Strings

In Python, strings are a sequence of characters and can be considered as a basic data structure. Though not explicitly classified as a data structure like lists or dictionaries, strings are an important part of data manipulation, and they are often used in various algorithms and data processing tasks.

**Key Properties of Strings in Python:**

1. **Immutable**: Strings in Python are immutable, meaning that once created, their contents cannot be changed. Any operation that modifies a string results in the creation of a new string.
2. **Sequence**: A string behaves like a sequence of characters (similar to lists or tuples), and you can access individual characters by their index.
3. **Indexed**: Strings are indexed, with the first character being at index 0.
4. **Supports slicing**: You can extract substrings using slice notation.

**Common Operations on Strings**

Here are some common operations you can perform on strings in Python:

**1. String Creation**

You can create a string by enclosing characters in single quotes (') or double quotes ("), both are equivalent.

python

Copy code

string1 = 'Hello'

string2 = "World"

**2. String Concatenation and Repetition**

You can concatenate strings using the + operator and repeat them using the \* operator.

python

Copy code

s1 = "Hello"

s2 = "World"

s3 = s1 + " " + s2 # Concatenation

print(s3) # Output: Hello World

s4 = "Hi" \* 3 # Repetition

print(s4) # Output: HiHiHi

**3. String Length**

The len() function returns the length of a string.

python

Copy code

string = "Python"

length = len(string)

print(length) # Output: 6

**4. Accessing Characters (Indexing)**

You can access individual characters in a string using indices. Negative indices count from the end of the string.

python

Copy code

string = "Hello"

print(string[0]) # Output: H

print(string[-1]) # Output: o

**5. Slicing**

Slicing allows you to get a substring of a string.

python

Copy code

string = "Hello, World!"

substring = string[0:5] # Extracts from index 0 to 4 (exclusive)

print(substring) # Output: Hello

substring = string[7:] # Extracts from index 7 to the end

print(substring) # Output: World!

substring = string[:5] # Extracts from the start to index 4 (exclusive)

print(substring) # Output: Hello

**6. String Methods**

Python provides a variety of built-in methods to manipulate strings:

* lower()/upper() – Converts the string to lowercase/uppercase.
* strip() – Removes leading and trailing whitespace.
* replace(old, new) – Replaces occurrences of a substring.
* split(separator) – Splits a string into a list based on a separator.
* join(iterable) – Joins a list of strings into a single string.

Example:

python

Copy code

string = " Hello, World! "

print(string.lower()) # Output: hello, world!

print(string.upper()) # Output: HELLO, WORLD!

print(string.strip()) # Output: Hello, World!

print(string.replace("World", "Python")) # Output: Hello, Python!

# Splitting a string into a list

words = string.split(", ")

print(words) # Output: ['Hello', 'World!']

# Joining a list of strings

new\_string = " ".join(words)

print(new\_string) # Output: Hello World!

**7. Finding Substrings**

You can use find() or index() to search for a substring in a string. find() returns the index of the first occurrence of the substring (or -1 if not found), while index() raises a ValueError if not found.

python

Copy code

string = "Hello, World!"

position = string.find("World") # Returns the starting index of "World"

print(position) # Output: 7

# If the substring is not found, find returns -1

position = string.find("Python")

print(position) # Output: -1

**8. String Formatting**

You can format strings using f-strings, format(), or the older % formatting.

python

Copy code

name = "Alice"

age = 30

# Using f-strings (Python 3.6+)

formatted\_string = f"Hello, my name is {name} and I am {age} years old."

print(formatted\_string)

# Using str.format()

formatted\_string2 = "Hello, my name is {} and I am {} years old.".format(name, age)

print(formatted\_string2)

# Using the older style (%)

formatted\_string3 = "Hello, my name is %s and I am %d years old." % (name, age)

print(formatted\_string3)

**9. String Immutability**

Since strings are immutable, you cannot modify a string directly. However, you can create a new string with the desired modification.

python

Copy code

string = "Hello"

new\_string = string.replace("e", "a") # Replaces 'e' with 'a'

print(new\_string) # Output: Hallo

**10. Escape Sequences**

You can use escape sequences in strings for special characters like newlines (\n), tabs (\t), and quotes.

python

Copy code

string = "Hello\nWorld"

print(string)

# Output:

# Hello

# World

string2 = "He said, \"Python is awesome!\""

print(string2) # Output: He said, "Python is awesome!"

**Advanced String Usage**

1. **String as a Data Structure in Algorithms**:  
   Strings are often used in algorithms that require pattern matching, text processing, or manipulation. For example:
   * Searching for substrings (e.g., the KMP algorithm).
   * Regular expressions for pattern matching.
2. **Text Parsing**:  
   Strings are used in applications that require text parsing, such as log file analysis, data extraction, etc.
3. **String Interning**:  
   Python optimizes memory by reusing identical string objects (string interning). This is useful in cases where there are a lot of identical strings (like keywords in a program).

python

Copy code

a = "hello"

b = "hello"

print(a is b) # Output: True (since Python reuses identical strings)

Top of Form

Bottom of Form