SI100B 期中考试核心速查表 (Cheatsheet)

---

1. 基础与数据类型 (Basics & Data Types)

1.1 运算符易错点 (Operator Pitfalls)

- `//` **(整除)**: 结果向下取整。 `9 // 4` 结果是 `2`； `-9 // 4` 结果是 `-3`。
- `%` **(取模)**: 结果的符号与**除数**相同。`10 % 3` 结果是 `1`； `-10 % 3` 结果是 `2`。
- `**` **(幂运算)**: `2 ** 3` 结果是 `8`。

1.2 字符串 `str` 技巧与方法

- **不可变性 (Immutability)**: 字符串一旦创建就不能修改。`s[0] = 'b'` 会报错。修改只能通过创建新字符串，如 `s = 'b' + s[1:]`。
- **高级切片 (Slicing)**: `s[start:stop:step]`
  - `s[::-1]`: **翻转字符串**，非常常用。
  - `s[7:1:-2]`: 从索引7到1，步长为-2（反向）。
  - 如果 `start >= stop` 且 `step` 为正，结果是空字符串 `''`。
- **f-string 格式化**:
  - `f'{val:.2f}'`: 保留两位小数。
  - `f'{val:.2%}'`: 转换为百分比并保留两位小数。
  - `f'{num:,}'`: 添加千位分隔符。
  - `f'{name:^10}'`: 居中对齐，总宽度10。
- **常用方法**:
  - `s.split(sep)`: 将字符串按 `sep` 分割成**列表**。默认按所有空白（空格、换行、制表符）分割。
  - `'sep'.join(list)`: 将**字符串列表**用 `sep` 连接成一个新字符串。**注意：列表元素必须是字符串。**

---

2. 列表 `list` 与元组 `tuple`

2.1 列表 `list` (可变)

- **修改列表的陷阱**:
  - **迭代时修改**: **绝对不要**在 `for e in L:` 循环中直接 `L.remove(e)` 或 `L.append(e)`。这会导致迭代器混乱，跳过元素或无限循环。
  - **正确做法**: 迭代列表的**副本** `for e in L[:]`，然后修改原列表 `L`。
- **别名 (Aliasing) vs 克隆 (Cloning)**:
  - `L2 = L1`: **别名**。`L1` 和 `L2` 指向**同一个对象**。修改 `L2` 会影响 `L1`。
  - `L2 = L1[:]`: **克隆 (浅拷贝)**。`L1` 和 `L2` 是**两个独立的对象**。修改 `L2` 不会影响 `L1` (除非有嵌套的可变对象)。
- **常用方法 (返回值是重点)**:
  - **添加**:
    - `L.append(elem)`: 在末尾添加**单个元素**，`elem` 可以是任何类型。**返回** `None`。
    - `L.extend(iterable)`: 将一个**可迭代对象的所有元素**逐个添加到末尾。**返回** `None`。
  - **删除**:

        - `L.pop(index)`: 删除并**返回**指定索引的元素（默认最后一个）。
        - `L.remove(value)`: 删除第一个匹配的 `value`。**返回** `None`。如果值不存在，抛出 `ValueError`。
        - `del L[index]`: 按索引删除，无返回值。
    - **排序**:
        - `L.sort()`: **原地**排序。**返回** `None`。
        - `sorted(L)`: **返回一个排好序的新列表**，原列表不变。

2.2 元组 `tuple` (不可变)

- **定义**: 关键是**逗号**，不是括号。
    - `t = (1,)` 或 `t = 1,`: 单元素元组。
    - `t = ()`: 空元组。
- **用途**:
    - **函数返回多个值**: `return x, y` 实际返回的是一个元组 `(x, y)`。
    - **安全的数据容器**: 不可变性保证了数据不会被意外修改。
    - **字典的键**: 因为不可变，元组可以作为字典的键。

---

3. 字典 `dict` (可变)

- **键 (Key) 的要求**: 必须是**不可变**类型 (`int`, `str`, `tuple` 等)。列表和字典不能作为键。
- **访问**:
    - `d[key]`: 如果 `key` 不存在，抛出 `KeyError`。
    - `d.get(key, default)`: 如果 `key` 不存在，返回 `default` 值 (默认为 `None`)，**不会报错**。
- **迭代**:
    - `for k in d:` 或 `for k in d.keys()`: 遍历键。
    - `for v in d.values()`: 遍历值。
    - `for k, v in d.items()`: **同时遍历键和值**，最常用。
- **词频统计模板**:

```
freq = {}
for item in data_list:
    if item in freq:
        freq[item] += 1
    else:
        freq[item] = 1
# 或者使用 .get()
# freq[item] = freq.get(item, 0) + 1
```

---

4. 函数 (Functions)

- `None` **返回值**: 如果函数没有 `return` 语句，它会隐式地 `return None`。
- **作用域 (LEGB Rule)**: 变量查找顺序：**L**ocal -> **E**nclosing -> **G**lobal -> **B**uilt-in。
- **可变默认参数陷阱**: **永远不要**使用列表或字典作为函数的默认参数，如 `def func(a, L=[])`。这个列表只会被创建一次，并在所有调用之间共享。

- **正确做法**:

```
def func(a, L=None):
    if L is None:
        L = []
    # ...
```

- **Lambda 匿名函数**: `lambda arguments: expression`
  - 只能包含一个表达式，其结果就是返回值。
  - 适用于作为高阶函数（如 `sorted`, `map`）的参数，实现临时、简单的逻辑。

---

5. 异常与断言 (Exceptions & Assertions)

5.1 异常处理

- `try...except`: 用于处理**可能发生**的运行时错误（用户输入、文件IO等），避免程序崩溃。
- **常见异常类型**:
  - `ValueError`: 类型正确，但值不合适。如 `int('abc')`。
  - `TypeError`: 操作用在了错误的类型上。如 `'2' + 2`。
  - `IndexError`: 序列索引越界。
  - `KeyError`: 字典中没有找到指定的键。
  - `ZeroDivisionError`: 除以零。
- `try-except-else-finally` **结构**:
  - `try`: 尝试执行的代码。
  - `except`: 如果 `try` 块中发生异常，执行此块。
  - `else`: 如果 `try` 块中**没有**发生异常，执行此块。
  - `finally`: **无论如何**都会执行的块，通常用于资源清理（如关闭文件）。
- `raise`: 主动抛出一个异常。`raise ValueError("错误信息")`。

5.2 断言 `assert`

- `assert condition, "message"`: 用于**调试**，检查程序内部**绝不应该发生**的错误。
- 如果 `condition` 为 `False`，抛出 `AssertionError`。
- **与异常的区别**: 断言是给程序员看的，用于保证代码逻辑的正确性。异常处理是给用户和外部环境的，用于处理不可预知的运行时问题。

---

6. 递归与面向对象 (Recursion & OOP)

6.1 递归 (Recursion)

- **两大要素**:
  1. **基本情况 (Base Case)**: 递归的终止条件，可以直接求解。
  2. **递归步骤 (Recursive Step)**: 将问题分解为更小的、同性质的子问题，并调用自身来解决。
- **栈溢出 (Stack Overflow)**: 如果递归没有正确的终止条件，或递归深度过大，会导致栈溢出错误。

6.2 面向对象 (OOP)

- `class`: 对象的蓝图。
- `self`: 代表实例本身，在方法中自动作为第一个参数传入。
- `__init__(self, ...)`: 构造方法，在创建实例时自动调用，用于初始化实例属性 (`self.x = ...`)。
- **实例属性 vs 类属性**:
  - **实例属性**: `self.name`，每个实例独有。
  - **类属性**: 直接在 `class` 下定义，所有实例共享。
- **特殊方法 (Dunder Methods)**:
  - `__str__(self)`: 定义 `print(obj)` 的输出，面向用户。
  - `__repr__(self)`: 定义直接输入 `obj` 时的输出，面向开发者。
  - `__eq__(self, other)`: 定义 `==` 的行为。
- **继承 (Inheritance)**: 子类可以继承父类的所有方法和属性，并可以重写 (override) 它们或添加新功能。`super().__init__(...)` 用于调用父类的构造方法。