Strings

Representation of text

To represent text, we use strings. In Python, this can be done by putting text between single quotes (‘) or double quotes (“).

Strings are a sequence of characters, basically an array. As with any array, you can use the same array-indexing logic to get a specific character of the string.

This means, that we can use positive and negative indexing to get any character.

String Manipulation

We have already known the + and * operators used with numbers. Similarly, we have a use for them with strings.

To concatenate, or join two strings together, we can use the + operator. This returns a new string, that contains the two joined strings after each other.

Similarly, we have the * operator, that we can use to put multiple copies of the same string after each other.

We also have two new operators, for checking if a string can be found in another string.

These are the in and not in operators.

Both of these checks if the first operand is in the second one and return true or false accordingly.

Built-in string and character functions

Python has built-in functions for working with strings and characters.

Ord converts a character to their integer representations. Chr does the inverse, converting an integer representation to a character. Len returns the length of a string. Str returns the string representation of any object.

Computers store data as numbers. To store text, one of the simplest scheme is called ASCII. ASCII covers the standard Latin alphabet and most used symbols, and only uses 7 bits to store each character, which means that it’s relatively small in size, while covering everything for most usecases.

If we pass in a character to the ord function, we get back it’s ASCII representation. If we pass in a character that is not in the ASCII table, for example special characters from other languages or emojis, the function returns the UNICODE representation, which is an extension of ASCII and covers every used character and symbol.

The chr function is the exact opposite of the ord function. It takes in an ASCII or UNICODE representation of a character and returns said character.

The len function is mostly self-describing, as it returns the length, or character count of the passed-in string.

The str function can be used to get the string representation of an object. This means, that for example if we pass in a number, we get the same number but as a string.

Getting Parts of Strings

You can get a part, or slice of a string, given by it's tart and end location. For this, we can use a similar indexing scheme than with arrays. The starting index is inclusive, and the ending index is exclusive, which means that the character at the starting index will be in the sliced string wile the character at the ending index won't.

If you want to get a slice of the string that starts from the first character, then you can leave out the starting index, as the default value for it is 0, which is the first character. Similar applies to the ending character.

You can also use negative index numbers, for example if you want to only cut off the last character.

If you add a third number to the slicing, you can get every nth character from the string, starting from the first one of the slice.

Leaving off the starting or ending indices has the same effect as previously.

Putting variables into strings

If we want to output some result that we got to the console, or to send it to somewhere, we might want to format it into a string. We have a few options on how we might achieve this.

The first one is to get the string representation of the variable using the str function, but we don't have to, instead we can use one other built-in way to achieve this.

Built-in String Methods

Strings in Python have several built-in functions, to speed up our work with them. We won't go over every one of them, as some of them have very niche usecases. All these functions return a new string and will not modify the original.

Most of them have very descriptive names.

The replace function takes in two strings as arguments. It searches for the first one and if it finds a match, it replaces it with the second one.

The capitalize function will convert the first character to upper case and everything else to lowercase.

The lower function will convert every character to lower case.

The upper function will convert every character to upper case.

The swapcase function looks at every single character, and if it's lower case, it will change it to upper case, and if it's upper case, will change it to lower case.

The title function will format the string as a title, meaning that it will convert the first letter of every word to upper case, and everything else to lower case. It's basically the same, as if we ran the capitalize function on every single word.

The count function takes in a string as argument, and counts every occurrence of it in the string, then returns the count.

The endswith function takes in a string as argument, and return true, if the original string is ending with it, otherwise it returns false.

The startwith function does the same, except it doesn't look at the end of the string, but at the start.

The find function takes in one string as argument, and it scans the string for a match, then returns the index of the starting character if it finds one. If no match can be found, it returns -1.

There are a few evaluating functions that can be useful when we ask for user inputs, or other applications.

The isalnum function return true if all characters in the string are alphanumeric, meaning that it only contains letters and numbers, otherwise false.

The isalpha function returns true if all characters are letters, otherwise false.

The isdigit function returns true if all characters are numbers, otherwise false.

Links:

https://realpython.com/python-strings