CSCI 141

Lecture 18
Strings: Slicing, String Methods, Comparison and `in` operators

Announcements

- A4 is due Friday.
- Bonus points for reviewing exams will be awarded and curved scores transferred to Canvas later this week.
- I corrected grading on a couple questions

Goals

- Know how Python interprets negative indices into strings.
- Know how to use slicing to get substrings
- Know how to use a few of the basic methods of string objects:
    - `upper, lower, find, replace`
- Understand the behavior of the following operators on strings:
    - <, >, ==, !=, in, and not in
    - Understand the meaning of lexicographic ordering
- Understand the meaning and implications of strings being immutable objects.

Last time…

- Review what we know already about strings:
  - the str type, + and * operators, len function

```
type("hello")

print("Hello")

"Hello" + “World"

len("abc")

"na" * 16 + " Batman!"
```

Last time...

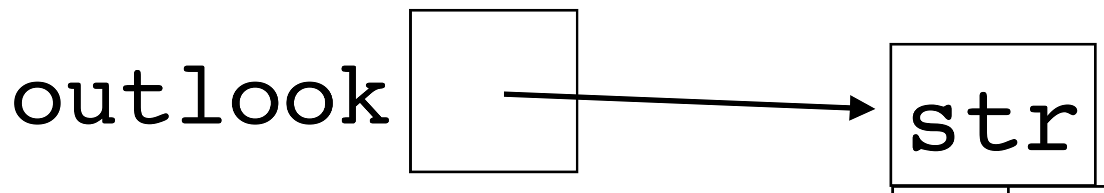
- Know how to iterate over tuples and **strings** using for loops

```
def remove_vowels(string):
    """ Print string, but with no vowels. Don't
        count y as a vowel.
        Pre: no upper case vowels.
    """
    result = ""
    for letter in string:
        # letter has the current letter in the string
        if not (letter == "a" or letter == "e" or letter == "i" \
                or letter == "o" or letter == "u"):
            result = result + letter
    return result
```

Last time…

- Know how to index into a string

```
outlook = “Summer is near”
```



Index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Value:	S	u	m	m	e	r		i	s		n	e	a	r

Indices in Python begin at 0.

Spaces are characters too!

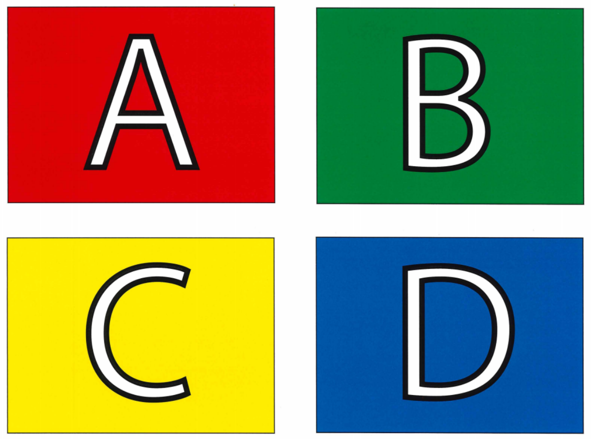
```
outlook[0] # => "S"
outlook[4] # => "e"
```

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

Indexing into Strings

Index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Value:	S	u	m	m	e	r		i	s		n	e	a	r

ABCD: What is the index of the last character of a string `s`?



```
A. len(s) - 1
B. len(s)
C. len(s) + 1
D. 42
```

A consequence of indexing - Another way to loop through strings:

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

is equivalent to

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

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
	S	u	m	m	e	r		i	s		n	e	a	r
Index:	-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]
```

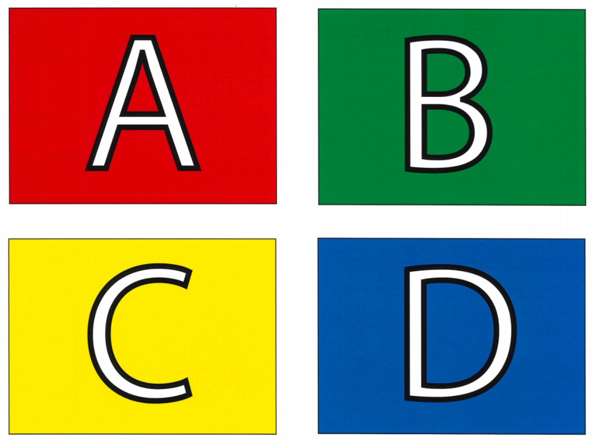
is equivalent to

```
a_string[len(a_string)-5]
```

Negative Indices!

```python
last_name = "wehrwein"

print(last_name[a] == last_name[b])
```



For which assignment of `a` and `b` does the above **not** print `True`?

```
A. a = 1        C. a = -8
   b = 5           b = -4


B. a = 1        D. a = -2
   b = 7           b =  6
```

Today’s Quiz

- 3 minutes

Today’s Quiz

- 3 minutes
- Working with a neighbor: do your answers agree? (2 minutes)

Worksheet - Exercise 1

```
def remove_comments(string):
    """ Return a copy of string, but with
        all characters starting with the first
        # symbol removed. If there is no # in
        the string, return input unchanged.
    """
```

Hint: use a `while` loop!

```
# Example:

remove_comments("a = b # assign b to a”))
# => "a = b “
```

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`*`]`

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

```
alph[0:5] # => “abcde"

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

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

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)`

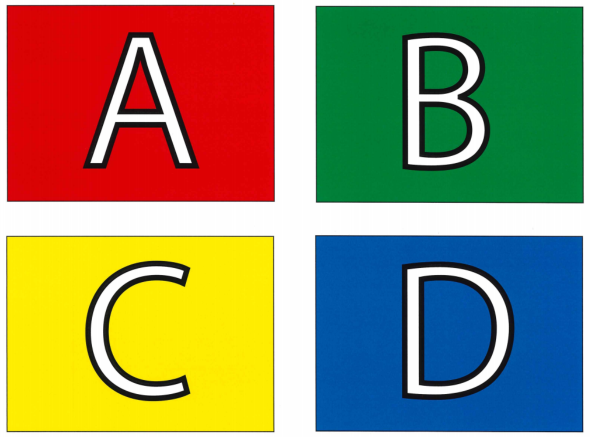
```
alph[:4] # => “abcd"
alph[5:] # => "fghij"
```

String Slicing: Exercise

```
last_name = "Wehrwein"
```

str

Ind	0	1	2	3	4	5	6	7
Val	W	e	h	r	w	e	i	n



Which of the above evaluates to `"in"`?

```
A. last_name[7:8]
B. last_name[6:-1]
C. last_name[-3:]
D. last_name[-2:8]
```

Strings are objects.

We've seen other objects before: turtles!

Turtles had methods:

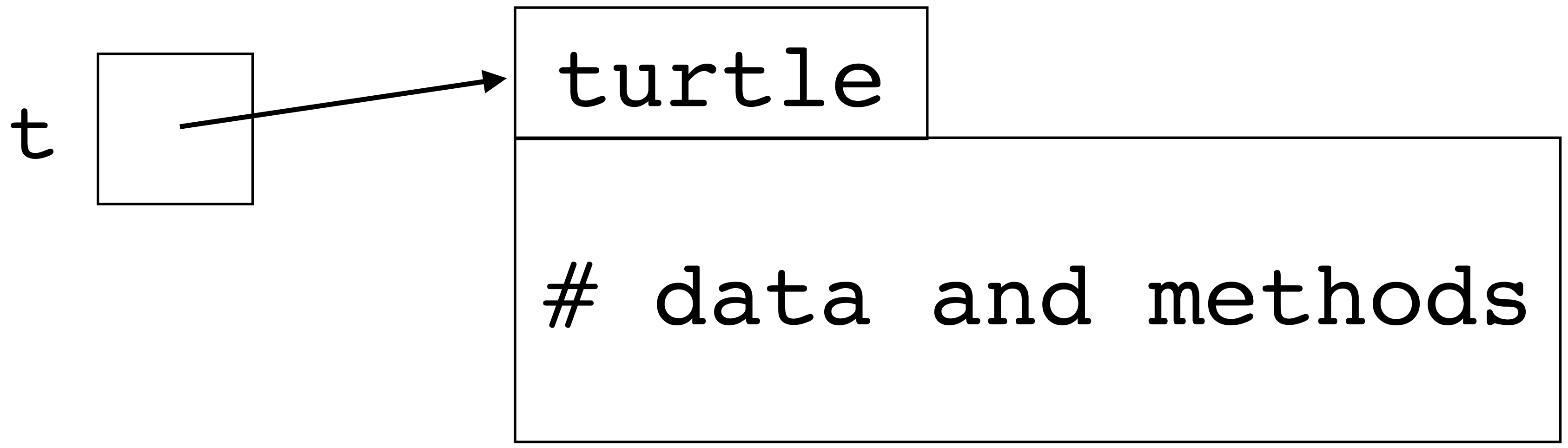
turtle module

module function (turtle constructor)

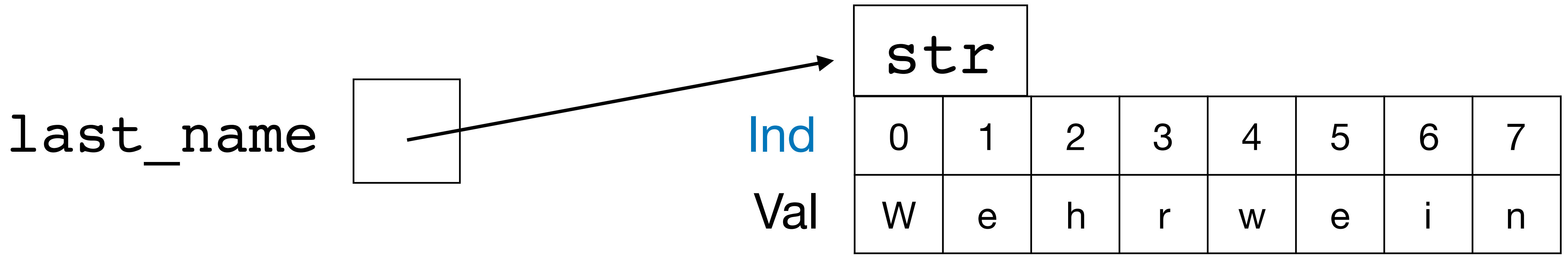
```
t = turtle.Turtle()
t.forward(100)
```

variable that refers to a turtle object

method of a turtle object



Strings are objects.



Ind	0	1	2	3	4	5	6	7
Val	W	e	h	r	w	e	i	n

Strings are objects too - they also have methods.

Turtles had methods:

variable that refers to a string object

a string literal

```
last_name = "Wehrwein"

last_name.upper()
```

method of a string object

Strings have many methods

here are a few of them:

Method	Parameters	Description
upper	none	Returns a string in all uppercase
lower	none	Returns a string in all lowercase
strip	none	Returns a string with the leading and trailing whitespace removed
count	item	Returns the number of occurrences of item
replace	old, new	Replaces all occurrences of old substring with new
find	item	Returns the leftmost index where the substring item is found, or -1 if not found

String methods: demo

`upper, lower, count, replace, find, strip`

String methods: demo

upper, lower, count, replace, find, strip

```
word = "Banana"
word.upper()
word.lower()
word.count("a")
word.replace("a", "A")

line = " snails are out "
line.find("s")
line.find("snails")
line.find("banana")
line.strip()

phrase = "WWU is in Bellingham"
phrase = phrase[:19] + phrase[19].upper()
```

String Methods: More

The textbook (Section 9.5) has a more complete listing of string methods:
http://interactivepython.org/runestone/static/thinkcspy/Strings/StringMethods.html

The Python documentation has full details of the `str` type and all its methods:

https://docs.python.org/3/library/stdtypes.html#str

You should know how to use `upper`, `lower`, `replace`, and `find`.

Worksheet - Exercise 2

```
phrase = "WWU is in Bellingham"
phrase = phrase[:19] + phrase[19].upper()
```

Write a function that capitalizes the last letter of **any** string:

```
def capitalize_last(in_str):
    """ Return a copy of in_str with its
        last letter capitalized.
    """
```

```
# Example:
capitalize_last(“Mix")) # => "MiX"
```

Worksheet - Exercise 3

Rewrite the function from Exercise 1 using the `find` method and slicing to avoid using a loop.

```python
def remove_comments(string):
    """ Return a copy of string, but with
        all characters starting with the first
        # symbol removed. If there is no # in
        the string, return input unchanged.
    """
```

Next time: Lists