C++与Python的对比 与 Python基本语法

1. 变量

区别： Python 不需要声明类型，它会自动推导，相当于前面加了C++的 `auto` 一样，并且也不需要分号

C++	Python
`int a = 10;`	`a = 10`
`float b = 3.14;`	`b = 3.14`
`string s = "hello";`	`s = "hello"`
`bool f = true;`	`f = True` (首字母需要大写与C++的true不一样)

2. 输入输出

输出：

C++	Python
`cout << "Hello";`	`print("Hello")`
`cout << "Num: " << a;`	`print("Num", a)`

输入：

Python 的输入默认是字符串类型，如果要进行计算，需要手动强制类型转化

C++	Python
`string s; cin >> s;`	`s = input()`
`int n; cin >> n;`	`n = int(input())` (必须强制类型转化为 `int` 才能用于数学计算)

3. 判断语句

区别：

1. 没有括号 `()` 包围条件
2. 行尾必须加冒号 `:`
3. **没有大括号 `{}`，全靠缩进（通常为4个空格，可以使用Tab按键来快速缩进）来表示某个代码块**
4. `else if` 变成了 `elif`

C++:

```cpp
if (x > 0) {
    cout << "正数";
} else if (x == 0) {
    cout << "零";
} else {
    cout << "负数";
}
```

Python:

```python
if x > 0:           # 没括号，有冒号
    print("正数")    # 必须缩进，表示是属于 x > 0 的这个代码块
elif x == 0:        # else if 变成 elif，并且回退缩进，从 x > 0 的代码块中退出
    print("零")
else:
    print("负数")
```

Python的缩进

你可以把缩进看作是**“文件目录的层级”**

- 缩进越深，说明它属于越内部的逻辑
- 一旦缩进退回来，说明那个逻辑块结束了

  C++ (用花括号 `{}` 决定范围)
  C++ 根本不在乎你有没有缩进，只要有 `{}` 就能跑。哪怕你写成一行也是对的（虽然很丑）

  ```cpp
  // C++ 即使写得乱七八糟，机器也能读懂，因为有 {}
  void fun() {
  if(isReady){
  do();
  }
  }
  ```

  Python (用缩进决定范围)
  Python 没有 `{}`，**必须**靠缩进排版。如果你乱写，程序直接报错，或者逻辑完全变了

  ```python
  # Python 必须写得整整齐齐
  def fun(): # 函数下也必须缩进，代表函数的代码块范围
      if is_ready:       # 冒号代表下面要开始缩进来确定它自己的代码块范围了
          do()      # 这行缩进了，说明它属于 if
      print("Done")      # 这行退回去一个缩进了，说明它不属于 if，但属于 fun 函数内部

  print("Over") # 没有缩进，代表它不属于任何逻辑代码块的内部
  ```

---

4. 循环

For 循环

区别： C++ 的 `for(int i=0; i<n; i++)` 在 Python 中用 `range(n)` 实现

C++:

```
// 循环 5 次：0, 1, 2, 3, 4
for (int i = 0; i < 5; i++) {
    cout << i;
}
```

Python:

```
# range(5) 就代表 [0, 1, 2, 3, 4] 左闭右开的区间
for i in range(5):
    print(i)
```

While 循环

区别： 同样是去掉括号，加上冒号。注意 Python 没有 `i++`，必须写 `i += 1`

C++:

```
while (i < 5) {
    cout << i;
    i++;  // 或者 ++i
}
```

Python:

```
while i < 5:
    print(i)
    i += 1  # Python 没有 ++ 运算符
```

---

5. 函数

核心区别：

1. **关键字**：C++ 用 `返回值类型` 开头，Python 统一用 `def` 开头
2. **参数**：Python **不需要** 写参数类型
3. **结构**：同样是去掉大括号 `{}`，行尾加冒号 `:`，内容靠 **缩进**

写法对比

1：带返回值的函数 (比如求和)

C++:

```cpp
// 必须声明返回类型(int)和参数类型(int)
int add(int a, int b) {
    return a + b;
}

// 调用
int result = add(5, 10);
```

Python:

```python
# 统一用 def，参数没类型，结尾加冒号
def add(a, b):
    return a + b  # 缩进

# 调用
result = add(5, 10)
```

2：无返回值的函数 (Void 类型)

C++:

```cpp
// 使用 void
void say_hello(string name) {
    cout << "Hello, " << name << endl;
}
```

Python:

```python
# 还是用 def，如果没有在内部写 return，就等于没有返回值
def say_hello(name):
    print(Hello, name)
```