Case Study 1 Exercises

Exercise 1

1/1 point (graded)

In Exercise 1, we will define the alphabet used in the cipher.

The sample code imports the `string` library has been imported. Create a string called `alphabet` consisting of the space character (`' '`) followed by (concatenated with) the lowercase letters. Note that we're only using the lowercase letters in this exercise.

Sample code:

```
import string
# write your code here!
```

What is the correct way to create the `alphabet` string using the `string` library?

`alphabet = string.ascii_lowercase`

`alphabet = string.ascii_lowercase + " "`

`alphabet = " " + string.ascii_letters`

`alphabet = " " + string.ascii_lowercase`

Correct

`alphabet = string.ascii_letters + " “`

```
Code = [
          alphabet = " " + string.ascii_lowercase
]
```

Exercise 2

1/1 point (graded)

In Exercise 2, we will define a dictionary that specifies the index of each character in `alphabet`.

Note that `alphabet` is as defined in Exercise 1. Create a dictionary with keys consisting of the characters in `alphabet` and values consisting of the numbers from 0 to 26. Store this as `positions`.

What is the value of the key `n` in the `positions` dictionary?
Answer = [14]

```
Code = [
 positions = {}
index = 0
for char in alphabet:
    positions[char] = index
    index += 1

print(positions['n'])
14
]
```

Exercise 3

1/1 point (graded)

In Exercise 3, we will encode a message with a Caesar cipher.

Note that `alphabet` and `positions` are as defined in Exercises 1 and 2. Use `positions` to create an encoded message based on `message` where each character

in `message` has been shifted forward by 1 position, as defined by `positions`.

Note that you can ensure the result remains within 0-26 using `result % 27`.

Store this as `encoded_message`.

Use this code to get started:

```
message = "hi my name is caesar"
# write your code here!
```

What is `encoded_message`?

Answer = [ijanzaobnfajtadbftbs]

Code = [

```
encoding_list = []
for char in message:
    position = positions[char]
    encoded_position = (position + 1) % 27
    encoding_list.append(alphabet[encoded_position])
encoded_message = "".join(encoding_list)

print(encoded_message)
ijanzaobnfajtadbftbs

]
```

Exercise 4

1/1 point (graded)

In this Exercise 4, we will define a function that encodes a message with any given encryption key.

Use `alphabet`, `position`, and `message` as defined in Exercises 1 through 3. Define a function `encoding` that

takes a message as input as well as an int encryption key `key` to encode a message with the Caesar cipher by shifting each letter in message by `key` positions.

Your function should return a string consisting of these encoded letters.

Use `encoding` to encode `message` using `key = 3` and save the result as `encoded_message`. Print `encoded_message`.

What is the new `encoded_message`?
Answer = [klcpacqdphclvcfdhvdu]

Code = [

```
def encoding(message, key = 0):
    encoding_list = []
    for char in message:
        position = positions[char]
        encoded_position = (position + key) % 27
        encoding_list.append(alphabet[encoded_position])
    encoded_string = "".join(encoding_list)
    return encoded_string

encoded_message = encoding(message, 3)
print(encoded_message)
klcpacqdphclvcfdhvdu
]
```

Exercise 5

1/1 point (graded)

In Exercise 5, we will decode an encoded message.

Instructions

- Use `encoding` to decode `encoded_message`.
- Store your encoded message as `decoded_message`.

- Print `decoded_message`. Does this recover your original message?

What key can be used to decode the message and recover the original message shifting backwards?
Answer = [-3]

Code = [

```
decoded_message = encoding(encoded_message, -3)
print(decoded_message)
hi my name is caesar


]
```