To what does a relative path refer?

```
- A relative path refers to the location of a file or directory relative to the current working directory or another specified location. It provides a path to a file or directory starting from a particular reference point rather than the root directory.
For example, if you have a file called "example.txt" located in a folder named "documents" that is within the current directory, the relative path to that file from the current directory might be "documents/example.txt".
```

What does an absolute path start with your operating system?

```
An absolute path in a file system typically starts with the root directory of the operating system.

In Windows, absolute paths start with a drive letter followed by a colon (e.g., "C:\Users\Username\Documents\file.txt").

Absolute paths specify the complete location of a file or directory from the root of the file system, regardless of the current working directory.
```

What do the functions os.getcwd() and os.chdir() do?

```
- The os.getcwd() function in Python's os module stands for "get current working directory."
```

In [1]:
```
import os

current_directory = os.getcwd()
print(current_directory)
```

```
C:\Users\rosha\INeuronAssignments
```

```
- The os.chdir() function, on the other hand, stands for "change directory."
```

What are the . and .. folders?

```
. (dot) represents the current directory. When used in a path, it refers to the directory you're currently in. For example, if you're in the directory /home/user/documents, . refers to the documents directory itself.
```

```
.. (dot-dot) represents the parent directory. It refers to the directory immediately above the current directory. For instance, if you're in the directory /home/user/documents, .. refers to the user directory, which is the parent directory of documents.
```

In C:\bacon\eggs\spam.txt, which part is the dir name, and which part is the base name?

```
"C:\bacon\eggs" is the directory name.
"spam.txt" is the base name.
```

What are the three “mode” arguments that can be passed to the open() function?

```
Read mode ('r')
write mode ('w')
append mode ('a')
```

What happens if an existing file is opened in write mode?

```
It truncates file to the zero length.
```

How do you tell the difference between read() and readlines()?

- read() Method:

```
Reads the entire content of the file as a single string (or bytes in binary mode), including newline characters (\n).
It reads the file's content from the current position of the file pointer until the end of the file or until the specified number of characters/bytes.
If no argument is provided, it reads the entire file. If a size argument is given (read(size)), it reads the specified number of characters/bytes.
```

In [ ]:
```
with open('file.txt', 'r') as file:
    content = file.read()  # Reads the entire file content as a string
    print(content)
```

- readlines() Method:

```
Reads the lines of the file and returns a list where each element represents a line in the file.
It reads the file content from the current position of the file pointer until the end of the file.
If called without arguments (readlines()), it reads all lines from the current position to the end of the file.
```

In [ ]:
```
with open('file.txt', 'r') as file:
    lines = file.readlines()  # Reads lines into a list
    for line in lines:
        print(line)
```

What data structure does a shelf value resemble?

In [8]:

```python
import shelve

# Creating and accessing a shelf
with shelve.open('my_data') as shelf:
    shelf['key1'] = 'value1'
    shelf['key2'] = [1, 2, 3]

    print(shelf['key1'])  # Accessing values by key
    print(shelf['key2'])

    # Getting keys, values, and items
    print(list(shelf.keys()))
    print(list(shelf.values()))
    print(list(shelf.items()))
```

```
value1
[1, 2, 3]
['key1', 'key2']
['value1', [1, 2, 3]]
[('key1', 'value1'), ('key2', [1, 2, 3])]
```

In [ ]:

```python

```