**Module 6 -** Strings

Contents

[6 Strings 1](#_5834k7go3cdb)

[6.1 What is a String? 2](#_59yz126h6hd2)

[6.2 String Creation and Initialization 2](#_nmo1a3qwd970)

[6.3 Accessing Characters in a String 2](#_18xgbj4b1s3y)

[6.4 String Slicing 3](#_87p5lgocio9t)

[6.5 String Methods 3](#_8i8nzmfu79sp)

[6.6 String Formatting 4](#_e8yduvvb20c0)

[6.7 String Comparison 5](#_7dhr411lfdh9)

[6.8 in Operator 6](#_fhk0t0gmq26x)

[6.9 Looping Through Strings 7](#_wmu5xwelur0m)

[6.10 References 9](#_pn24hm7gox2u)

# 

## Strings

[Strings](https://www.w3schools.com/python/python_strings.asp) are one of the most essential data types in Python. They are used to store and manipulate text and come with a powerful set of built-in features and methods.

## What is a String?

In Python, a [string](https://www.w3schools.com/python/python_strings.asp) is a sequence of Unicode characters enclosed within either single quotes ' or double quotes ". You can also use triple quotes (''' or """) for multi-line strings. They are used to store and manipulate text such as words, sentences, or even entire documents.

## String Creation and Initialization

Python strings are immutable, which means once a string is created, it cannot be changed. Instead, operations on strings create new ones. [Initializing a string variable](https://www.geeksforgeeks.org/python/how-to-initialize-a-string-in-python/) is straightforward and can be done in several ways.

|  |
| --- |
| str1 = 'Hello'  str2 = "World"  str3 = ' ' 'Hello! Welcome to the world of programming.' ' '  str4 = str(123)  str5 = str(3.14159)  text = str1+str2 #Hello World  line =str4\*4 #123123123123 |

The + will concatenate two strings, and the \* will repeat a string as shown above.

## Accessing Characters in a String

[Each character in a string has an index](https://www.programiz.com/python-programming/string) starting from 0. Entering a negative number as the index will count backwards from the last character in the string.

|  |
| --- |
| word = "Python"  print(word[0]) # P  print(word[3]) # h  print(word[-1]) # n |

## String Slicing

You can extract a substring using [slicing](https://www.tutorialspoint.com/python/python_slicing_strings.htm). The number in front of the : symbol represents the starting position of the slice and the second represents the position of the character at the end. You can also use negative numbers to remove characters as shown in the final example.

|  |
| --- |
| text = "Hello, world!"  print(text[0:5]) # Hello  print(text[1:3]) # el  print(text[7:]) # world!  print(text[:]) # Hello, world!  print(text[:-1]) # Hello, world |

## String Methods

Python provides a wide variety of [built-in methods](https://www.geeksforgeeks.org/python/python-string-methods/) to manipulate strings.

**Common Methods:**

* **strip():** By default, the strip method removes any whitespaces at the beginning and end from the string.
* **lower():** The lower method simply converts and uppercase letters into lowercase letters.
* **upper():** The upper method works the same way, but instead converts lowercase letters to uppercase.
* **replace():** This method replaces any substring given in the first parameter that exists in the full string with the substring given in the second parameter.
* **split():** The split method is used to create a list of words that exist in the string. By default split() will consider whitespace as the separator and all characters before and after each whitespace will be different values in the list created.
* **find():** The find method returns the place in the string where the given parameter first occurs. You can also specify the starting point and endpoint for this method to check through. If the value is not found in the string, it will return -1.
* **index():** The index method is very similar to find(), the only difference is that if the value is not found this method will raise an exception which means any line of code after the index method will not be run.
* **count():** The count method simply returns the number of times a given value appears in the string. You can also specify a start and end point if desired.
* **startswith():** This method returns a boolean value (True or False) corresponding to whether or not the string starts with the given value. Again, a start and end point can be specified.
* **capitalize():** The capitalize method returns a string with the first letter of the original string in uppercase, and the rest lowercase.
* **len():** The len method returns an integer representing the number of characters listed in the string.

|  |
| --- |
| s = " Hello, World! "  print(s.strip()) # "Hello, World!"  print(s.lower()) # " hello, world! "  print(s.upper()) # " HELLO, WORLD! "  print(s.replace("World", "Python")) # " Hello, Python! "  print(s.split(",")) # [' Hello', ' World! ']  print(s.find("World")) # 8  print(s.count("l")) # 3  print(s.startswith(" He")) # True  print(s.capitalize()) # " hello, world! "  print(len(s)) # 15 |

## 

## String Formatting

Formatting allows variables to be inserted into strings. This makes them more readable and allows you to concatenate strings without numbers.

* [**f-strings**](https://www.geeksforgeeks.org/python/formatted-string-literals-f-strings-python/)**:** F-strings are currently the recommended way to handle string values especially when printing them. This allows you to insert variables within the string by putting the variable inside the {} symbols.
* [**str.format()**](https://www.w3schools.com/python/ref_string_format.asp)**:** This method allows you to only put {} in the string with nothing inside, then at the end of the string you can add .format() and put the variables you want to insert within the parentheses, separated by commas, respective to their position in the string.
* [**%formatting**](https://www.tutorialspoint.com/python/string_formatting_operator.htm)**:** %formatting isn't used as frequently, but works similarly to str.format(). You can add %s to represent a string variable and %d for an integer, then add % (x, y) to indicate which variables you want to insert.

* [**str()**](https://www.w3schools.com/python/ref_func_str.asp)**:** The str() function simply converts the given variable (in this case an integer) into a string.

|  |
| --- |
| name = "Neda"  age = 30  # f-string (recommended)  print(f"My name is {name}, and I am {age} years old.")  # str.format()  print("My name is {}, and I am {} years old.".format(name, age))  # % formatting  print("My name is %s, and I am %d years old." % (name, age))  # %s → string, %d → integer  #str() method  print("My name is",name,"and I am "+str(age)+" years old.")  ### notice how using , after a string to insert a variable automatically separates it with a space but + does not. Keep this in mind if you notice your output has unexpected whitespace or a lack of intended whitespace. |

## String Comparison

[String comparisons](https://www.digitalocean.com/community/tutorials/python-string-comparison) use standard comparison operators and are case-sensitive. You can [compare strings](https://www.geeksforgeeks.org/python/string-comparison-in-python/) in Python using the equality (==) and comparison (<, >, !=, <=, >=) operators. For example, this could be used to check if a password is equal to the correct value.

|  |
| --- |
| fruit = 'Apple'  print(fruit == 'Apple') #True  print(fruit != 'Apple') #False  print(fruit < 'Apple') #False  print(fruit > 'Apple') #False  print(fruit <= 'Apple') #True  print(fruit >= 'Apple') #True |

## in Operator

If you want to check whether a particular substring exists in the string or not use the Python [string in operator](https://sparkbyexamples.com/python/python-string-in-operator-with-examples/). If the substring is present in the string, True is returned, otherwise False is returned. It is case-sensitive.

|  |
| --- |
| text = "Hello, world!"  if "world" in text:  print("The word 'world' is in the text.")    else:  print("The word 'world' is not in the text")  ### See how the result changes when you take "world" out of the text variable |

## Looping Through Strings

Looping through strings in Python can be achieved using several methods, with the for loop being the most common approach. This allows you to check each character individually. For example, you could use this to check the number of vowels in a string.

* [Iterating Directly over Characters:](https://www.geeksforgeeks.org/python/iterate-over-characters-of-a-string-in-python/) Loops through a string and prints each character every iteration.

|  |
| --- |
| str = "Python"  for char in str:  print(char) ###P  y  t  h  o  n |

There are other ways of iterating over the characters of a string:

* [Iterating by Index using range(len())](https://www.geeksforgeeks.org/python/iterate-over-a-list-in-python/): This returns indexes of each character in the string.

|  |
| --- |
| str = "Python"  for i in range(len(str)):  print(i, str[i]) #0 P  #1 y  #2 t  #3 h  #4 o  #5 n  ###for every character within the range of the string's length,  prints the index followed by the character at that same index |

* [Iterating with enumerate()](https://www.geeksforgeeks.org/python/iterate-over-characters-of-a-string-in-python/): enumerate gives both the index and the character simultaneously. This is a cleaner way when you need all characters and their index.

|  |
| --- |
| str = "Python"  for i, char in enumerate(str):  print(f"{i}: {char}") #0 P  #1 y  #2 t  #3 h  #4 o  #5 n  ###for every character within the range of the string's length,  prints the index followed by the character at that same index |

* **Using a while loop:** Using a while loop to iterate through strings allows you more control over how much of the string is gone through.

|  |
| --- |
| str = "Hello"  index = 0  while index < len(str) - 2:  char = str[index]  print(char)  index += 1  ### H  e  l |

Programming Exercises

1. **Character Frequency Checker**

Write a program that asks the user to enter a string and a character. Use a loop to count how many times the chosen character appears in the string.

1. **Reverse a String**

Write a program that asks the user to enter a string. Use a loop to reverse the string and display the reversed result.

1. **Vowel Counter**

Write a program that asks the user to enter a sentence. Use a loop to count the number of vowels (a, e, i, o, u) in the sentence and display the total.

1. **Palindrome Checker**

Write a program that asks the user to enter a word. Use a loop to check whether the word is a palindrome (reads the same forwards and backwards). Display a message indicating whether or not it is a palindrome.

1. **Word Lengths**

Write a program that asks the user to enter a sentence. Use a loop to display each word in the sentence along with its length.

1. **Middle Slice**

Write a program that asks the user to enter a word. Use slicing to display the middle three characters of the word.

1. **Uppercase and Lowercase**

Write a program that asks the user to enter a sentence. Use string methods to display the sentence in all uppercase letters, then in all lowercase letters.

1. **Word Splitter**

Write a program that asks the user to enter a sentence. Use the .split() method to break the sentence into words. Use a loop to display each word on a new line.

1. **Longest Word**

Write a program that asks the user to enter a sentence. Split the sentence into words and use a loop to find the longest word. Display the word and its length.

1. **Count Words of Given Length**

Write a program that asks the user to enter a sentence and a number. Split the sentence into words and use a loop to count how many words have exactly the given length. Display the count.

*This OER module incorporates and adapts content from CSCI1301 (****Introduction to Computer Programming, Application Development, Computer Science I & II****) Open Course**materials by Myungjae Kwak, Jonathan Jenkins, Joobum Kim, and Kevin Floyd from Middle Georgia State University, available on the ALG Manifold platform. Updates have been made to align the material with our course objectives, instructional approach, and current technological practices.*

## 6.10 References

<https://www.w3schools.com/python/python_strings.asp>

<https://www.geeksforgeeks.org/python/how-to-initialize-a-string-in-python/>

<https://www.programiz.com/python-programming/string>

<https://www.tutorialspoint.com/python/python_slicing_strings.htm>

<https://www.geeksforgeeks.org/python/python-string-methods/>

<https://www.geeksforgeeks.org/python/formatted-string-literals-f-strings-python/>

<https://www.w3schools.com/python/ref_string_format.asp>

<https://www.tutorialspoint.com/python/string_formatting_operator.htm>

<https://www.w3schools.com/python/ref_func_str.asp>

<https://www.digitalocean.com/community/tutorials/python-string-comparison>

<https://www.geeksforgeeks.org/python/string-comparison-in-python/>

<https://sparkbyexamples.com/python/python-string-in-operator-with-examples/>

<https://www.geeksforgeeks.org/python/iterate-over-characters-of-a-string-in-python/>

<https://www.geeksforgeeks.org/python/iterate-over-a-list-in-python/>

<https://www.geeksforgeeks.org/python/iterate-over-characters-of-a-string-in-python/>

Programming Exercise Solutions

1.

|  |
| --- |
| text = input("Enter a string: ")  char = input("Enter a character to count: ")  count = 0  for c in text:  if c == char:  count += 1  print(f"The character '{char}' appears {count} times in the string.") |

2.

|  |
| --- |
| text = input("Enter a string: ")  reversed\_text = ""  for char in text:  reversed\_text = char + reversed\_text  print("Reversed string:", reversed\_text) |

3.

|  |
| --- |
| text = input("Enter a sentence: ")  vowels = "aeiouAEIOU"  count = 0  for char in sentence:  if char in vowels:  count += 1  print("Number of vowels:", count) |

4.

|  |
| --- |
| word = input("Enter a word: ")  reversed\_word = ""  for char in word:  reversed\_word = char + reversed\_word  if word == reversed\_word:  print(f"{word} is a palindrome.")  else:  print(f"{word} is not a palindrome.") |

5.

|  |
| --- |
| text = input("Enter a sentence: ")  words = text.split()  for word in words:  print(f"{word}: {len(word)}") |

6.

|  |
| --- |
| word = input("Enter a word with 4 or more characters: ")  if len(word) < 3:  print("Word needs to be >3 letters long”)  else:  mid = len(word) // 2  middle\_three = word[mid - 1:mid + 2]  print("Middle three characters:", middle\_three) |

7.

|  |
| --- |
| text = input("Enter a sentence: ")  print("Uppercase:", text.upper())  print("Lowercase:", text.lower()) |

8.

|  |
| --- |
| text = input("Enter a sentence: ")  words = text.split()  for word in words:  print(word) |

9.

|  |
| --- |
| text = input("Enter a sentence: ")  words = text.split()  longest = ""  for word in words:  if len(word) > len(longest):  longest = word  print(f"The longest word is '{longest}' with length {len(longest)}.") |

10.

|  |
| --- |
| text = input("Enter a sentence: ")  x = int(input("Enter a word length to count: "))  words = text.split()  count = 0  for word in words:  if len(word) == x:  count += 1  print(f"There are {count} words with length {n}.") |