JÖNKÖPING UNIVERSITY

School of Engineering

STRINGS IN PYTHON

Peter Larsson-Green

Jönköping University

Autumn 2018

STRINGS

Represents a sequence of characters.

- Expressions creating strings:

```
"This is a string."  →  This is a string.
'This is a string.'  →  This is a string.
"""This is a         →  This is a
string covering         string covering
multiple lines."""      multiple lines.
```

- The + operator can be used to concatenate strings:

```
"This is " + 'a string!'  →  This is a string!
```

STRINGS

```
"Winter" == "Summer"   → False
"Winter" == "winter"   → False
"Hello"  <  "Hi"       → True
4 * 'ab'               → "abababab"
"b" in 'abc'           → True
"bc" in 'abc'          → True
"cb" not in 'abc'      → True
```

STRINGS ARE SEQUENCES

A string is a sequence of characters.

- Each character in the string has an index.

abc	=	a	b	c
Index:		0	1	2
		-3	-2	-1

- Expression retrieving a character at specified index:

```
<str-expr>[ <index-expr> ]
```

```
"abc"[0]    → a        "abc"[2]      → c
"abc"[1]    → b        len("abc")    → 3
```

JÖNKÖPING UNIVERSITY
School of Engineering

ITERATING OVER STRINGS

```
name = "Alice"
for c in name:
  print(c)
```

```
A
l
i
c
e
```

```
name = "Alice"
for i in range(len(name)):
  print(str(i) +" "+ name[i])
```

```
0 A
1 l
2 i
3 c
4 e
```

EXAMPLE

```
def reverse(string):
  reversed = ""
  for c in string:
    reversed = c + reversed
  return reversed
```

```
def sum(numbers):
  sum = 0
  for n in numbers:
    sum = sum + n
  return sum
```

```
reverse("abc")   → cba
reverse("12345") → 54321
```

STRINGS ARE OBJECTS

- Objects have methods.

```
<expr>.method()
```

- Some string methods:

```
"abc abc".capitalize()   → Abc abc
"abc abc".count("b")     → 2
"abc abc".islower()      → True
```

- Strings are immutable.

SOME MORE STRING METHODS

```
"AbC aBc".lower()
```

→ abc abc

```
"abc abc".replace("c ", "xx")
```

→ abxxabc

```
"abc abc".startswith("ab")
```

→ True

```
"AbC aBc".swapcase()
```

→ aBc AbC

```
"Abc abc".upper()
```

→ ABC ABC

```
help(str)
```

SLICING

Extracting a sub sequence of a sequence.

```
Indexes:  01234
name = "Alice"
```

```
<seq-expr> [ : ]
```

```
name[:] → "Alice"
```

```
<seq-expr> [ <expr> : ]
```

```
name[2:] → "ice"
```

```
<seq-expr> [ : <expr> ]
```

```
name[:2] → "Al"
```

SLICING

Extracting a sub sequence of a sequence.

```
Indexes:  01234
name = "Alice"
```

```
<seq-expr> [ <expr> : <expr> ]
```

```
name[1:3]  → "li"
```

```
<seq-expr> [ <expr> : <expr> : <expr> ]
```

```
name[1:4:2]  → "lc"
name[::2]    → "Aie"
name[2::2]   → "ie"
```

```
name[-2::]    → "ce"
name[3:1:-1]  → "ci"
name[::-1]    → "ecilA"
```