Activity - Work with strings in Python

Introduction

Security analysts work with a lot of string data. For example, some security analysts work on creating and updating IDs such as employee IDs and device IDs, which are commonly represented as strings. As another example, certain network activity will be stored as string data. Becoming comfortable working with strings in Python is essential for the work of a security analyst.

In this lab, you'll practice creating Python code and working with strings. You'll work with an employee ID, a device ID, and a URL, all represented as string data.

Tips for completing this lab

Scenario

You're working as a security analyst, and you are responsible for writing programs in Python to automate updating employee IDs, extracting characters from a device ID, and extracting components from a URL.

Task 1

In your organization, employee IDs are currently either four digits or five digits in length. In this task, you're given a four-digit numeric employee ID stored in a variable called `employee_id`. Convert this to a string format and store the result in the same variable. Later, you'll update this employee ID string so that it complies with a new standardized format.

Complete the following code. Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
# Assign `employee_id` to a four digit number as an initial value

employee_id = 4186

# Display the data type of `employee_id`

print(type(employee_id))

# Reassign `employee_id` to the same value but in the form of a string

employee_id = str(4186)

# Display the data type of `employee_id` now

print(type(employee_id))
```

```
<class 'int'>
<class 'str'>
```

Hint 1

Hint 2

Question 1

What do you observe about the data type of `employee_id` the first time it's displayed? What do you observe about the data type of `employee_id` the second time it's displayed (after the variable is reassigned)?
[Double-click to enter your responses here.]

Task 2

Imagine that you have just been informed of a new criteria for employee IDs. They must all be five digits long for standardization purposes.

In this task, you will write a conditional statement that displays a message if the length of the employee ID is less than five digits.

Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
# Assign `employee_id` to a four digit number as an initial value

employee_id = 4186

# Reassign `employee_id` to the same value but in the form of a string

employee_id = str(employee_id)

# Conditional statement that displays a message if the length of `employee_id` is less than five digits

out =len(employee_id)

if out < 5:
   print("This employee ID has less than five digits. It does not meet length requirements.")
```

```
This employee ID has less than five digits. It does not meet length requireme
nts.
```

Hint 1

Hint 2

Hint 3

Task 3

In this task, you'll build upon the previous code. If an employee ID is only four digits, you'll use concatenation to create a five-digit employee ID number.

Concatenation is a process that allows you to merge strings together. The addition operator (`+`) in Python allows you to concatenate two strings.

Write an `if` statement that evaluates whether the length of `employee_id` is less than `5`. When the condition evaluates to `True`, reassign `employee_id` by concatenating `"E"` in front of the four-digit employee ID to create a five character employee ID. Then, display `employee_id` again. Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
# Assign `employee_id` to a four digit number as an initial value

employee_id = 4186

# Reassign `employee_id` to the same value but in the form of a string

employee_id = str(employee_id)

# Display the `employee_id` as it currently stands

print(employee_id)

# Conditional statement that updates the `employee_id` if its length is less than 5 digits

out =len(employee_id)

if out < 5:
   employee_id_out = "0" + employee_id

# Display the `employee_id` after the update

print (employee_id_out)
```

```
4186
04186
```

Hint 1

Hint 2

Hint 3

Task 4

Now you'll move on to the next part of your task. Imagine that the characters in a device ID convey technical information about the device. You'll need to extract characters in specific positions from the device ID. Start off by extracting the fourth character.

The variable `device_id` represents a device ID containing alphanumeric characters; it's already stored as a string.

Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
# Assign `device_id` to a string that contains alphanumeric characters

device_id = "r262c36"

# Extract the fourth character in `device_id` and display it

print(device_id[3])
```

```
2
```

Hint 1

Hint 2

Hint 3

Task 5

Now you will also need to extract the first through the third characters in the device ID. So take a slice of the device ID. You can achieve this using bracket notation in Python. Then, display the slice to examine the result.

Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
# Assign `device_id` to a string that contains alphanumeric characters

device_id = "r262c36"

# Extract the first through the third characters in `device_id` and display the result

print(device_id[0:3])
```

```
r26
```

Hint 1

Hint 2

Hint 3

Task 6

You'll now proceed to the last part of your task. This involves extracting components of a URL.

You'll work with string indices to display various components of a URL that's stored in the URL variable. First, you'll extract and display the protocol of the URL and the `://` characters that follow it using string slicing. Consider that the protocol is in the secure format of `https` when determining the indices for your slice.

Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
# Assign `url` to a specific URL

url = "https://exampleURL1.com"

# Extract the protocol of `url` along with the syntax following it, display the result

print (url[0:8])
```

```
https://
```

Hint 1

Hint 2

Hint 3

Task 7

Later in this lab, you'll extract the domain extension. To prepare for this, use the `.index()` method to identify the index where the domain extension `.com` is located in the given URL.

Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
# Assign `url` to a specific URL

url = "https://exampleURL1.com"

# Display the index where the domain extension ".com" is located in `url`

com_index = url.index(".com")
print(com_index)
```

```
19
```

Hint 1

Apply the `.index()` method to `url` in order to get the appropriate index. The `.index()` method takes in a substring, and if that substring is located in the original string, it returns the index where that substring starts to occur in the original string.

Hint 2

Call `url.index()`, and inside the parentheses, pass in the targeted domain extension as a string.

Task 8

It's a good idea to save important data in variables when programming. This allows for quick and easy tracking and reuse of information.

Store the output of the `.index()` method in a variable called `ind`, which is short for index. This index represents the position where the domain extension `".com"` starts in the `url`. Be sure to replace the `### YOUR CODE HERE ###`with your own code before you run the following cell. Note that running this cell will not produce an output.

```
# Assign `url` to a specific URL

url = "https://exampleURL1.com"

# Assign `ind` to the output of applying `.index()` to `url` in order to extract the starting index of ".com" in `url`

ind = url.index(".com")
print(ind)
```

```
19
```

Hint 1

Hint 2

Task 9

You can use string slicing to also extract the domain extension of a URL. To do so, you can create a slice. The starting index should be the `ind` variable. This contains the index where the domain extension begins. The ending index should be `ind + 4` (since `".com"` is four characters long). Sometimes, like in this situation, it's easier to express the ending index in relation to the starting index. Examine the following code, run it as is, and observe the output.

```
# Assign `url` to a specific URL
```

```python
url = "https://exampleURL1.com"

# Assign `ind` to the output of applying `.index()` to `url` in order to extract the starting index of ".com" in `url`

ind = url.index(".com")

# Extract the domain extension in `url` and display it

print(url[ind:ind+4])
```

```
.com
```

Question 2

What does this code output and why?
[Double-click to enter your responses here.]

Task 10

Finally, extract the website name from the given URL using string slicing and the `ind` variable that you defined earlier. In the given URL, the website name is `"exampleURL1"`. Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

```python
# Assign `url` to a specific URL

url = "https://exampleURL1.com"

# Assign `ind` to the output of applying `.index()` to `url` in order to extract the starting index of ".com" in `url`

ind = url.index(".com")

# Extract the website name in `url` and display it

print(url[6:19])
```

```
//exampleURL1
```

Hint 1

Hint 2