

text as a sequence of characters, indexed, sliced, and never changed in place



A string is a sequence

- a string is an ordered sequence of characters, written in single or double quotes
- every character has a position, called its **index**, counted from the left starting at 0
- negative indices** count from the right instead, starting at -1 for the last character
- `len(s)` reports how many characters a string holds

$s =$

0	1	2	3	4	5
P	Y	T	H	O	N

six cells holding the letters of PYTHON in order, each labelled above with its positive index, counted from the left starting at 0

$s =$

-6	-5	-4	-3	-2	-1
P	Y	T	H	O	N

the same six cells, this time labelled above with the negative index each one also carries, counting from the right and ending at -1

Indexing and slicing *

- `s[i]` gives the single character sitting at index `i`
- `s[start:stop]` gives the slice from start up to, but not including, stop
- a missing start defaults to 0, a missing stop defaults to the end of the string
- `s[start:stop:step]` adds a step, `s[::-1]` with no start or stop and a step of -1 reverses the whole string

Indexing and slicing `s = 'PYTHON'`

Q. `s = 'PYTHON'`, find `s[0]`, `s[-1]`, `s[2:5]`, `s[:3]`, `s[::-1]`

- `s[0]` is the character at index 0, that's P
- `s[-1]` is the last character, counting from the right, that's N
- `s[2:5]` runs from index 2 up to but not including index 5, that's THO
- `s[:3]` has no start, so it begins at 0 and stops before index 3, that's PYT
- `s[::-1]` has no start or stop and a step of -1, walking the whole string backward, that's NOHTYP

$s =$

P	Y	T	H	O	N
---	---	---	---	---	---

the same six cells, one of them, index 0, highlighted, the character `s[0]` actually picks out

Answer **P, N, THO, PYT, NOHTYP**

Strings are immutable

- a string can never be changed in place, `s[0] = 'z'` raises a `TypeError`, there is no way around it
- every string method that looks like it changes a string actually builds and returns a BRAND NEW string, leaving the original untouched
- `s = s.upper()` looks like a change, but it's really `s` being rebound to a new string, the old one is simply discarded

a string method never edits the string it's called on, it hands back a new one and leaves the original exactly as it was

