Scenerio

In this scenario, I'm a security analyst developing Python programs to automate three key tasks: updating employee IDs, extracting characters from device IDs, and parsing URL components. These scripts will enhance our organization's data management, streamline workflows, and improve security protocols.

Task 1

In this task, I'm working with a four-digit numeric employee ID stored in the variable employee_id. My job is to convert this ID to a string format and store it back in the same variable. This conversion is the first step in preparing our employee IDs for a new standardized format that our organization plans to implement. The process involves using Python's string conversion function to transform the numeric ID into a string, setting the stage for future modifications to the employee ID system.

```
: # 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(employee_id)

# Display the data type of `employee_id` now

print(type(employee_id))
```

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

```
Initially, employee_id is an integer. After conversion, it becomes a string.
```

In this task, I need to address a new requirement for employee IDs. The company now requires all IDs to be five digits long for standardization. My job is to write a conditional statement in Python that checks the length of an employee ID. If the ID is less than five digits, the program should display a message indicating this. This check will help identify IDs that don't meet the new standard and need to be updated.

```
# 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

if len(employee_id) < 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 requirements.
```

In this task, I'm expanding on the previous code to handle four-digit employee IDs. I need to write an if statement that checks if the employee_id length is less than 5. If true, I'll use string concatenation to add "E" at the beginning of the existing ID, creating a five-character ID. This involves using the + operator to combine "E" with the current employee_id. After making this change, I'll display the updated employee_id to confirm the modification.

```
# 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

### YOUR CODE HERE ###:
if len(employee_id) < 5:
    employee_id = "E" + employee_id

# Display the `employee_id` after the update

print (employee_id)
```

```
4186
E4186
```

Task 4

In this task, I'm working with device IDs that contain important technical information encoded in their characters. My job is to extract specific characters from these IDs, starting with the fourth character. I'll be using a string variable called device_id, which already contains an alphanumeric device ID. To accomplish this, I'll need to use string indexing in Python to access and extract the desired character from the device_id string.

```
# 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
```

Task 5

In this task, I need to extract the first three characters from the device ID string using Python's string slicing feature. I'll create a slice from index 0 to 3 and display it to verify the result.

```
# 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
```

Task 6

In this task, I'm extracting components of a URL stored in the 'URL' variable. I'll use string slicing to display the protocol part, which includes 'https://' and its following characters.

```
# 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://
```

Task 7

In this task, I'm using Python's .index() method to locate the position of the ".com" extension in a URL. This will help me prepare for extracting the domain extension later.

```
# Assign `url` to a specific URL
url = "https://exampleURL1.com"

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

```
19
```

Task 8

In this task, I'm storing the index of ".com" in the URL string into a variable called 'ind'. This involves using the .index() method and assigning its result to 'ind'. Saving this index in a variable will make it easier to use this information later in the program.

```
# 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 `ur

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

Task 9

In this project, I'm focusing on extracting the domain extension from a URL using string slicing. I'll create a slice that starts at the index stored in the 'ind' variable, which marks the beginning of the domain extension. The slice will end at 'ind + 4' to capture the full ".com" (four characters). This approach demonstrates how to use relative indexing, where the end point is defined in relation to the start point. By running the provided code, I'll be able to see how this slicing technique effectively isolates the domain extension from the rest of the URL.

What does this code output and why?

```
The code outputs ".com" because it extracts a slice from the URL string. The slice starts at index 20, where ".com"
begins, and ends at index 24, capturing the four characters ".com". This is achieved by using url[ind:ind+4], where
ind is 20, the starting index of ".com".
```

Task 10

In this task, I'm extracting the website name from the URL using string slicing. I'll use the ind variable I defined earlier, which marks the start of ".com", to help determine the slice boundaries. My goal is to isolate "exampleURL1" from the full URL string. This involves creating a slice that starts after the "https://" prefix and ends just before the ".com" extension.

```
# 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 `ur

ind = url.index(".com")

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

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

```
exampleURL1
```

Key Takeaways

1) You can extract specific characters from a device ID using Python's string indexing
2) To extract the protocol part of a URL you can use string slicing like url[0:8].
3) The .index() method can be used to find the starting index of a domain extension like ".com" in a URL.
4) You can extract the domain extension using string slicing with the index found by .index(), like url[ind:ind+4].