Computing

Lesson 2: Dictionary Challenge

Programming Part 6: Dictionaries and Datafiles

Rebecca Franks

[1] *Materials from the Teach Computing Curriculum created by the National Centre for Computing Education*

Code Snippets

Creates an empty dictionary and adds the key-value pairing A : C to it

```
caesar = {}

caesar["A"] = "C"
```

Creates a dictionary that holds alphabet letters with the corresponding encrypted pair

```
caesar = {"A": "C",
          "B": "D",
          "C": "E"}

caesar["D"] = "F" # adds the key-value
pairing to the dictionary

print(caesar)
```

Code Snippets

Converts an ASCII value to its equivalent character

```
chr(66) # this will return the character B
```

Iterates through a string

```
word = "pizza"
for letter in word:
    print(letter)
```

Concatenates a string with another string value

```
letter = "a"
anotherletter = "b"
letter = letter + anotherletter # letter is assigned ab
```

Scenario

A program needs to be created that allows the user to enter a piece of plain text that will be encrypted using a user defined key.

The program should:

- Prompt for the encryption key
- Prompt for the plain text to be encrypted
- Store the plain text in uppercase
- Encrypt the plain text message
- Display the encrypted text for the user

The encryption will be based on a Caesar’s cipher. Each letter entered as plain text will be shifted forwards using the number given as the encryption key.

Scenario

Example program input and output:

```
What is the encryption key?
3
Enter your text to encrypt
Hello world
KHOOR ZRUOG
>>>
```

In this example the encryption key entered is 3. Each letter from the plain text `Hello world` is shifted over by 3 letters of the alphabet.

Scenario

The program should work in the following way:

1. Prompt the user to enter an `encryption_key` between 1 and 25.
2. Use a **function** to populate a **dictionary** that will form the Caesar cipher. The function should:
   a. Accept the `encryption_key` as an argument.
   b. Create a dictionary called `caesar`
   c. Add the space (“ “) character to the dictionary
   d. For each letter of the alphabet it should create a dictionary pair that contains the **plaintext** letter as the key and the **encrypted letter** as the **key-value** pair.
   e. If the **encrypted letter** is higher than **Z** then it should **go back 26 letters** to mimic a Caesar’s wheel.
   f. **Return** the caesar dictionary to the main program

Scenario

3. Prompt the user to enter their **plaintext** message
4. Ensure the **message** is stored in uppercase
5. Generate the **ciphertext** by taking each **letter** in **plaintext** and generating a new **encrypted letter** based on the cipher dictionary.
6. Display the **ciphertext** to the user

Task: Create your program

- Using the guidance in the scenario, create your Caesar cipher encryption program.
- Use the code snippets on page one and any other programs that you have created to support you with this activity.
- Remember to test often and break the problem down into smaller steps if they become too challenging. For example, instead of encrypting the entire message see if you can encrypt a single character.

Note: You completed a similar challenge in **Programming Part 5, Lesson 3: String Handling II** which might provide some support for this challenge.

Task: Test your program

- Test all aspects of your program by entering boundary, normal and erroneous data.
- Add any appropriate data validation checks to your program.