was long before the voice telephone was invented (in 1876), however it was still used for radio communications until as recently as 1999.

Morse Code is also used by ships as a simple way to signal to each other (without needing any special equipment to receive a message) and to send messages to shore, by using special lights with shutters on, or by using mirrors to reflect the sun.

Perhaps the Python Pirate used these codes to communicate with other pirates and with allies on the land. No wonder no one could catch him even when he had to come into dock, since he would always know if it was a trap!

## Using Morse Code:

See if you can make use of this code, perhaps try sending your own secret messages or use it to read a message from another group.

1. Make a copy of the “`piratecode.py`” script and save it as “`morsecode.py`”.
2. In this new file replace the `letter2code` dictionary with the new version (shown above).
3. Make a copy of the “`pythonpirate.py`” script and save it as “`pythonmorse.py`”.
4. In this new file replace “`import piratecode as CODE`” with “`import morsecode as CODE`” so we can use the new codes!
5. To send your own messages, you can use the “`PlayMessage()`” function:

```
if __name__ == "__main__":  
    PlayMessage("H e l l o")
```

You may find that you need to add extra space between your letters since it can take a lot of practice to be able to identify where one letter ends and another starts.

6. To read messages, it is recommended you write down each \*/- as you see them and get the message repeated until you are sure it is correct. You can use the original “`main()`” function to enter, playback and decode the message. Or you can use the “`DecodeCodedMessage()`” function directly:

```
print(DecodeCodedMessage("***** * *-** *-** ---"))
```

## What next?

Try to adapt the program to allow you to enter messages to send and to help decode any messages you receive. Perhaps change the “`main()`” function so it asks if you want to send a message or decode a message.

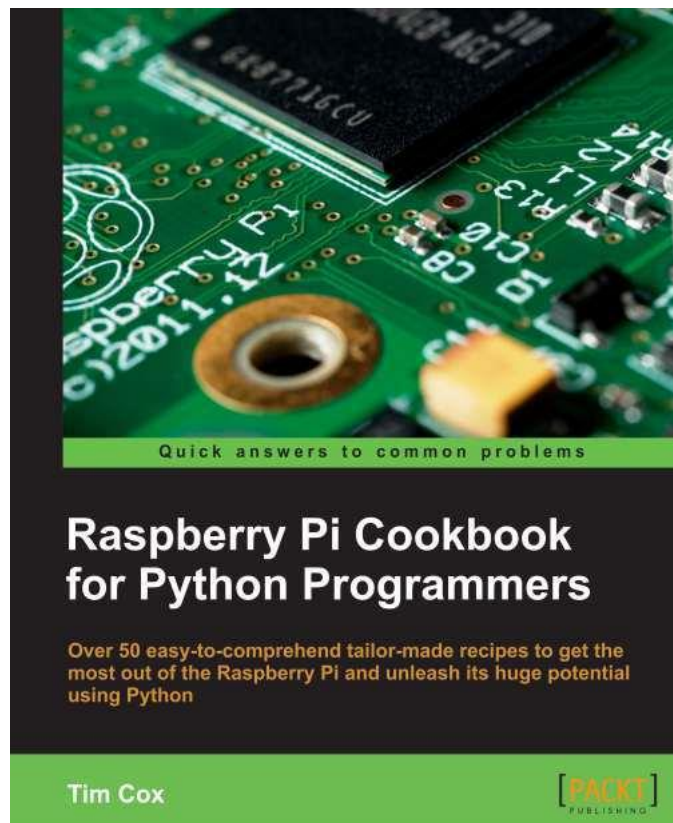
---

If you have enjoyed yourself, don't forget you can buy a Pi-Stop after the workshop and continue with the other workshops which are on my website – [www.pihardware.com](http://www.pihardware.com).

I also have other kits and materials available to use with the Raspberry Pi.

Be sure to check out my book, which will show you how to get the most out of using the Raspberry Pi:

*The book's 400+ pages cover everything from setting up the Raspberry Pi and your first “Hello World” program, right through to creating 3D worlds, interfacing with hardware, controlling the camera module and building robots.*



<http://goo.gl/dmVtsc>