STRING AND CHARACTER DATA IN PYTHON

STRING AND CHARACTER DATA IN PYTHON

What will you learn in this tutorial?

- How to use operators with strings
- How to access and extract portions of strings
- Methods to manipulate and modify string data
- How to use two other Python objects to represent raw byte data
    - `bytes` and `bytearray` Objects

TABLE OF CONTENTS

1. **Section 1: String Manipulation**

▶ **1.1 Intro and Course Overview**

1.2 String Operators

1.3 Built-in String Functions

1.4 String Indexing

1.5 String Slicing

1.6 Interpolating Variables Into a String

1.7 Modifying Strings

2. Section 2: Built-in String Methods

3. Section 3: `bytes` Objects

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. Section 2: Built-in String Methods
   - 2.1 String Methods Overview
   - 2.2 Case Conversion
   - 2.3 Find and Seek
   - 2.4 Character Classification
   - 2.5 String Formatting
   - 2.6 Converting Between Strings and Lists
3. Section 3: `bytes` Objects
   - 3.1 `bytes` Objects Overview
   - 3.2 Defining a Literal `bytes` Object
   - 3.3 Defining a `bytes` Object with the Built-in `bytes()` Function
   - 3.4 Operations on `bytes` Objects
   - 3.5 `bytearray` Objects
   - 3.6 Conclusion and Course Review

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
   - **1.1 Intro and Course Overview**
   - ▶ **1.2 String Operators**
   - 1.3 Built-in String Functions
   - 1.4 String Indexing
   - 1.5 String Slicing
   - 1.6 Interpolating Variables Into a String
   - 1.7 Modifying Strings
2. Section 2: Built-in String Methods
3. Section 3: `bytes` Objects

STRING OPERATORS

A couple of operators that can be used on numeric operands can be applied to strings as well

- **The `+` Operator**
- **The `*` Operator**

And a membership operator that can be used with strings

- **The `in` Operator**

STRING OPERATORS

The + Operator

- **Concatenates strings**
    - Returns a string consisting of the operands joined together

STRING OPERATORS

The `*` Operator

- **Creates multiple copies of a string**
    - Returns a string consisting of `n` concatenated copies of a string

STRING OPERATORS

The `in` Operator

- **A membership operator that can be used with strings**
    - Returns `True` if the first operand is contained within the second
    - Returns `False` otherwise
    - Also can be used as `not in`

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
   - **1.1 Intro and Course Overview**
   - **1.2 String Operators**
   - ▶ **1.3 Built-in String Functions**
   - 1.4 String Indexing
   - 1.5 String Slicing
   - 1.6 Interpolating Variables Into a String
   - 1.7 Modifying Strings
2. Section 2: Built-in String Methods
3. Section 3: `bytes` Objects

BUILT-IN STRING FUNCTIONS

A few functions built-in to the Python interpreter that work with strings

Function	Description
`chr()`	Converts an integer to a character
`ord()`	Converts a character to an integer
`len()`	Returns the length of a string
`str()`	Returns a string representation of an object

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
   - **1.1 Intro and Course Overview**
   - **1.2 String Operators**
   - **1.3 Built-in String Functions**
   - ▶ **1.4 String Indexing**
   - 1.5 String Slicing
   - 1.6 Interpolating Variables Into a String
   - 1.7 Modifying Strings
2. Section 2: Built-in String Methods
3. Section 3: `bytes` Objects

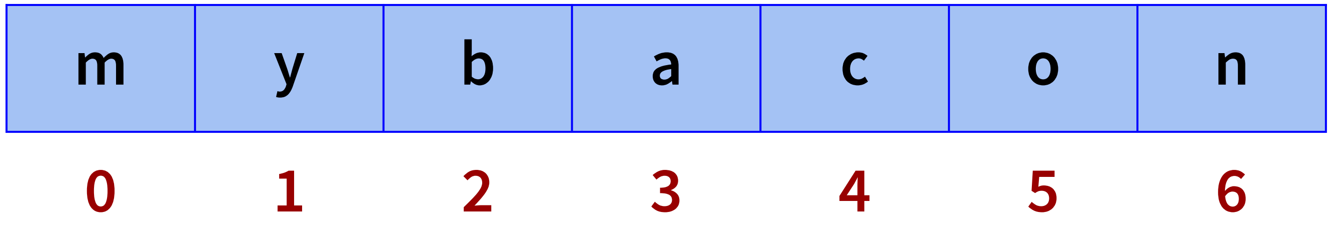
STRING INDEXING

Strings are ordered sequences of character data

- **Individual characters of a string can be accessed directly using a numeric index**
    - String indexing in Python is zero-based
        - The first character in the string has index 0
        - The next has index 1 … and so on
        - The index of the last character will be the length of the string minus one.

STRING INDEXING

An example



String Indices

STRING INDEXING

Negative indexing

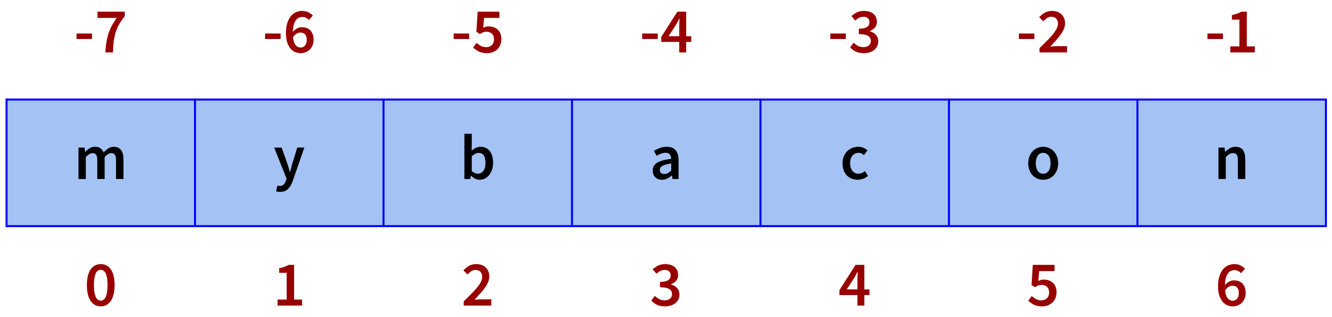


TABLE OF CONTENTS

1. **Section 1: String Manipulation**
    - **1.1 Intro and Course Overview**
    - **1.2 String Operators**
    - **1.3 Built-in String Functions**
    - **1.4 String Indexing**
    - ▶ **1.5 String Slicing**
    - 1.6 Interpolating Variables Into a String
    - 1.7 Modifying Strings
2. Section 2: Built-in String Methods
3. Section 3: `bytes` Objects

STRING SLICING

Indexing syntax that extracts substrings from a string

- **If `s` is a string `s[m:n]` returns the portion of `s`**
    - Starting with position `m`
    - And up to but not including position `n`

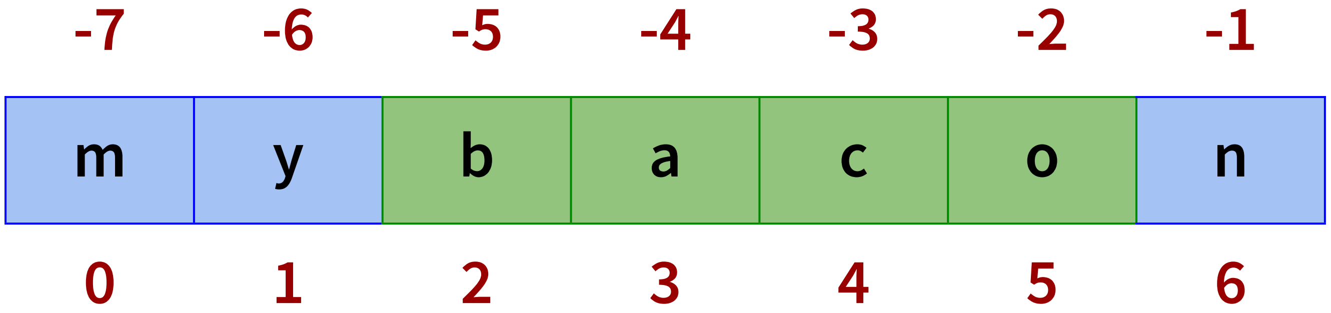
STRING SLICING

Omitting the first and/or last index

- **Omitting the first index `s[:n]` starts the slice at the beginning of the string**
- **Omitting the last index `s[m:]` extends the slice from the first index `m` to the end of the string**
- **Omitting both indexes `s[:]` returns the entire original string**
    - It's not a copy, it's a reference to the original string

STRING SLICING

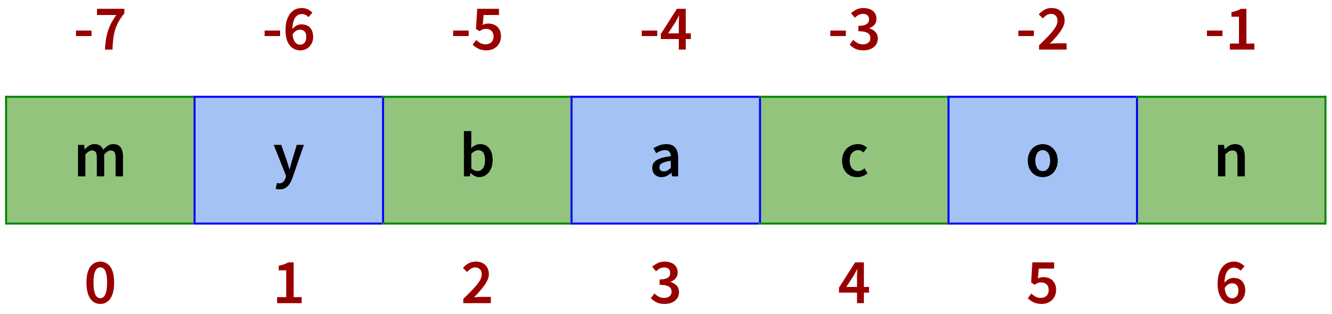
Negative indexing works as well



STRING SLICING

Specifying a Stride in a String Slice

- **Adding an additional : and a third index designates a stride (also called a step)**
- **For the slice** `[0:7:2]`



STRING SLICING

Specifying a Stride in a String Slice

- **For the slice** [1:7:2]

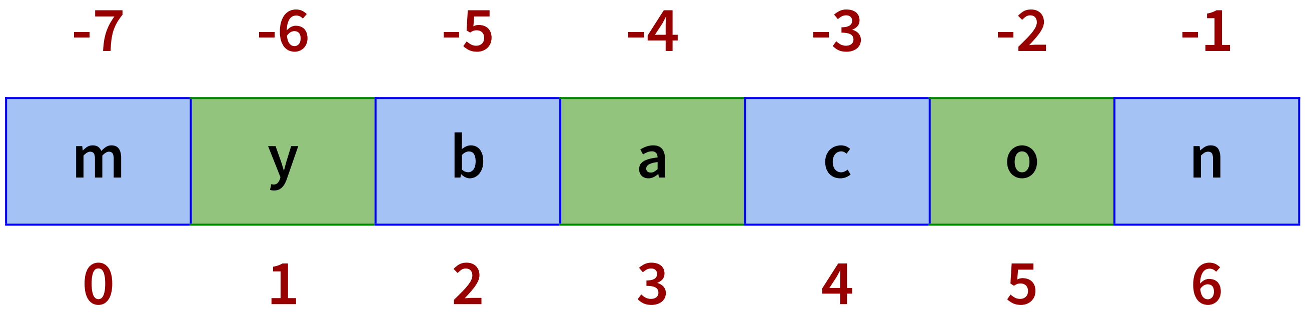


TABLE OF CONTENTS

1. **Section 1: String Manipulation**
   - **1.1 Intro and Course Overview**
   - **1.2 String Operators**
   - **1.3 Built-in String Functions**
   - **1.4 String Indexing**
   - **1.5 String Slicing**
   - ▶ **1.6 Interpolating Variables Into a String**
   - 1.7 Modifying Strings
2. Section 2: Built-in String Methods
3. Section 3: `bytes` Objects

INTERPOLATING VARIABLES INTO A STRING

Formatted String Literal - nicknamed the f-string

- **Covered in much more depth in the course**
  Python 3's f-Strings: An Improved String Formatting Syntax
- **A quick preview with the feature - Variable Interpolation**

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
   - 1.1 **Intro and Course Overview**
   - 1.2 **String Operators**
   - 1.3 **Built-in String Functions**
   - 1.4 **String Indexing**
   - 1.5 **String Slicing**
   - 1.6 **Interpolating Variables Into a String**
   - ▶ 1.7 **Modifying Strings**
2. Section 2: Built-in String Methods
3. Section 3: `bytes` Objects

MODIFYING STRINGS

Can you modify a string?

- **Strings are immutable**
- **Making a copy instead**

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. **Section 2: Built-in String Methods**

▶ **2.1 String Methods Overview**

2.2 Case Conversion

2.3 Find and Seek

2.4 Character Classification

2.5 String Formatting

2.6 Converting Between Strings and Lists

3. Section 3: `bytes` Objects

BUILT-IN STRING METHODS

Methods are similar to functions

- **A method is a specialized type of callable procedure that is tightly associated with an object.**
- **Like a function, a method is called to perform a distinct task**
- **But it is invoked on a specific object and has knowledge of its target object during execution**
- `obj.foo(<args>)`

BUILT-IN STRING METHODS

Categories of String Methods

- **Case Conversion**
- **Find and Seek**
- **Character Classification**
- **String Formatting**
- **Converting Between Strings and Lists**

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. **Section 2: Built-in String Methods**
   - **2.1 String Methods Overview**
   - ▶ **2.2 Case Conversion**
   - 2.3 Find and Seek
   - 2.4 Character Classification
   - 2.5 String Formatting
   - 2.6 Converting Between Strings and Lists
3. Section 3: `bytes` Objects

STRING METHODS - CASE CONVERSION

- `str.capitalize()`
- `str.lower()`
- `str.swapcase()`
- `str.title()`
- `str.upper()`

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. **Section 2: Built-in String Methods**
   - **2.1 String Methods Overview**
   - **2.2 Case Conversion**
   - ▶ **2.3 Find and Seek**
   - 2.4 Character Classification
   - 2.5 String Formatting
   - 2.6 Converting Between Strings and Lists
3. Section 3: `bytes` Objects

STRING METHODS - FIND AND SEEK

- `str.count(<sub>[, <start>[, <end>]])`
- `str.endswith(<sub>[, <start>[, <end>]])`
- `str.startswith(<sub>[, <start>[, <end>]])`
- `str.find(<sub>[, <start>[, <end>]])`
- `str.rfind(<sub>[, <start>[, <end>]])`
- `str.index(<sub>[, <start>[, <end>]])`
- `str.rindex(<sub>[, <start>[, <end>]])`

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. **Section 2: Built-in String Methods**
   - **2.1 String Methods Overview**
   - **2.2 Case Conversion**
   - **2.3 Find and Seek**
   - ▶ **2.4 Character Classification**
   - 2.5 String Formatting
   - 2.6 Converting Between Strings and Lists
3. Section 3: `bytes` Objects

STRING METHODS - CHARACTER CLASSIFICATION

- `str.isalnum()`
- `str.isalpha()`
- `str.isdigit()`
- `str.isidentifier()`
- `iskeyword(<str>)`
  - Not a string method
  - A function imported from the keyword module
- `str.isprintable()`
- `str.isspace()`
- `str.istitle()`
- `str.islower()`
- `str.isupper()`
- `str.isascii()`
  (introduced python 3.7)

CHARACTER CLASSIFICATION

`str.isidentifier()`

- Determines whether the target string is a valid Python identifier
- What is a Python identifier?
    - A name that is used to define a variable, function, class, or some other type of object
    - Must begin with an alphabetic character or underscore (_)
        - Can be a single character
        - Can be followed by any alphanumeric or the underscore
        - Cannot have other punctuation characters

CHARACTER CLASSIFICATION

Python Keywords

False	break	else	if	not	while
True	class	except	import	or	with
None	continue	finally	in	pass	yield
and	def	for	is	raise	
as	del	from	lambda	return	
assert	elif	global	nonlocal	try	

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. **Section 2: Built-in String Methods**
   - **2.1 String Methods Overview**
   - **2.2 Case Conversion**
   - **2.3 Find and Seek**
   - **2.4 Character Classification**
   - ▶ **2.5 String Formatting**
   - 2.6 Converting Between Strings and Lists
3. Section 3: `bytes` Objects

STRING METHODS - STRING FORMATTING

- `str.center(<width>[, <fill>])`
- `str.expandtabs(tabsize=8)`
- `str.ljust(<width>[, <fill>])`
- `str.rjust(<width>[, <fill>])`
- `str.lstrip([<chars>])`
- `str.rstrip([<chars>])`
- `str.strip([<chars>])`

STRING METHODS - STRING FORMATTING

Continued

- `str.replace(<old>, <new>[, <count>])`
- `str.zfill(<width>)`

TABLE OF CONTENTS

1. Section 1: String Manipulation
2. Section 2: Built-in String Methods
   - 2.1 String Methods Overview
   - 2.2 Case Conversion
   - 2.3 Find and Seek
   - 2.4 Character Classification
   - 2.5 String Formatting
   - ▶ 2.6 Converting Between Strings and Lists
3. Section 3: `bytes` Objects

CONVERTING FROM STRINGS AND LISTS

These methods operate on or return **iterables**

- **A general Python term for a sequential collection of objects**

Many of these methods return either a list or a tuple, which are very similar collections of ordered objects, with a couple of differences

- **List**
  - **enclosed in square brackets - [ ]**
  - **mutable**
- **Tuple**
  - **enclosed in parentheses - ( )**
  - **immutable**

CONVERTING BETWEEN STRINGS AND LISTS

- `str.join(<iterable>)`
- `str.partition(<sep>)`
- `str.rpartition(<sep>)`
- `str.split(sep=None, maxsplit=-1])`
- `str.rsplit(sep=None, maxsplit=-1])`

CONVERTING BETWEEN STRINGS AND LISTS

Continued

- `str.splitlines([<keepends>])`

Escape Sequence	**Character**	**Escape Sequence**	**Character**
`\n`	Newline	`\x1d`	Group Separator
`\r`	Carriage Return	`\x1e`	Record Separator
`\r\n`	Carriage Return + Line Feed	`\x85`	Next Line (C1 Control Code)
`\v or \x0b`	Line Tabulation	`\u2028`	Unicode Line Separator
`\f or \x0c`	Form Feed	`\u2029`	Unicode Paragraph Separator
`\x1c`	File Separator		

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. **Section 2: Built-in String Methods**
3. **Section 3: `bytes` Objects**

▶ **3.1 `bytes` Objects Overview**

3.2 Defining a Literal `bytes` Object

3.3 Defining a `bytes` Object with the Built-in `bytes()` Function

3.4 Operations on `bytes` Objects

3.5 `bytearray` Objects

3.6 Conclusion and Course Review

`bytes` OBJECTS OVERVIEW

The `bytes` Object

- **One of the core built-in types for manipulating binary data**
- **A `bytes` object is an immutable sequence of single byte values**
- **Each element in a `bytes` object is a small integer in the range of `0` to `255`**

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. **Section 2: Built-in String Methods**
3. **Section 3: `bytes` Objects**

▶ **3.1 `bytes` Objects Overview**

3.2 Defining a Literal `bytes` Object

3.3 Defining a `bytes` Object with the Built-in `bytes()` Function

3.4 Operations on `bytes` Objects

3.5 `bytearray` Objects

3.6 Conclusion and Course Review

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. **Section 2: Built-in String Methods**
3. **Section 3: `bytes` Objects**

3.1 `bytes` Objects Overview

▶ **3.2 Defining a Literal `bytes` Object**

3.3 Defining a `bytes` Object with the Built-in `bytes()` Function

3.4 Operations on `bytes` Objects

3.5 `bytearray` Objects

3.6 Conclusion and Course Review

DEFINING A LITERAL `bytes` OBJECT

A `bytes` literal is defined similarly to a string literal

- **Requires an additional `'b'` prefix**
- **Single, double, or triple quoting mechanisms can be used**
- **Only ASCII characters are allowed**
    - **Any character value greater than 127 must be specified using an appropriate escape sequence**
- **The `'r'` prefix can be used to disable processing of escape sequences**

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. **Section 2: Built-in String Methods**
3. **Section 3: `bytes` Objects**

3.1 **`bytes` Objects Overview**
3.2 **Defining a Literal `bytes` Object**
▶ **3.3** **Defining a `bytes` Object with the Built-in `bytes()` Function**
3.4 Operations on `bytes` Objects
3.5 `bytearray` Objects
3.6 Conclusion and Course Review

DEFINING `bytes` OBJECT WITH `bytes()`

The `bytes()` function also creates a `bytes()` object

- `bytes(<s>, <encoding>)`
  - **Creates a bytes object from a string**
- `bytes(<size>)`
  - **Creates a bytes object consisting of null (`0x00`) bytes**
- `bytes(<iterable>)`
  - **Creates a bytes object from an iterable**

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. **Section 2: Built-in String Methods**
3. **Section 3: `bytes` Objects**
   - **3.1 `bytes` Objects Overview**
   - **3.2 Defining a Literal `bytes` Object**
   - **3.3 Defining a `bytes` Object with the Built-in `bytes()` Function**
   - ▶ **3.4 Operations on `bytes` Objects**
   - 3.5 `bytearray` Objects
   - 3.6 Conclusion and Course Review

OPERATIONS ON `bytes` OBJECTS

`bytes` objects support the common sequence operations

- **The `in` and `not in` operators**
- **Concatenation `(+)` and replication `(*)` operators**
- **Indexing and slicing**
- **Built-in functions**
    - `len()` `min()` `max()`
- **Many of the methods for string objects are valid for `bytes` objects**
- `bytes.fromhex(<s>)` and `b.hex()`

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. **Section 2: Built-in String Methods**
3. **Section 3: `bytes` Objects**

   3.1 **`bytes` Objects Overview**

   3.2 **Defining a Literal `bytes` Object**

   3.3 **Defining a `bytes` Object with the Built-in `bytes()` Function**

   3.4 **Operations on `bytes` Objects**

   ▶ 3.5 **`bytearray` Objects**

   3.6 Conclusion and Course Review

`bytearray` OBJECTS

`bytearray` objects are another type of binary sequence

- **Differences**
  - **There is no dedicated syntax for defining a `bytearray` literal**
  - **A `bytearray` is always created using the `bytearray()` built-in function**
  - **`bytearray` objects are mutable**

CONGRATULATIONS
YOU'VE COMPLETED THE COURSE!

STRING AND CHARACTER DATA
IN PYTHON

TABLE OF CONTENTS

1. **Section 1: String Manipulation**
2. **Section 2: Built-in String Methods**
3. **Section 3: `bytes` Objects**

3.1 `bytes` Objects Overview
3.2 Defining a Literal `bytes` Object
3.3 Defining a `bytes` Object with the Built-in `bytes()` Function
3.4 Operations on `bytes` Objects
3.5 `bytearray` Objects
▶ 3.6 Conclusion and Course Review

THANK YOU!

PRACTICE WITH WHAT YOU HAVE LEARNED