STRINGS

CREATING A STRING

- assign a sequence of characters to a variable. The output will be displayed below the cell. You can directly type and execute the code in a code cell without needing the `**print()**` statements as Jupyter automatically displays the output of the last line of code in a cell.

  Here's how you can create a string:

```
# Create a string using single quotes
string_with_single_quotes = 'Hello, world!'

# Create a string using double quotes
string_with_double_quotes = "Hello, world!"

# Create a string using triple quotes for multiline strings
multiline_string = """
This is a multiline string.
It can span across multiple lines.

# Display the strings
print(string_with_single_quotes)
print(string_with_double_quotes)
print(multiline_string)
```

ACCESSING CHARACTERS IN THE STRING

- Using indexing you can access individual characters in a string. In Python, strings are zero-indexed, meaning the first character has an index of 0, the second character has an index of 1, and so on. It will output the characters at the specified indices in the string. Remember that attempting to access an index beyond the length of the string will result in an `**IndexError**`.

  Here's how you can access characters in a string:

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

# Access the first character
first_character = my_string[0]
```

```
# Access the fifth character
fifth_character = my_string[4]

# Access the last character
last_character = my_string[-1]  # Negative indexing starts from
the end of the string

# Display the characters
print("First character:", first_character)
print("Fifth character:", fifth_character)
print("Last character:", last_character)
```

REMOVING SPACE FROM A STRING

- You can use the `**replace()**` function or the `**split()**` and `**join()**` methods. Both methods will produce the same output, removing all spaces from the original string.

- **Here's how you can do it:**

  Using `replace()` function:

```
# Define a string with spaces
my_string = "Hello, world! How are you?"

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

# Display the string without spaces
print(no_spaces)
```

- **Using `split()` and `join()` methods:**

```
# Define a string with spaces
my_string = "Hello, world! How are you?"

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

# Display the string without spaces
print(no_spaces)
```

PYTHON STRING METHODS

- These are some common and available string methods in Python. It has many methods available for manipulating strings in Python. You can find more in the Python documentation.

  1. `**capitalize()**`: Converts the first character of the string to uppercase.
  2. `**upper()**`: Converts all characters in the string to uppercase.
  3. `**lower()**`: Converts all characters in the string to lowercase.
  4. `**title()**`: Converts the first character of each word to uppercase, and the rest to lowercase.
  5. `**strip()**`: Removes leading and trailing whitespace from the string.
  6. `**lstrip()**`: Removes leading whitespace from the string.
  7. `**rstrip()**`: Removes trailing whitespace from the string.
  8. `**startswith(prefix)**`: Checks if the string starts with the specified prefix.
  9. `**endswith(suffix)**`: Checks if the string ends with the specified suffix.
  10. `**replace(old, new)**`: Replaces all occurrences of the old substring with the new substring.
  11. `**split(separator)**`: Splits the string into a list of substrings based on the specified separator.
  12. `**join(iterable)**`: Joins the elements of an iterable (such as a list) into a single string, using the string as a separator.
  13. `**find(substring)**`: Searches for the first occurrence of the specified substring and returns its index. Returns -1 if the substring is not found.
  14. `**index(substring)**`: Similar to `find()`, but raises a ValueError if the substring is not found.
  15. `**count(substring)**`: Returns the number of occurrences of the specified substring in the string.
  16. `**isdigit()**`: Checks if all characters in the string are digits.
  17. `**isalpha()**`: Checks if all characters in the string are alphabetic.
  18. `**isalnum()**`: Checks if all characters in the string are alphanumeric.
  19. `**isspace()**`: Checks if all characters in the string are whitespace.
  20. `**startswith(prefix)**`: Checks if the string starts with the specified prefix.

PYTHON AND JUPYTER NOTEBOOK

LAUNCH JUPYTER NOTEBOOK

- By using the command line interface.
  Here's how you can do it:

1. Open your command prompt or terminal.
2. Navigate to the directory where you want to store your Jupyter Notebook files, or where you have existing notebooks.
3. Once you're in the desired directory, simply type the following command and press Enter:
   jupyter notebook

   This command will start the Jupyter Notebook server, and a new tab will open in your default web browser displaying the Jupyter Dashboard, where you can create new notebooks or open existing ones.

   In Jupyter Notebook, you can also launch a new notebook from the dashboard by clicking on "New" and selecting "Python 3" (or any other available kernel).

   Additionally, if you're using an integrated development environment (IDE) such as Anaconda or VSCode, there may be specific features or buttons within the IDE to launch Jupyter Notebook directly.

   Once the Jupyter Notebook is running, you can start coding in Python within the notebook cells.

Using Anaconda Navigator (if you have Anaconda installed):

- If you have Anaconda installed, you can also launch a Jupyter Notebook using Anaconda Navigator.

1. Open Anaconda Navigator from your Start menu or launch it using the command line.

2. In Anaconda Navigator, you'll see an option for Jupyter Notebook. Click on it to launch Jupyter.

OPEN A NOTEBOOK FILE

1. **Using the Jupyter Notebook Dashboard:**
   - Open your web browser and navigate to the URL displayed in your command prompt/terminal when you launch the Jupyter Notebook.
   - In the Jupyter Notebook dashboard, navigate to the directory where your notebook file (.ipynb extension) is located.
   - Click on the notebook file you want to open. This will open the notebook in a new tab within the Jupyter Notebook interface.

2. **Using the Command Line:**
   - Open your command prompt (Windows) or terminal (Mac/Linux).
   - Navigate to the directory where your notebook file is located using the **`cd`** command.
   - Once you're in the desired directory, type the following command and press Enter: **jupyter notebook notebook_name.ipynb**
   - Replace `notebook_name.ipynb` with the actual name of your notebook file.
   - This command will open the specified notebook file directly in Jupyter Notebook.

Once the notebook file is opened, you can view its contents, execute code cells, and make edits as needed.

START WRITING A JUPYTER NOTEBOOK

To start writing in a Jupyter Notebook, you can follow these steps:

1. **Open a Jupyter Notebook:**
   - Launch Jupyter Notebook using one of the methods by command line or Anaconda Navigator.

- Once the Jupyter Notebook opens in your web browser, you'll see the dashboard displaying your files and folders.

2. Create a New Notebook:

- Click on the **"New"** button located in the top-right corner of the dashboard.
- From the dropdown menu, select **"Python 3" (or any other kernel you prefer)** to create a new Python notebook.

3. Writing in a Notebook:

- You'll see a new tab opened with the title **"Untitled.ipynb" or similar.**
- Click on the first cell, which is a code cell by default, to start writing Python code.
- To change the cell type to markdown for writing text, press **`Esc`** to exit editing mode, then press **`M`** to change the cell type to markdown or you can use the dropdown menu in the toolbar.
- Once the cell type is set to markdown, you can start typing text in the cell.
- To execute the cell and render the markdown, press **`Shift + Enter`.**

4. Writing and Executing Code:

- If you want to write and execute Python code, simply type your code in a code cell and press **`Shift + Enter`** to execute it.
- The output (if any) will be displayed below the code cell.

5. Adding New Cells:

- To add a new cell, click on the **"+"** button in the toolbar.
- You can also use the keyboard shortcut **`Esc` + `B`** to insert a new cell below the current one or **`Esc` + `A`** to insert a new cell above the current one.

6. Saving Your Work:

- Jupyter Notebook automatically saves your work periodically, but you can also manually save by clicking on

the floppy disk icon in the toolbar or by pressing **`Ctrl + S` (`Cmd + S` on Mac).**