15-strings-II

March 17, 2020

1 Functions on Strings

Python is a language particularly useful when it comes to string processing [1]. This is because it provides a variety of useful functions for string processing and transformation. In this lecture we will look at a few of them that will make our lives much easier when programming. To know more about other string functions available in python, check the documentation here.

All of the function we will see are part of the string *class*, which means they are called using the dot `.` notation: if `s` is a variable containing a string, and `f()` is a string function, we call `f` on `s` like this: `s.f()`. Depending on the function, it may take arguments.

Observe that **none of the functions below modify the string**.

[1] As a consequence, it is also a great language for file processing, since files are, most of the times, nothing more than really long strings.

1.1 split

The `split()` function splits the string into parts. If we pass no parameters to it, it will split the string at whitespaces and linebreaks, and return a list of the parts. Alternatively, we can pass to it a string parameter indicating what it should split on. For example, if `s = "hello,world"` then calling `s.split(",")` will return the list `["hello", "world"]`.

Another related function that might be useful is `splitlines()`, which will return a list with all the lines of the string.

```
In [1]: s = """Philosophy is the study of general and fundamental questions
        about existence, knowledge, values, reason, mind, and language.
        Such questions are often posed as problems to be studied or resolved.
        Philosophical methods include questioning, critical discussion,
        rational argument, and systematic presentation. Classic philosophical
        questions include: Is it possible to know anything and to prove it?
        What is most real? Philosophers also pose more practical and concrete
        questions such as: Is there a best way to live? Is it better to be
        just or unjust (if one can get away with it)? Do humans have free will?"""

        sentences = s.split(".") # Splits at final stops
        words = s.split()        # Splits at blank spaces and linebreaks
        lines = s.split("\n")    # Same as s.splitLines()

        print(sentences)
```

```
        print(words)
        print(lines)
```

```
['Philosophy is the study of general and fundamental questions\nabout existence, knowledge, val
['Philosophy', 'is', 'the', 'study', 'of', 'general', 'and', 'fundamental', 'questions', 'about
['Philosophy is the study of general and fundamental questions', 'about existence, knowledge, v
```

1.2 strip

The `strip()` function removes spurious content from the beginning and end of a string. When called without parameters, it removes whitespaces.

```
In [2]: s = """      Hello world

        """

        s_clean = s.strip()
        print(s_clean)
```

```
Hello world
```

When called with a string as a parameter, it removes all characters that are in that string. Intuitively, if we use `r` as a parameter, as in `s.strip(r)`, then `r` acts like a set of characters that must be removed from the beginning or end of `s`. Note that, since it is a set, they will be removed in any order. As soon as a character that is not in `r` is encountered, the stripping stops.

```
In [3]: s = "www.anAwesomeSite.com"
        s_stripped = s.strip("w.com") # Note that the inner w and m are not removed
        print(s_stripped)
```

```
anAwesomeSite
```

1.3 lower & upper

The `lower` and `upper` functions return the string where its aphabetic characters are replaced by their lower- or upper-case versions, respectively. All other characters remain the same.

```
In [4]: s = "1984 is a dystopian novel by English novelist George Orwell."
        sU = s.upper()
        print(sU)
```

```
1984 IS A DYSTOPIAN NOVEL BY ENGLISH NOVELIST GEORGE ORWELL.
```

```
In [5]: sl = s.lower()
        print(sl)
```

```
1984 is a dystopian novel by english novelist george orwell.
```

1.4 replace

The `replace` function replaces substrings in a string (I bet you did not see that one coming!)

```
In [6]: s = "These are the times."
        s1 = s.replace("the", "interesting") # Note that capitalization matters!
        s2 = s1.replace(".", "!")
        print(s2)
```

```
These are interesting times!
```

Alternatively, we can pass an integer `n` as the third argument of `replace`, and it will only replace the first `n` occurrences of the substring.

```
In [7]: s = "These are the times to stay together."
        s1 = s.replace("the", "interesting", 1) # Replaces only the first 'the' (and leaves th
        print(s1)
```

```
These are interesting times to stay together.
```

1.5 startswith & endswith

The functions `startswith(prefix)` and `endswith(suffix)` return a boolean indicating whether the string starts with `prefix` or ends with `suffix`.

```
In [8]: s = "In the beginning it seemed that it would never end"
        sw = s.startswith("In")
        ew = s.endswith("ever end")
        print("Starts with 'In'?", sw)
        print("Ends with 'ever end'?", ew)
```

```
Starts with 'In'? True
Ends with 'ever end'? True
```

1.6 find

The function `find(ss)` returns the first index where substring `ss` occurs. Alternatively, you can use `find(ss,start,end)`, where start and end are integers indicating the slice where `ss` should be searched for. If you want to search until the end of the string, you can ommit `end`.

```
In [9]: s = "ACTGTCTGATGTACCCT"

        tg_first = s.find("TG")
        print("First occurence of TG:", tg_first)

         # Starting to search from the next position where we found the first occurrence
        tg_second = s.find("TG", tg_first+2)
        print("Second occurence of TG:", tg_second)
```

```
First occurence of TG: 2
Second occurence of TG: 6
```

If the substring `ss` does not occur in `s`, the function `find` returns -1. Another function which serves the same purpose is `index` (like the one for lists). The only difference is that `index` raises a `ValueError` if the substring is not found.

1.7 Exercise 1: count occurrences

The function `count(ss)` returns the number of times the substring `ss` occurs in a string, however, it only counts *non-overlapping* occurrences. That means that, for example, `"AAA".count("AA")` will return 1. Implement the function `countOverlapping(s, ss)` that counts the number of times `ss` occurrs in `s` considering overlapping occurrences.

For example, `countOverlapping("AAA", "AA")` should return 2.

```
In [10]: def countOverlapping(s, ss):
             return 42
```

1.8 Exercise 2: fixing messages

People have become sloppy when it comes to typing on the internet. There are a lot of abbreviations, capitalization is not respected, nor is punctuation. Your job in this task is to fix internet messages. Implement the function `fix(M)` that takes a message `M` as a string for the input, and returns another message where the content of `M` is the same, but the following is fixed:

- There is a period at the end of the message.
- All punctutations (.,!?) must be followed by one space.
- The first letter of each sentence is capitalized.
- The following abbreviations are spelled out in full:
- u -> you
- ofc -> of course
- thru -> through
- ppl -> people
- 2 -> too
- 8 -> ate

For example, let

```
M1 = "it is already halfway thru the semester and u have no idea what is
going on? rest assured u are not alone!a lot of ppl go thru the same desperation
ofc.What u need to do is take a step back and figure out what u don't understand
and why...it is not 2 l8 to recover. U can do this"
```

then `fix(M1)` should return:

```
"It is already halfway through the semester and you have no idea what is
going on? Rest assured you are not alone! A lot of people go through the same
desperation of course. What you need to do is take a step back and figure out
what you don't understand and why... It is not too late to recover. You can do
this."
```

You may find it easier to split this task into functions, one for each fix, and use them to implement `fix(M)`.

```
In [11]: def fix(M):
             return ""
```