📚 Python String Formatting Basics

Brief Overview

This note covers **Python string formatting** and was created from the Lecture 5 : Python f-Strings & .format() Explained | Make Your Prints Look Pro | #2025 #python YouTube video. This 11-minute lecture explains how to use the older .format() method and the newer f-Strings for inserting variables into text, how to reorder placeholders with indices, how to round numbers using format specifiers (e.g., {:.2f}), and why f-Strings are generally more readable and faster.

Key Points

- Understand the difference between .format() and f-Strings for inserting values.
- Learn to reorder placeholders using index numbers.
- Format numbers with precision specifiers (e.g., {:.2f}).
- Appreciate the readability and performance benefits of f-Strings.

📄 Variables for Formatting

- name = "Jane"
- surname = "Tracy"
- age = 18

> **Format string** – a string that contains placeholders (curly brackets) which are replaced by variable values when the string is formatted.

🛠️ .format() Method (outdated)

Syntax

```
"{} {} is {} years old".format(name, surname, age)
```

- Curly brackets {} act as **placeholders**.
- Values are inserted **positionally** (first placeholder → first argument, etc.).

Index-based placeholders

```
"{0} {2} is {1} years old".format(name, age, surname)
```

Placeholder	Refers to
{0}	name
{1}	age
{2}	surname

- Changing the index order lets you rearrange output without altering the argument list.

✨ f-Strings (newest)

Syntax

```
f"{name} {surname} is {age} years old"
```

- Variable names are placed **directly** inside {}.
- No need for a separate .format() call or a period after the string.

> **f-String** – a string prefixed with f that allows inline expression evaluation inside {}.

Comparison

Feature	.format()	f-String
Syntax	"{} {}".format(a, b)	f"{a} {b}"
Inline variable use	No (needs positional or named args)	Yes (write variable name)
Readability	Moderate	High
Performance	Slightly slower	Faster

🔢 Formatting Numbers (Rounding)

General specifier

:{precision}f – rounds a floating-point number to the given number of digits after the decimal point.

Example: round to two decimal places (hundredths)

```
print("{:.2f}".format(3.456778))   # → 3.46
```

- :.2f means **"format as a float with 2 digits after the decimal."**
- Rounding follows standard rules: look at the next digit; if ≥ 5, increase the last retained digit by 1.

Same with an f-String

```
f"{3.456778:.2f}"
```

- Produces the identical result 3.46.

Rounding logic (illustrative)

- **.1f** → nearest **tens** (first digit after the decimal).
- **.2f** → nearest **hundredths** (second digit).
- **.3f** → nearest **thousandths** (third digit), and so on.

📚 Reference Resource

- **piformat.info** – an online reference for Python string formatting details and specifiers.