CS PROFESSIONAL ELECTIVE | FINALS: ASSIGNMENT #3

MATILLA, CARL ANDRIE D. | BSCS 3C

CREATE A STRING

In a Jupyter Notebook, you may make a string by just assigning a string of characters, surrounded in single(‘ ‘) or double quotes (""), to a variable.

Example:

```
# Using single quotes
my_string_single = 'Hello, World!'

# Using double quotes
my_string_double = "Hello, World!"

# Displaying the strings
print(my_string_single)    # Output: 'Hello, World!'
print(my_string_double)   # Output:  "Hello, World!"
```

- You may run each of these code cells individually in Jupyter Notebook to check out the results. Just confirm that the Jupyter Notebook is running a Python kernel.

ACCESSING CHARACTERS IN THE STRING

In Python, accessing characters within a string is simple. Indexing allows you to get individual characters inside a string. Strings in Python are zero-indexed, which means that the index of the first character is 0, the index of the second character is 1, and so on. Negative indices are another option; in this case, -1 denotes the final character, -2 the next-to-last character, and so on.

Example:

```
# Define a string
my_string = "Hello, World!"
```

```
# Accessing individual characters using positive indices
print(my_string[0])  # Output: H
print(my_string[7])  # Output: W

# Accessing individual characters using negative indices
print(my_string[-1])  # Output: !
print(my_string[-6])  # Output: W
```

- In this example, the string "Hello, World!" is created and saved in the variable 'my_string'. Individual characters can be accessed using positive and negative indices, and the pertinent characters are printed.

REMOVE SPACE FROM A STRING

Use the 'replace()' or 'split()' methods, followed by 'join()', to eliminate spaces from a text in Python.

'replace()':

```
# Define a string with spaces
my_string = "Hello, World! This is a string with spaces."

# Remove spaces using replace()
my_string_without_spaces = my_string.replace(" ", "")

# Print the string without spaces
print(my_string_without_spaces) #Output: Hello,World!Thisisastringwithspaces.
```

'split()' & 'join()':

```
# Define a string with spaces
my_string = "Hello, World! This is a string with spaces."

# Remove spaces using split() and join()
my_string_without_spaces = "".join(my_string.split())

# Print the string without spaces
print(my_string_without_spaces) #Output: Hello,World!Thisisastringwithspaces.
```

PYTHON STRING METHODS

There are several built-in string functions in Python that let you work with strings in different ways. The following are a few popular string methods:

1. **'capitalize()':** changes the string's initial character to uppercase and the remaining characters to lowercase.

```
my_string = "hello, world!"
print(my_string.capitalize())  # Output: Hello, world!
```

2. **'upper()':** Makes every character in the string capital.

```
my_string = "hello, world!"
print(my_string.upper())  # Output: HELLO, WORLD!
```

3. **'lower()':** Lowercases each character in the string.

```
my_string = "HELLO, WORLD!"
print(my_string.lower())  # Output: hello, world!
```

4. **'strip()':** eliminates the string's leading and trailing whitespaces.

```
my_string = "   hello, world!   "
print(my_string.strip())  # Output: hello, world!
```

5. **"replace()":** Changes a substring's occurrences to another substring.

```
my_string = "hello, world!"
print(my_string.replace("hello", "hi"))  # Output: hi, world!
```

6. **"split()":** Divides a string into a list of substrings according to a delimiter (whitespace is the default).

```
my_string = "hello, world!"
print(my_string.split(","))  # Output: ['hello', ' world!']
```

7. **'join()':** Uses the original string as a delimiter to unite components of an iterable (such as a list) into a single string.

```
my_list = ['hello', 'world']
print(" ".join(my_list))  # Output: hello world
```

8. **"find()":** Gives back the substring's lowest index within the string (-1 if not found).

```
my_string = "hello, world!"
print(my_string.find("world"))  # Output: 7
```
