[301] Strings

Tyler Caraza-Harter

Learning Objectives Today

String Basics

- Comparison
- Common functions

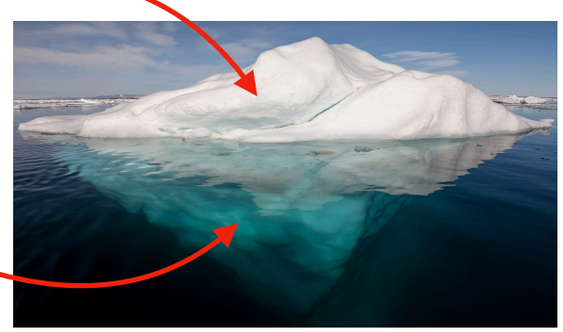
Method Syntax

Sequences (a string is an example of a sequence)

- len
- indexing
- slicing
- for loop

what we've learned about strings so far

what we'll learn today



https://naturalfiberproducers.com/store-3/suri-silk-yarn/

Today's Outline

Comparison

String Methods

Sequences

Slicing

for loop over sequence

Comparison

1 < 2 → True

(because 1 is before 2)

200 < 100 → False

(because 200 is NOT before 100)

Comparison

1 < 2

True

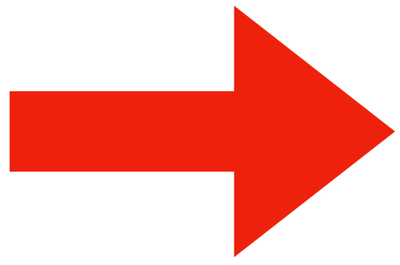
(because 1 is before 2)

200 < 100

False

(because 200 is NOT before 100)

“cat” < “dog”



Python can also compare strings

Comparison

1 < 2

True

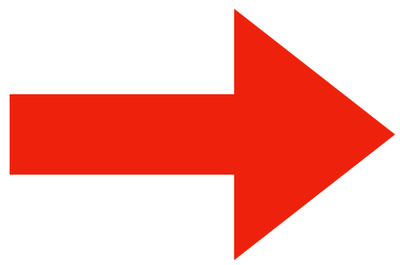
(because 1 is before 2)

200 < 100

False

(because 200 is NOT before 100)

“cat” < “dog”



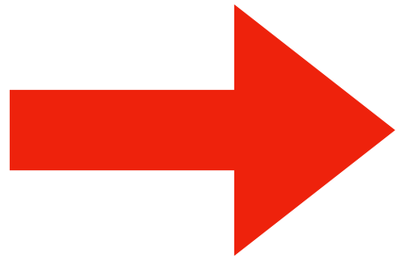
True

(because “cat” is before “dog” in the dictionary)

Python can also compare strings

Comparison

“dog” < “doo doo”

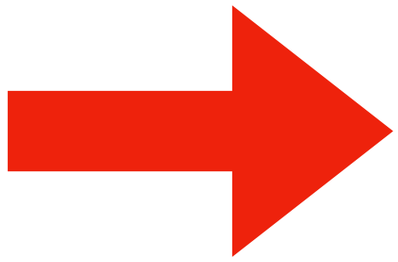


???

What about strings that start with the same letter?

Comparison

“dog” < “doo doo”



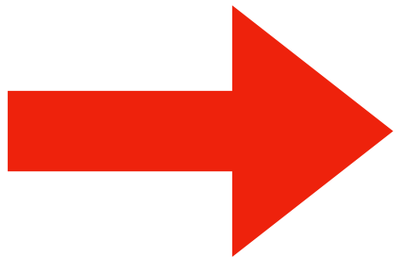
???

What about strings that start with the same letter?

Look for the first letter that’s different, and compare those.

Comparison

“dog” < “doo doo”



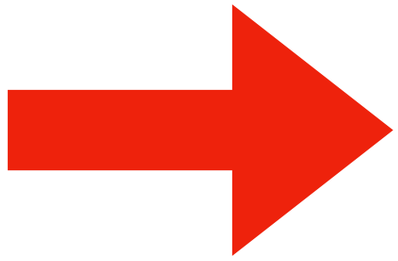
True

What about strings that start with the same letter?

Look for the first letter that’s different, and compare those.

Comparison

“dog” < “doo doo”



True

There are three "gotchas":

1. case (upper vs. lower)
2. digits
3. prefixes

1. Case rules

“A” < “B” < … < “Y” < “Z” **makes sense**

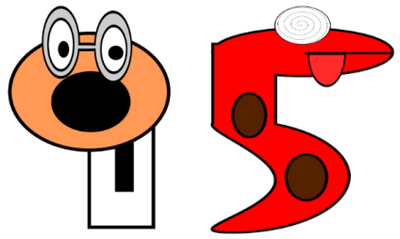
“a” < “b” < … < “y” < “z” **makes sense**

“C” < “b”
“Z” < “a”

upper case is before lower

less intuitive

2. Pesky digits



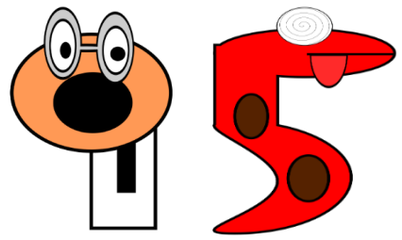
“0” < “1” **makes sense**

“8” < “9” **makes sense**

“11” < “2”
“100” < “15”

less intuitive

2. Pesky digits



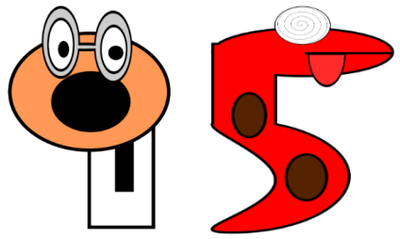
“0” < “1” **makes sense**

“8” < “9” **makes sense**

“11” < “2”
“100” < “15”

remember to find the FIRST difference,
and base everything on that

2. Pesky digits



“0” < “1” **makes sense**

“8” < “9” **makes sense**

“11” < “2”
“100” < “15”

remember to find the FIRST difference,
and base everything on that

3. Prefixes

String 1: bat
String 2: batman

3. Prefixes

String 1: bat

String 2: batman

3. Prefixes

String 1: bat

String 2: batman

“” < “m”, so String 1 is first:

“bat” < “batman”

Today's Outline

Comparison

String Methods

Sequences

Slicing

for loop over sequence

What is a method?

A special function associated variable/value

```
>>> msg = "hello"
>>>
```

What is a method?

A special function associated variable/value

```
>>> msg = "hello"
>>> len(msg)
```

len is a normal function,
it returns number
of characters in string.

It returns the number of
characters in a string

What is a method?

A special function associated variable/value

```
>>> msg = "hello"
>>> len(msg)
5
>>>
```

What is a method?

A special function associated variable/value

```
>>> msg = "hello"
>>> len(msg)
5
>>> msg.isdigit()
```

isdigit is a special function, called a method, that operates on the string in msg.

It returns a bool, whether the string is all digits

What is a method?

A special function associated variable/value

```
>>> msg = "hello"
>>> len(msg)
5
>>> msg.isdigit()
```

```
str.isdigit(msg)
```

type of msg

method in str
(similar to module)

equivalent

isdigit is a special function,
called a method, that operates
on the string in msg.

It returns a bool, whether the
string is all digits

What is a method?

A special function associated variable/value

```
>>> msg = "hello"
>>> len(msg)
5
>>> msg.isdigit()
False
>>>
```

What is a method?

A special function associated variable/value

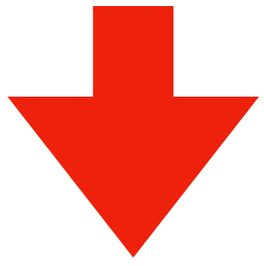
```
>>> msg = "hello"
>>> len(msg)
5
>>> msg.isdigit()
False
>>>
```

Both the regular function (len) and method (isdigit) are answering
a question about the string in msg, but we call them slightly differently

Function/method styles you might see...

```
import math

x = len("hi")
y = math.ceil(2.1)       # math is a module
z = str.isdigit("hi")    # str is a type
```




```
z = "hi".isdigit()
```

What is a method?

A special function associated variable/value

```
>>> msg = "hello"
>>> len(msg)
5
>>> msg.isdigit()
False
>>> msg.upper()
'HELLO'
```

is upper a regular function or a method?

What is a method?

A special function associated variable/value

```
>>> msg = "hello"
>>> len(msg)
5
>>> msg.isdigit()
False
>>> msg.upper()
'HELLO'
```

methods can be called with literal values as well as with values in variables

What is a method?

A special function associated variable/value

```
>>> msg = "hello"
>>> len(msg)
5
>>> msg.isdigit()
False
>>> msg.upper()
'HELLO'
```

methods can be called with literal values as well as with values in variables

What is a method?

A special function associated variable/value

```
>>> msg = "hello"
>>> len(“301”)
3
>>> “301”.isdigit()
True
>>> “Hello World”.upper()
‘HELLO WORLD’
```

methods can be called with literal values as well as with values in variables

String Method	Purpose
s.upper()	change string to all upper case
s.lower()	opposite of upper()
s.strip()	remove whitespace (space, tab, etc) before and after
s.lstrip()	remove whitespace from left side
s.rstrip()	remove whitespace from right side
s.format(args…)	replace instances of "{}" in string with args
s.find(needle)	find index of needle in s
s.startswith(prefix)	does s begin with the given prefix?
s.endswith(suffix)	does s end with the given suffix?
s.replace(a, b)	replace all instances of a in s with b

Quick demos…

Today's Outline

Comparison

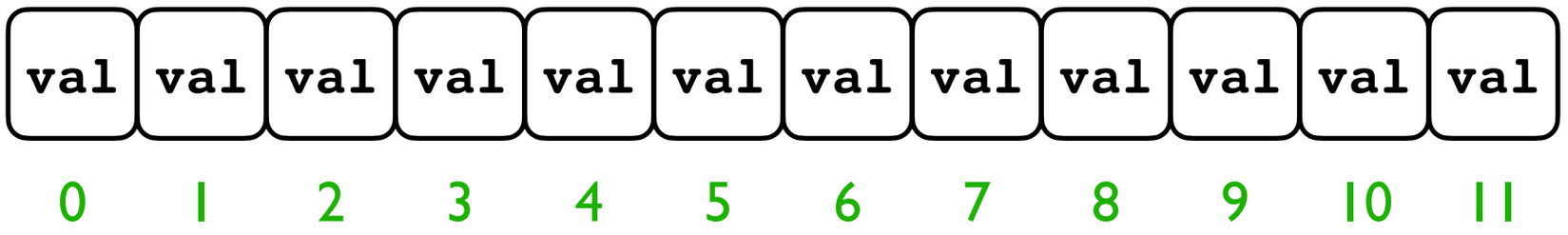
String Methods

Sequences

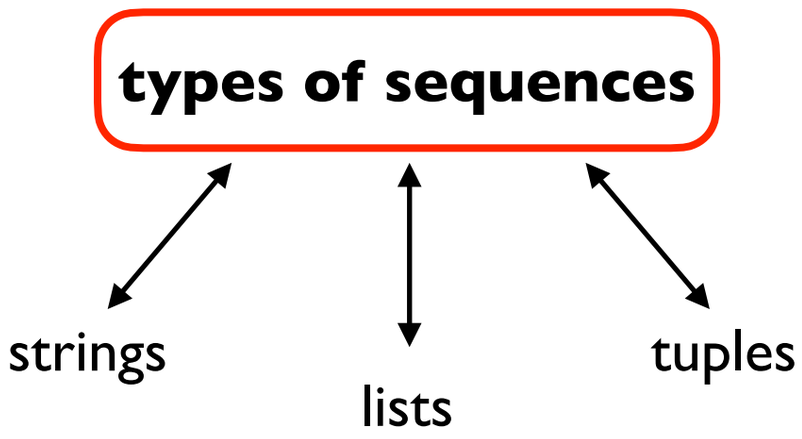
Slicing

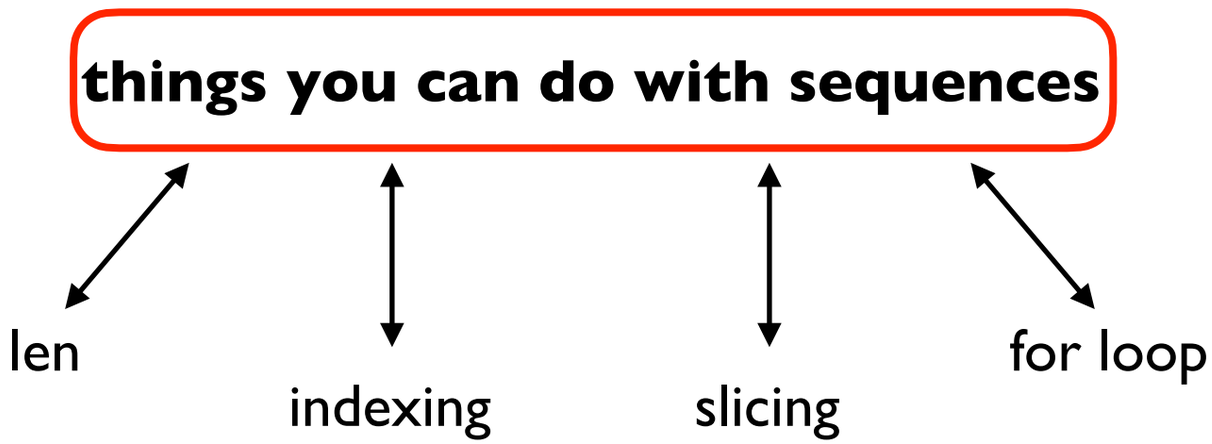
for loop over sequence

Python Sequences

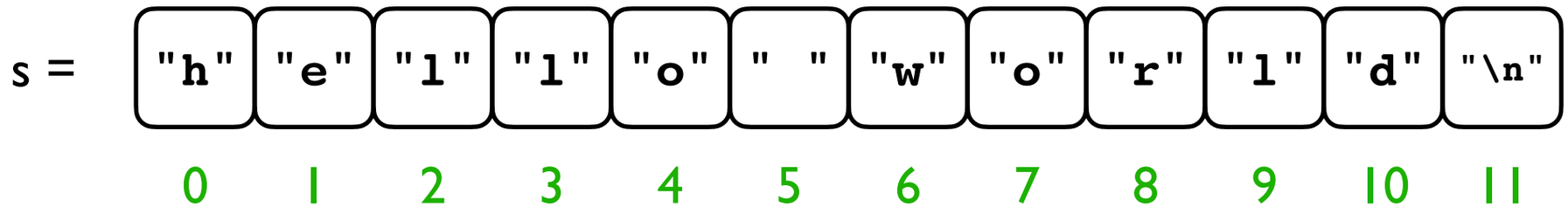


Definition: a *sequence* is a collection of numbered/ordered values

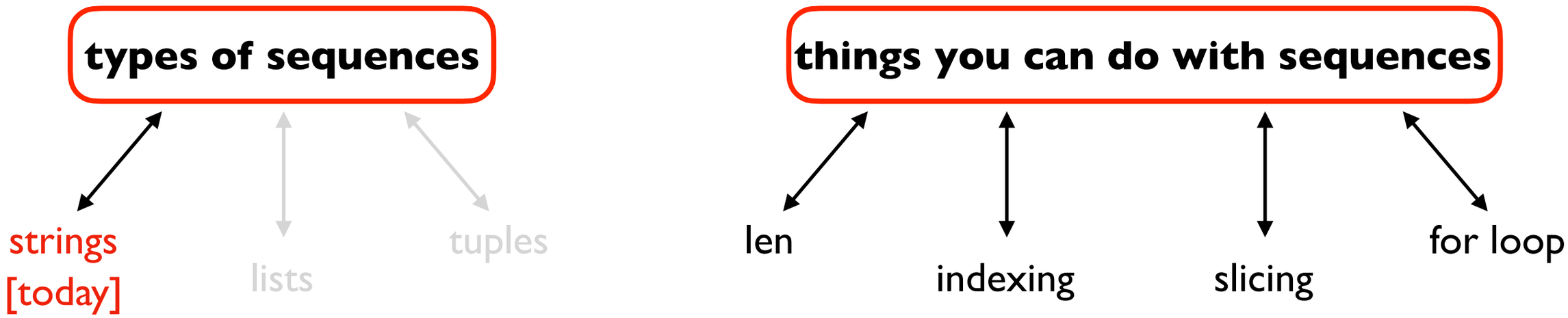




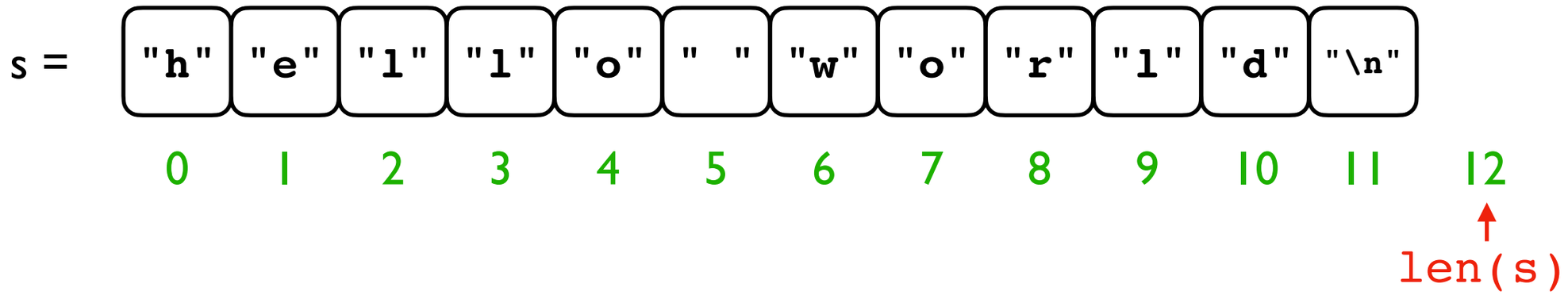
Python Sequences



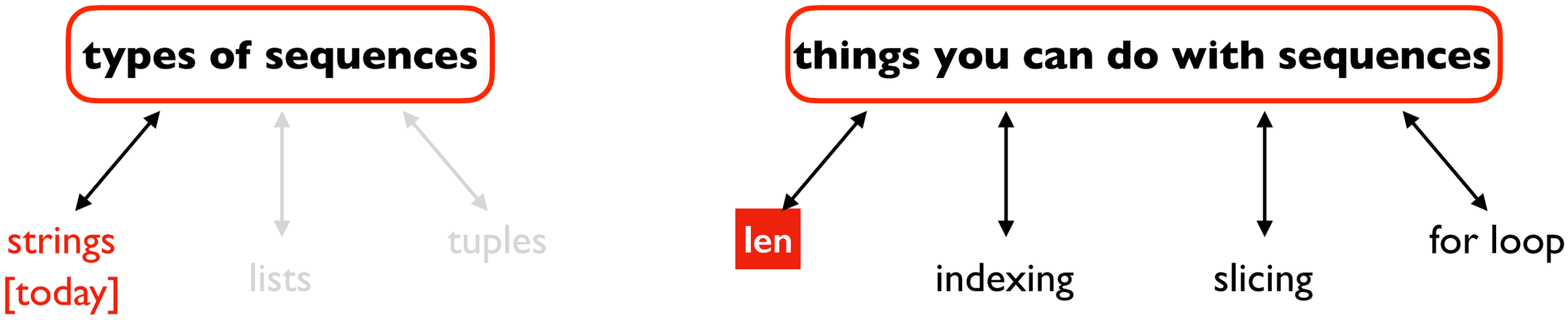
Definition: a *string* is a sequence of one-character strings



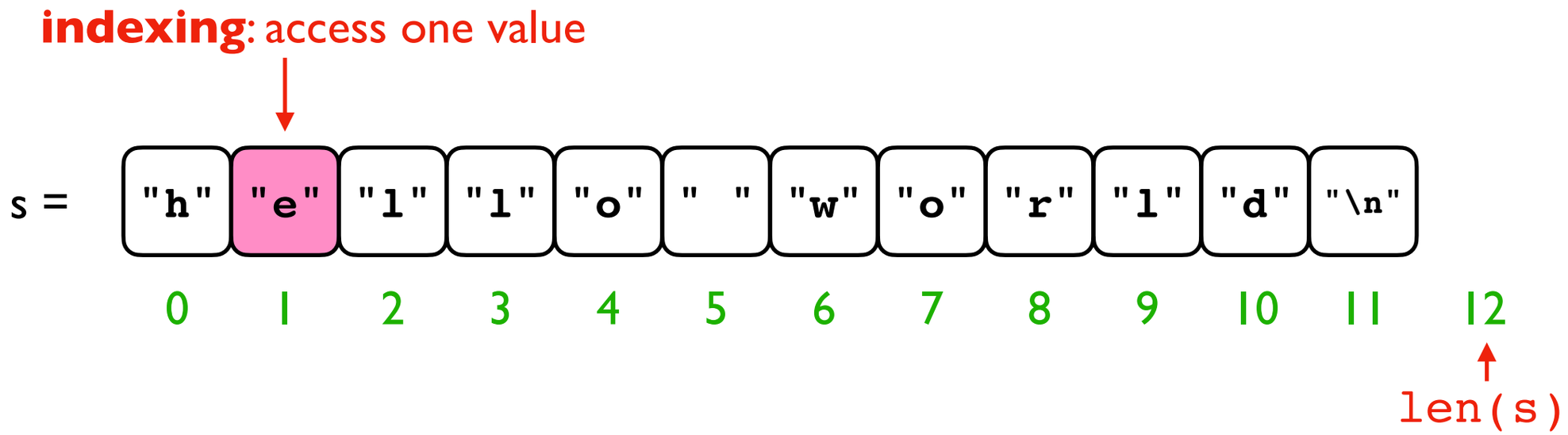
Python Sequences



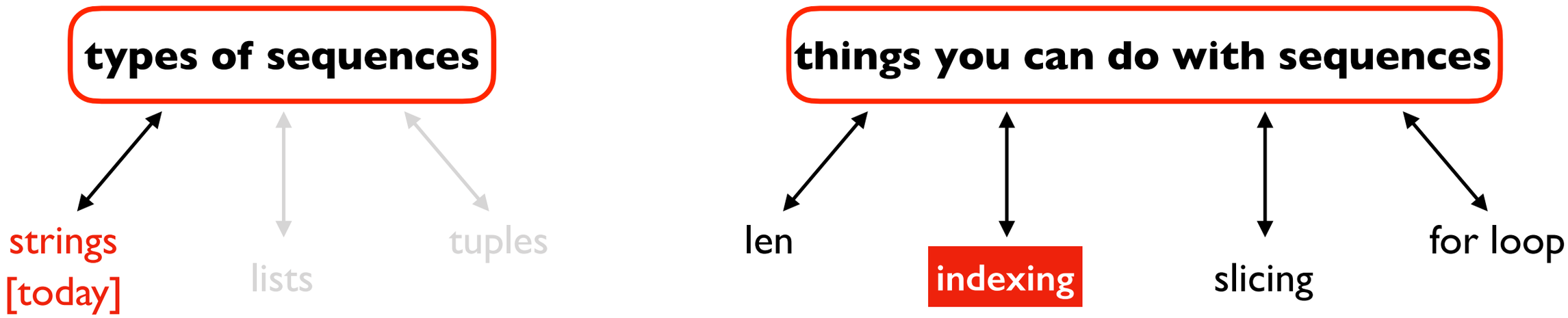
Definition: a *string* is a sequence of one-character strings



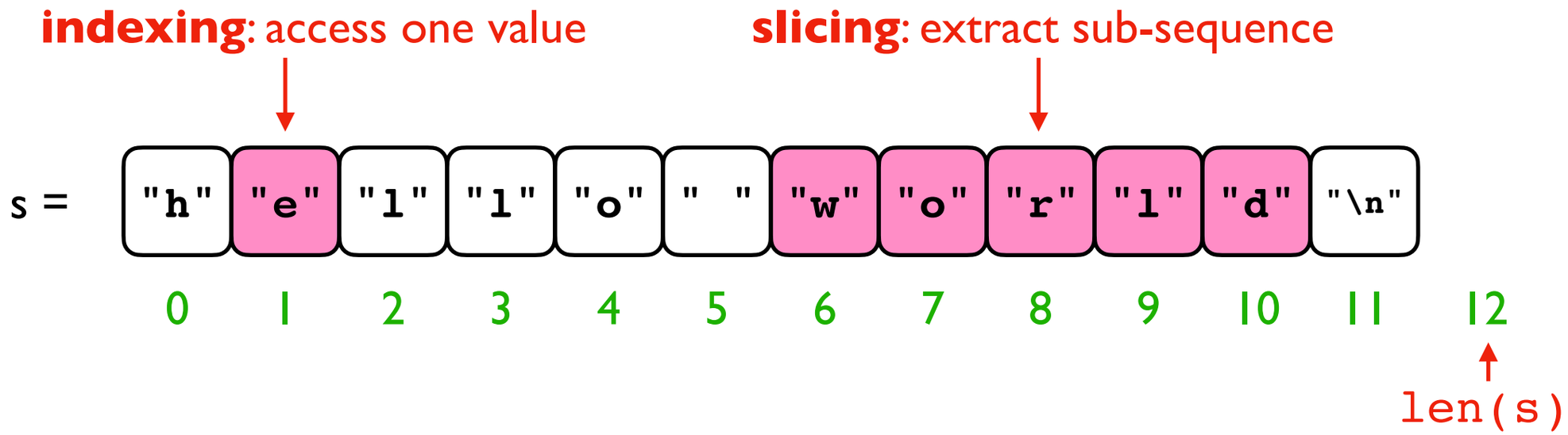
Python Sequences



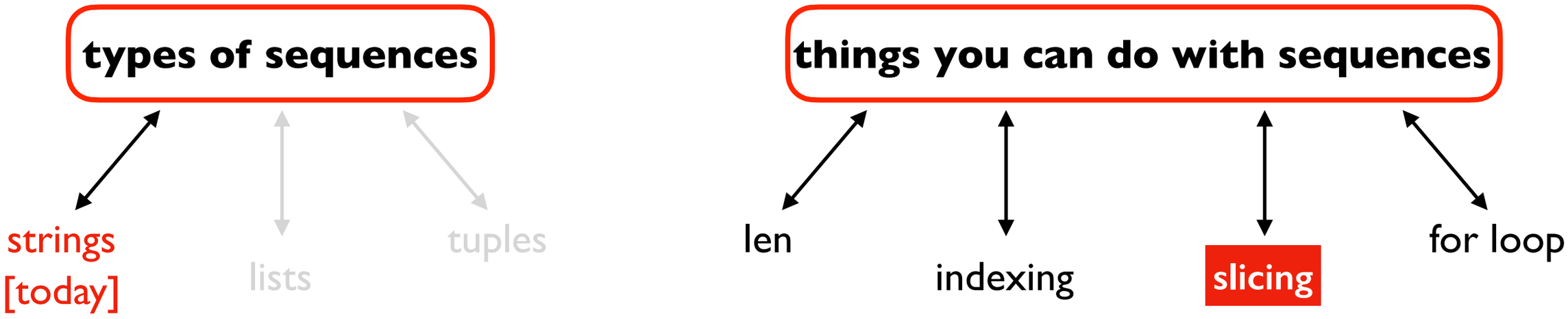
Definition: a *string* is a sequence of one-character strings



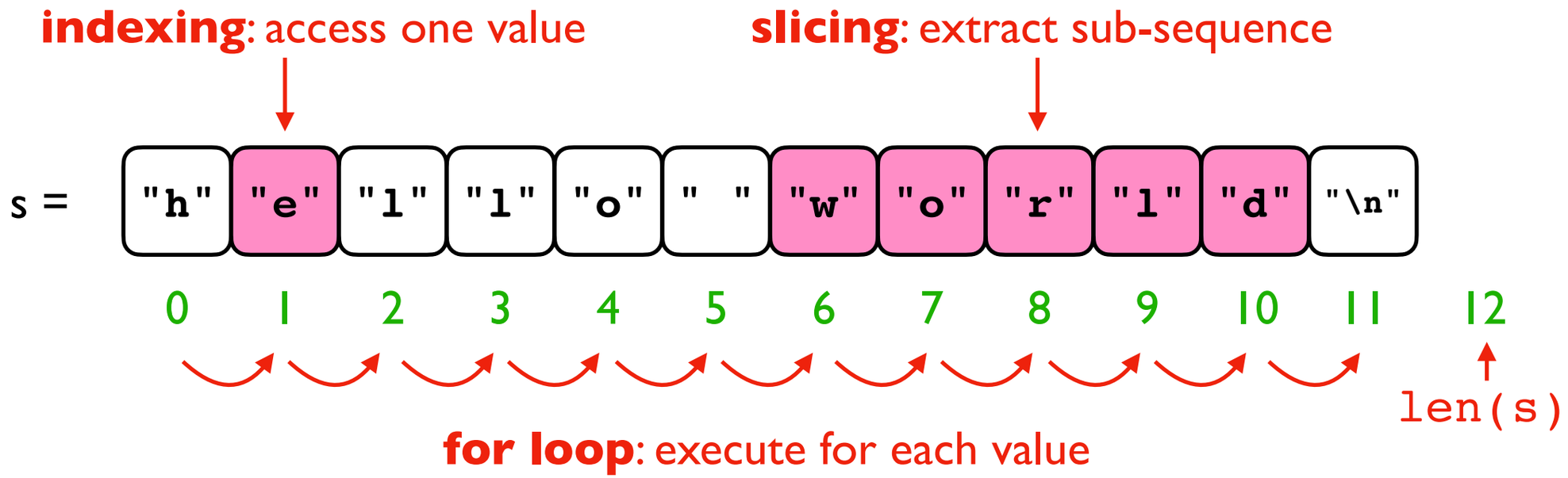
Python Sequences



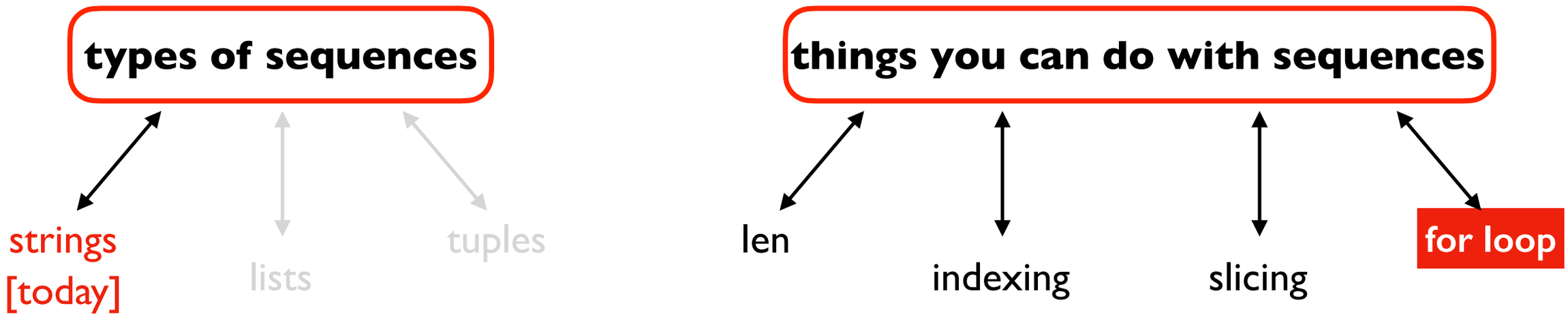
Definition: a *string* is a sequence of one-character strings



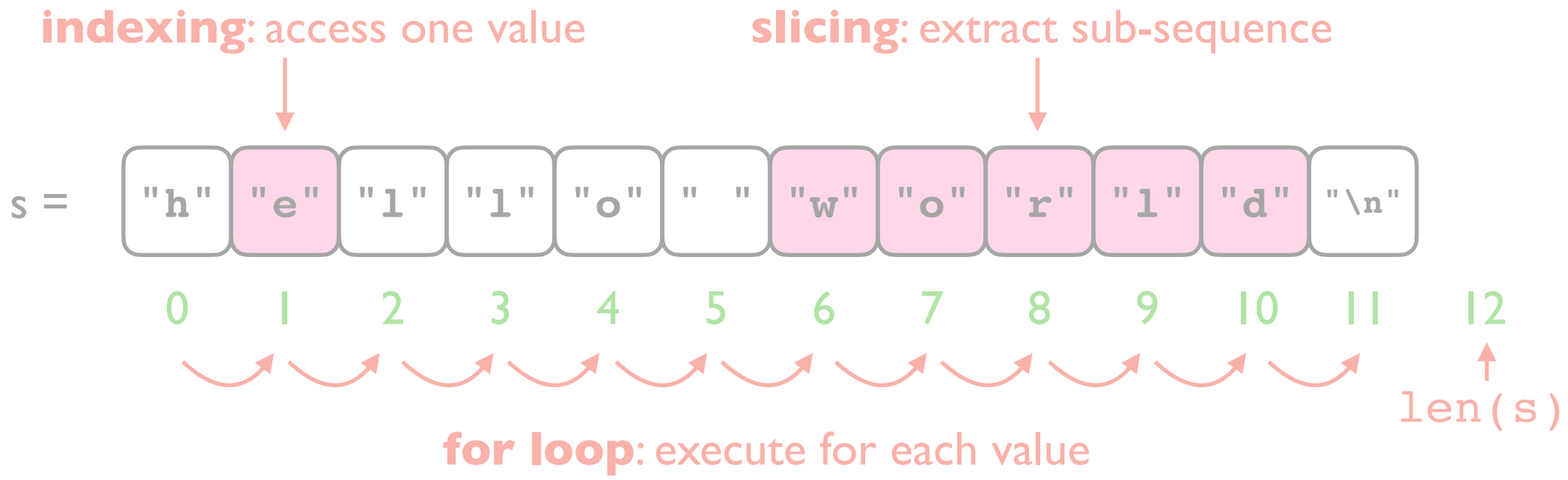
Python Sequences



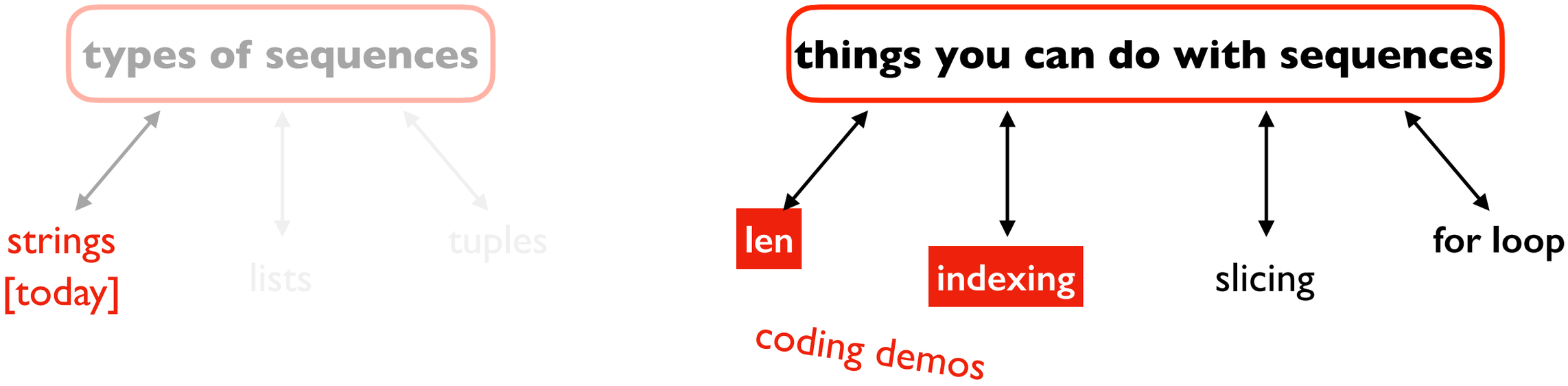
Definition: a *string* is a sequence of one-character strings



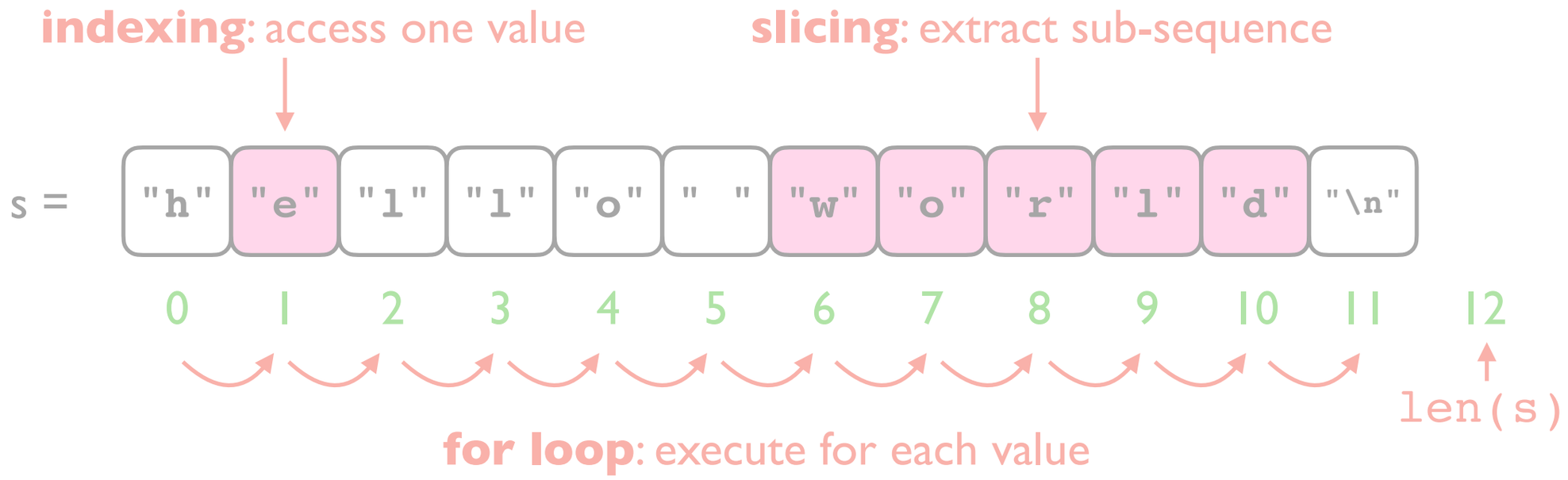
Python Sequences



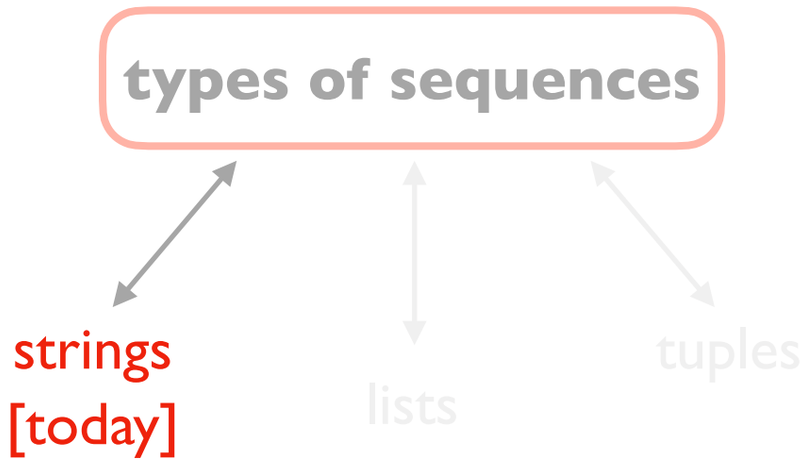
Definition: a *string* is a sequence of one-character strings

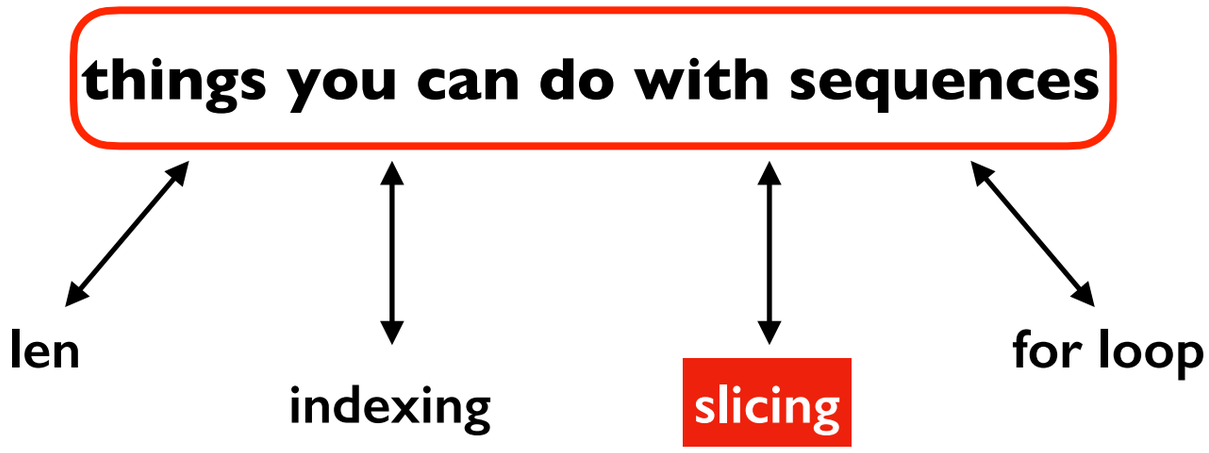


Python Sequences



Definition: a *string* is a sequence of one-character strings





Today's Outline

Comparison

String Methods

Sequences

Slicing

for loop over sequence

Indexing

	0	1	2	3	4
S:	**P**	**I**	**Z**	**Z**	**A**

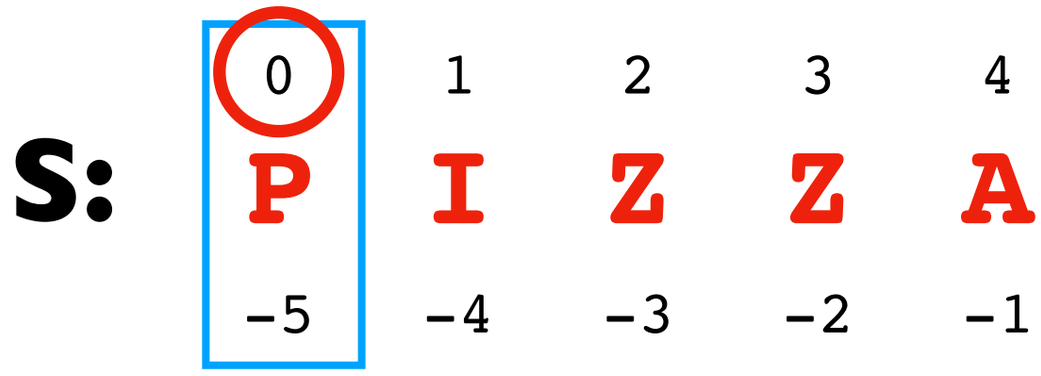
Code:

```
S = “PIZZA”
```

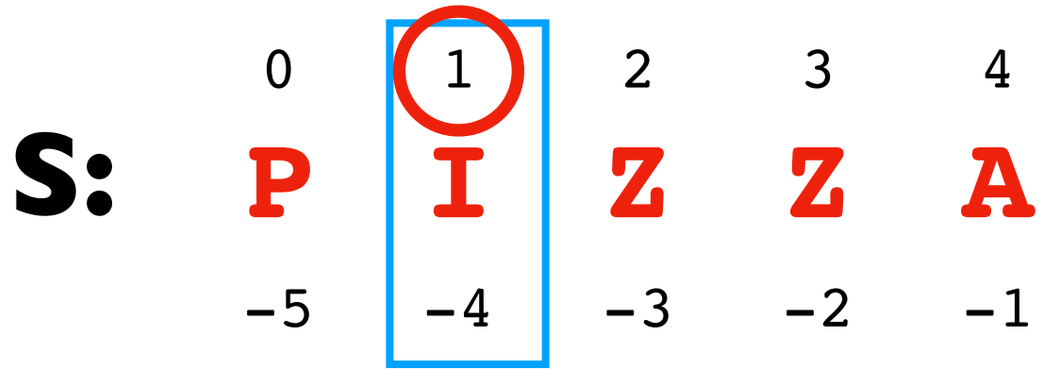
Indexing

	0	1	2	3	4
S:	**P**	**I**	**Z**	**Z**	**A**
	-5	-4	-3	-2	-1

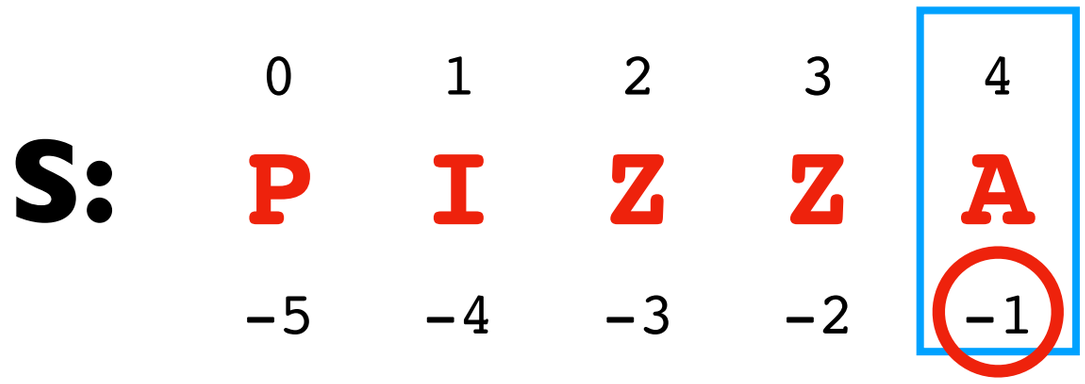
Indexing



Indexing

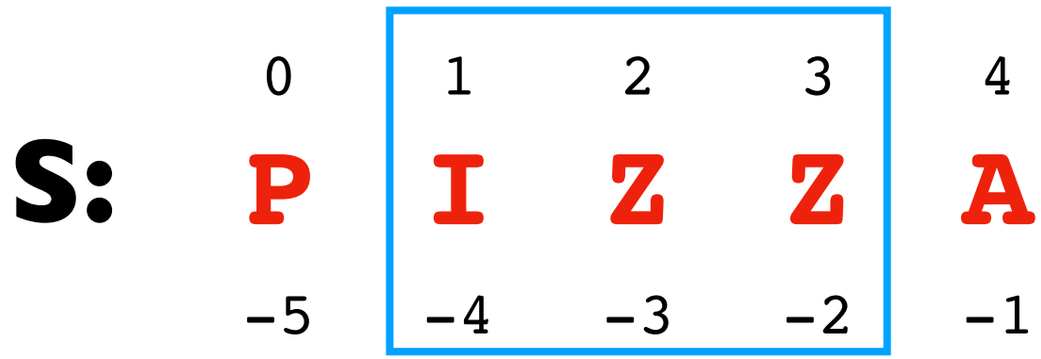


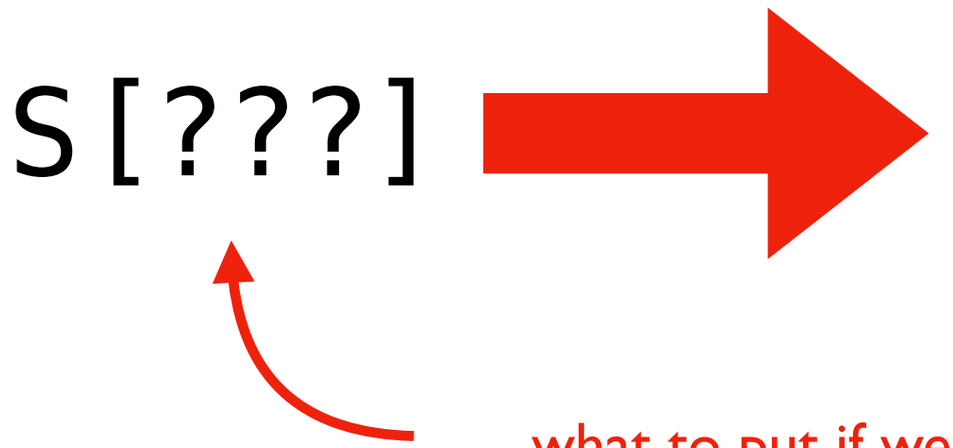
Indexing



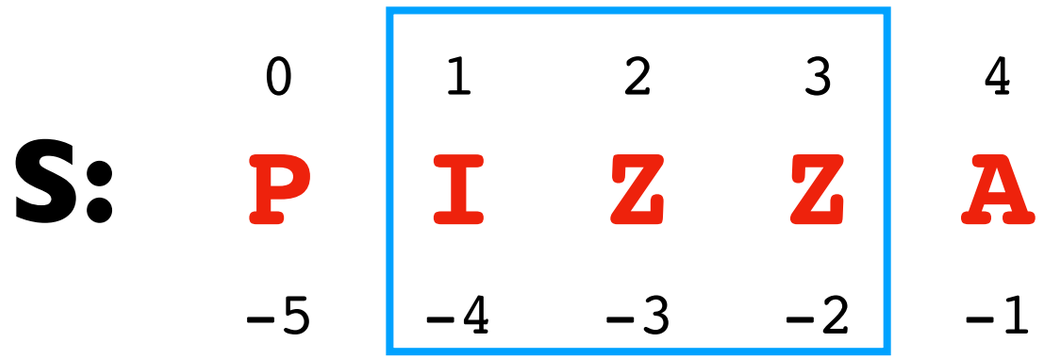
S[-1] → “A”

Slicing

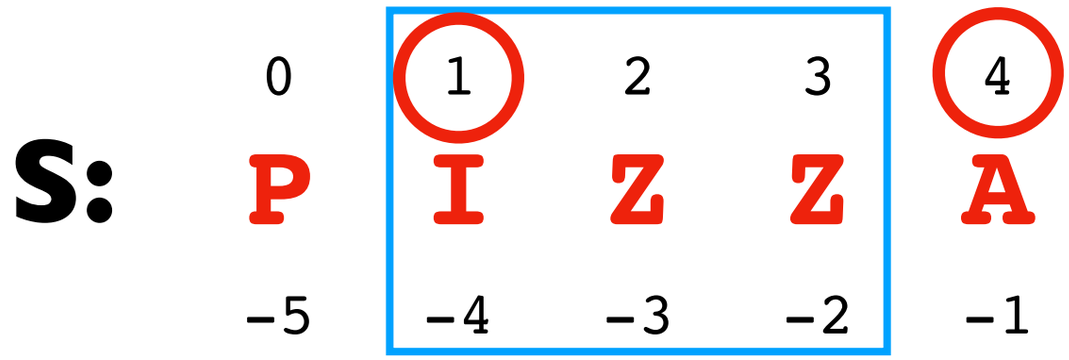

S:
0
1
2
3
4
P
I
Z
Z
A
-5
-4
-3
-2
-1


S[???]
“IZZ”
what to put if we want multiple letters,
like “IZZ”?

Slicing

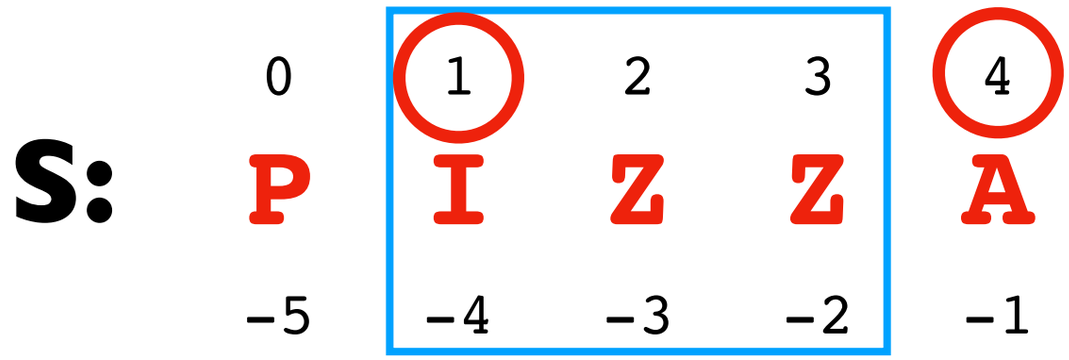


Slicing



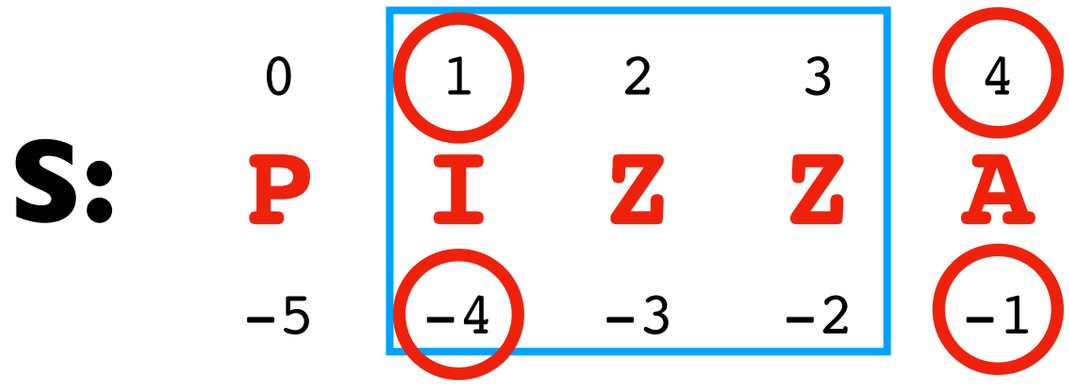
Slicing

start is “inclusive”
end is “exclusive”



S[1:4] → “IZZ”

Slicing

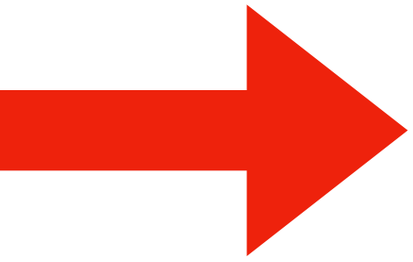


Many different slices give the same result:
S[1:4] == S[1:-1] == S[-4:4] == S[-4:-1]

Slicing

	0	1	2	3	4
S:	**P**	**I**	**Z**	**Z**	**A**
	-5	-4	-3	-2	-1

```
S[1:100]
```



“IZZA”

Slices don’t complain about out-of-range numbers.
You just don’t get data for that part

Slicing

	0	1	2	3	4
S:	**P**	**I**	**Z**	**Z**	**A**
	-5	-4	-3	-2	-1

`S[50:100]` → “”

Slices don’t complain about out-of-range numbers.
You just don’t get data for that part

Slicing

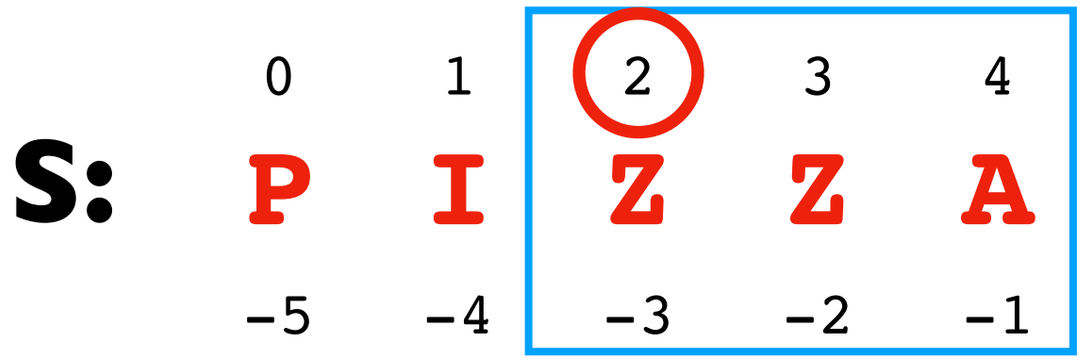
S:

0	1	2	3	4
P	I	Z	Z	A
-5	-4	-3	-2	-1

`S[ : 2]` → “PI”

Feel free to leave out one of the numbers in the slice

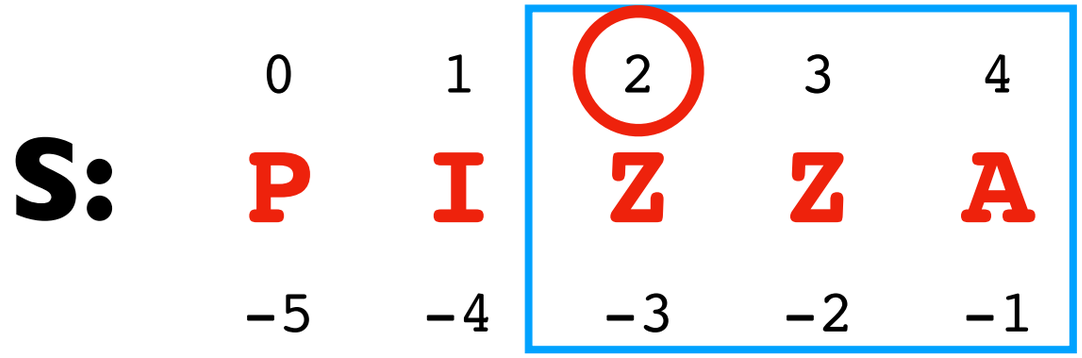
Slicing



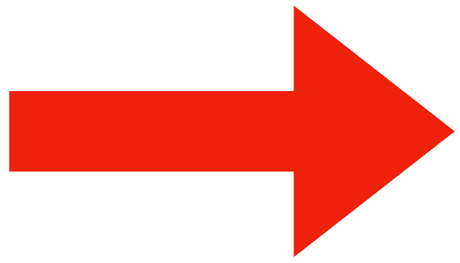
`S[2 : ]` → `"ZZA"`

Feel free to leave out one of the numbers in the slice

Slicing



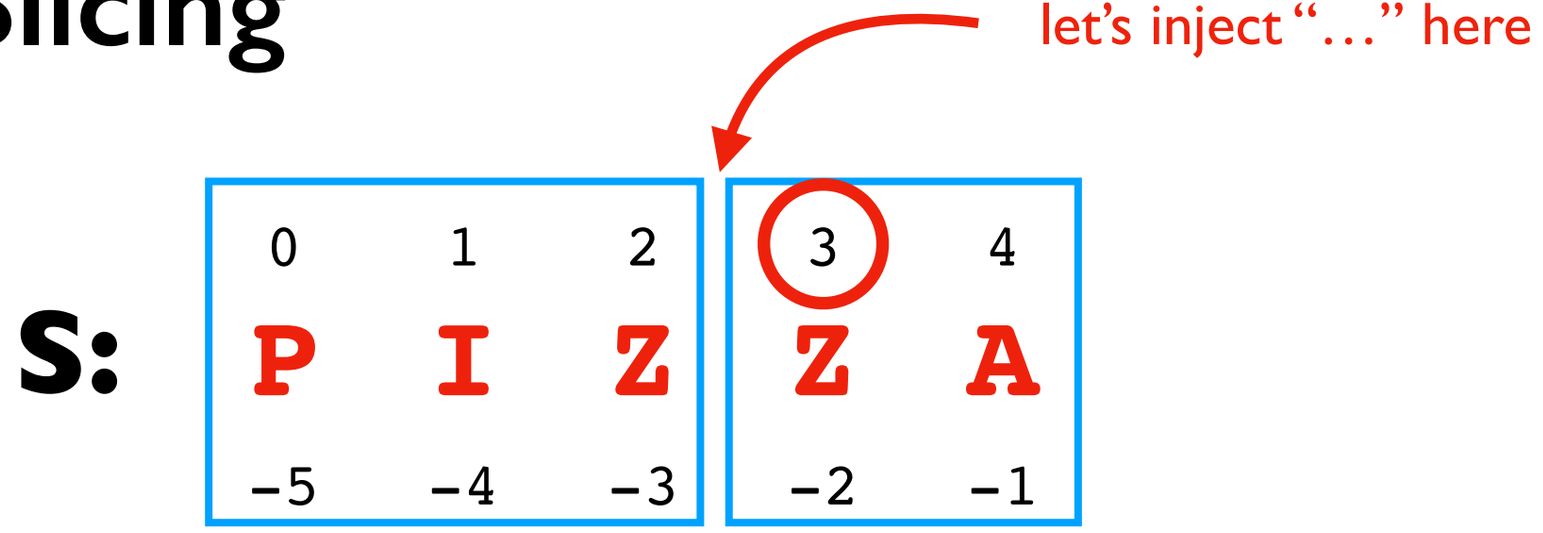

```
S[2 : ]
```



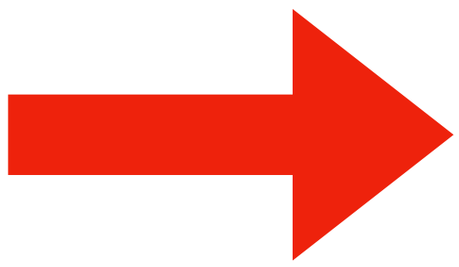
“ZZA”

Inclusive start and exclusive end makes it easier to split and inject things

Slicing




```
S[:3] + "…" + S[3:]
```



```
"PIZ...ZA"
```

Inclusive start and exclusive end makes it easier to split and inject things

Today's Outline

Comparison

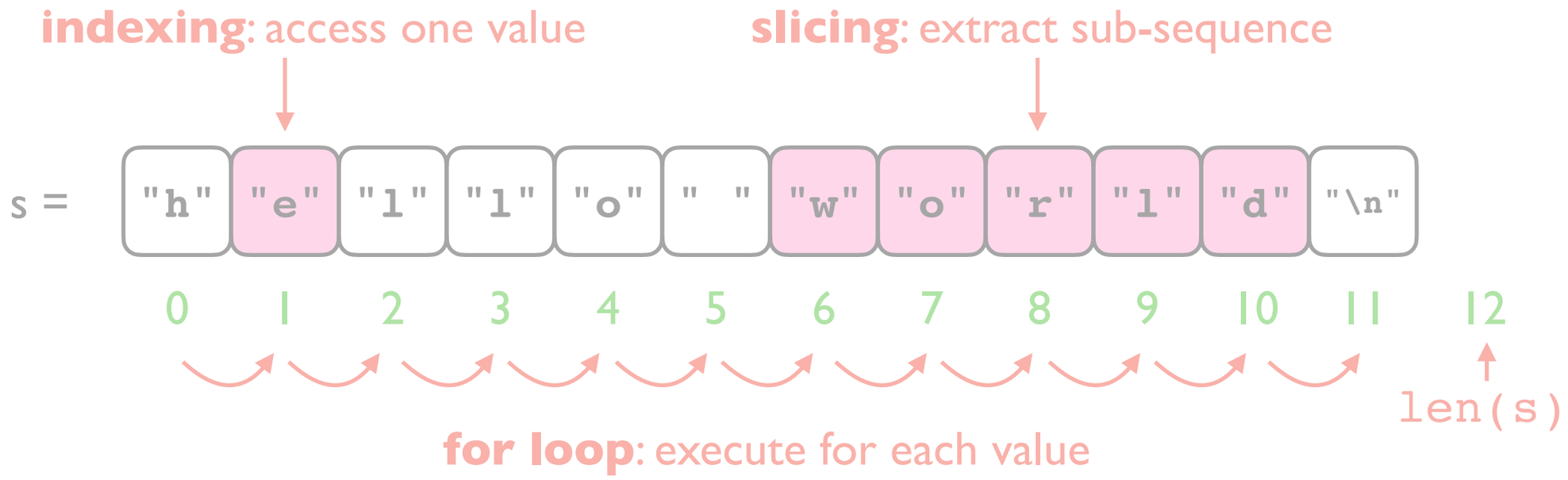
String Methods

Sequences

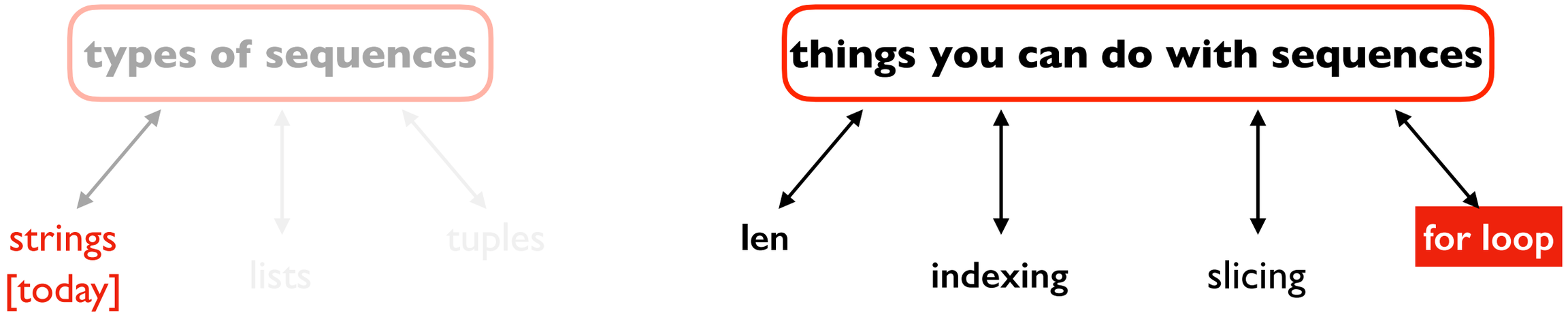
Slicing

for loop over sequence

Python Sequences



Definition: a *string* is a sequence of one-character strings



for syntax

```
msg = "howdy"

for i in range(len(msg)):
    letter = msg[i]
    print(letter)
```

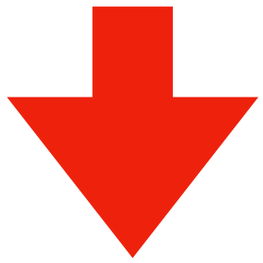
we often want to loop over
all the items in a sequence

for syntax

```
msg = "howdy"

for i in range(len(msg)):
    letter = msg[i]
    print(letter)
```

we often want to loop over all the items in a sequence



shortcut

```
for letter in msg:
    print(letter)
```