CSCI 141

Scott Wehrwein

String Manipulation - Indexing and Slicing

Goals

- Know how to index into a string
- Know how Python interprets negative indices into strings.
- Know how to use slicing to get substrings

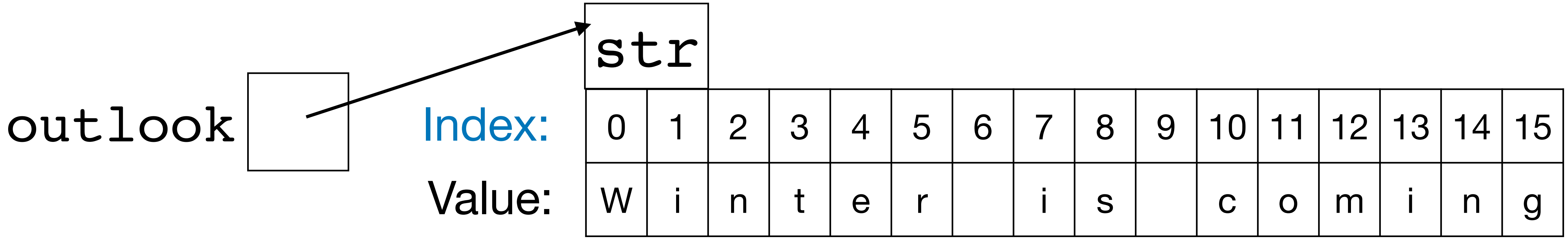
Indexing into Strings

(just smaller strings!)

Strings are collections of individual characters.
We can get access to an individual character by index.

```
outlook = “Winter is coming”
```

How is this stored in memory?



Index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value:	W	i	n	t	e	r		i	s		c	o	m	i	n	g

Indices in Python begin at 0.

Syntax:

```
outlook[0] # => "W"
outlook[4] # => "e"
```

Spaces are characters too!

```
outlook[6] # => " "
```

Indexing

gives us other ways to loop through strings:

```
for letter in a_string:
    print(letter, end="")
```

is equivalent to

```
for i in range(len(a_string)):
    print(a_string[i], end="")
```

and also

```
i = 0
while i < len(a_string):
    print(a_string[i], end="")
    i += 1
```

Nifty Python Feature: Negative Indices

Negative indices count backwards from len(s):

Index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	W	i	n	t	e	r		i	s		c	o	m	i	n	g
Also Index:	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Two possible ways to remember how this works:

-1 is always the last character, and indices count backwards from there.

`a_string[-5]`

is equivalent to

`a_string[len(a_string)-5]`

Slicing: indexing substrings

```
alph = "abcdefghij"
alph[0] # => "a"
alph[4] # => "e"
```

`str`

Ind	0	1	2	3	4	5	6	7	8	9
Val	a	b	c	d	e	f	g	h	i	j

What if I want to "index" more than one character at a time?

```
alph[???] # => "cdef"
```

Slicing: indexing substrings

```
alph = "abcdefghij"
alph[0] # => "a"
alph[4] # => "e"
```

str

Ind	0	1	2	3	4	5	6	7	8	9
Val	a	b	c	d	e	f	g	h	i	j

index of first character

1 + index of last character

Slicing syntax: `string[`*`start`*`:`*`end`*`]`

```
alph[2:6] # => "cdef"
```

just like the `range` function: the end index is **not** included

```
alph[0:10] # => "abcdefghij"

alph[5:-2]  # => "fgh"
```

not like the `range` function: negative indices don't make empty substrings

Slicing: indexing substrings

```
alph = "abcdefghij"
```

str

Ind	0	1	2	3	4	5	6	7	8	9
Val	a	b	c	d	e	f	g	h	i	j

index of first character   1 + index of last character

Slicing syntax: `string[`*`start`*`:`*`end`*`]`

If omitted, *`start`* defaults to 0   If omitted, *`end`* defaults to `len(string)`

```
alph[:4] # => “abcd"
alph[5:] # => "fghij"
```

String Slicing: Demo

String Slicing: Demo

- `s = "fibonacci"`
- Positive indices: s[1:3]
- Negative indices!? s[-4:9]
- Leaving out start/endpoint: s[:6], s[4:]
- Indices past the end in a slice: s[1:21]
- Single indices past the end: s[9], s[21]
- Loop over a slice of a string

```
for c in s[2:6]:
    print(c, "!", sep="", end="")
```