1.search():

Python regex re.search() method looks for occurrences of the regex pattern inside the entire target string and returns the corresponding Match Object instance where the match found.

The re.search() returns only the first match to the pattern from the target string. Use a re.search() to search pattern anywhere in the string.

Table of contents

How to use re.search()

Regex search example – look for a word inside the target string

Regex search example find exact substring or word

When to use re.search()

Search vs. findall

Regex search groups or multiple patterns

Search multiple words using regex

Case insensitive regex search

How to use re.search()

Before moving further, let’s see the syntax of it.

Syntax:

re.search(pattern, string, flags=0)



The regular expression pattern and target string are the mandatory arguments, and flags are optional.

pattern: The first argument is the regular expression pattern we want to search inside the target string.

string: The second argument is the variable pointing to the target string (In which we want to look for occurrences of the pattern).

flags: Finally, the third argument is optional and it refers to regex flags by default no flags are applied.

There are many flags values we can use. For example, the re.I is used for performing case-insensitive searching. We can also combine multiple flags using bitwise OR (the | operator).



Return value :

The re.search() method returns a Match object(i.e.,re.Match).

This match object contains the following two items.

The tuple object contains the start and end index of a successful match.



Second, it contains an actual matching value that we can retrieve using a group() method.



If the re.search() method fails to locate the occurrences of the pattern that we want to find or such a pattern doesn’t exist in a target string it will return a None type.



Now, Let’s see how to use re.search().

Python search regex pattern

Regex search example – look for a word inside the target string

Now, let’s see how to use re.search() to search for the following pattern inside a string.

Pattern: \w{8}

What does this pattern mean?

The \w is a regex special sequence that represents any alphanumeric characterr such as letters uppercase or lowercase, digits as well as the underscore character.

Then the 8 inside curly braces mean the characters have to occur exactly 8 times in a row inside the target string

In simple words, it means to search any eight-letter word

"Emma is a baseball player who was born on June 17, 1993."

As we can see in the above string baseball is the first eight-letter word inside the target string, so we should get the baseball as an output.

import re

# Target String

target\_string = "Emma is a baseball player who was born on June 17"

# search() for eight-letter word

result = re.search(r"\w{8}", target\_string)

# Print match object

print("Match Object", result)

# output re.Match object; span=(10, 18), match='baseball”

# print the matching word using group() method

print("Matching word: ", result.group())

# Output : 'baseball'

Let’s understand the above example.

First of all, I used a raw string to specify the regular expression pattern. As you may already know, the backslash has a special meaning in some cases because it may indicate an escape character or escape sequence. To avoid that we used raw string.

Also, we are not defining and compiling this pattern beforehand (like the compile method), The practice is to write the actual pattern in the string format.

Next, we wrote a regex pattern to search for any eight-letter word inside the target string.

Next, we passed this pattern to re.search() method to looks for occurrences of the pattern and it returned the re.Match object.

Next, we used the group() method of a re.Match object to retrieve the exact match value i.e., baseball.

Regex search example find exact substring or word

In this example, we will find substring “ball” and “player” inside a target string.

import re

# Target String

target\_string = "Emma is a baseball player who was born on June 17, 1993."

# find substring 'ball'

result = re.search(r"ball", target\_string)

# Print matching substring

print(result.group())

# output : 'ball'

# find exact word/substring surrounded by word boundary

result = re.search(r"\bball\b", target\_string)

if result:

print(result)

# output : None

# find word 'player'

result = re.search(r"\bplayer\b", target\_string)

print(result.group())

# output : 'player'

When to use re.search()

The search() method will always match and return only the first occurrence of the pattern from the target string.

Use it when you want to find the first match. The search method is useful for a quick match. I.e., As soon as it gets the first match, it will stop its execution. You will get performance benefits.

Also, please use it when you want to check the possibility of a pattern in a long target string.

Avoid using the search() method in the following cases

To search all occurrence to a regular expression, please use the findall() method instead.

To search at the start of the string, Please use the match() method instead. Also, read regex search() vs. match()

If you want to perform search and replace operation in Python using regex, please use the re.sub() method.

Search vs. findall

Both search and findall method servers the different purpose/use case when performing regex pattern matching in Python.

As you know, the search method scans the entire string to look for a pattern and returns only the first match. I.e., As soon as it gets the first match, it stops its execution.

On the other hand, The findall() method returns all matches to the pattern.

So use the findall() method to search all occurrence/possible matches to a regular expression.

One more difference, the search method returns a Match object which consists of the start and end index of a successful match and the actual matching value that we can retrieve using a group() method.

On the other hand, the findall() method returns all the matches in the form of a Python list.

Regex search groups or multiple patterns

In this section, we will learn how to search for multiple distinct patterns inside the same target string. Let’s assume, we want to search the following two distinct patterns inside the target string at the same time.

A ten-letter word

Two consecutive digits

To achieve this, Let’s write two regular expression patterns.

Regex Pattern 1: \w{10}

It will search for any six-letter word inside the target string

Regex Pattern 2: \d{2}

Now each pattern will represent one group. Let’s add each group inside a parenthesis ( ). In our case r"(\w{10}).+(\d{2})"

On a successful search, we can use match.group(1) to get the match value of a first group and match.group(2) to get the match value of a second group.

Now let’s see how to use these two patterns to search any six-letter word and two consecutive digits inside the target string.

Example to search multiple patterns

import re

target\_string = "Emma is a basketball player who was born on June 17."

# two group enclosed in separate ( and ) bracket

result = re.search(r"(\w{10}).+(\d{2})", target\_string)

# Extract the matches using group()

# print ten-letter word

print(result.group(1))

# Output : basketball

# print two digit number

print(result.group(2))

# Output : 17

Let’s understand this example

We enclosed each pattern in the separate opening and closing bracket.

I have added the .+ metacharacter before the second pattern. the dot represents any character except a new line and the plus sign means that the preceding pattern is repeating one or more times. So .+ means that before the first group, we have a bunch of characters that we can ignore

Next, we used the group() method to extract two matching values.

Note: The group() method returned two matching values because we used two patterns.

Also, read search for a regex pattern within a text file.

Search multiple words using regex

Let’s take another example and search any three words surrounded by space using regex. Let’s search words “emma”, “player”, “born” in the target string.

Use | (pipe) operator to specify multiple patterns.

import re

str1 = "Emma is a baseball player who was born on June 17, 1993."

# search() for eight-letter word surrounded by space

# \b is used to specify word boundary

result = re.findall(r"\bEmma\b|\bplayer\b|\bborn\b", str1)

print(result)

# Output : ['Emma', 'player', 'born']

Case insensitive regex search

There is a possibility that the string contains lowercase and upper case words or words with a combination of lower case and uppercase letters.

For example, you want to search a word using regex in a target string, but you don’t know whether that word is in uppercase or lowercase letter or a combination of both. Here you can use the re.IGNORECASE flag inside the search() method to perform case-insensitive searching of a regex pattern.

Example:

import re

# Target String

target\_string = "Emma is a Baseball player who was born on June 17, 1993."

# case sensitive searching

result = re.search(r"emma", target\_string)

print("Matching word:", result)

# Output : None

print("case insensitive searching")

# using re.IGNORECASE

result = re.search(r"emma", target\_string, re.IGNORECASE)

print("Matching word:", result.group())

# Output : 'Emma'