Python Strings: Pre-Class Reading

Unit 3 - Lesson 2 - Part 2 | Bergen Tech CTE

📚 Purpose of This Reading

Before class, please read this document carefully. It covers the syntax differences between JavaScript and Python strings. By completing this reading, you'll be ready to jump straight into coding practice during class time!

Estimated Time: 10-12 minutes
What to do: Read through, try the examples in a Python REPL, and complete the self-check at the end.

🔄 The Big Picture

Good news: The concepts you learned in JavaScript transfer directly to Python! Strings work the same way—they're immutable, zero-indexed, and support similar operations.

What changes: Mostly just syntax (how you write it). The logic stays the same.

📊 JavaScript → Python: Quick Reference Table

Study this table first. It's your cheat sheet!

What You Want to Do	JavaScript Syntax	Python Syntax
Get length of string	`text.length`	`len(text)`
Convert to uppercase	`text.toUpperCase()`	`text.upper()`
Convert to lowercase	`text.toLowerCase()`	`text.lower()`

What You Want to Do	JavaScript Syntax	Python Syntax
Remove leading/trailing spaces	`text.trim()`	`text.strip()`
Check if contains substring	`text.includes("word")`	`"word" in text`
Find position of substring	`text.indexOf("word")`	`text.find("word")`
Extract part of string	`text.slice(0, 5)`	`text[0:5]`
Split into array/list	`text.split(" ")`	`text.split(" ")`
Replace text	`text.replace("old", "new")`	`text.replace("old", "new")`

🔍 Deep Dive: Key Differences

1. Getting Length: `.length` vs `len()`

JavaScript:

```javascript
let username = "alex_2024";
console.log(username.length);  // 9
```

Python:

```python
username = "alex_2024"
print(len(username))  # 9
```

Why the difference? Python uses the same `len()` function for all collections (strings, lists, dictionaries). This keeps the language consistent!

Try it:

```python
password = "MyP@ssw0rd!"
char_count = len(password)
print(f"Your password is {char_count} characters long")
# Your password is 11 characters long
```

---

2. Case Conversion: Drop the "to" Prefix

JavaScript:

```
let color = "blue";
console.log(color.toUpperCase());  // "BLUE"
console.log(color.toLowerCase());  // "blue"
```

Python:

```
color = "blue"
print(color.upper())  # "BLUE"
print(color.lower())  # "blue"
```

Pattern: Python drops the "to" prefix. Shorter = faster to type!

Try it:

```
team_name = "wildcats"
print(team_name.upper())  # "WILDCATS"

title = "THE GREAT GATSBY"
print(title.lower())  # "the great gatsby"
```

---

3. Trimming Whitespace: `trim()` vs `strip()`

JavaScript:

```
let userInput = "   hello world   ";
console.log(userInput.trim());  // "hello world"
```

Python:

```
user_input = "   hello world   "
print(user_input.strip())  # "hello world"
```

Bonus Python powers:

```python
data = "   extra spaces   "

print(data.strip())   # "extra spaces" - both ends
print(data.lstrip())  # "extra spaces   " - left only
print(data.rstrip())  # "   extra spaces" - right only

# Remove specific characters
url = "https://example.com"
print(url.strip("https://"))  # "example.com"
```

---

4. Checking Contents: `includes()` vs `in`

JavaScript:

```javascript
let message = "Welcome to Python class";
console.log(message.includes("Python"));  // true
console.log(message.includes("Java"));    // false
```

Python:

```python
message = "Welcome to Python class"
print("Python" in message)  # True
print("Java" in message)    # False
```

Why it's better: Python's `in` keyword reads like English! "Is 'Python' in message?"

Try it:

```python
bio = "I love coding in JavaScript and Python"

if "Python" in bio:
    print("Found Python!")

if "Ruby" not in bio:
    print("Ruby is not mentioned")
```

---

5. Finding Position: `indexOf()` vs `find()`

JavaScript:

```javascript
let sentence = "The quick brown fox";
console.log(sentence.indexOf("quick"));  // 4
console.log(sentence.indexOf("dog"));    // -1
```

Python:

```python
sentence = "The quick brown fox"
print(sentence.find("quick"))  # 4
print(sentence.find("dog"))    # -1
```

⚠️ **Important:** Both return `-1` if not found (NOT an error). Always check with `in` first!

```python
# Good practice:
text = "Python programming"
search_word = "program"

if search_word in text:
    position = text.find(search_word)
    print(f"Found '{search_word}' at index {position}")
else:
    print(f"'{search_word}' not found")
```

---

6. Slicing: `slice()` vs Bracket Notation

JavaScript:

```javascript
let phrase = "Computer Science";
console.log(phrase.slice(0, 8));   // "Computer"
console.log(phrase.slice(9, 16));  // "Science"
```

Python:

```
phrase = "Computer Science"
print(phrase[0:8])   # "Computer"
print(phrase[9:16])  # "Science"
```

Python slicing is powerful:

```
subject = "Mathematics"
#          0123456789... <- index positions

print(subject[0:4])    # "Math" (indices 0-3)
print(subject[:4])     # "Math" (omit start = from beginning)
print(subject[4:])     # "ematics" (omit end = to end)
print(subject[-4:])    # "tics" (negative = from end)
print(subject[::2])    # "Mteais" (every 2nd character)
```

Key concept: `[start:end]` - start is **included**, end is **excluded**

Try it:

```
full_name = "Sarah Johnson"
first_name = full_name[:5]      # "Sarah"
last_name = full_name[6:]       # "Johnson"
initials = full_name[0] + full_name[6]  # "SJ"
```

---

7. What Stays the Same!

These work identically in both languages:

Splitting strings:

```
// JavaScript
let ingredients = "flour,sugar,eggs,butter";
let items = ingredients.split(",");
// ["flour", "sugar", "eggs", "butter"]
```

```python
# Python
ingredients = "flour,sugar,eggs,butter"
items = ingredients.split(",")
# ["flour", "sugar", "eggs", "butter"]
```

Replacing text:

```javascript
// JavaScript
let oldUrl = "example.com/page";
let newUrl = oldUrl.replace("example", "mysite");
// "mysite.com/page"
```

```python
# Python
old_url = "example.com/page"
new_url = old_url.replace("example", "mysite")
# "mysite.com/page"
```

🎯 New Python Methods to Know

These don't exist (or work differently) in JavaScript:

`title()` - Capitalize First Letter of Each Word

```python
book = "to kill a mockingbird"
print(book.title())  # "To Kill A Mockingbird"

movie = "the lord of the rings"
print(movie.title())  # "The Lord Of The Rings"
```

`capitalize()` - Capitalize Only First Letter

```
city = "new york"
print(city.capitalize())  # "New york" (only first letter)

sentence = "python is awesome"
print(sentence.capitalize())  # "Python is awesome"
```

`startswith()` and `endswith()`

```
filename = "document.pdf"
print(filename.startswith("doc"))   # True
print(filename.endswith(".pdf"))    # True
print(filename.endswith(".docx"))   # False

url = "https://google.com"
print(url.startswith("https://"))   # True
print(url.startswith("http://"))    # False
```

`count()` - Count Occurrences

```
lyrics = "la la la la la"
print(lyrics.count("la"))  # 5

dna_sequence = "ATCGATCGATCG"
print(dna_sequence.count("AT"))  # 3

paragraph = "Python is great. I love Python. Python rocks!"
print(paragraph.count("Python"))  # 3
```

Checking String Content

```
code = "ABC123"
print(code.isalnum())    # True - letters and numbers only

password = "secret123"
print(password.isalpha())    # False - has numbers
print(password.isdigit())    # False - has letters

age = "25"
print(age.isdigit())     # True - only numbers

shout = "HELLO"
print(shout.isupper())   # True - all uppercase

whisper = "hello"
print(whisper.islower()) # True - all lowercase
```

🔗 Method Chaining

Both JavaScript and Python let you chain methods:

JavaScript:

```
let tag = "  #JavaScript  ";
let clean = tag.trim().toLowerCase();
console.log(clean);  // "#javascript"
```

Python:

```
tag = "  #JavaScript  "
clean = tag.strip().lower()
print(clean)  # "#javascript"
```

Even better - multiple chains:

```
raw_title = "  the MATRIX reloaded  "
clean_title = raw_title.strip().lower().title()
print(clean_title)  # "The Matrix Reloaded"
```

How it works: Each method returns a string, so you can immediately call another method on that result.

Try it:

```
messy_data = "  PRODUCT-123  "
cleaned = messy_data.strip().lower().replace("-", "_")
print(cleaned)  # "product_123"
```

💡 F-Strings: Python's Template Literals

JavaScript:

```
let product = "laptop";
let price = 899;
console.log(`The ${product} costs $${price}`);
```

Python:

```
product = "laptop"
price = 899
print(f"The {product} costs ${price}")
```

Pattern: Use `f"..."` and put variables in `{curly braces}`. Works exactly like JavaScript template literals!

Try it:

```
name = "Jordan"
score = 95
grade = "A"
print(f"{name} scored {score}% and earned an {grade}")
# Jordan scored 95% and earned an A

# You can even do math inside f-strings!
quantity = 3
price = 12.99
print(f"Total: ${quantity * price:.2f}")
# Total: $38.97
```

⚠️ Critical Concepts: What You MUST Remember

1. Strings Are Immutable

You **cannot** change a string in place:

```
word = "cat"
word[0] = "b"  # ❌ TypeError! Can't change strings directly

# Instead, create a new string:
word = "b" + word[1:]  # ✓ "bat"
# OR use replace:
word = word.replace("c", "b")  # ✓ "bat"
```

2. Methods Return New Strings

String methods don't modify the original:

```
original = "Python"
original.lower()  # ❌ Result is lost!
print(original)   # Still "Python"

# You must save the result:
original = original.lower()  # ✓ Correct
print(original)  # "python"
```

Real example:

```
username = "ALEX123"
username.lower()  # This does nothing!
# username is still "ALEX123"

username = username.lower()  # Save it!
# Now username is "alex123"
```

3. Python Is Case-Sensitive

```
"Java" in "I love JavaScript"  # False! (capital J vs lowercase j)
"java" in "I love JavaScript"  # False! (lowercase doesn't match capital)

# Best practice: normalize case first
language = "JavaScript"
print("java" in language.lower())  # True!

# Common use case: case-insensitive comparison
search = "python"
languages = ["JavaScript", "Python", "Ruby", "Java"]

for lang in languages:
    if search.lower() == lang.lower():
        print(f"Found {lang}!")
```