Sequential types: strings and lists

Lecture 02.05

by Marina Barsky

What types do you know?

string functions and operators

`str`	**`str(42)`** returns **`'42'`**	converts anything to a string
`len`	**`len('42')`** returns **`2`**	gets the string’ s length
`+`	**`'XL' + 'II'`** returns **`'XLII'`**	*concatenates* strings
`*`	**`'VI'*7`** returns **`'VIVIVIVIVIVIVI'`**	*repeats* strings

string functions and operators

str	`str(42)` returns `'42'`	converts anything to a string
len	`len('42')` returns `2`	gets the string's length
+	`'XL' + 'II'` returns `'XLII'`	*concatenates* strings
*****	`'VI'*7` returns `'VIVIVIVIVIVIVI'`	*repeats* strings

Given these strings

```
s1 = "ha"
s2 = "t"
```

What are the following strings?

`s1 + s2`	`hat`
`2*s1 + s2 + 2*(s1+s2)`	`hahathathat`

String surgery

`s = 'Python is fun'`

0 1 2 3 4 5 6 7 8 9 10 11 12

s[ ] *indexes* into the string, returning a one-character string

index

s[0] returns 'P'

s[8] returns 's'

s[11] returns 'u'

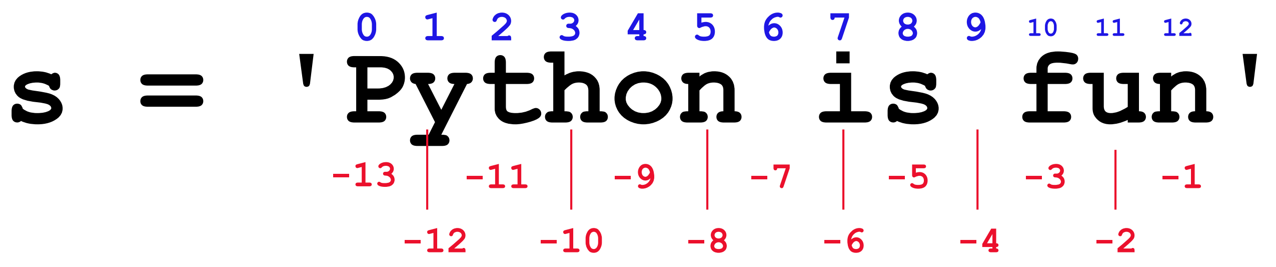
What returns **'n'**? s[5] s[12]

len(s) returns 13

s[len(s)] returns ERROR

python != English

Negative indices...



```
s = 'Python is fun'
```

Negative indices count ***backwards*** from the end!

`s[-1]`	returns	`'n'`
`s[-10]`	returns	`'h'`
`s[-0]`	returns	`'P'`

Slicing

what if you want a bigger piece of the pie???

```
      0 1 2 3 4 5 6 7 8 9 10 11 12
s = 'Python is fun'
 -13  -11  -9  -7  -5  -3  -1
    -12  -10  -8  -6  -4  -2
```

s[:] *slices* the string, returning a **substring**

the first index is the first character of the slice

the second index is **ONE AFTER** the last character

`s[0:4]`	returns	`'Pyth'`
`s[2:6]`	returns	`'thon'`
`s[10:]`	returns	`'fun'`
`s[:]`	returns	`'Python is fun'`

Slicing

```
      0 1 2 3 4 5 6 7 8 9 10 11 12
s = 'Python is fun'
    -13 -11 -9  -7  -5  -3  -1
      -12 -10 -8  -6  -4  -2
```

`s[ : ]` *slices* the string, returning a **substring**

What are these slices?

`s[10:-1]`	`'fu'`
`s[-6:-4]`	`'is'`

How do you get:

`'hon'`	`s[3:6]`
`'honey'`	`s[3:6] + 'ey'`

Skip-Slicing

if you don't want your neighbor to get any...

```
        0  1  2  3  4  5  6  7  8  9  10 11 12
s = 'Python is fun'
      -13  -11   -9   -7   -5   -3   -1
        -12   -10   -8   -6   -4   -2
```

s[: :] ***skip-slices***, returning a subsequence

the third index is the "stride" length it defaults to 1

s[0:10:2] returns **'Pto s'**

s[12:9:-1] returns **'nuf'**

What skip-slice returns **'tin'** s[2:13:5]

s[0::7] + 'e' returns **'pie'**

s[::-1] returns **'nuf si nohtyP'**

Lists → collections of ***any*** data

Lists are more general than strings:
strings are always sequences **of characters**,
whereas lists can contain values **of any type**

```
L = [ 3.14, [2,40], 'third', 42 ]
```

Lists → collections of *any* data

```
L = [ 3.14, True, 'third', 42 ]
```

Commas separate elements.

Square brackets tell python you want a list.

```
L = [ 3.14, [2,40], 'third', 42 ]
```

You can have a list in a list!

len, indexing, slicing

```
L = [ 3.14, [2,40], 'third', 42 ]
```

`len(L)`	`L[0]`	`L[0:1]`
length	**indexing**	**slicing**

How could you extract from **L** `'hi'`

List operators

+

concatenation

Joins two lists

```
>>> P = [ 3.14, [2,40], 'third', 42]
>>> R = ['a','b','c']
>>> P + R
[3.14, [2, 40], 'third', 42, 'a', 'b', 'c']
```

*****

multiplication

Repeats list a number of times

```
>>> lst = [1,2,3]
>>> lst * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

The *in* operator – membership testing for lists and strings

```
>>>'i' in 'alien'                              True

>>> 3*'i' in 'alien'                           False

>>> 'i' in 'team'                              False

>>> 'cs' in 'physics'                          True

>>> 'sleep' not in ‘CMPT 100'                  True

>>> 42 in [41,42,43]                           True

>>> 42 in [ [42], '42' ]                       False
```

Mutable and immutable sequences

Strings are immutable (read-only)

Once a string is created, individual elements of string cannot be changed!

```
>>> st = 'ABC'
>>> st[0]
'A'
>>> st[0]='B'
Traceback (most recent call last):
  File "<pyshell#33>", line 1, in <module>
    st[0]='B'
TypeError: 'str' object does not support item
assignment
```

Mutable and immutable sequences

Lists are mutable (read and write)

Individual items or entire slices can be replaced through assignment statements

```
>>> lst = ['A', 'B', 'C']
>>> lst
['A', 'B', 'C']
>>> lst[0] = 'B'
>>> lst
['B', 'B', 'C']
```

Raising and razing lists -1

```
pi = [3,1,4,1,5,9]

L = [ 'pi', "isn't", [4,2] ]
```

What is **len(pi)**	6
What is **len(L)**	3
What is **len(L[1])**	**5**
What is **pi[2:4]**	[4,1]
What slice of **pi** is **[3,1,4]**	pi[0:3]
What slice of **pi** is **[3,4,5]**	pi[::2]

Raising and razing lists - 2

```
pi = [3,1,4,1,5,9]

L = [ 'pi', "isn't", [4,2] ]
```

What is **pi[0]*(pi[1] + pi[2])** **15**

What is **pi[0]*(pi[1:2] + pi[2:3])** **[1,4,1,4,1,4]**

Raising and razing strings

```
L = [ 'pi', "isn't", [4,2] ]
M = 'You need parentheses for chemistry !'
     0   4   8   12  16  20  24  28  32
```

What is **L[0]**	'pi'
What is **L[0:1]**	['pi']
What is **L[0][1]**	'i'
What slice of **M** is **'try'**	M[31:34]
What is **M[9:15]**	'parent'
What is **M[::5]**	**'Yeah cs!'**