#📚 Python f-Strings & .format()

### Brief Overview

This note covering **Python string formatting** was created from the Lecture 5 : Python f-Strings & .format() Explained | Make Your Prints Look Pro | #2025 #python YouTube video. It covers the print function basics, old-style .format() usage, new-style f-strings, formatting specifiers, and rounding rules.

### Key Points

- Learn how the print function handles multiple arguments and line endings.
- Master old-style .format() with indexed placeholders and reordering.
- Use new-style f-strings for concise inline variable insertion.
- Apply formatting specifiers like .2f to round numbers to desired precision.

---

## 📄 Print Function Basics

> The **print()** function displays text or variable values to the console.

- Can accept multiple arguments separated by commas.
- Automatically adds a newline unless end is specified.

---

## 🔧 String Formatting Overview

### 🏛️ Old-style .format() method

> A **format string** is a regular string that contains **curly-bracket placeholders** ({}) which are replaced by values supplied to the .format() method.

- Placeholders are filled **in order** of the arguments.
- Index numbers can be used to **reorder** or **repeat** values.

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

```
# Simple ordered placement
print("{} {} is {} years old".format(name, surname, age))
# Output: Jane Tracy is 18 years old

# Using explicit indexes
print("{1} {0} is {2} years old".format(name, surname, age))
# Output: Tracy Jane is 18 years old

# Repeating a placeholder
print("{0} {0} is {2}".format(name, surname, age))
# Output: Jane Jane is 18 years old
```

- **Index reference table**

| Index | Variable | Example Placeholder |
|---|---|---|
| 0 | name | {0} |
| 1 | surname | {1} |
| 2 | age | {2} |

### ✨ New-style f-strings

> An **f-string** is a string literal prefixed with f that allows **direct insertion** of variable names inside {}.

- No need for a separate .format() call.
- Supports the same format specifiers as .format().

```
print(f"{name} {surname} is {age} years old")
# Output: Jane Tracy is 18 years old
```

- Variables can be **reordered** simply by changing their placement inside the braces.

---

## 📊 Formatting Specifiers (Rounding)

> A **format specifier** controls how a value is presented, such as number of decimal places.

- Syntax: {value:.type}
- :.2f → round to **2 decimal places** (nearest hundredths).

```
number = 3.456778
print("{:.2f}".format(number))   # Output: 3.46
print(f"{number:.2f}")           # Output: 3.46
```

### Rounding Rules Illustrated

| Specifier | Target digit | Next digit examined | Result (example = 3.45678) |
|---|---|---|---|
| :.1f | 1st decimal | 2nd decimal (5) | 3.5 (round up) |
| :.2f | 2nd decimal | 3rd decimal (6) | 3.46 (round up) |
| :.3f | 3rd decimal | 4th decimal (7) | 3.457 (round up) |
| :.4f | 4th decimal | 5th decimal (7) | 3.4568 (round up) |

- If the next digit is **≥ 5**, add 1 to the target digit; otherwise, leave it unchanged.

---

## 🌐 Helpful Resource

> **piformat.info** – an online reference for Python string formatting syntax and examples.

- Use it to look up obscure specifiers or to verify formatting behavior without memorizing every detail.