

1 print 函数

Python 的 `print()` 函数是编程中最基础且高频使用的工具之一，它用于将数据输出到控制台或文件。以下从基础语法、核心功能到高级用法进行详细解析，并结合实际场景说明其灵活性。

1.1 基本语法与参数解析

`print()` 函数的语法为：

```
1 print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

- `*objects`：要输出的一个或多个对象（如字符串、数字、列表等），用逗号分隔。例如：`print("Hello", 123)`。
- `sep`：对象之间的分隔符，默认为空格。例如：`print("A", "B", sep="-")` 输出 `A-B`。
- `end`：输出结束时的字符，默认为换行符。例如：`print("Hello", end=" ")` 后接 `print("World")` 会输出 `Hello World`。
- `file`：指定输出目标，默认为标准输出（控制台）。可设置为文件对象，如 `print("日志", file=open("log.txt", "w"))`。
- `flush`：强制刷新缓冲区，默认 `False`。设为 `True` 可立即输出（如实时进度条）。

1.2 核心功能与使用场景

1. 多对象输出与格式控制

- 默认行为：多个参数用空格分隔，末尾换行。

```
1 print("Python", "3.12", "发布") # 输出：Python 3.12 发布
```

- 自定义分隔符：通过 `sep` 参数实现特殊格式。

```
1 print("2025", "04", "18", sep="-") # 输出：2025-04-18
```

- 自定义结束符：避免自动换行。

```
1 print("加载中", end="")
2 print("完成!") # 输出：加载中完成!
```

2. 格式化输出 • 旧式 % 格式化：

```
1 print("年龄：%d，姓名：%s" % (25, "张三")) # 输出：年龄：25，姓名：张三
```

- `str.format()` 方法：

```
1 print("坐标：({}, {})".format(120.5, 30.2)) # 输出：坐标：(120.5, 30.2)
```

- f-string（推荐）：Python 3.6+ 支持，简洁高效。

```
1 name = "李四"
2 score = 95.5
3 print(f"{name}的成绩是{score}分") # 输出：李四的成绩是95.5分
```

3. 文件操作 将输出写入文件：

```
1 with open("output.txt", "w") as f:  
2     print("数据已保存", file=f) # 文件内容：数据已保存
```

1.3 高级用法与技巧

1. 动态刷新输出（实时显示）通过 `flush=True` 实现即时输出，适用于进度条或实时日志：

```
1 import time  
2 for i in range(5):  
3     print(f"进度：{i+1}/5", end="\r", flush=True) # 覆盖同一行显示  
4     time.sleep(1)
```

2. 控制颜色与样式 结合 ANSI 转义序列实现彩色输出（仅限支持 ANSI 的终端）：

```
1 print("\033[31m红色文字\033[0m 默认颜色") # 输出红色文字
```

3. 自定义对象的输出 通过重写 `__str__` 方法控制对象的打印格式：

```
1 class Person:  
2     def __init__(self, name):  
3         self.name = name  
4     def __str__(self):  
5         return f"用户：{self.name}"  
6 print(Person("王五")) # 输出：用户：王五
```

1.4 常见问题与注意事项

1. 换行问题

若需连续输出不换行，设置 `end=""`：

```
1 print("Hello", end="")  
2 print("World") # 输出：HelloWorld
```

2. 分隔符误用

若未用逗号分隔参数，会触发字符串拼接：

```
1 print("Python" "3.12") # 输出：Python3.12
```

3. 格式化类型错误

确保占位符与变量类型匹配（如 `%d` 对应整数，`%s` 对应字符串）。

1.5 总结

`print()` 函数虽简单，但通过灵活组合参数（如 `sep`、`end`、`file`）和格式化方法，能满足从基础调试到复杂日志记录的需求。掌握其高级用法（如实时刷新、颜色控制）可显著提升开发效率。建议结合具体场景选择最合适的输出策略。