Level 5

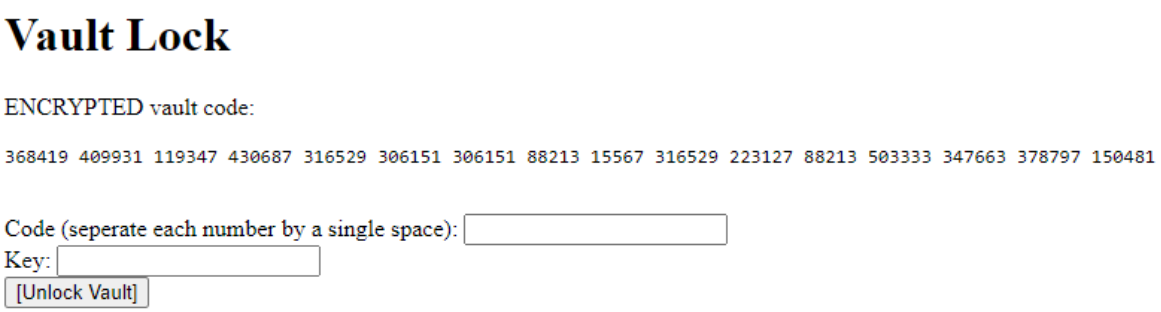
We are presented with the following code:

`app.py` file:

```
from flask import Flask, request

app = Flask(__name__)


def encrypt(vault_code, key): # vault_code is a list of integers, and key is an integer
    encrypted_vault_code = []

    for digit in vault_code:
        enc_digit = digit * key
        encrypted_vault_code.append(enc_digit)

    return encrypted_vault_code


@app.route('/')
def index():
    with open('encrypted_code.txt', mode='r') as encrypted_code:
        return f'''
        <h1>Vault Lock</h1>
        <p>ENCRYPTED vault code:<p>
        <pre>{encrypted_code.read()}</pre><br>
        <form action="/attempt-unlock" method="POST">
            <label>Code (seperate each number by a single space): <input type="text" name="code"/></label><br>
            <label>Key: <input type="text" name="key"/></label><br>
            <input type="submit" value="[Unlock Vault]"/>
        </form>
        '''

@app.route('/attempt-unlock', methods=['POST'])
def unlock():
    with open('encrypted_code.txt', mode='r') as code:
        try:
            inputted_code = request.form['code']
            inputted_code = [int(c) for c in inputted_code.split(' ')] # convert the inputted code into a list of integers

            inputted_key = int(request.form['key'])
            if inputted_key < 2:
                return '<p>The key must be 2 or greater.</p>'

            code_to_match = code.readline()

            encrypted_inputted_code = encrypt(inputted_code, inputted_key)
            encrypted_inputted_code = ' '.join([str(c) for c in encrypted_inputted_code])

            if encrypted_inputted_code == code_to_match:
                with open('flag.txt', mode='r') as flag:
                    return f'<h1>[Vault Unlocked!]</h1><p>Here is the flag:</p><pre>{flag.read()}</pre>'
            else:
                return '<h1>The vault won\'t budge...</h1>'
        except:
            return '<p>Invalid inputs. Try again.</p>'

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=False)
```

This appears to be a web-application to access some vault.

You are given the ENCRYPTED vault code, somewhere to input the actual code, and the key, in order to unlock the vault.



If we take a look at the code, we notice that the vault code you input gets encrypted using the key you also inputted. Then it's compered with the encrypted vault code above. You will be let in the vault only if they match.

```
code_to_match = code.readline()

encrypted_inputted_code = encrypt(inputted_code, inputted_key)
encrypted_inputted_code = ' '.join([str(c) for c in encrypted_inputted_code])

if encrypted_inputted_code == code_to_match:
    with open('flag.txt', mode='r') as flag:
        return f'<h1>[Vault Unlocked!]</h1><p>Here is the flag:</p><pre>{flag.read()}</pre>'
else:
    return '<h1>The vault won\'t budge...</h1>'
```

The code and key you input are encrypted using the `encrypt` function defined at the top of the code:

```
def encrypt(vault_code, key): # vault_code is a list of integers, and key is an integer
    encrypted_vault_code = []

    for digit in vault_code:
        enc_digit = digit * key
        encrypted_vault_code.append(enc_digit)

    return encrypted_vault_code
```

Let's go through this function to understand what it does:

1. The function takes as arguments both the inputted vault code and key.
2. We define an empty list called `encrypted_vault_code`.
3. For every single number (don't know why I called it digit) in the inputted vault code:
    a. Multiply it by the inputted key.

    b. Add the product to the `encrypted_vault_code` list.
4. Once the code has gone through every number in the code, return the `encrypted_vault_code` list.

So from what we can see, to encrypt a vault code with a key, you simply multiply every number in the vault code by the key.

Since the vault is encrypting your inputted code with your inputted key, and comparing it to the encrypted code above, we need to find a set of numbers (our code), that when multiplied by another number (our key), the multiplied set of numbers matches the encrypted code.

IN OTHER WORDS: we need to find a common divisor amongst all numbers in the ENCRYPTED code. This common divisor will be our key. And the code will be all the encrypted numbers divided by the key we found.

Here is a Python script that does the following:

```
encrypted_code = input('Encrypted Vault Code (split each number by a space): ')
encrypted_code = [int(c) for c in encrypted_code.split(' ')]

possible_keys = []

for key in range(1000, 9999+1):
    valid_key = True
    for number in encrypted_code:
        if number % key != 0:
            valid_key = False
            break

    if valid_key:
        possible_keys.append(key)


print('Possible keys:', possible_keys)

for key in possible_keys:
    current_valid_code = [int(i / key) for i in encrypted_code]
    print('For key:', key, ', the vault code is:', ' '.join([str(i) for i in current_valid_code]))
```

The range of keys to try is [1000, 9999] since this is the range of possible generated keys. You can see this if you read the `generate_code.py` file, though this is not mandatory for solving this problem. You could have just set the range to [1,10000] for example and the script would still work.

When you run the python script and input the encrypted code:

```
Possible keys: [5189]
For key: 5189 , the vault code is: 71 79 23 83 61 59 59 17 3 61 43 17 97 67 73 29
```

Boom! It worked!

The vault code is: `71 79 23 83 61 59 59 17 3 61 43 17 97 67 73 29`

And the key is: `5189`

NOTE: this code and key will most likely not work on your instance of the challenge since a random encrypted vault code is generated when starting the challenge.

Upon trying out what we got into the vault, it works! We got the flag.