SI100B Python Cheet Sheet

1. Python 基础语法

运算符

- 算术:

`+, -, *, **, / (浮点除), // (整除), % (取余)`

- 比较:

`==, !=, >, <, >=, <=`

- 逻辑:

`and, or, not`

数据类型

- 不可变类型 (Immutable):

`int, float, bool, str, tuple, NoneType`

- 可变类型 (Mutable):

`list, dict`

类型转换

```
float(3)        # 3.0
int(3.9)        # 3 (截断)
round(3.9)      # 4 (四舍五入)
str(123)        # "123"
```

2. 字符串操作

基本操作

```
s = "Hello"
len(s)          # 5
s + " World"    # "Hello World"
s * 3           # "HelloHelloHello"
```

切片和索引

```
s = "ShanghaiTech"
s[0]            # 'S' (索引从0开始)
s[0:5]          # "Shang" (切片: 左开右闭)
s[::2]          # "SagaTc" (步长2)
s[::-1]         # 反转字符串
```

字符串方法

```
s.upper()       # 大写
s.lower()       # 小写
list(s)         # 逐字母分割成列表
s.split(" ")    # 按指定字符分割成列表
" ".join(list) # 列表转字符串
```

f-string

```
f"Hello {name}"          # 基本插值
f"{value:.2f}"           # 保留两位小数

# e.g.占位2个空格、制表对齐、输出不换行
print(f"{j}x{i}={i*j:2}\t", end="")
```

3. 控制结构

条件语句

```
if condition1:
    # 代码块
elif condition2:
    # 代码块
else:
    # 代码块
```

循环结构

```
# while循环
while condition:

# for循环
for i in range(0, 10, 2): # 0,2,4,6,8

# 遍历元素
for char in "hello":
    print(char)
```

循环控制

```
break     # 立即退出循环
continue  # 跳过当前迭代
```

4. 列表(List)

创建和基本操作

```
L = [2, 'a', 4, [1,2]]   # 混合类型
len(L)                   # 4
L[1:3]                   # ['a', 4] (切片)
L[3:1:-1]                # [[1, 2], 4]
del L[index]             # 删除元素
for value, index in enumerate(L): # 同时遍历
Lnew = [expressions for elem in L if con]
```

列表方法 (会改变原列表)

```
L.append(element)      # 添加元素
L.extend([list])       # 扩展列表
L.pop(index)           # 移除并返回元素 (默认最后)
L.remove(element)      # 移除第一个匹配元素
L.sort()               # 排序
L.reverse()            # 反转
L.clear()              # 清空列表
L.insert(index, item) # 插入元素
```

别名 vs 克隆

```
import copy
L1 = [1,2,3]
L2 = L1                  # 别名 (指向同一对象)
L3 = L1[:]               # 克隆 (新对象)
L4 = copy.copy(L1)       # 浅拷贝 (等同于L1[:])
L5 = copy.deepcopy(L1) # 深拷贝 (拷贝嵌套结构)
```

注意

```
# e.g.遍历时修改列表
for elem in L:
    if condition:
        L.remove(elem) # 可能导致错误!
```

```
# e.g. 列表排序 按多个值排序
# 升序
L.sort(key = lambda i: (value, index))
# 降序（但index小的在前）
L.sort(key = lambda i: (value, index))
      # 先按index升序
L.sort(key = lambda i: (value), reverse = True) # 再按value降序

# 或者取负数
L.sort(key = lambda i: (-value, index))
```

5. 函数

函数定义

```
def function_name(parameters):
    """
    docstring
    """
    return value  # None as default
```

函数特性

- 函数是一等对象：可赋值、作参数、作返回值
- 每个函数调用创建新的临时环境
- 参数传递：对于可变对象是引用传递
  - `*args  #用于传入多个参数 e.g. a tuple`

6. 字典(Dictionary)

基本特性

```
# 键值对映射，无序集合
d = {"key1": "value1", "key2": "value2"}
d = dict(key1 = "value1", key2 = "value2")
len(d)         # 键值对数量
"key1" in d    # 检查键是否存在（返回True/False）
```

核心操作

```
# 访问和修改
d["key"] = value       # 添加/修改键值对
value = d["key"]       # 获取值（键不存在会报错）
value = d.get("key", default)  # 安全获取（可设默认值）

# 删除操作
del d["key"]           # 删除键值对
value = d.pop("key")   # 删除并返回值
d.clear()              # 清空字典
```

重要方法

```
d.keys()      # 所有键的视图
d.values()    # 所有值的视图
d.items()     # 所有键值对的视图

# 遍历字典
for key in d:                    # 遍历键
for value in d.values():         # 遍历值
for key, value in d.items():     # 同时遍历键值对
```

注意事项

- 字典键必须是不可变对象
- 每个键只能出现一次，重复赋值会覆盖旧值
- 字典无序（Python 3.7+ 为插入顺序）
- 字典在内存中占用空间比列表大，但查找速度更快（接近 O(1)）

```
# 错误示例
d = {[1,2]: "value"}    # 报错！列表不可哈希

# 正确示例
d = {(1,2): "value"}    # 元组可作为键
```

7. 异常处理

基本结构

```
try:
    # 可能出错的代码
except ExceptionType:
    # 异常处理
else:
    # 无异常时执行
finally:
    # 总是执行（清理代码）
```

断言

```
assert condition, "错误信息"
```

报错

```
raise ExceptionType("错误信息")
```

8. 重要概念和技巧

浮点数精度问题

```
# 不要用==比较浮点数
abs(a - b) < 1e-9  # 正确比较方式
```

算法思想

- 穷举枚举：猜测-验证模式
- 二分查找：每次取其半
- 递归思维：将问题分解为相同子问题

调试技巧

- 写一点测一点，不要一次性写完整程序
- 使用 Python Debugger 逐步调试
- 善用 print 语句输出中间结果

9. 常用内置函数

```
len(obj)       # 长度
type(obj)      # 类型
max/min()      # 最大/最小值
sum()          # 求和
sorted()       # 返回新排序列表（不改变原列表）
map(function, iterable) # 将function作用到iterable的每个元素，返回一个map对象迭代器
```