Intro to Strings

September 14, 2023

1 Introduction to Strings in Python

Strings are one of the most fundamental data types in Python and many other programming languages. In essence, a string is a sequence of characters. These characters can be letters, numbers, symbols, whitespace, or any combination thereof.

In programming terminology, a single character, such as 'a' or '1', is often referred to as a `char`. While some languages differentiate between a string and a char data type, in Python, both are considered strings regardless of their length.

1.1 Defining Strings

In Python, strings can be defined in several ways:

1. **Single Quotes:** This is a common way to define strings. For instance, `'Hello, World!'`.

   ```
   greeting = 'Hello, World!'
   ```

2. **Double Quotes:** They function the same way as single quotes. It's mostly a matter of preference or to handle cases where the string itself contains a single quote. Example: `"I'm learning Python!"`.

   ```
   statement = "I'm learning Python!"
   ```

3. **Triple Quotes:** These can be triple single quotes `'''` or triple double quotes `"""`. They're used to define multiline strings or strings that span across several lines. They're also commonly used for docstrings (documentation strings) in functions.

   ```
   paragraph = """This is a long string that spans
   across multiple lines. It's useful for large
   blocks of text."""
   ```

Regardless of which style you choose, it's essential to remain consistent throughout your code.

In conclusion, strings are a versatile data type in Python, and understanding how to define and manipulate them is foundational to many programming tasks, from simple output messages to complex text processing operations.

2 Title: Intro to Strings

How to define a sting 1. "Hello world!" 1. 'Hello world!' 1. """ Hello world """

Why? The python creator chose it

[37]:
```
'same'
```

[37]:
```
'same'
```

[38]:
```
"same"
```

[38]:
```
'same'
```

[41]:
```
"""same"""
```

[41]:
```
'same'
```

[36]:
```
'''same'''
```

[36]:
```
'same'
```

3 From Char to String

Basically, a string is like an array of a single character stings, but we will get to arrays later.

[47]:
```
my_char = 's'
```

[48]:
```
my_char2 = 'S'
```

[45]:
```
my_string = 'YaY WoW'
```

[46]:
```
my_string
```

[46]:
```
'YaY WoW'
```

[49]:
```
my_string
```

[49]:
```
'YaY WoW'
```

4 String Manipulation Based on Indexes

Strings in Python are sequences of characters, and each character has a unique position, or index, in the sequence. These indexes allow us to access, extract, and manipulate specific parts of the string. The indexing starts from `0` for the first character and goes up by `1` for each subsequent character.

4.1 Accessing a Single Character in a String

You can access a specific character in a string by referring to its index number. Use square brackets for slicing along with the index to obtain the character.

Example:

```
text = "Python"
print(text[0])  # Outputs: P
print(text[3])  # Outputs: h
```

4.2 Accessing a Part of a String (Slicing)

Python allows you to "slice" a string, which means extracting a part of it. You can specify where to start the slicing, and where to end.

The syntax is:

`string[start:end]`

Note: The character at the start index is included, but the character at the end index is not.

Example:

```
text = "Python"
print(text[1:4])  # Outputs: yth
```

You can also omit the start or end index:

- If you omit the start index, Python will start from the beginning.

  ```
  print(text[:4])  # Outputs: Pyth
  ```

- If you omit the end index, Python will go until the end.

  ```
  print(text[2:])  # Outputs: thon
  ```

4.3 Getting the End of a String

To get characters from the end of a string, you can use negative indices:

- `-1` refers to the last character, `-2` refers to the second last character, and so on.

Example:

```
text = "Python"
print(text[-1])    # Outputs: n
print(text[-3:])   # Outputs: hon
print(text[-4:-1]) # Outputs: tho
```

4.4 In Summary

String manipulation based on indexes is a powerful way to access and modify specific parts of your strings in Python. By understanding the index system and the concept of slicing, you can effectively work with parts of strings to fit your application's requirements.

[50]:
```
# New Operator []
my_string[0]
```

[50]:
```
'Y'
```

[51]:
```
print(my_string)
```

```
YaY WoW
```

[55]:
```
my_string[3]
```

[55]:
```
' '
```

[59]:
```
print(my_string[0:4])
print(my_string[4])
```

```
YaY
W
```

[66]:
```
my_string[9]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-66-a8aa16518d4b> in <cell line: 1>()
----> 1 my_string[9]

IndexError: string index out of range
```

[60]:
```
print(my_string[:4])
print(my_string[0:4])
```

```
YaY
YaY
```

[61]:
```
my_string[-1]
```

[61]:
```
'W'
```

[67]:
```
my_string[-9]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-67-93719b2ddbf7> in <cell line: 1>()
----> 1 my_string[-9]

IndexError: string index out of range
```

[72]:
```
my_string[-3:7]
```

[72]:
```
'WoW'
```

[80]:
```
# Write code that show the string without the space in between YaY and WoW
new_string = my_string[:3] + my_string[-3:]
new_string
```

[80]: 'YaYWoW'

[84]:
```
my_string[:3]
```

[84]: 'YaY'

[83]:
```
my_string[3:]
```

[83]: ' WoW'