CSCI 141

Scott Wehrwein

String Methods

Goals

- Know how to use a few of the basic methods of string objects:
  - `upper, lower, find, replace`
- Be able to look up and make use of other string methods as needed.

Strings are objects.

We've seen other objects before: turtles!

Turtles had methods:

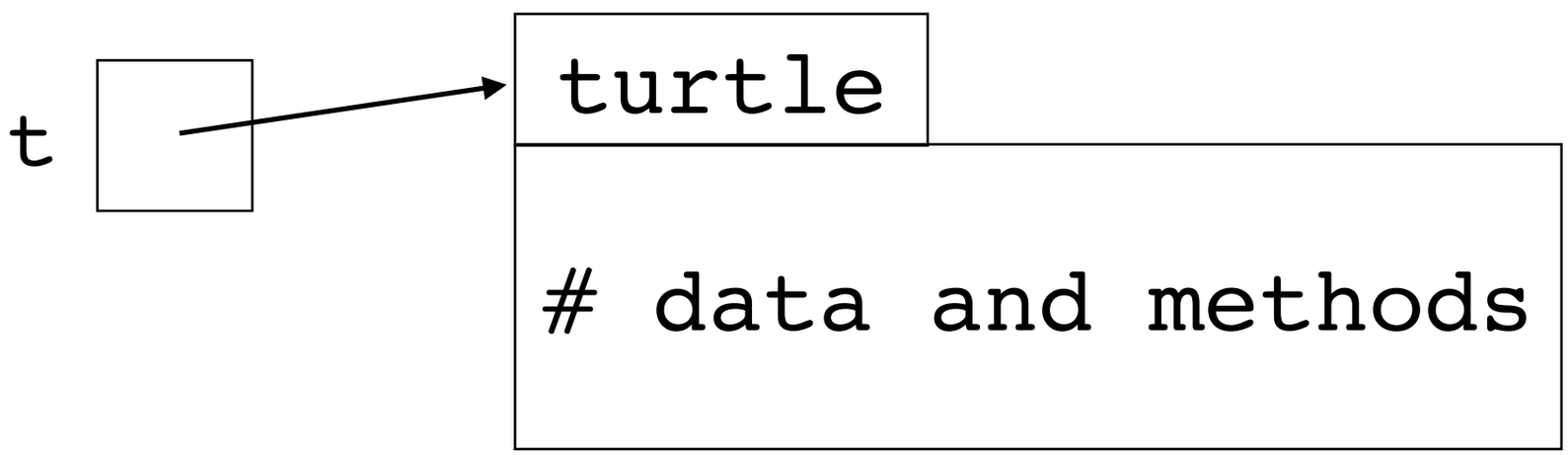
turtle module

module function
(turtle constructor)

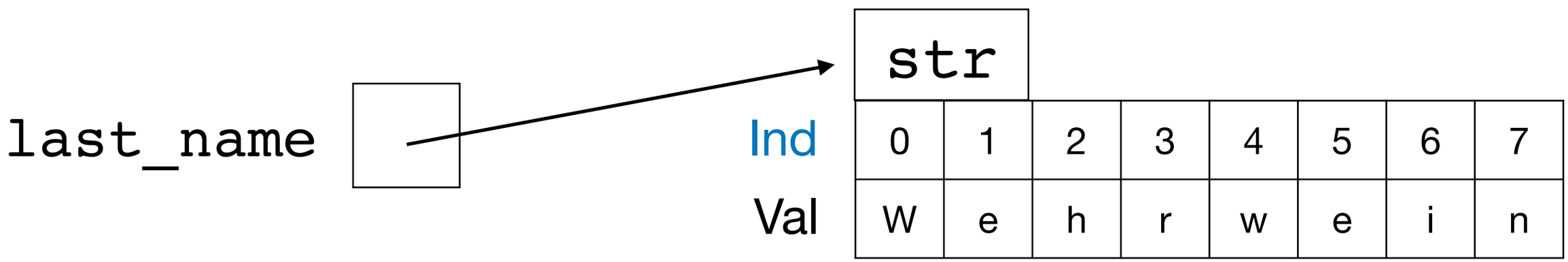
```
t = turtle.Turtle()
t.forward(100)
```

variable that refers to a turtle object

method of a turtle object



Strings are objects.



Strings are objects too - they also have methods.

variable that refers to a string object

a string literal

```
last_name = "Wehrwein"
```

Methods can be called directly on the literal string, too:

```
"Wehrwein".upper()
```

```
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()
line.strip().upper()

word = "Bellingham"
word = word[:9] + word[9].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`.

String Methods

Problem: write an expression to determine if a string `user_input` contains the word `"yes"`, with any capitalization and with any amount of spaces.

`user_input`

`=> " Y  eS "`

String Methods

Problem: write an expression to determine if a string `user_input` contains the word `"yes"`, with any capitalization and with any amount of spaces.

```
user_input.replace(" ", "")
```

```
=> "YeS"
```

String Methods

Problem: write an expression to determine if a string `user_input` contains the word `"yes"`, with any capitalization and with any amount of spaces.

```
user_input.replace(" ", "").lower()
```

```
=> "yes"
```

String Methods

Problem: write an *expression* to determine if a string `user_input` contains the word `"yes"`, with any capitalization and with any amount of spaces.

```
user_input.replace(" ", "").lower() == "yes"
```

```
=> " Y  eS ".replace(" ", "").lower() == "yes"
=> "YeS".lower() == "yes"
=> "yes" == "yes"
=> True
```

dot (method call) operators are evaluated left-to-right!

Effects vs Return Values, again.

Most turtle methods **change the state** of the turtle object they're called on:

```
t.forward(100) # actually moves t forward
```

Most string methods return a **new** string with the given modifications:

```
s = "BOO"
s.lower() # => "boo"
print(s)  # prints BOO
t = s.lower() # if you want "boo", save it
```

Why is this? Because strings can't be modified. Try this:

```
s = "Scott"
s[3] = "o" # error
```