第三章 列表、字典与格式化

3.1 格式化字符串 f-string

3.1.1 概述与基本格式

概述 格式化字符串 f-string 是以 f 前缀开头，并在字符串中使用{}包裹**变量和表达式**的一种方式。f-string 提供了灵活的格式控制，如对齐、填充字符、指定宽度、控制小数位数等。

基本格式 `f"str{var:control}"`

3.1.2 对齐与填充

左对齐 `print(f"|{name:-<10}|")` 数字是字符宽度，符号前加符号实现填充符号

右对齐 `print(f"|{name:->10}|")`

居中对齐 `print(f"|{name:-^10}|")`

3.1.3 数字格式化

3.1.3.1 整数

假设 `num =123`

统一宽度 `print(f"{num:05d}")` `"00123"` 总宽度固定为 5，**前导 0 补齐**

千位分隔符 `print(f'{1234567:,}')` `"1,234,567"` 添加千位分隔符

综合示例 `print(f"No.{i:02d}: {i * 1000:8,d}")` 编号前导 0，总宽度 8，千位分隔

3.1.3.2 小数

小数位数 `print(f"{pi:.2f}")` `3.14` 保留两位小数

总宽度 `print(f"{pi:08.2f}")` `00003.14` 总宽度 8，前导 0 补齐，小数点后两位

综合示例 `print(f"No.{i:04d}: {num:=^16.3f}")` 总宽度 16，居中对齐，小数点后三位

3.2 列表的高级用法

3.2.1 列表解析

用途 一种简洁的创建列表的方法，它通过一行代码生成一个新的列表。

语法格式 **[表达式 for 变量 in 可迭代对象]** 例如：`squares = [x**2 for x in range(5)]`

输出：`[0, 1, 4, 9, 16]`

嵌套循环 列表解析可以支持嵌套循环来生成元素组合，嵌套循环可以生成两个列表的所有组合。

`combinations = [(x,y) for x in range(3) for y in range(3)]`

输出：`[(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)]`

条件解析 在列表解析中添加条件，使其只生成满足条件的元素。

`even_squares = [x**2 for x in range(10) if x%2==0 and x>4]` 输出：`[0, 4, 16, 36, 64]`

`even_squares = [x**2 for x in range(10) if x%2==0 if x>4]` 更推荐连用 if 判断

条件判断 可以在列表推导中使用 if- else 语句，根据条件为每个元素生成不同的值

[表达式 1 if 条件 else 表达式 2 for 变量 in 可迭代对象]

`numbers = [1,2,3,4,5]` `result = ["even" if x%2==0 else "odd" for x in numbers if x>3]`

3.2.2 enumerate

用途 用于在**遍历列表时同时获取索引和元素**。它返回的是**一个包含索引和值的元组**。

实例

```
letters = ["a","b","c","d","e"]
for index, letter in enumerate(letters):
    print(f"{index}:{letter}")
0:a     1:b     2:c     3:d     4:e
```

或使用：`index_letters = [(index,letter) for index,letter in enumerate(letters)]`

3.2.3 zip

用途 用于将**多个可迭代对象**中的元素**打包成一个元组**，生成新的迭代器。

实例

```
nums = [1, 2, 3]
letters = ['a', 'b', 'c']
for num, letter in zip(nums, letters):
    print(num, letter)
```

3.3 字典的高级用法

字典解析 字典拥有与列表相似的字典解析方法。 `squares = {x: x**2 for x in range(5)}`

字典合并 `d1 = {'a': 1, 'b': 2}` `d2 = {'c': 3, 'd': 4}` `merged = d1 | d2`

默认值 `person = {"name": "Alice"}` `person.setdefault("age", 25)`

enumerate 使用 `enumerate()`枚举方法为键添加索引

`keys = {"a","b","c"}` `values = [1,2,3]`

`indexed_dict = {i: value for i, value in enumerate(values)}` 输出：`{0: 1, 1: 2, 2: 3}`

zip 使用 `zip` 生成字典，将两个列表中的元素对应起来

`zipped_dict = dict(zip(keys,values))` 输出：`{'a': 1, 'b': 2, 'c': 3}`

实例 以台风路径数据为例，存储多个字段的数据，包括纬度、经度、海平面气压（slp）和风速（wind）

```
# 定义台风路径数据
typhoon_data = {
    "lat": [15.2, 16.0, 16.8, 17.5],
    "lon": [120.5, 121.0, 121.5, 122.0],
    "slp": [1005, 1003, 1002, 1000],
    "wind": [35, 40, 45, 50]
}
# 遍历字典并输出每个字段
for key, values in typhoon_data.items():
    print(f"{key}: {values}")
# 计算最大风速和对应的经纬度
max_wind = max(typhoon_data["wind"])
index = typhoon_data["wind"].index(max_wind)
max_lat = typhoon_data["lat"][index]
max_lon = typhoon_data["lon"][index]

print(f"最大风速: {max_wind} knots, 位置: ({max_lat}, {max_lon})")
最大风速: 50 knots, 位置: (17.5, 122.0)
```