Day 2 Session 1

Strings and their methods

Strings

A string is a data structure in Python that represents a sequence of characters. It is an immutable data type, meaning that once you have created a string, you cannot change it. Strings are used widely in many different applications, such as storing and manipulating text data, representing names, addresses, and other types of data that can be represented as text.

Example:

```
print("A Computer Science portal for geeks" )
```

Output:

```
A Computer Science portal for geeks
```

String slicing

String slicing in Python is about obtaining a sub-string from the given string by slicing it respectively from start to end.

Python slicing can be done in two ways:

- Using a slice() method
- Using the array slicing [::] method

Method 1: Using the slice() method

The slice() constructor creates a slice object representing the set of indices specified by range(start, stop, step).

Syntax:

- *slice(stop)*
- *slice(start, stop, step)*

Parameters: start: *Starting index where the slicing of object starts.* ***stop:*** *Ending index where the slicing of object stops.* ***step:*** *It is an optional argument that determines the increment between each index for slicing.* ***Return Type:*** *Returns a sliced object containing elements in the given range only.*

Method 2: Using the List/array slicing [::] method

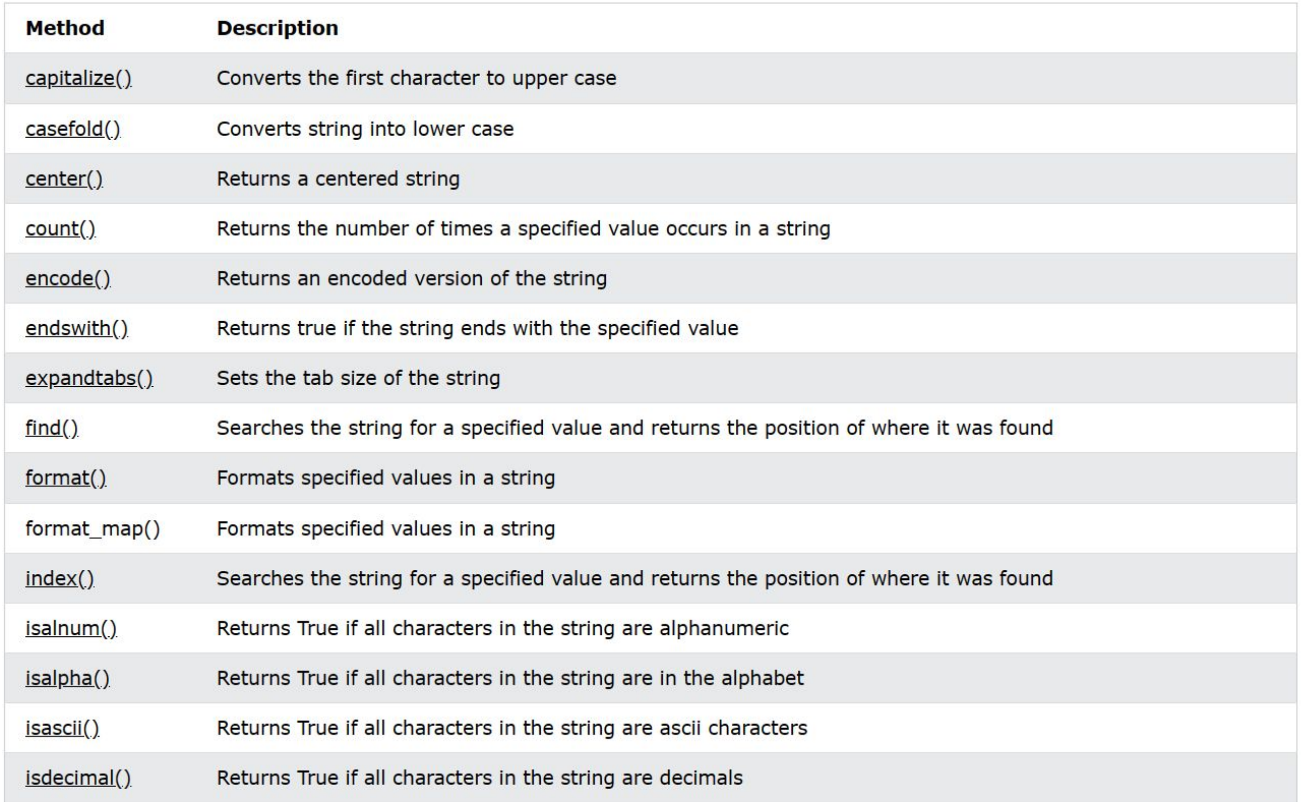
In Python, indexing syntax can be used as a substitute for the slice object. This is an easy and convenient way to slice a string using list slicing and Array slicing both syntax-wise and execution-wise. A start, end, and step have the same mechanism as the slice() constructor.

Below we will see **string slicing in Python with examples.**

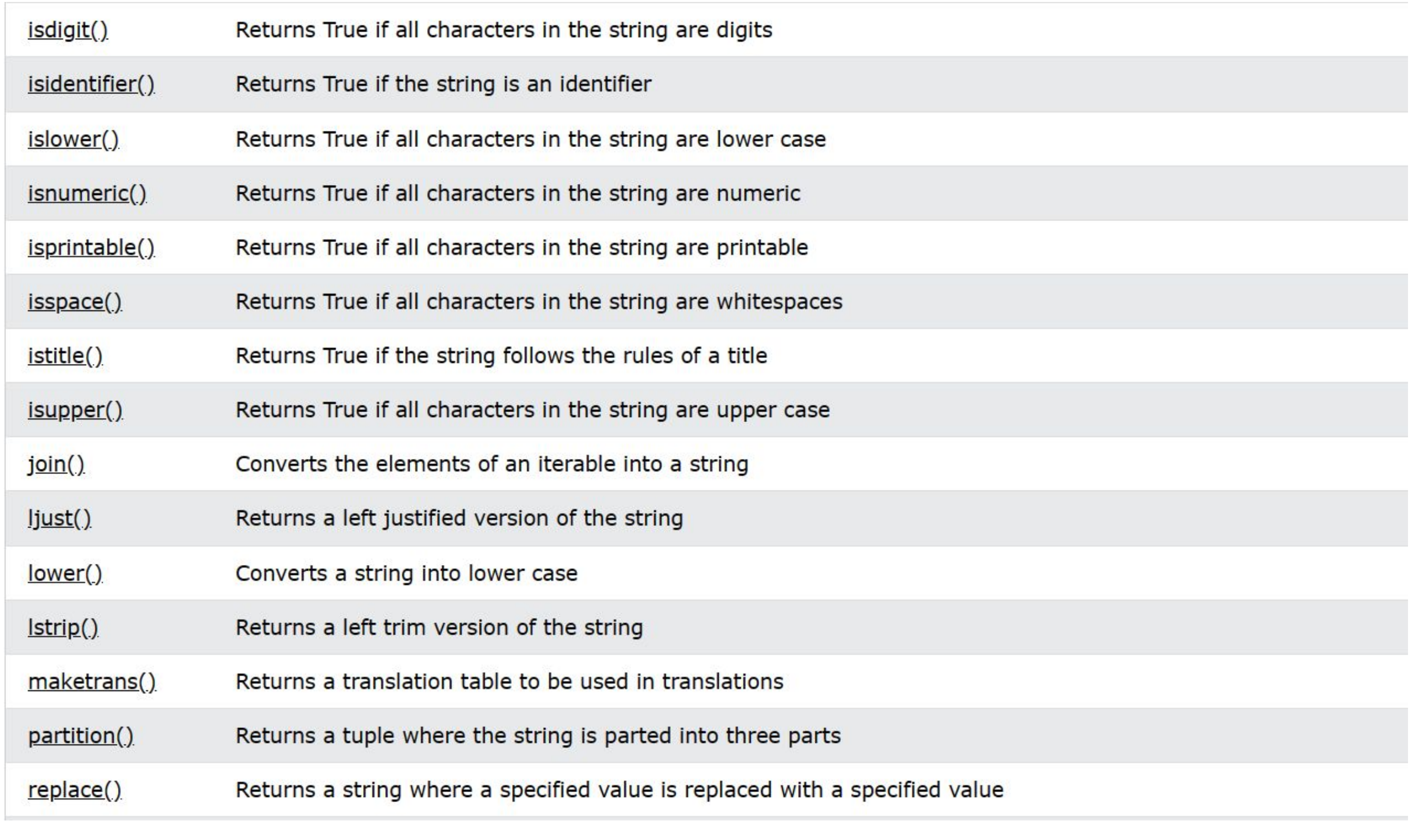
Syntax

```
arr[start:stop]         # items start through stop-1

arr[start:]             # items start through the rest of the array

arr[:stop]              # items from the beginning through stop-1

arr[:]                  # a copy of the whole array

arr[start:stop:step]    # start through not past stop, by step
```

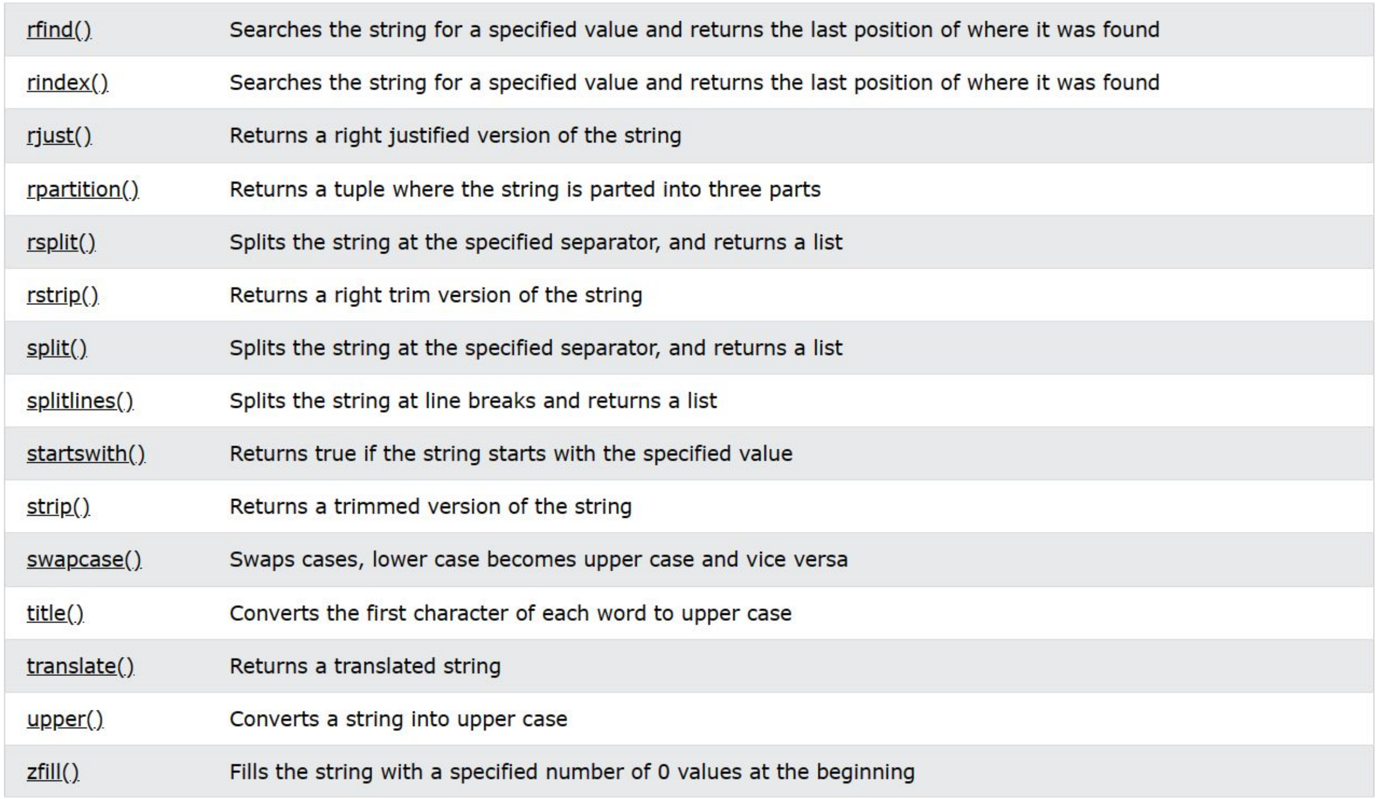
String Methods



Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isascii()	Returns True if all characters in the string are ascii characters
isdecimal()	Returns True if all characters in the string are decimals



isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Converts the elements of an iterable into a string
ljust()	Returns a left justified version of the string
lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value



rfind()	Searches the string for a specified value and returns the last position of where it was found
rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case
zfill()	Fills the string with a specified number of 0 values at the beginning

String Program Example

Write a Python program to calculate the length of a string

Code:

```
def string_length(str1):
  count = 0
  for char in str1:
    count += 1
    return count print(string_length('Computers'))
```

Output:

9