**✅ 1. Output Formatting — Formatowanie danych wyjściowych**

**🔍 Czym jest output formatting?**

Formatowanie danych wyjściowych to sposób, w jaki prezentujemy dane (np. liczby, tekst, daty) na ekranie lub w pliku, żeby były **czytelne, zrozumiałe i estetyczne**. Przykład: prezentowanie danych w tabeli, ograniczenie liczby miejsc po przecinku, wyrównanie tekstu.

W Pythonie mamy kilka sposobów formatowania:

1. str.format()
2. f-stringi (od Pythona 3.6)
3. %-formatting (starszy styl, rzadziej używany)

**📌 1. str.format() — klasyczne podejście**

print("Hello, {}. You have {} new messages.".format("Szymon", 5))

🔍 Działa tak, że {} to miejsce, gdzie wstawiana jest dana, a .format() dostarcza wartości.

➡️ Można wskazać indeksy:

print("First: {0}, Second: {1}, Again first: {0}".format("A", "B"))

➡️ Lub użyć nazwanych argumentów:

print("User: {name}, Score: {points}".format(name="Szymon", points=98))

➡️ Formatowanie liczb (np. 2 miejsca po przecinku):

pi = 3.14159265

print("Pi: {:.2f}".format(pi)) # Pi: 3.14

**📌 2. f-stringi — nowoczesne i bardzo wygodne (Python 3.6+)**

name = "Szymon"

messages = 5

print(f"Hello, {name}. You have {messages} new messages.")

➡️ Można używać wyrażeń w środku:

x = 10

print(f"x squared is {x\*\*2}") # x squared is 100

➡️ Formatowanie liczb:

pi = 3.14159265

print(f"Pi to 3 decimal places: {pi:.3f}")

➡️ Wyrównanie tekstu:

for x in range(1, 4):

print(f"Row {x:<5} | Value: {x\*10:>4}")

📌 Efekt:

Row 1 | Value: 10

Row 2 | Value: 20

Row 3 | Value: 30

* < – wyrównanie do lewej
* > – do prawej
* ^ – do środka

**📌 3. %-formatting (starsze, podobne do C)**

name = "Szymon"

print("Hello, %s!" % name)

print("Pi: %.2f" % 3.14159)

📛 Obecnie **zaleca się używanie f-stringów lub str.format()**, a % zostawić tylko do starszego kodu.

**💡 Formatowanie danych tabelarycznych**

data = {"apples": 10, "bananas": 5, "oranges": 12}

for fruit, count in data.items():

print(f"{fruit:<10} | {count:>3}")

🧾 Wynik:

apples | 10

bananas | 5

oranges | 12

**🔧 Zaawansowane: liczby binarne, szesnastkowe**

number = 42

print(f"Binarnie: {number:b}")

print(f"Szesnastkowo: {number:x}")

**🧠 Podsumowanie:**

| **Sposób** | **Czy polecany?** | **Od wersji Pythona** | **Przykład** |
| --- | --- | --- | --- |
| str.format() | ✅ Tak | 2.6+ | "Hello {}".format(name) |
| f-string | ✅ Tak | 3.6+ | f"Hello {name}" |
| %-formatting | ❌ Nie | starszy styl | "Hello %s" % name |

**✅ Do samodzielnego przećwiczenia:**

# 1. Wydrukuj swoje imię i wiek używając f-stringa

# 2. Wypisz tabelę 5 osób z imieniem i wynikiem (np. 'Adam', 92), z wyrównaniem

# 3. Pokaż liczbę Pi z dokładnością do 4 miejsc po przecinku

# 4. Zamień liczbę 255 na zapis binarny i heksadecymalny

**✅ Zadanie 1. Wydrukuj swoje imię i wiek używając f-stringa**

name = "Szymon"

age = 30

print(f"My name is {name} and I am {age} years old.")

**✅ Zadanie 2. Wypisz tabelę 5 osób z imieniem i wynikiem, z wyrównaniem**

results = [

("Adam", 92),

("Beata", 85),

("Cezary", 78),

("Dorota", 90),

("Ewa", 88)

]

print(f"{'Name':<10} | {'Score':>5}")

print("-" \* 20)

for name, score in results:

print(f"{name:<10} | {score:>5}")

📄 Wynik:

Name | Score

--------------------

Adam | 92

Beata | 85

Cezary | 78

Dorota | 90

Ewa | 88

**✅ Zadanie 3. Pokaż liczbę Pi z dokładnością do 4 miejsc po przecinku**

pi = 3.1415926535

print(f"Pi with 4 decimal places: {pi:.4f}")

**✅ Zadanie 4. Zamień liczbę 255 na zapis binarny i heksadecymalny**

number = 255

print(f"Binary: {number:b}")

print(f"Hexadecimal: {number:x}")

📄 Wynik:

Binary: 11111111

Hexadecimal: ff