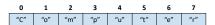


Accessing Characters Review

Strings are stored character by character. Each character in a string is numbered by its position:



The numbers shown here above the characters are called *indices* (singular: index) or *positions*.

3

		N	ega	ativ	e In	dic	es		
0	ive inde ularly u	0			aracte	rs nea	r the e	end of	a string.
	0	1	2	3	4	5	6	7	
	-8	-7	-6	-5	-4	-3	-2	-1	
	"C"	"o"	"m"	"p"	"u"	"t"	"e"	"r"	
		ind la		-	-6] bot string		r to °	'm''	
									4

String Indices

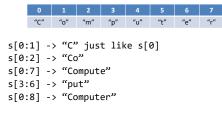
Two ways to use square brackets

- 1 number inside -> gives you 1 character of a string
 - s[0] gives you the first character in s
 - If s = "Computer", s[0] gives you 'C'
- 2 numbers inside (separated by a colon) -> gives you a substring or string slice

String Slicing

- <u>Slice</u>: span of items taken from a sequence, known as *substring*
 - Slicing format: string[start : end]
 - Expression will return a string containing a copy of the characters from *start* up to, but not including, *end*
 - If *start* not specified, 0 is used for start index
 - If end not specified, len(string) is used for end index
 - Slicing expressions can include a step value and negative indexes relative to end of string

String Slicing s[a:b] gives you a substring of s starting from index a and ending at index b-1.



Indices Don't have to be Literal Numbers Say we have this code:

```
s = input("Type in a string: ")
x = int(len(s) / 2)
print s[0:x])
```

What does this print?

More Fun with Indices

- · Examples using negative indices
- A negative index counts from the right side of the string, rather than from the left

s = "Compute	er″
--------------	-----

<pre>print(s[-1])</pre>	#prints	r
<pre>print(s[-3:len(s)])</pre>	#prints	ter
print(s[1:-1])	#prints	ompute

More Fun with Indices

- Slices don't need both left and right indices
- Missing left -> use 0 [far left of string]
- Missing right -> use len(s) [far right of string]

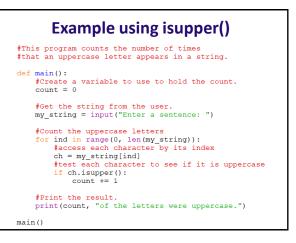
```
s = "Computer"
```

```
print(s[1:])
                   #prints omputer
print(s[:5])
print(s[-2:])
```

#prints Compu #prints er

String Testing Methods

Method	Description
isalnum()	Returns true if the string contains only alphabetic letters or digits and is at least one character in length. Returns false otherwise.
isalpha()	Returns true if the string contains only alphabetic letters, and is at least one character in length. Returns false otherwise.
isdigit()	Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.
islower()	Returns true if all of the alphabetic letters in the string are lowercase, and the string contains at least one alphabetic letter. Returns false otherwise.
isspace()	Returns true if the string contains only whitespace characters, and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines (n) , and tabs (t) .
isupper()	Returns true if all of the alphabetic letters in the string are uppercase, and the string contains at least one alphabetic letter. Returns false otherwise.
	string contains at reast one appraisede retter. Rettillis faise other wise.



String Modification Methods

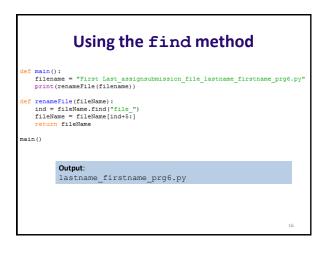
Method	Description	
lower()	Returns a copy of the string with all alphabetic letters converted to lowercase. Any character that is already lowercase, or is not an alphabetic letter, is unchanged.	
lstrip()	Returns a copy of the string with all leading whitespace characters removed. Leading whitespace characters are spaces, newlines (\n), and tabs (\t) that appear at the beginning of the string.	
lstrip(char)	The char argument is a string containing a character. Returns a copy of the string with all instances of char that appear at the beginning of the string removed.	
rstrip()	Returns a copy of the string with all trailing whitespace characters removed. Trailing whitespace characters are spaces, newlines (\n), and tabs (\t) that appear at the end of the string.	
<pre>rstrip(char)</pre>	The char argument is a string containing a character. The method returns a copy of the string with all instances of char that appear at the end of the string removed.	
strip()	Returns a copy of the string with all leading and trailing whitespace characters removed.	
<pre>strip(char)</pre>	Returns a copy of the string with all instances of <i>char</i> that appear at the beginning and the end of the string removed.	
upper()	Returns a copy of the string with all alphabetic letters converted to uppercase. Any character that is already uppercase, or is not an alphabetic letter, is unchanged.	

Example	
<pre>shape = input("Enter shape: Sphere or Cube ")</pre>	
<pre>#Ensures that all letters in shape are lowercase shape = shape.lower()</pre>	
<pre>if shape == 'sphere' or shape == 'cube': validShape = True else: validShape = False</pre>	
14	

More String Methods

Method	Description		
endswith(substring)	The substring argument is a string. The method returns true if the string ends with substring.		
find(substring)	The substring argument is a string. The method returns the lowest index in the string where substring is found. If substring is not found, the method returns -1 .		
replace(old, new)	The old and new arguments are both strings. The method returns a copy of the string with all instances of old replaced by new.		
<pre>startswith(substring)</pre>	The substring argument is a string. The method returns true if the string starts with substring.		

15



18

Testing, Searching, and Manipulating Strings

- You can use the in operator to determine whether one string is contained in another string
 - General format: *string1* in *string2*
 - *string1* and *string2* can be string literals or variables referencing strings

17

 Similarly you can use the not in operator to determine whether one string is not contained in another string In-Class Lab