图解Python基础语法

原创 王海华 模型视角 2025年08月25日 11:30 上海

Python作为一门简洁易学的编程语言，以其优雅的语法和强大的功能深受程序员喜爱。无论是初学者入门编程，还是资深开发者快速构建原型，Python都是不二之选。

我在最近教学中也涉及这部分内容，顺便进行了Python基础语法的梳理。

本文将结合图解系统地介绍Python的基础语法，帮助读者建立扎实的编程基础。

1. Python简介与特点

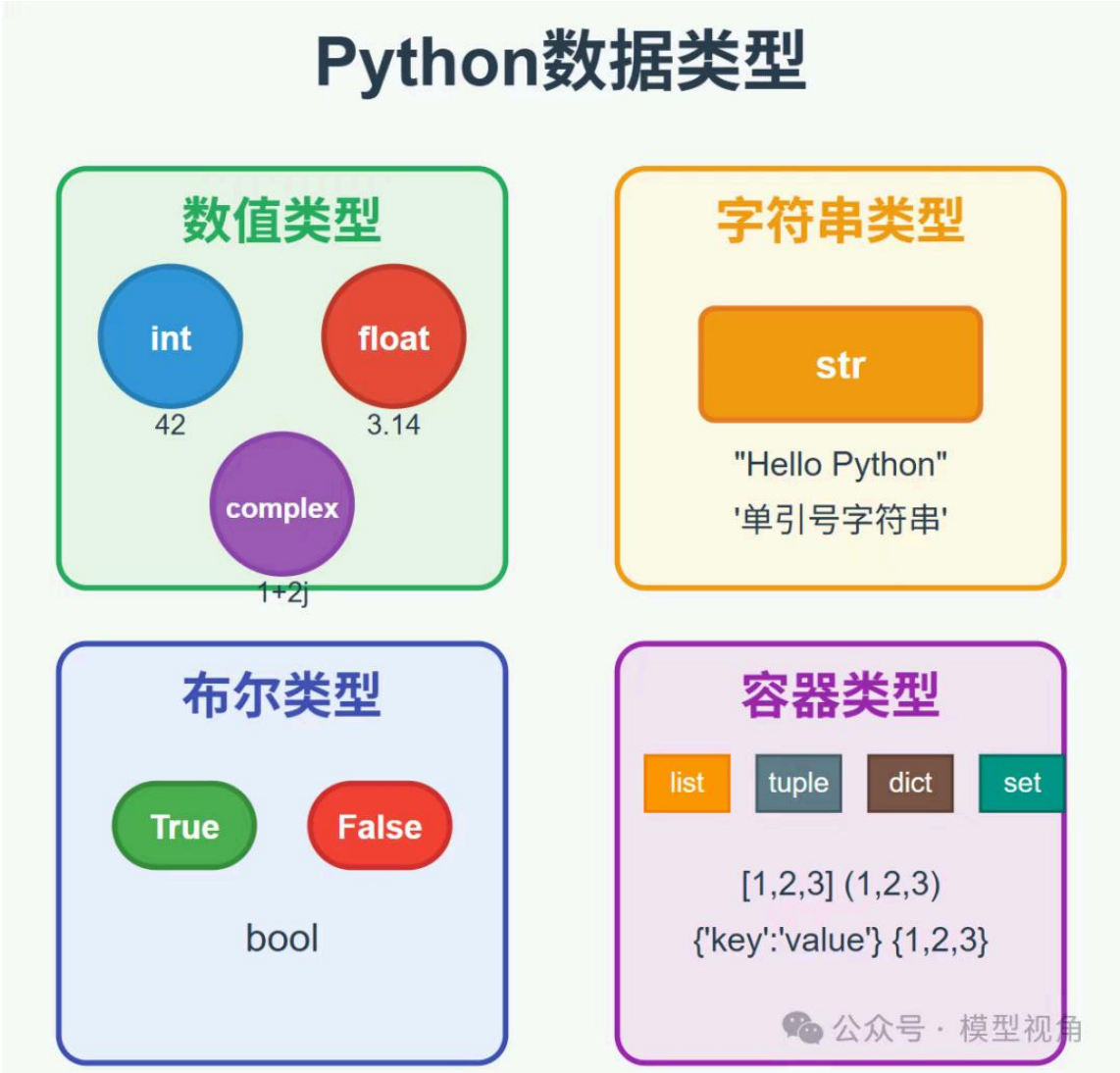
Python由吉多·范罗苏姆(Guido van Rossum)于1989年发明，是一种**解释型、面向对象、动态数据类型的高级程序设计语言**。Python的设计哲学强调代码的可读性，其语法清晰明了，能够用更少的代码表达相同的意思。

Python的核心特点包括：简单易学、跨平台兼容、丰富的标准库、活跃的社区支持以及广泛的应用领域。从Web开发到数据科学，从人工智能到自动化脚本，Python都能发挥重要作用。

2. 变量与数据类型

在Python中，变量是用来存储数据的容器。与其他编程语言不同，**Python不需要显式声明变量类型**，解释器会根据赋值自动推断类型。这种特性被称为动态类型。

变量命名遵循特定规则：**必须以字母或下划线开头，不能以数字开头**；只能包含字母、数字和下划线；区分大小写；不能使用Python关键字。



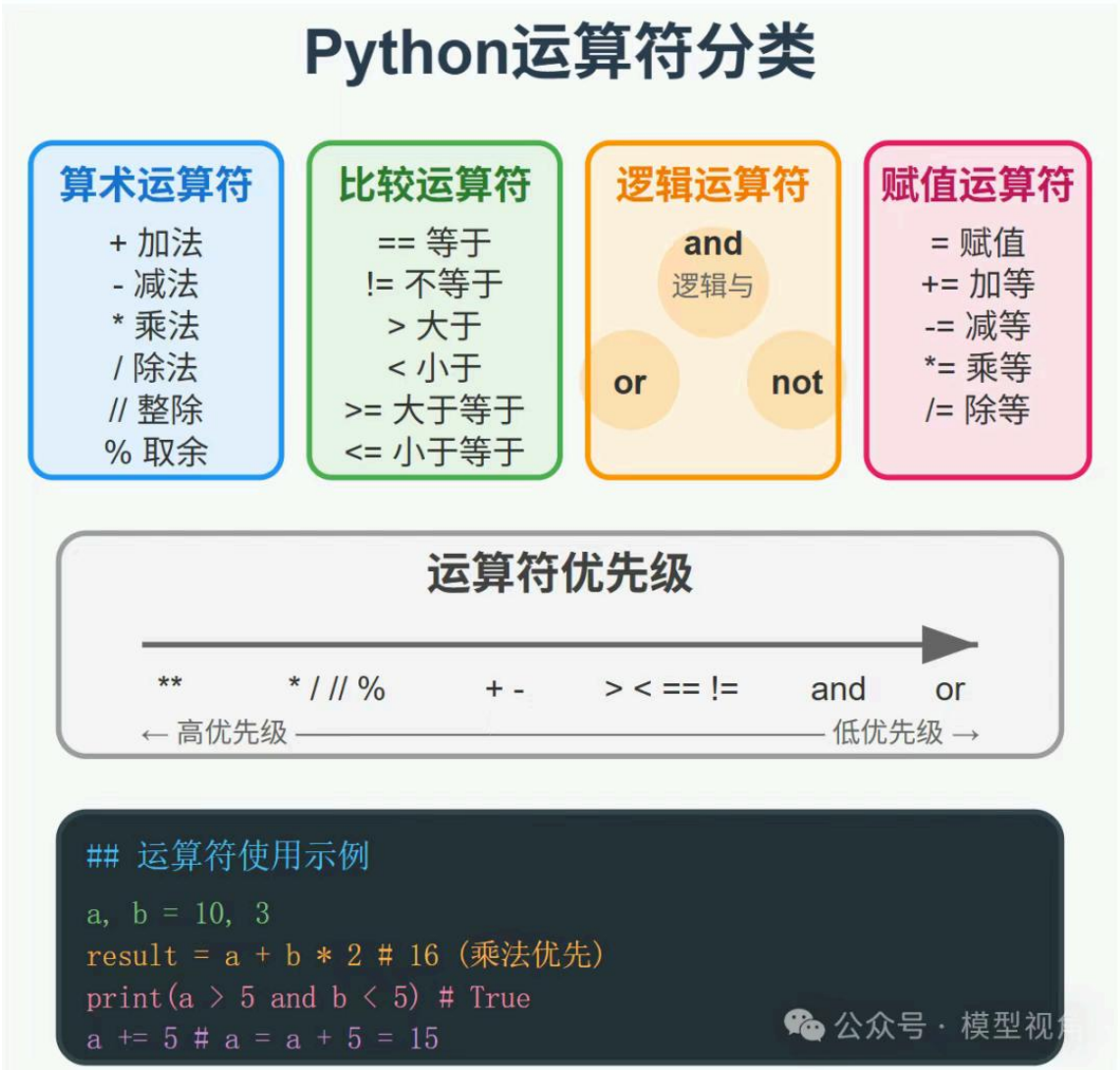
Python的**基本数据类型包括数值类型、字符串类型和布尔类型**。数值类型又分为整数(int)、浮点数(float)和复数(complex)。字符串(str)用于存储文本数据，可以使用单引号、双引号或三引号定义。布尔类型(bool)只有True和False两个值。

```
# 变量赋值示例
name = "Python"        # 字符串
age = 30               # 整数
height = 175.5         # 浮点数
is_student = True      # 布尔值
```

3. 运算符详解

Python提供了丰富的运算符来执行各种操作。运算符可以分为算术运算符、比较运算符、逻辑运算符、赋值运算符和身份运算符等几大类。

算术运算符包括加(+)、减(-)、乘(*)、除(/)、整除(//)、取余(%)和幂(**)运算。比较运算符用于比较两个值，包括等于(==)、不等于(!=)、大于(>)、小于(<)、大于等于(>=)和小于等于(<=)。



逻辑运算符包括逻辑与(and)、逻辑或(or)和逻辑非(not)，用于组合布尔表达式。赋值运算符不仅包括基本的等号赋值(=)，还有复合赋值运算符如+=、-=、*=等。

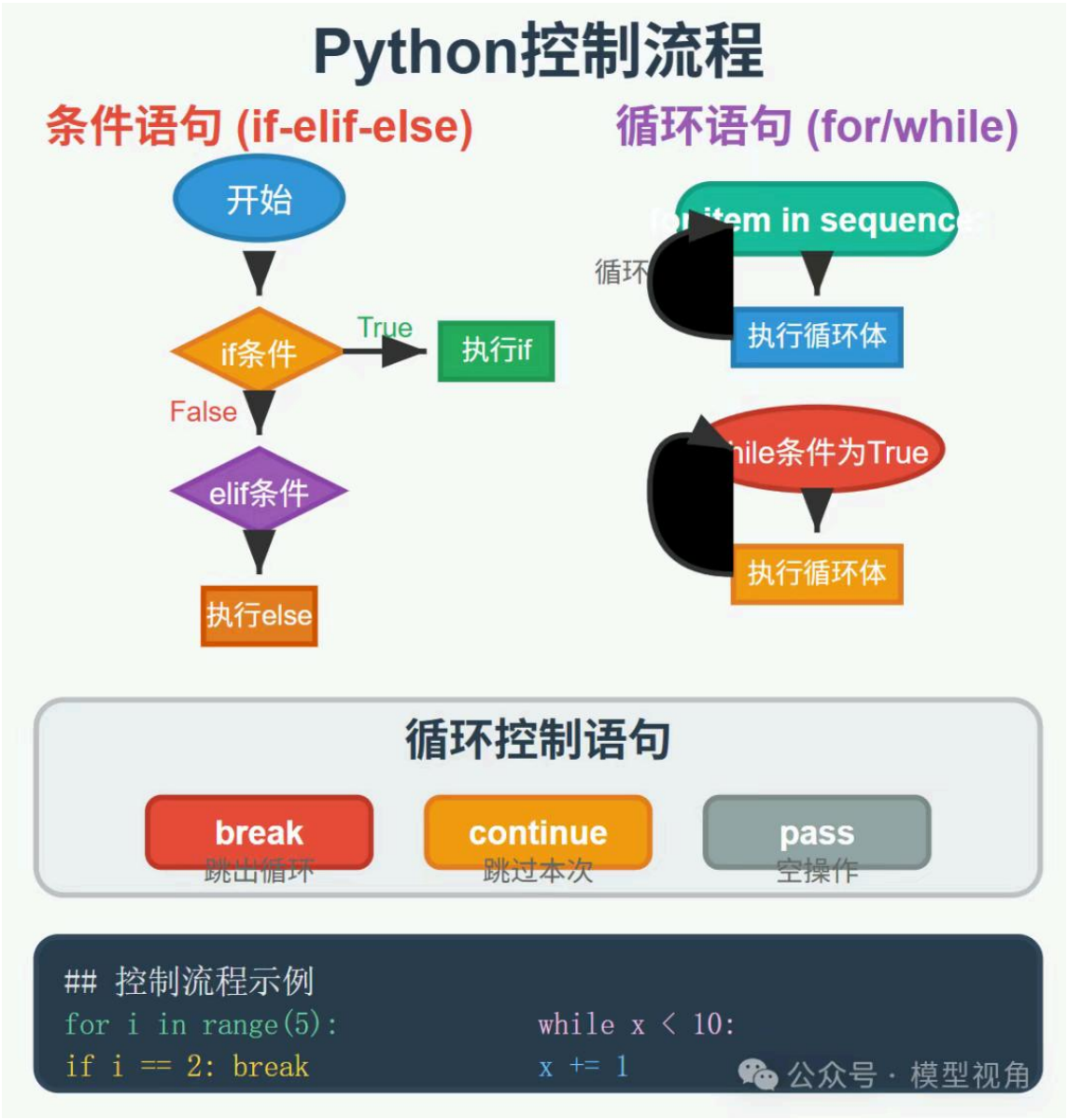
```
# 运算符使用示例
a = 10
b = 3

print(a + b)    # 13 (加法)
print(a // b)   # 3 (整除)
print(a ** b)   # 1000 (幂运算)
print(a > b)    # True (比较)
print(a > 5 and b < 5)  # True (逻辑运算)
```

4. 控制流程结构

程序的执行流程控制是编程的核心概念。Python提供了条件语句和循环语句来控制程序的执行路径。

条件语句使用if、elif和else关键字。if语句的基本语法是"if 条件:"，当条件为真时执行缩进的代码块。elif用于检查多个条件，else用于处理所有条件都不满足的情况。Python使用缩进来表示代码块，这是其语法的独特之处。



循环语句包括for循环和while循环。for循环通常用于遍历序列(如列表、字符串)，while循环在条件为真时重复执行代码块。循环中可以使用break语句提前退出循环，使用continue语句跳过当前迭代。

```
# 条件语句示例
score = 85
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
else:
    grade = "C"

# 循环语句示例
```

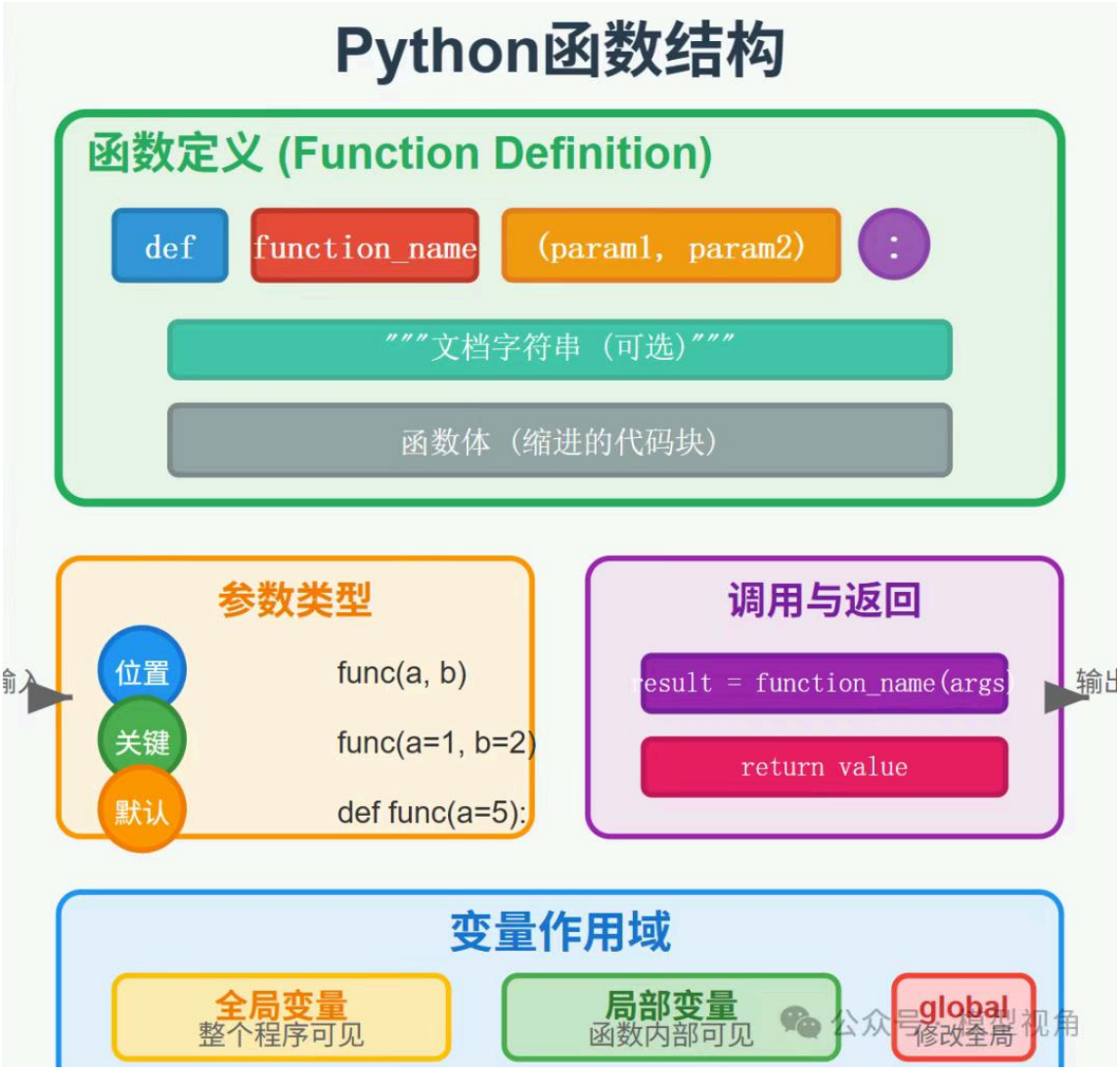
```
for i in range(5):
    print(i)

count = 0
while count < 3:
    print(f"Count: {count}")
    count += 1
```

5. 函数定义与调用

函数是组织代码的基本单元，它将相关的语句组织在一起执行特定任务。Python使用def关键字定义函数，函数可以接受参数并返回值。

函数定义的基本语法是"def 函数名(参数):"，函数体使用缩进表示。**函数可以有位置参数、关键字参数、默认参数和可变参数**。位置参数按顺序传递，关键字参数通过参数名传递，默认参数在调用时可以省略。



函数的作用域概念很重要。在函数内部定义的变量是局部变量，只在函数内部可见。全局变量在整个程序中都可以访问，如果需要在函数内修改全局变量，需要使用global关键字。

```python
# 函数定义示例
def greet(name, greeting="Hello"):
    """问候函数"""
    return f"{greeting}, {name}!"

def calculate_area(length, width):
    """计算矩形面积"""
    return length * width

# 函数调用
message = greet("Alice")
area = calculate_area(5, 3)
```

6. 数据结构概览

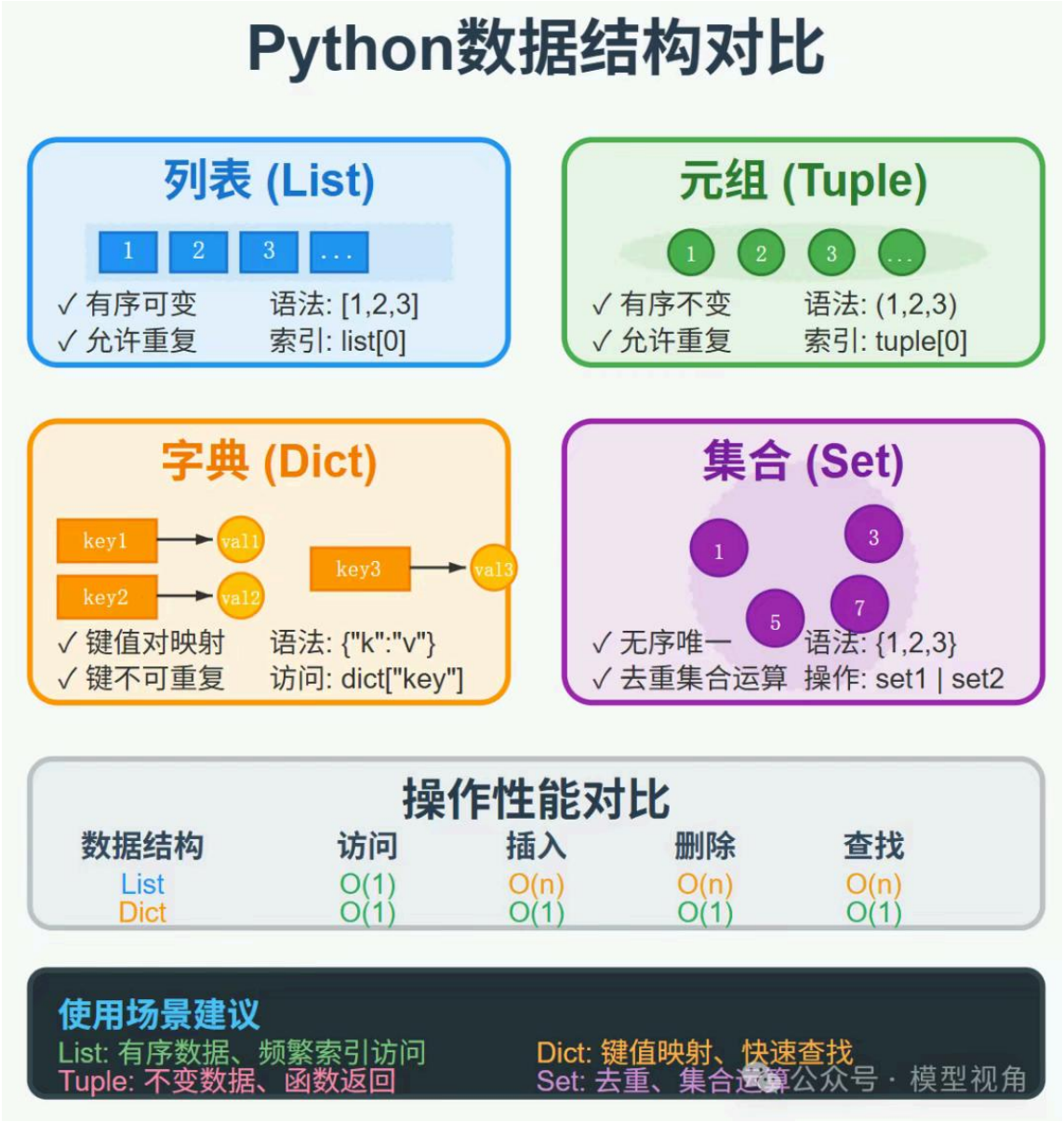
Python提供了多种内置数据结构，每种都有其特定的用途和特点。**主要的数据结构包括列表、元组、字典和集合**。

列表(list)是有序的可变序列，用方括号[]定义，可以存储不同类型的数据。列表支持索引访问、切片操作、添加删除元素等操作。列表是Python中最常用的数据结构之一。

元组(tuple)是有序的不可变序列，用圆括号()定义。虽然元组不能修改，但可以用于函数返回多个值、作为字典的键等场景。

字典(dict)是无序的键值对集合，用花括号{}定义。字典通过键来访问值，键必须是不可变类型。字典在查找、插入和删除操作上都有很好的性能。

集合(set)是无序的不重复元素集合，用花括号{}或set()函数定义。集合支持数学中的交集、并集、差集等操作，常用于去重和集合运算。




```
# 数据结构示例
my_list = [1, 2, 3, "hello"]       # 列表
my_tuple = (1, 2, 3)               # 元组
my_dict = {"name": "John", "age": 25}  # 字典
my_set = {1, 2, 3, 3}              # 集合(自动去重)

# 常用操作
my_list.append(4)                  # 列表添加元素
print(my_dict["name"])             # 字典访问值
print(len(my_set))                 # 集合长度
```

7. 异常处理机制

程序执行过程中可能遇到各种错误，Python提供了完善的异常处理机制来优雅地处理这些错误。**异常处理使用try、except、else和finally语句**。

try块包含可能引发异常的代码，except块处理特定的异常类型，else块在没有异常时执行，finally块无论是否有异常都会执行。这种机制让程序能够从错误中恢复，提供更好的用户体验。

常见的异常类型包括ValueError、TypeError、IndexError、KeyError等。程序员也可以使用raise语句主动抛出异常，或定义自定义异常类。

```
# 异常处理示例
try:
    num = int(input("请输入一个数字: "))
    result = 10 / num
    print(f"结果: {result}")
except ValueError:
    print("输入的不是有效数字")
except ZeroDivisionError:
    print("除数不能为零")
finally:
    print("程序执行完毕")
```

8. 模块与包的使用

Python的强大之处在于其丰富的标准库和第三方模块。模块是包含Python代码的文件，包是包含多个模块的目录。通过import语句可以导入模块中的功能。

标准库提供了大量实用模块，如os(操作系统接口)、sys(系统参数)、math(数学函数)、random(随机数生成)、datetime(日期时间处理)等。这些模块大大扩展了Python的功能。

第三方模块通过pip工具安装，PyPI(Python Package Index)是Python包的官方仓库。流行的第三方库包括requests(HTTP库)、numpy(数值计算)、pandas(数据分析)、matplotlib(数据可视化)等。

```
# 模块导入示例
import math
from datetime import datetime
import random as rand

# 使用标准库功能
print(math.pi)                      # 圆周率
current_time = datetime.now()       # 当前时间
random_num = rand.randint(1, 100)   # 随机数
```

Python的基础语法虽然简单，但蕴含着强大的编程思想。掌握这些基础知识是深入学习Python的关键。变量和数据类型为数据存储提供基础，运算符实现各种计算操作，控制流程结构化程序逻辑，函数实现代码复用，数据结构高效组织数据，异常处理保证程序健壮性。

学习编程是一个实践的过程，建议读者在掌握理论知识的同时，多动手编写代码，通过实际项目加深理解。Python社区活跃，资源丰富，是学习编程的理想选择。随着对Python基础语法的深入理解，读者将能够利用这门优雅的语言解决更多实际问题，开启程序设计的精彩旅程。

推荐几本与大语言模型原理和应用有关的书：

编程语言 · 目录

上一篇
实用教程：10个提高Python编程效率的技巧

下一篇
图解Python重要第三方库

写留言

朔游在海滨 广东 8月25日 3

python确实好用，在matlab和vs code中都很方便。

作者赞过

下午茶与老人... 安徽 8月28日 1

胶水语言你以为嘞

陈学文 广东 8月25日 1

请问怎么学？需要什么样的电脑？具体教程以及需要下载的软件在什么网站？

模型视角 作者 8月25日 2

上述问题可以在微信公众号后台私信，会有ai结合往期文章内容对你的问题进行回答

纯蓝 河南 8月29日 1

直接问ai，学了就能工作

yh7921 天津 8月29日 1

漂亮的图解

续写青春 浙江 8月29日 1

图片结合的真好

朔游在海滨 广东 8月29日

达成共识