Project 8: Cracking coded messages

Last session we created programs that can code or decode messages by supplying a certain **key**.

These programs worked like this:

- Take a character that is in the message, convert it into a number.
- Add the key to this number.
- Now convert this new number back into a character.
- Repeat this process for every character in the message.

In this session we are going to make an adapted version of this program that can be used to crack other messages that were made using a Caesar Cypher. Even if we don't know what key was used to encrypt it.

In order to do this we will need to give our programs the ability to create key values to try.

It should try the lowest possible key value (1) and then try to decrypt the message. If the message has been decrypted by the key value of 1 then that's great. We have cracked it. If not, it needs to increase the key value to 2 and try again.

Keep repeating this, increasing the key and trying again until eventually the code has been cracked!

How can our program know if the code has been cracked? The text that is output will change from gibberish to something that is sensible.

Our program at this stage is not able to recognise English words so it will not know if the result it has produced trying to decode the message with a certain key value makes sense. In this case we will need to tell the program with each attempt if it has been successful or not.

Here is how the **algorithm** for our adapted cracking program will work.

- Input the encrypted message.
    - Automatically make a key value to try out, then to use this key to try to decrypt each character in the message.
    - Ask the user if this attempt at decrypting the message makes sense.
    - If it makes sense, stop the loop running. If not then repeat these steps with another key.
- If the user has said the message makes sense, display the current key value, the original encrypted message and the decrypted message.

Programming Challenge: Cracker Program

Your original Caesar program should look something like this:

```
message = input("Enter the message you want to encrypt")
encrypted = ""
key = int(input("Input key (+ve number for encryption or -ve for decryption): "))
for index in range(0,len(message)):
        number = ord(message[index])
        number = number + key
        newCharacter = chr(number)
        encrypted = encrypted + newCharacter
print(encrypted)
```

You need to create a way of automatically creating a key to try out. This key value should start at 1, try to use that to crack the coded message. We can use a for loop to do this.

```
for key in range (1, 26):
```

Keep repeating this until the user says the answer makes sense.

```
    answer = input("Does this make sense? Y/N")
    if answer == "Y" :
        break
```

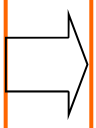
Once the user has said the program has decrypted the message it needs to stop running the loop, output the key value, the original encrypted message and the final decrypted message.

```
print("encrypted message was: " + message)
print ("decrypted message was: " + cracked)
print("key was: " + str(key))
```

Here in purple is how you would incorporate these new lines and edit the original program. Make sure you get the indentation correct. Save this as "**cracker.py**"

```
message = input("Enter the message you want to crack")

for key in range(1,26):
    cracked = ""
    for index in range(0, len(message)):
            number = ord(message[index])
            number = number - key
            newCharacter = chr(number)
            cracked = cracked + newCharacter
    print(cracked)
```



```
answer = input("Does this make sense? Y/N")
    if answer == "Y" :
        break

print("encrypted message was: " + message)
print ("decrypted message was: " + cracked)
```