Strings In Python (part-1)

In Python, a string is a sequence of characters enclosed within either single (' '), double (" "), or triple (''' ' ''', """ """ or ''' ''') quotes. Strings are used to represent text data and are one of the fundamental data types in Python

- Strings are sequence of Characters
- In Python specifically, strings are a sequence of Unicode Characters
- Immutable: Strings in Python are immutable, meaning once you create a string, you cannot change its content. You can create new strings based on the original string, but the original string remains unchanged.

1. Creating Strings

In [ ]:
```
# create string with the help of single quotes
s = 'Hello World'
print(s)
```

```
Hello World
```

In [ ]:
```
# create string with the help of Double quotes
s = "Hello World"
print(s)
```

```
Hello World
```

when we use single and double quotes?

-> In Python, you can create strings using both single quotes (') and double quotes ("), and they are functionally equivalent. The choice between using single or double quotes mainly depends on your personal preference or specific coding style guidelines.

If your string contains a quotation mark character (either a single quote or double quote), you can use the other type of quote to define the string without escaping the inner quotes. This can make your code more readable:

In [ ]:
```
single_quoted_string = 'He said, "Hello!"'
double_quoted_string = "He said, 'Hello!'"
# here we use single and double quotes
print(single_quoted_string)
print(double_quoted_string)
```

```
He said, "Hello!"
He said, 'Hello!'
```

In [ ]:
```
# multiline strings
s = '''hello'''
s = """hello"""
s = str('hello')
print(s)
```

```
hello
```

2. Accessing Substrings from a String

Indexing

Indexing in Python refers to the process of accessing individual elements or characters within a sequence, such as a string. Strings in Python are sequences of characters, and you can access specific characters in a string using indexing. Indexing is done using square brackets [ ], and Python uses a zero-based indexing system, meaning the first element has an index of 0, the second element has an index of 1, and so on.

we have two types of Indexing:

1. Positive indexing
2. Negative Indexing

In [ ]:
```
s = "Hello World"

# Accessing individual characters using positive indexing
print(s[0])  # Accesses the first character, 'H'
print(s[1])  # Accesses the second character, 'e'
```

```
H
e
```

In [ ]:
```
# Accessing individual characters using negative indexing (counting from the end)
print(s[-1])  # Accesses the last character, 'd'
print(s[-2])  # Accesses the second-to-last character, 'l'
```

```
d
l
```

Slicing

In Python, slicing is a technique used to extract a portion of a string, list, or any other sequence-like data structure. When it comes to strings, slicing allows you to create a new string by specifying a range of indices to extract a substring from the original string.

In [ ]:
```
# string[start:end]
```

- string: The original string you want to slice.
- start: The index from which the slicing begins (inclusive).
- end: The index at which the slicing ends (exclusive).

In [ ]:
```
s = 'hello world'
print(s[1:5]) # from 1 index that is e to 4 index=o
print(s[:6]) # from 0 to 5
print(s[1:]) # from 1 to all
print(s[:]) # all string
print(s[-5:-1]) # with negative slicing
```

```
ello
hello
ello world
hello world
worl
```

In [ ]:
```
# we use step function in slicing
s = 'hello world'
print(s[6:0:-2])
print(s[::-1])
```

```
wol
dlrow olleh
```

In [ ]:
```
s = 'hello world'
print(s[-1:-6:-1])
```

```
dlrow
```

3. Editing and Deleting in Strings

In [ ]:
```
s = 'hello world'
s[0] = 'H'
# it throws error
# Python strings are immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_4564\2237226474.py in <module>
      1 s = 'hello world'
----> 2 s[0] = 'H'
      3
      4 # Python strings are immutable

TypeError: 'str' object does not support item assignment
```

Immutable: Strings in Python are immutable, meaning once you create a string, you cannot change its content. You can create new strings based on the original string, but the original string remains unchanged.