String Indexing in Python

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.

However, Python does not have a character data type, a single character is simply a string with a length of 1.

Square brackets **[start : end : step]** can be used to access elements of the string using the parameters “start”, “end” and “step”.

Example:

We want to display the first digit of a credit card we use only [start] position:

```
creditcard = "5024 1234 5678 2468"
# string pos  0123456789...

# to print the first digit
print(creditcard[0])
```

To display the first 4 digits, display positions 0 – 3

The **start** index is inclusive, but the **end** index is NOT included.

```
creditcard = "5024 1234 5678 2468"
# string pos  0123456789...

# to print the first digit
print(creditcard[0:4])  # <-- Start index is inclusive | end
index exclusive
```

print(creditcard[0:4]) could also be written **print(creditcard[:4])** ← Python assumes we are starting at the beginning of the string.

If we want to print the last half of the card number, the first digit will be in the 10th place. We’d use **print(creditcard[10:])** ← Python assumes a blank after the colon means we want everything through to the end.

If we want to print the last digit we can use the ‘**negative index**’ position where the last digit’s position is deemed to be **-1**, second last digit is at **-2** and so on.

To display the last four digits:

```
creditcard = "5024 1234 5678 2468"
# string pos  0123456789...

# to print the first digit
print(creditcard[-4:])  # position -4 is 4th from the end
```

The step parameter specifies the increment between each index in the slice.

If we wish to display every 3rd character of the entire string we'd use:

print(creditcard[0:19:3]) ← where [start:end:step]

print(creditcard[::3]) ← will do the same thing, Python presumes start to end.

If we wanted every second character of the centre 8 numbers (9 characters)

```
creditcard = "5024 1234 5678 2468"
# string pos  0123456789...

print(creditcard[5:14:2])
```

We expect from the positions indicated an output of **13 68**:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
5	**0**	**2**	**4**		**1**	**2**	**3**	**4**		**5**	**6**	**7**	**8**		**2**	**4**	**6**	**8**
					^		**^**		**^**		**^**		**^**					

Exercise:

Write code to display a user's inputted credit card number securely with the following output:

Your credit card number is: XXXX-XXXX-XXXX-2468

To print out a string in reverse order, we simply make the '**step**' parameter -1

```
backwardtext = input("Enter some input to be reversed: ")

print(backwardtext[::-1])
```

Exercise:

Write code to input a user's email address and separate it into a domain name and username. The output should look like:

Your username is bob.down

Your email domain is gmail.com

Solution:

```
email = input("Enter your email address: ")

split = email.find("@")  # <-- could also use email.index("@")

username = email[:split]     # <-- or email[0:split]
domain = email[split + 1:]  # <-- + 1 excludes @ from domain

print(f"Your username is: {username} ")
print(f"Your domain is: {domain}")
```

On line 3 we could use

split = email.find(“@”) ← if the user inputs an invalid email omitting ‘@’ - split = -1

split = email.index(“@”) ← if the user inputs an invalid email omitting ‘@’ - ERROR

You could also write this code with one less line omitting the “split’ variable – it’s a matter of personal preference and ease of readability:

```
email = input("Enter your email address: ")

username = email[:email.find("@")]
domain = email[email.find("@") + 1:]

print(f"Your username is: {username} ")
print(f"Your domain is: {domain}")
```