Theory: Search in a string

34 minutes 6 / 8 problems solved

Start practicing

One of the essential skills when working with data is to be able to search it and locate specific bits of information. Working with textual data in Python, you may need to get some information about its content: whether it includes a specific **substring** (i.e. part of the string), where this substring is, or how many times it occurs in the text. In this topic, we will learn how to do it.

§1. Membership testing

We'll start with the first question: how can we define if there's a specific pattern in our string? One way to do it is called **membership testing**, and it is implemented with the help of the operators `in` and `not in`. When we write `pattern in string`, the left operand should be a string, and membership test checks if `string` contains `pattern` as a substring.

If membership test returns `True`, this means that there exists a position in `string` starting from which you can read the pattern in the string.

```
print("apple" in "pineapple")  # True
print("milk" in "yogurt")      # False
```

Interestingly, an empty string is considered to be a substring of any string.

```
print('' in '')           # True
print('' not in "lemon")  # False
```

§2. Boolean search in a string

Apart from knowing that a substring just occurs in the string, we can determine that the string *starts* or *ends* with a specific pattern. Methods `startswith()` and `endswith()` return `True` if the pattern is found and `False` otherwise.

```
email = "email_address@something.com"
print(email.startswith("www."))          # False
print(email.endswith("@something.com"))  # True
```

Optional values for `start` and `end` that bound the search area can be added: `string.startswith(pattern, start, end)`. When we specify only one additional element, it's automatically considered as `start`.

```
email = "my_email@something.com"
print(email.startswith("email", 2))  # False
print(email.startswith("email", 3))  # True
```

In the example above, when we specified the `start` argument as **2**, we limited the search to the substring `"_email@something.com"`, which actually doesn't start with `"email"`. Then we fixed this off-by-one mistake by setting start to **3**.

Note that the substring bound by the start and end indexes *does* include the character with the start index but *does not* include the element with the end index.

```
email = "my_email@something.com"
print(email.endswith("@", 5, 8))  # False
print(email.endswith("@", 5, 9))  # True
```

The substring defined for the search in the first case is `"ail"`, while in the second one it's `"ail@"`.

§3. Element position

Current topic:

- ✓ Search in a string 4★

Topic depends on:

- ✓ Indexes 7★ — Stage 3
- ✓ Declaring a function 10★ — Stage 1
- ✓ If statement 16★ — Stage 1

Table of contents:

Now, as we know how to check if a string contains a substring, starts or ends with it, let's learn how to define the exact position of the substring. We can use the methods `find()` or `index()` to do so:

```
best = "friend"

print(best.find("i"))   # 2
print(best.index("i"))  # 2
```

They work absolutely the same except that the former returns `-1` if it can't find the given element, while the latter raises `ValueError`.

```
print(best.find("u"))   # -1
print(best.index("u"))  # ValueError
```

So, all the examples with `find()` below will work with `index()` as well.

We can search both for single characters and for longer substrings. In the latter case, the index of the **first character** of the substring is returned.

```
print(best.find("end"))  # 3
```

In the string `friend`, the substring `end` occupies positions from **3** to **5**, and the start index is returned. Keep in mind that both methods return only the index of the first occurrence of the element we search for.

```
magic = "abracadabra"
print(magic.find("ra"))  # 2
```

However, we can additionally specify an interval for searching, just as with the boolean search: `string.find(pattern, start, end)`.

```
print(magic.find("ra", 5))      # 9
print(magic.find("ra", 5, 10))  # -1
```

Once again, the end index is not included in the search area.

Alternatively, we can use methods `rfind()` and `rindex()` to search backward from the end of the string.

```
print(magic.rfind("ra"))  # 9
print(magic.rindex("a"))  # 10
```

§4. Element number

Finally, it's often useful to count how many times an element (a char or a substring) occurs in the string, and for this, we can use the method `count()`.

```
magic = "abracadabra"

print(magic.count("abra"))  # 2
print(magic.count("a"))     # 5
```

§5. Conclusions

In this topic, we have examined different aspects of searching through a string and learned how to locate specific patterns. Now you will be able:

- To test for membership in a text,
- To check that the string starts or ends with a specific pattern,
- To find the exact position of a substring,
- To count how many times a pattern occurs in the text.

This knowledge will be helpful in real-world tasks, so let's practice it!

Report a typo

366 users liked this theory. **3** didn't like it. **What about you?**

Start practicing

Comments (12) Hints (0) Useful links (0) Show discussion