Introduction

In this course, you'll learn about...

1. The **string** `.format()` **method**
2. The **formatted string literal**, or **f-string**

TABLE OF CONTENTS

The Python String `.format()` method

- A method for “interpolating” Python values into pre-built strings
- Useful for creating dynamic text
- `<template>.format(<pos_args>, <keyword_args>)`
- The template string can contain *replacement fields*, which can be as simple as a pair of curly braces (`{}`) or as complex as a dictionary access or a reference to an object field (`{keyword.field[“dict_key”]}`)
- Replacement fields are then substituted for by the arguments to the `.format()` method

TABLE OF CONTENTS

Replacement fields: name and conversion

- Replacement fields have their own syntax following the format `{[<name>][!<conversion>][:<format_spec>]}`
- **Name:** Either nothing, an index in the list of positional arguments, or a keyword to access a keyword argument; support list indices, dictionary key references, or object attributes.
- **Conversion:** Indicates the method used to convert the object to a string—`s` for `str()` (this is the default), `r` for `repr()`, and `a` for `ascii()`.

TABLE OF CONTENTS

Replacement fields: format specification

- `{[<name>][!<conversion>][:<format_spec>]}`
- The format spec itself has a detailed sub-specification: `:[[<fill>]<align>][<sign>][#][0][<width>][<group>][.<prec>][<type>]`

Replacement fields: format specification

`<fill>`	Specifies how to pad values that don't occupy the entire field width	`{:x>6}`
`<align>`	Specifies how to justify values that don't occupy the entire field width	`{:x>6}`
`<sign>`	Controls whether a leading sign is included for numeric values	`{:>+6}`
`#`	Selects an alternate output form for certain presentation types	`{:>6#x}`
`0`	Causes values to be padded on the left with zeros instead of spaces	`{:x>06}`
`<width>`	Specifies the minimum width of the output	`{:x>6}`
`<group>`	Specifies a grouping character for numeric output	`{:x>6,}`
`.<prec>`	Specifies the number of digits after the decimal point, or the maximum output width for string presentation types	`{:x>6.4}`
`<type>`	Specifies the presentation type (what type to convert the output to)	`{:x>6b}`

TABLE OF CONTENTS

Nested replacement fields

- Replacement fields can be created dynamically
- Replacement names, conversions, and format specs are all fair game
- This allows you to create functions and classes that can format their own output in whatever way best suits your data

TABLE OF CONTENTS

F-Strings

- “f-strings,” or formatted string literals, are a convenient shorthand for the `template.format()` method
- An f-string simply looks like a normal string with an “f” in front of it, e.g. `f”This is an f-string with a {replacement}”`
- Replacements in f-strings generally have the same rules for format specifiers and conversion as those in `.format()` templates
- However, f-string expressions can’t be empty, and they can’t contain backslashes (\) or comments

Conclusion

In this course, you learned about...

1. The **string** `.format()` **method**
2. The `[name][!conversion][:format_spec]` syntax
3. The **formatted string literal**, or **f-string**