String in Python

what is string in Python?

- Definition: In Python, a string is a sequence of characters enclosed in single quotes ('...'), double quotes ("..."), or triple quotes ('''...''' or """..."""). Triple quotes are used for multi-line strings.
- Immutability: Strings in Python are immutable, which means once a string is created, it cannot be modified. Operations that seem to modify a string actually create a new string.
- Indexing and Slicing: Strings are indexed sequences of characters, where indexing starts at 0. You can access individual characters using indices and extract substrings using slicing.
- Concatenation and Repetition: You can concatenate strings using the + operator and repeat them using the * operato.
- String Methods: Python provides a rich set of built-in methods for string manipulation.
- Formatting: Strings can be formatted using the % operator, the str.format() method, or f-strings

Example

```
my_string = 'Hello, World!'
my_string = "Hello, World!"
my_string = '''Hello, World!'''
my_string = """Hello, World!"""
```

Represntation of String

```
string1 = 'This is a string.'
string2 = 'string with single quotes.'
string3 = "12345"
string4 = "Hello world!"
string5 = "python is fun!"
```

String Operations:

1.Concatenation:

String concatenation means add strings together

```
string1 = 'This is a string.'
string2 = 'string with single quotes.'
string3 = "12345"
string4 = "Hello world!"
string5 = "python is fun!"
result = string1 +" " + string2+" "+ string3 +" "+ string4 +" "+
string5
print(result)
```

```
This is a string. string with single quotes. 12345 Hello world! python
is fun!
```

2.Repetiton: In Python we can repeat a string by using the * operator.

```
string6 = 'This is a string. \n'
string7 = 'string with single quotes. \n'
string8 = "12345 \n"
string9 = "Hello world! \n"
string10 = "python is fun! \n"
result1 = string6 * 4
print(result1)
result2 = string7 * 4
print(result2)
result3 = string8 * 4
print(result3)
result4 = string9 * 4
print(result4)
result5 = string10 * 4
print(result5)
```

```
This is a string.
This is a string.
This is a string.
This is a string.

string with single quotes.
string with single quotes.
string with single quotes.
string with single quotes.

12345
12345
12345
```

```
12345

Hello world!
Hello world!
Hello world!
Hello world!

python is fun!
python is fun!
python is fun!
python is fun!
```

Python String index

- The index method find the first occurrence of thr specified value. it raise an exception if the value is not found.

Example:

```
string500 = "In the heart of the bustling city, where skyscrapers stretch towards the clouds and the streets are alive with the rhythm of urban life, a sense of wonder and excitement permeates the air. The diverse culture and vibrant energy create a mosaic of experiences that captivate the senses. As the sun sets, the city transforms into a dazzling spectacle of lights and sounds, offering endless opportunities for adventure and discovery. Each corner holds a new story, waiting to be explored and cherished by those who wander through its dynamic landscape."
print(string500[7])
print(string500[21])
print(string500[47])
print(string500[105])
print(string500[129])
print(string500[201])
print(string500[249])
print(string500[311])
print(string500[395])
print(string500[409])
print(string500[418])
print(string500[420])

h
u
a

n
c
```

```
p

n
e
d
d
```

Python string slicing

slicing is about obtain a sub-string from the given string by slicing ,it can respectively from start to end.

Example:

```
string500 = "In the heart of the bustling city, where skyscrapers stretch towards the clouds and the streets are alive with the rhythm of urban life, a sense of wonder and excitement permeates the air. The diverse culture and vibrant energy create a mosaic of experiences that captivate the senses. As the sun sets, the city transforms into a dazzling spectacle of lights and sounds, offering endless opportunities for adventure and discovery. Each corner holds a new story, waiting to be explored and cherished by those who wander through its dynamic landscape."
print(string500[0:10])
print(string500[109:122])
print(string500[200:243])
print(string500[320:350])
print(string500[400:420])
print(string500[-11:-2])
```

```
In the hea
h the rhythm
 culture and vibrant energy create a mosaic
ms into a dazzling spectacle o
s for adventure and
 landscap
```

Python string Formatting

1.Formating string using % operator (old style formatting)

It is oldest method of string formatting

```
name = "John"
age = 25
print("%s is %d years old." % (name, age))
```

```
name = "Alice"
age = 30
print("%s is %d years old." % (name, age))
highscore = 100
print("The high score is %d" % highscore)
```

```
John is 25 years old.
Alice is 30 years old.
The high score is 100
```

2.How to Formate string using formate() method:

The format() method in Python is used to insert values into a string, allowing you to create a dynamic string with placeholders that are replaced by the values you provide.

```
name = "John"
age = 25
print("{} is {} years old.".format(name, age))
name = "Alice"
age = 30
print("{} is {} years old.".format(name, age))
highscore = 100
print("The high score is {}".format(highscore))
```

```
John is 25 years old.
Alice is 30 years old.
The high score is 100
```

3.Understanding python f-string:

In Python, f-strings, introduced in Python 3.6, provide a concise and efficient way to embed expressions inside string literals. The "f" in f-string stands for "formatted."

```
name = "John"
age = 25
print(f"{name} is {age} years old.")
name = "Alice"
age = 30
print(f"{name} is {age} years old.")
highscore = 100
print(f"The high score is {highscore}")
```

```
John is 25 years old.
Alice is 30 years old.
The high score is 100
```

Python String methods

1.Upper():

Converts all characters to uppercase.

```
text = "hello world"
print(text.upper())

HELLO WORLD
```

2.lower():

Converts all characters to lowercase.

```
text = "HELLO WORLD"
print(text.lower())

hello world
```

3.Capitalize():

Capitalies the first character of the string.

```
text = "hello world"
print(text.capitalize())

Hello world
```

4.Title():

Capitalize the first letter of each word.

```
text = "hello world"
print(text.title())

Hello World
```

5.Strip():

Remove leading and trailing whitespace.

```
text = "   hello world   "
print(text.strip())

hello world
```

6.lstrip():

Removes leading whitespace.

```
text = "   hello world   "
print(text.lstrip())
```

```
hello world
```

7.rstrip():

Removes trailing whitespace.

```
text = "   hello world   "
print(text.rstrip())
```

```
   hello world
```

8.split():

splits the string into a list by spaces

```
text = "hello world"
print(text.split())
```

```
['hello', 'world']
```

9.join():

join the elements of an iterable into a single string using the string as a separator

```
text = "hello world"
print("".join(text))
```

```
hello world
```

10.replace(old,new):

Replaces a substring with another substring.

```
text = "hello world"
print(text.replace("hello", "hi"))
```

```
hi world
```

11.find(substring):

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

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

```
6
```

12.rfind(substring):

Returns the index of the last occurrence of the substring or -1 if not found

```
text = "hello world"
print(text.rfind("world"))
```

```
6
```

13.count(substring):

Returns the number of times a substring occursin the string

```
text = "hello world"
print(text.count("l"))
```

```
3
```

14.Startswith(prefix):

Checks if strings with the specified prefix.

```
text = "hello world"
print(text.startswith("hello"))
```

```
True
```

15.endswith(suffix):

check if the string ends with the specified suffx.

```
text = "hello world"
print(text.endswith("world"))
```

```
True
```

16.format(*args,*kwargs):**

Formats the string using placeholder ({})

```
text = "Hello, {}. Welcome to {}."
print(text.format("Adam", "python"))
```

```
Hello, Adam. Welcome to python.
```

17.format_map(mapping):

formats the string using a dictionary.

```
text = "Hello, {name}. Welcome to {place}."
print(text.format_map({"name": "Adam", "place": "python"}))
```

```
Hello, Adam. Welcome to python.
```

18.isalpha():

checks if all characters are alphabetic.

```
text = "hello"
print(text.isalpha())

True
```

19.isalnum():

checks if all characters are alphanumeric.

```
text = "hello123"
print(text.isalnum())

True
```

20.isdigit():

checks if all characters are digits.

```
text = "123"
print(text.isdigit())

True
```

21.islower():

checks if all characters are lowercase.

```
text = "hello"
print(text.islower())

True
```

22.isupper():

checks if all characters are uppercase.

```
text = "HELLO"
print(text.isupper())

True
```

23.isspace():

check if all characters are whitespace

```
text = "   "
print(text.isspace())
```

```
True
```

24.swapcase():

swaps case:lowercase becomes uppercase and vice versa.

```
text = "Hello World"
print(text.swapcase())
```

```
hELLO wORLD
```

25.patition(separator):

splits the string into three parts:before the separator,the separator itself,and after.

```
text = "Hello World!"
print(text.partition(" "))
```

```
('Hello', ' ', 'World!')
```

26.rpartition(separator):

splits the string into three parts,starting from the last occurrence of the separator.

```
text = "Hello World!"
print(text.rpartition(" "))
```

```
('Hello', ' ', 'World!')
```

27.zfill(width):

pads the string with zeros on the left, to the specified width.

```
text = "42"
print(text.zfill(5))
```

```
00042
```

28.ljust(width,filchar):

left-justifies the string in a field of the specified width.

```
text = "hello"
print(text.ljust(10, "*"))
```

```
hello*****
```

29.rjust(width,filchar):

right-justifies the string in a field of the specified width.

```
text = "hello"
print(text.rjust(10, "*"))
```

```
*****hello
```

**30.center(width,fillchar):

center the string in a field of the specified width

```
text = "hello"
print(text.center(10, "*"))
```

```
**hello***
```

31.len():

returns length of the string.

```
text = "hello"
print(len(text))
```

```
5
```