Girls' Programming Network

Cryptography N

Create a Caesar and Vigenère Cipher Cracker!

TUTORS ONLY

Part 1: The English Dictionary

Full code Lesson 1

The code should look like this (no bonuses):

```
dictionary = set()
with open("dictionary.txt") as f:
    for line in f:
        dictionary.add(line.strip())
```

Part 2: Counting Words

TUTOR TIPS

Make sure students test their individual functions as they go.

Here you could get them to add lines like these to see if it counts only real english words:

```
print(count_english("i like pie"))  >> prints 3
print(count_english("i dont like glarble"))  >> prints 3 (if used "I" and "don't" would print 1
```

Full code Lesson 2

They should have a function that looks like this:

```
def count_english(text):
    words = text.split()
    counter = 0
    for word in words:
        if word in dictionary:
            counter = counter + 1
    return counter
```

Part 3: Time to crack!

TUTOR TIPS

They should have a function that looks like this:

Previously we needed to encrypt and decrypt. But now we only need to decrypt.

Students can either use their same code from before and fix to always be in decrypt mode

```
def caesar_decrypt(message, key):
    mode = "d"
    if mode == "d":
        key = -1 * key
    new_message = ""
    for current_letter in message:
        ...
```

Or they can remove the unnecessary if statement to clean up their code

```
def caesar_decrypt(message, key):
    key = key * -1
    new_message = ""
    for current_letter in message:
        ...
```

TUTOR TIPS

They should have a function that looks like this:

```
# At the top of the file
alphabet = "abcdefghijklmnopqrstuvwxyz"

...

# Anywhere in the file
def caesar_decrypt(message, key):
    key = key * -1
    new_message = ""
    for current_letter in message:
        if current_letter in alphabet:
            current_index = alphabet.index(current_letter)
            new_index = (current_index + key) % 26
            new_letter = alphabet[new_index]
            new_message = new_message + new_letter
        else:
            new_message = new_message + current_letter
    return new_message
```

Part 4: Which is the best key?

TUTOR TIPS

Looking for this logic to track the best key:

```
decrypted_text = caesar_decrypt(message, key)
count = count_english(decrypted_text)
if count > best_count:
    best_key = key
    best_count = count
```

Task 4.4: Return the best key!

TUTOR TIPS

In theory we could have kept track of the best decrypted message too, but we've separated that out just for the flow of the workbook. But students can make changes to make this into a function that returns the best decryption, rather than just the best key.

Task 4.5: Test your function!

TUTOR TIPS

Get students to use their caesar ciphers to create their own test cases.
For instance they could do this:

```
find_best_key("khoor zruog")
```

Hopefully it will return a key of 3. (It translates to hello world)

Full code Lesson 4

They should have a function that looks like this:

```
def find_best_key(message):
    best_key = 0
    best_count = 0
    for key in range(26):
        decrypted_text = caesar_decrypt(message, key)
        count = count_english(decrypted_text)
        if count > best_count:
            best_key = key
            best_count = count
    return best_key
```

Part 5: Putting it all together

Task 5.5: Calling your main function

TUTOR TIPS

The line they are most likely to forget is this one:

```
main()
```

Which will mean their code doesn't run anything, because we just have function definitions.

TUTOR TIPS

They should have a function that looks like this:

```
def main():
    with open("message.txt") as f:
        message = f.read().strip()
    key = find_best_key(message)
    decrypted_text = caesar_decrypt(message, key)
    print(decrypted_text)

main()
```

Keys (Part 6)

Filename	Keyword
vig1.txt	lamp
vig2.txt	fly
vig3.txt	up
vig4.txt	taylor
vig5.txt	hunger

Complete Code looks like:

TUTOR TIPS

The code should look like this:

```
alphabet = "abcdefghijklmnopqrstuvwxyz"

dictionary = set()
with open("dictionary.txt") as f:
    for line in f:
        dictionary.add(line.strip())

def count_english(text):
    words = text.split()
    counter = 0
    for word in words:
        if word in dictionary:
            counter = counter + 1
    return counter


def find_best_key(message):
    best_key = 0
    best_count = 0
    for key in range(26):
        decrypted_text = caesar_decrypt(message, key)
        count = count_english(decrypted_text)
        if count > best_count:
            best_key = key
            best_count = count
    return best_key


def caesar_decrypt(message, key):
    key = key * -1
    new_message = ""
    for current_letter in message:
        if current_letter in alphabet:
            current_index = alphabet.index(current_letter)
            new_index = (current_index + key) % 26
            new_letter = alphabet[new_index]
            new_message = new_message + new_letter
        else:
            new_message = new_message + current_letter
    return new_message

def main():
    with open("message.txt") as f:
        message = f.read().strip()
    key = find_best_key(message)
    decrypted_text = caesar_decrypt(message, key)
    print(decrypted_text)

main()
```

Part 6: Extension: Vigenère Cracker

Solution

```
alphabet = "abcdefghijklmnopqrstuvwxyz"

keywords = []
with open("keys.txt") as f:
    for line in f:
        keywords.append(line.strip())

dictionary = set()
with open("dictionary.txt") as f:
    for line in f:
        dictionary.add(line.strip())

def count_english(text):
    words = text.split()
    counter = 0
    for word in words:
        if word in dictionary:
            counter = counter + 1
    return counter

def find_best_key(message):
    best_key = 0
    best_count = 0
    for key in keywords:
        print(key)
        decrypted_text = vigenere_decrypt(message, key)
        count = count_english(decrypted_text)
        if count > best_count:
            best_key = key
            best_count = count
            print(decrypted_text)
    return best_key

def caesar_letter(letter, key_num):
    current_index = alphabet.index(letter)
    new_index = (current_index + key_num) % 26
    new_letter = alphabet[new_index]
    return new_letter

def vigenere_decrypt(message, key):
    key_nums = []
    for letter in key:
        key_num = alphabet.index(letter)
        key_num = -1 * key_num
        key_nums.append(key_num)

    new_message = ""
    count = 0
    for current_letter in message:
        if current_letter in alphabet:
            key_num = key_nums[count % len(key_nums)]
            new_letter = caesar_letter(current_letter, key_num)
            new_message = new_message + new_letter
            count = count + 1
        else:
            new_message = new_message + current_letter
    return new_message

def main():
    with open("message_vigenere.txt") as f:
        message = f.read().strip()

    key = find_best_key(message)
    decrypted_text = vigenere_decrypt(message, key)
    print("best", key)
    print(decrypted_text)

main()
```