

Essential Data Skills for Business Analytics

Lecture 7: Strings in Python

Decision, Operations & Information Technologies
Robert H. Smith School of Business
Spring, 2020

pythonTM

The string data type

- Text is represented in programs by the *string* data type.
- A string is a sequence of characters enclosed within quotation marks (") or apostrophes (')

The string data type

- `>>> str1 = "Hello"`
- `>>> str2 = 'spam'`
- `>>> print (str1, str2)`
- Hello spam
- `>>> type(str1)`
- `<class 'str'>`
- `>>> type(str2)`
- `<class 'str'>`

The string data type

- Getting a string as input

```
>>> firstName = input("Please enter your name: ")
```

```
Please enter your name: John
```

```
>>> print ("Hello", firstName)
```

```
Hello John
```

The string data type

- We can access the individual characters in a string through *indexing*.
- The positions in a string are numbered from the left, **starting with 0**.
- The general form is <string_name>[expr], where the value of expr determines which character is selected from the string.

The string data type

H	e	l	l	o		B	o	b
0	1	2	3	4	5	6	7	8

```
>>> greet = "Hello Bob"
```

```
>>> greet[0]
```

```
'H'
```

```
>>> print (greet[0], greet[2], greet[4])
```

```
H l o
```

```
>>> x = 8
```

```
>>> print (greet[x - 2])
```

```
B
```

The string data type

H	e	l	l	o		B	o	b
0	1	2	3	4	5	6	7	8

- In a string of n characters, the last character is at position $n-1$ since we start counting with 0.
- We can index from the right side using negative indexes.

```
>>> greet[-1]
```

```
'b'
```

```
>>> greet[-3]
```

```
'B'
```

The string data type

- Indexing returns a string containing a single character from a larger string.
- We can also access a contiguous sequence of characters, called a *substring*, through a process called *slicing*.

The string data type

- Slicing:
`<string>[<start>:<end>]`
- start and end should both be integers
- The slice contains the substring beginning at position start and runs up to **but doesn't include** the position end.

The string data type

H	e	l	l	o		B	o	b
0	1	2	3	4	5	6	7	8

```
>>> greet[0:3]
```

```
'Hel'
```

```
>>> greet[5:9]
```

```
' Bob'
```

```
>>> greet[:5]
```

```
'Hello'
```

```
>>> greet[5:]
```

```
' Bob'
```

```
>>> greet[:]
```

```
'Hello Bob'
```

The string data type

- If either expression is missing, then the start or the end of the string are used.
- Can we put two strings together into a longer string?
- *Concatenation* “glues” two strings together (+)
- *Repetition* builds up a string by multiple concatenations of a string with itself (*)

The string data type

```
>>> "spam" + "eggs"
```

'spameggs'

```
>>> "Spam" + "And" + "Eggs"
```

'SpamAndEggs'

```
>>> 3 * "spam"
```

'spamspamspam'

```
>>> "spam" * 5
```

'spamspamspamspamspam'

```
>>> (3 * "spam") + ("eggs" * 5)
```

'spamspamspameggseggseggseggseggss'

The string data type

Operator	Meaning
+	Concatenation
*	Repetition
<string>[]	Indexing
<string>[:]	Slicing
len(<string>)	Length
for <var> in <string>	Iteration through characters

String methods

- `s.capitalize()` – Copy of `s` with only the first character capitalized
- `s.title()` – Copy of `s`; first character of each word capitalized
- `s.center(width)` – Center `s` in a field of given width
- `s.count(sub)` – Count the number of occurrences of `sub` in `s`

String methods

- `s.find(sub)` – Find the first position where `sub` occurs in `s`
- `s.join(list)` – Concatenate list of strings into one large string using `s` as separator.
- `s.ljust(width)` – Like center, but `s` is left-justified
- `s.lower()` – Copy of `s` in all lowercase letters
- `s.lstrip()` – Copy of `s` with leading whitespace removed

String methods

- `s.replace(oldsub, newsub)` – Replace occurrences of `oldsub` in `s` with `newsub`
- `s.rfind(sub)` – Like `find`, but returns the right-most position
- `s.rjust(width)` – Like `center`, but `s` is right-justified
- `s.rstrip()` – Copy of `s` with trailing whitespace removed
- `s.split()` – Split `s` into a list of substrings
- `s.upper()` – Copy of `s`; all characters converted to uppercase