**Python Strings**

This is an introduction to Python strings, as used in the CodingBat [Python practice problems](http://codingbat.com/python), specifically in the [String-1](http://codingbat.com/python/String-1) and [String-2](http://codingbat.com/python/String-2) sections.

**Python String Basics**

In Python code, a string is written withing double quotes, e.g. "Hello", or alternately within single quotes like 'Hi'. Use the len(s) function to get the length of a string, and use square brackets to access individual chars inside the string. The chars are numbered starting with 0, and running up to length-1.

s = 'Hello'

len(s) ## 5

## Chars are numbered starting with 0

s[0] ## 'H'

s[1] ## 'e'

s[4] ## 'o' -- last char is at length-1

s[5] ## ERROR, index out of bounds

Python strings are "immutable" which means a string can never be changed once created. Use + between two strings to put them together to make a larger string

s = 'Hello'

t = 'Hello' + ' hi!' ## t is 'Hello hi!'

**String Slices**

A "slice" in Python is powerful way of referring to sub-parts of a string. The syntax is **s[i:j]**meaning the substring starting at index i, running up to but not including index j.

s = 'Hello'

# 01234 ## Showing the index numbers for the 'Hello'

s[1:4] ## 'ell' -- starting at 1, up to but not including 4

s[0:2] ## 'He'

If the first slice number is omitted, it just uses the start of the string, and likewise if the second slice number is omitted, the slice runs through the end of the string.

s = 'Hello'

# 01234

s[:2] ## 'He', omit first number uses start of string

s[2:] ## 'llo', omit second number uses end of string

Use the slice syntax to refer to parts of a string and + to put parts together to make bigger strings.

a = 'Hi!'

b = 'Hello'

# Compute c as the first 2 chars of a followed by the last 2 chars of b

c = a[:2] + b[len(b) - 2:]

A bad index in Python, e.g. s[100] for the string "Hello", is a runtime error. However, slices work differently. If an index number in a slice is out of bounds, it is ignored and the slice uses the start or end of the string.

**Negative Index** As an alternative, Python supports using negative numbers to index into a string: -1 means the last char, -2 is the next to last, and so on. In other words -1 is the same as the index len(s)-1, -2 is the same as len(s)-2. The regular index numbers make it convenient to refer to the chars at the start of the string, using 0, 1, etc. The negative numbers work analogously for the chars at the end of the string with -1, -2, etc. working from the right side.

s = 'Hello'

# -54321 ## negative index numbers

s[-2:] ## 'lo', begin slice with 2nd from the end

s[:-3] ## 'He', end slice 3rd from the end

**Python String Loops**

One way to loop over a string is to use the range(n) function which given a number, e.g. 5, returns the sequence 0, 1, 2, 3 ... n-1. Those values work perfectly to index into a string, so the loop for i in range(len(s)): will loop the variable i through the index values 0, 1, 2, ... len(s)-1, essentially looking at each char once.

s = 'Hello'

result = ''

for i in range(len(s)):

# Do something with s[i], here just append each char onto the result var

result = result + s[i]

Since we have the index number, i, each time through the loop, it's easy to write logic that involes chars near to i, e.g. the char to the left is at i-1.

Another way to refer to every char in a string is with the regular for-loop:

s = 'Hello'

result = ''

for ch in s: ## looping in this way, ch will be 'H', then 'e', ... through the string

result = result + ch

This form is an easy way to look at each char in the string, although it lacks some of the flexibility of the i/range() form above.

[CodingBat.com](http://codingbat.com/) code practice.