



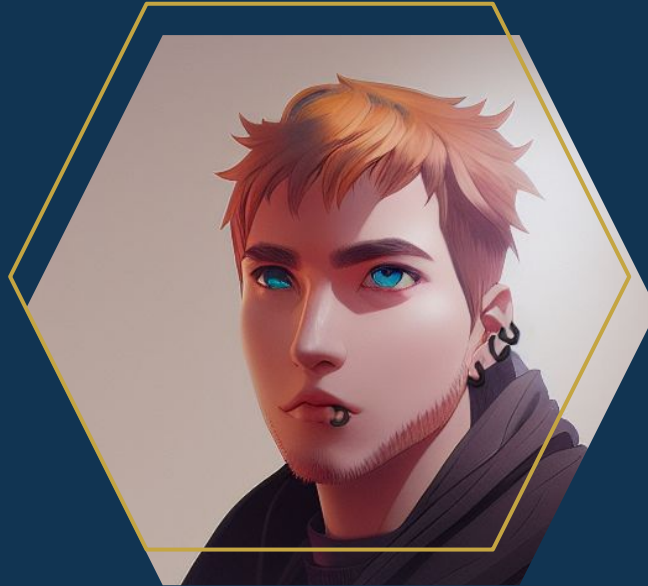
SE Bootcamp

Hyperiondev

The String Data Type

Welcome

Your Lecturer for This Session



Logan Meadows

Objectives

- Learn how to store and manipulate text using the String data type.

What Are “Strings”?

- ★ Strings are essentially any data made up of a sequence of letters or other characters.
- ★ Simply put, strings are just characters that have been “strung” together.

The String Data Type

- ★ Strings in Python are detected by quotation marks (" ") or inverted commas (' ').
- ★ Examples:

```
quotation_str = "The quick brown fox jumps over the lazy dog"  
inverted_comma_str = 'Strings are rather useful, what do you think?'
```

Concatenation of Strings

- ★ Strings can be added to one another. This is referred to as **concatenation**.
- ★ Example:

```
name = "Pieter"  
surname = "Parker"  
  
full_name = name + surname  
full_name = name + " " + surname
```

String Methods

- ★ String methods are ways to express an action in programming.
- ★ This is done through actions (also known as **methods**) that the code gets to do something.
 - Within the brackets of the method are its **arguments**.
 - Arguments are extra information given to the method.

String Methods cont.

★ Python has a number of built-in string methods:

★ `upper()`

★ `lower()`

★ `capitalize()`

★ `len()`

★ `strip()`

★ `join()`

★ `split()`

★ `replace()`

len() Example

- ★ The `len()` method will simply output the length value of a string.
- ★ Example:

```
message = "batman"  
message_len = len(message)  
print(message_len)  
  
# Result >> 6
```

upper() Example

- ★ The `upper()` method will take a string and convert all the characters to uppercase.
- ★ Example:

```
message = "PyThOn Is FuN"  
new_message = message.upper()  
print(new_message)  
  
# Result >> "PYTHON IS FUN"
```

lower() Example

- ★ The `lower()` method will take a string and convert all the characters to lowercase.
- ★ Example:

```
message = "PyThOn Is FuN"  
new_message = message.lower()  
print(new_message)  
  
# Result >> "python is fun"
```

capitalize() Example

- ★ The `capitalize()` method will take a string and convert the first letter to uppercase and the rest of the characters to lowercase, should there be any other uppercase characters.
- ★ Example:

```
message = "PyThOn Is FuN"  
new_message = message.capitalize()  
print(new_message)  
  
# Result >> "Python is fun"
```

strip() Example

- ★ The `strip()` method, will remove a symbol from a string. If you provide no argument for the method it will simply remove any blank spaces from the string.
- ★ Keep in mind that `strip()` will only remove from the **ends of the string**.
- ★ Example:

```
message = "****They've*taken*the*hobbits*to*Eisenguard!****"
message_strip = message.strip("*")
print(message_strip)

# Result >> "They've*taken*the*hobbits*to*Eisenguard"
```

split() Example

- ★ The `split()` method, will split a string by a symbol. However, once the split occurs the string will then be placed in what's called a **list**, which **can be indexed**.
- ★ Example:

```
message = "The-king-of-iron-fist"
message_split = message.split("-")
print(message_split)

# Result >> ["The", "king", "of", "iron", "fist"]
```

join() Example

- ★ The `join()` method will take a list of strings, and concatenate said strings to form one string.
- ★ Example:


```
list_example = ["The", "king", "of", "iron", "fist"]  
list_join = " ".join(list_example)  
print(list_join)  
  
# Result >> "The king of iron fist"
```

replace() Example

- ★ The `replace()` method will replace any specified character in a string with a new one. Keep in mind that `replace()` requires **two arguments**.
- ★ Example:

```
message = "Hey!you!over!there!"  
message_replace = message.replace("!", " ")  
print(message_replace)  
  
# Result >> "Hey you over there"
```


Indexing Strings



The diagram illustrates string indexing for the word "Hello". The word is displayed in a large, black, serif font. Below each character, there are two rows of indices. The first row shows positive indices starting from 0 for 'H' and ending at 4 for 'o'. The second row shows negative indices starting from -5 for 'H' and ending at -1 for 'o'. The characters are centered above their respective index pairs.

Character	Index
H	0
e	1
l	2
l	3
o	4

Character	Index
H	-5
e	-4
l	-3
l	-2
o	-1

Strings are basically a list of characters. An example here would be the word "Hello", which consists of the characters H+e+l+l+o.

String Slicing

- ★ String slicing is a way of **extracting multiple characters** from a string based on their **index position**.
- ★ Important to remember that this is done **character by character**, not word by word.
- ★ Example:

```
string = "Hello"  
  
string_idx = string[3]  
print(string_idx)  
  
# Result >> "l"  
  
string_slice = string[0:3]  
print(string_slice)  
  
# Result >> "Hel"
```

Escape Characters

- ★ Python uses the **backslash** (`\`) as an escape character.
- ★ The backdash is used as a marker to inform the compiler that the **next character has a special use / meaning**.
- ★ The backdash combined with specific other characters is known as an **escape sequence**.

Escape Characters cont.

- ★ Some useful escape characters:
 - `\n` - New line
 - `\t` - Tab space
 - `\s` - Space
- ★ The escape character can also be used for quoting in a string.
- ★ By placing a `backslash` in front of a quotation mark, you can tell the compiler to avoid terminating the string.

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic, should you have any.



Hyperiondev

Thank You for Joining Us