String

{ Collection Data Type }

Python Strings

We have used strings quite a bit up to this point. In this lesson, we further expand our knowledge of strings.

A string is a sequence of characters.

Computers do not deal with characters, they deal with numbers (binary). Even though you may see characters on your screen, internally, it is stored and manipulated as a combination of 0's and 1's.

This conversion of character to a number is called encoding, and the reverse process is decoding. ASCII and Unicode are some of the popular encoding used.

In Python, a string is a sequence of Unicode characters. Unicode was introduced to include every character in all languages and bring uniformity in encoding.

How to Create Strings?

Strings can be created by enclosing characters inside a single quote or double quotes. Even triple quotes can be used in Python but it is generally used to represent multiline strings.

```
my_string = 'Hello'
print(my_string)

my_string = "Hello"
print(my_string)

my_string = '''Hello'''
print(my_string)

# triple quotes string can extend
multiple lines
my_string = """Hello, welcome to
           the world of Python"""
print(my_string)
```

Output

```
Hello
Hello
Hello
Hello, welcome to
           the world of Python
```

Access Characters in a String

Similar to lists and tuples, we can access characters in a string by using the index operator `[]`. Here's how:

```
my_string = 'Python'

# first character
print(my_string[0])    # Output: 'P'

# fifth character
print(my_string[4])    # Output: 'o'
```

Note that indexing starts from 0, not 1.

Hence, the first character is `my_string[0]`, the second character is `my_string[1]` and so on.

Negative Indexing

Python allows negative indexing for its sequences.

The index of the last character is -1, the index of the second last character is -2 and so on.

```
my_string = 'Learning Path'

# last item
print(my_string[-1])    # Output: 'h'

# second last item
print(my_string[-2])    # Output: 't'

# 9th last item
print(my_string[-9])    # Output: 'n'
```

Slicing a String

We can access a range of characters in a string using the slicing operator `:`.

```
my_string = 'Python'

# 2nd to 4th characters
print(my_string[1:4])    # Output:
'yth'

# beginning to 4th characters
print(my_string[0:4])    # Output:
'Pyth'

# beginning to 4th characters
print(my_string[:4])    # Output:
'Pyth'

# 3rd to third last characters
print(my_string[2:-2])    # Output:
'th'
```

Slicing can be best visualized by considering the index to be between the elements as shown below.

If we want to access a range, we need the

Deleting a String

Strings are immutable. It's not possible to change or delete characters of a string.

However, you can delete the string entirely using the `del` keyword.

```
my_string = 'Python'
del my_string

print(my_string)

# NameError: name 'my_string' is not defined
```

Python String Operations

There are many operations that can be performed with a string which makes it one of the most used data types in Python.

Concatenation of Two Strings

You can use the `+` operator to concatenate two strings.

```
first_name = 'learning'
last_name = 'path'

full_name = first_name + ' ' + last_name
print(full_name)    # Output: learning
path
```

Repeat Two Strings

You can repeat a string using the `*` operator.

```
my_string = 'learning'
print(my_string * 3)  # learning learning
learning
```

Iterating through a String

You can iterate through a sequence (including strings) using a `for` loop.

Here's a program to count the number of `'l'` in a string.

```
count = 0

for letter in 'Hello World':
    if(letter == 'l'):
        count += 1

print(count,'letters found')
```

Output

```
3 letters found
```

Here, we iterate through all characters of a string `'Hello World'` using a `for` loop.

In each iteration, we checked if a letter is `'l'` or not. If the letter is `'l'`, we increase the value of `count` by 1.

Check whether Substring Exists

We can test if a substring exists within a string or not, using the `in` keyword.

```
result = 'a' in 'Hello'
print(result)     # False

result = 'o' in 'Hello'
print(result)     # True

result = 'ell' in 'Hello'
print(result)     # True

result = 'lo' in 'Hello'
print(result)     # True
```

Python String Methods

Strings are very common in most programming languages (including Python). There are numerous string methods in Python that make working with strings easier.

Some of the commonly used string methods are:

Methods	Description
`format()`	formats the given string into a nicer output in Python
`lower()`	returns a string with all uppercase characters converted to lowercase characters
`upper()`	returns a string with all lowercase characters converted to uppercase characters
`join()`	returns a string concatenated with the elements of an iterable
`split()`	breaks the string at the separator and returns a list of strings.
`find()`	returns the index of the first occurrence of the substring (if found)
`replace()`	returns a copy of the string where all occurrences of a substring are replaced with another substring.

Escape Sequence

If we want to print a text like - `He said, "What's there?"` - we can neither use single quotes nor double quotes. This will result in `SyntaxError` as the text itself contains both single and double quotes.

```
print("He said, "What's there?"")  # Error
print('He said, "What's there?"')  # Error
```

One way to get around this problem is to use triple quotes.

```
print('''He said, "What's there?"''')
# This works
```

Alternatively, we can use escape sequences.

Escape Sequence (Part II)

An escape sequence starts with a backslash and is interpreted differently.

If we use single quotes to represent a string, all the single quotes inside the string must be escaped. Similar is the case with double quotes. Here is how it can be done to represent the above text.

```
# escaping single quotes
print('He said, "What\'s there?"')

# escaping double quotes
print("He said, \"What's there?\"")
```

Some of the commonly used escape sequences are:

- `\\` - backslash
- `\n` - line feed (moving one line forward)
- `\'` - single quote
- `\"` - double quote
- `\t` - horizontal tab
- `\v` - vertical tab

Try to use these sequences yourself in a program.