String Basics

Strings are one of the fundamental Python data types. The term data type refers to what kind of data a value represents. Strings are used to represent text.

A string in Python is a sequence of characters enclosed within single quotes ('), double quotes ("), or triple quotes ''' or """).

Example:

```
single_quote_str = 'Hello, World!'

double_quote_str = "Hello, World!"

triple_quote_str = """Hello, World!"""
```

In Python, you can use single quotes ('), double quotes ("), or triple quotes ''' or """) to define strings. While they serve the same basic purpose of creating string literals, they have distinct uses and properties.

Single Quotes (')

- Single quotes are useful for creating simple string literals.
- They can be used interchangeably with double quotes.
- If your string contains a single quote, you can avoid escaping it by using double quotes to define the string.

```
single_quote_str = 'Hello, World!'

with_apostrophe = "It's a beautiful day!"  # Using double quotes to include a single quote
```

Double Quotes (")

- Double quotes function similarly to single quotes and can be used interchangeably.
- They are handy when your string contains double quotes, allowing you to avoid escaping them by using single quotes to define the string.

```
double_quote_str = "Hello, World!"
with_double_quotes = 'She said, "Hello, World!"'  # Using single quotes to include double quotes
```

Triple Quotes (''' or """)

- Triple quotes are primarily used for multi-line strings.
- They are also useful for writing string literals that include both single and double quotes without needing to escape them.
- Triple quotes are often used for docstrings, which are multi-line comments at the beginning of functions, classes, or modules.

```
multi_line_str = '''This is a multi-line string that spans multiple lines.'''
doc_string_example = """This function demonstrates
how to use triple quotes for
multi-line strings and docstrings."""
string_with_quotes = '''He said, "It's a beautiful day!"'''
```

```
>>> type("Hello, World")
<class 'str'>
```

The output <class 'str'> indicates that the value "Hello, World" is an instance of the str data type.

Basic Operations

Concatenation

You can concatenate (combine) strings using the + operator.

```
str1 = "Hello"
str2 = "World"
result = str1 + ", " + str2 + "!"
print(result)  # Output: Hello, World!
```

Repetition

You can repeat a string using the * operator.

```
str1 = "Ha"
result = str1 * 3
print(result)  # Output: HaHaHa
```

Length

You can find the length of a string using the len() function.

```
str1 = "Hello, World!"
print(len(str1))  # Output: 13
```

String Indexing

Each character in a string has a numbered position called an index.

You can access the character at the nth position by putting the number n between two square brackets ([]) immediately after the string:

```
>>> flavor = "fig pie"
>>> flavor[1]                # 'i'
```

If you try to access an index beyond the end of a string, then Python raises an IndexError:

```
>>> flavor[9]
Traceback (most recent call last):
File "<pyshell#4>", line 1, in <module>
flavor[9]
IndexError: string index out of range
```

To get the last character use -1 and onwards to get string in reverse order

```
last_character = user_input[-1]
```

Slicing

Slicing allows you to extract a substring from a string. The syntax for slicing is string[start:end:step].

start: The starting index (inclusive).

end: The ending index (exclusive).

step: The step size (optional).

```
str1 = "Hello, World!"
print(str1[0:5])   # Output: Hello
print(str1[7:12])  # Output: World
print(str1[::2])   # Output: Hlo ol!
print(str1[::-1])  # Output: !dlroW ,olleH
```

Strings Are Immutable

Strings are immutable, which means that you can't change them once you've created them. For instance, see what happens when you try to assign a new letter to one particular character of a string:

```
>>> word = "goal"
>>> word[0] = "f"
Traceback (most recent call last):
File "<pyshell#16>", line 1, in <module>
word[0] = "f"
TypeError: 'str' object does not support item assignment
```

Common String Methods

upper()

Converts all characters to uppercase.

```
str1 = "hello"
print(str1.upper())  # Output: HELLO
```

lower()

Converts all characters to lowercase.

```
str1 = "HELLO"
print(str1.lower())  # Output: hello
```

capitalize()

Capitalizes the first character of the string.

```
str1 = "hello world"
print(str1.capitalize())  # Output: Hello world
```

title()

Converts the first character of each word to uppercase.

```
str1 = "hello world"
print(str1.title())  # Output: Hello World
```

strip()

Removes leading and trailing whitespace.

```
str1 = "   hello world   "
print(str1.strip())  # Output: hello world
```

split()

Splits the string into a list of substrings based on a delimiter (default is space).

```
str1 = "hello world"
print(str1.split())  # Output: ['hello', 'world']
```

join()

Joins a list of strings into a single string with a specified delimiter.

```
words = ['hello', 'world']
print(' '.join(words))  # Output: hello world
```

find()

Returns the index of the first occurrence of a substring. Returns -1 if the substring is not found.

str.find(sub[, start[, end]])

sub: The substring to be searched for within the string.

start (optional): The starting index from where the search should begin.

end (optional): The ending index where the search should terminate.

If the substring is found, find() returns the index of the first occurrence.

If the substring is not found, find() returns -1.

```
str1 = "hello world"
print(str1.find("world"))  # Output: 6
```

replace()

Replaces all occurrences of a substring with another substring.

str.replace(old, new, count)

- old: The substring to be replaced.
- new: The substring to replace the old substring with.
- count (optional): The maximum number of occurrences to replace. If omitted, all occurrences are replaced.

```
str1 = "hello world"
print(str1.replace("world", "Python"))  # Output: hello Python
```

Quiz

1. What does the count() method do?
   - A) It counts the number of times a substring appears in a string.
   - B) It counts the total number of characters in a string.
   - C) It counts the number of words in a string.
   - D) It counts the number of vowels in a string.
2. What will be the output of the following code?

```
str1 = "   python   "
print(str1.lstrip() + "is fun")
```

   A) "python is fun" B) " python is fun"
   C) "python is fun" D) "python is fun"

3. What will be the output of the following code?

```
str1 = "hello world"
print(str1.find("world"))
```

   A) 6 B) -1 C) 5 D) "world"

4. What is the result of the following code?

```
s = "hello"
print(s.replace("l", "x", 1))
```

   A) "hexxo" B) "hexx" C) "hexlo" D) "helxo"

5. Given the string s = "hello", what does s.find("z") return?

   A) 0 B) -1 C) False D) Error

6. What does the following code output?

```
s = "abcdef"
print(s[1:5])
```

   A) "abcd" B) "bcde" C) "bcdef" D) "abcde"

Correct Answer

1. Correct Answer: A) It counts the number of times a substring appears in a string.
2. Correct Answer: A) "python is fun"
3. Correct Answer: A) 6
4. Correct Answer: D) "helxo"
5. Correct Answer: B) -1
6. Correct Answer: B) "bcde"