Multi Value Data Types

02 June 2025 16:13

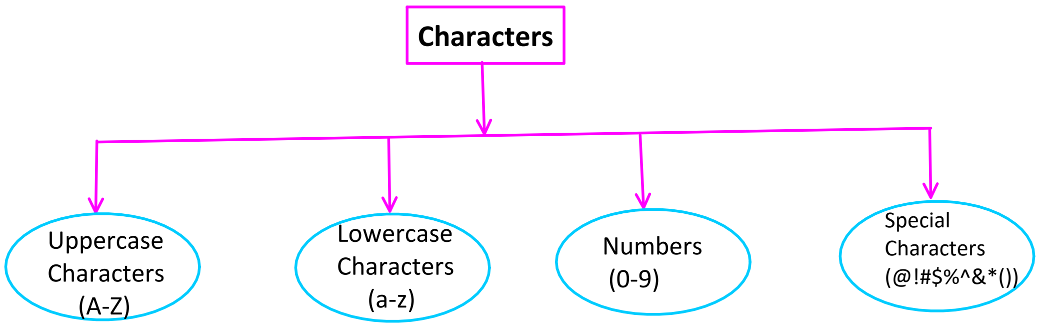
- We can called Multi value data type as a Collection Data Type.
- Whenever we store More than one value to a Variable then we can called it is a "Collection".

Multi value Data Types got classified into 5 types:
1.String
2.List
3.Tuple
4.Set
5.Dictionary

1.String collection:

- String is a collection of characters and we can represent using "str".

Q. What is characters?


Characters
Uppercase Characters (A-Z)
Lowercase Characters (a-z)
Numbers (0-9)
Special Characters (@!#$%^&*())

- String is a collection of characters which are enclosed between pai of ' ' or " " or ''' '''.

```
Syntax: Variable= 'val1 val2 val3........value n'
                        (OR)
        Variable= "val1 val2 val3........value n"
                        (OR)
        Variable= '''val1 val2 val3........value n'''
```

Q. Why We should go with this many Syntaxes?

Example: S='Happy father's day'
S
Output=>SyntaxError: unterminated string literal (detected at line 1)

[Note:-In the above example, the string contains an extra single quote (') inside the word "father's". Since the string starts and ends with a single quote, Controller gets confused and interprets the quote in "father's" as the end of the string. This causes a syntax error because the string is not properly closed.]

- To overcome this problem we are going to use (" ") double qoutes.

Example: P=" Happy Mother's Day"
P
Output:" Happy Mother's Day"

Example :

```
q='hi
SyntaxError: unterminated string literal (detected at line 1)
q="hello
SyntaxError: unterminated string literal (detected at line 1)
```

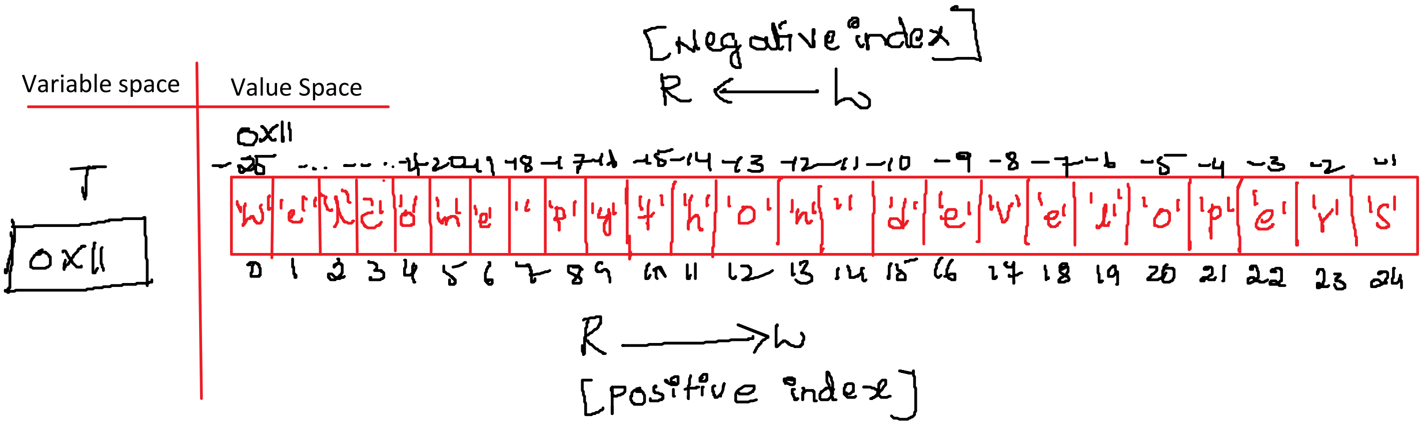
[Note: In the above examples, we are trying to write string data across multiple lines using single (') and double (") quotes. However, Python does not allow line breaks inside strings defined this way, which results in a SyntaxError: unterminated string literal.]
To overcome this issue and allow multi-line strings, we can use **triple quotes** (''' or """)

```
Example: R=''' hai
            Good morning
            Bye'''
      R
  Output: ' hai\nGood morning\nBye'      [Note: \n indicates NextLine]
```

- The Default value of String :- ' ' or " " or ''' ''' [internally False].

```
Example: a='Dear Beautiful Students'
           type(a)=>to know the what data storing
         Output:<Class 'str'>
           len(a)=>to find out the length
         Output: 6
          bool(a)=>to find out default value
         Output: True
```

Example: T=' Welcome python developers'



Indexing:

- Indexing is used when we want to access individual values or characters from a given string or collection.

Definition:

- Indexing is defined as assigning a sub-address to each element in a string or collection, allowing direct access to individual elements.

Types of Indexing:

1.Positive indexing:-

- In positive indexing, we traverse the collection from **left to right**.
- Indexing starts from 0 and goes up to length - 1.

2.Negative indexing:-

- In negative indexing, we traverse the collection from **right to left**.
- Indexing starts from -1 and goes up to -length.

Syntax for to fetch the character from string or collection:

Syntax: variable name[index value]

Example: T['23']
  Output: 'r'
  T[0]
  Output: 'W'
  T[-4]
  Output: 'p'

Why Do We Need Two Types of Indexes?

- If we know the exact length of the string or collection, we can access any character using positive indexing (e.g., T[23] for the last character).
- However, when we do not know the length of the string, we use negative indexing. For example, T[-1] always gives the last character of the string, regardless of its length

Q. Can We Modify Characters in a String?

Syntax: variable name[index value]=new value

Example: T[22]='a'
  Output :Error

Mutable vs. Immutable Collections

- **Mutable Collection:**
  A collection is **mutable** if its elements can be changed after it is created.
- **Immutable Collection:**
  A collection is **immutable** if its elements cannot be modified after creation.

Note: Strings are an example of an immutable collection in Python.