

人工智能程序设计

python



```
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
    turtle.circle(40, 80/2)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```



人工智能程序设计

3.6 元组

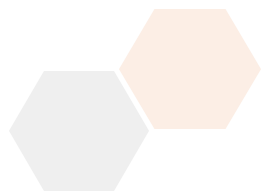
北京石油化工学院 人工智能研究院

刘 强

3.6 元组

元组 (Tuple) 是Python中另一种重要的序列类型

它与列表非常相似，但有一个关键区别：元组是不可变的 (immutable) 。



3.6.1 元组的创建与特点

创建元组

使用圆括号创建元组

```
coordinates = (3, 5)
```

```
colors = ("红", "绿", "蓝")
```

```
mixed = (1, "Python", 3.14, True)
```

```
print(coordinates) # 输出: (3, 5)
```

```
print(colors)      # 输出: ('红', '绿', '蓝')
```

```
print(mixed)       # 输出: (1, 'Python', 3.14, True)
```

3.6.1 元组的创建与特点

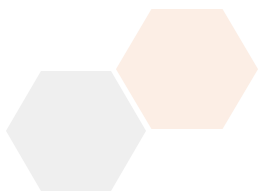
特殊情况

创建只有一个元素的元组时，必须在元素后面加逗号，否则Python会认为这只是普通的括号：

单元素元组（注意逗号）

```
single_tuple = (42,) # 注意：必须有逗号
```

```
print(single_tuple) # 输出：(42,)
```



3.6.1 元组的创建与特点

不使用括号创建元组

元组打包 (tuple packing)

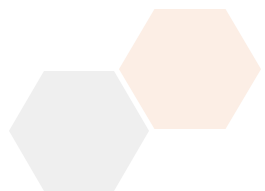
point = 10, 20

person = "张三", 25, "工程师"

print(point) # 输出: (10, 20)

print(person) # 输出: ('张三', 25, '工程师')

print(type(point)) # 输出: <class 'tuple'>



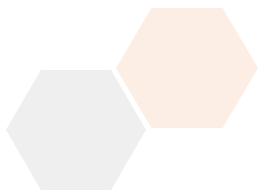
3.6.1 元组的创建与特点

使用tuple()函数

从列表创建元组

```
list_to_tuple = tuple([1, 2, 3, 4])
```

```
print(list_to_tuple) # 输出: (1, 2, 3, 4)
```



3.6.2 元组的操作

访问元组元素

元组的访问方式与列表完全相同：

```
fruits = ("苹果", "香蕉", "橙子", "葡萄")
```

```
## 索引访问
```

```
print(fruits[0]) # 输出: 苹果
```

```
print(fruits[-1]) # 输出: 葡萄
```

```
## 切片操作
```

```
print(fruits[1:3]) # 输出: ('香蕉', '橙子')
```

```
print(fruits[:2]) # 输出: ('苹果', '香蕉')
```

```
print(fruits[::2]) # 输出: ('苹果', '橙子')
```

3.6.2 元组的操作

元组解包 (Tuple Unpacking)

元组解包是将元组中的元素分别赋值给多个变量的操作：

基本解包

```
point = (3, 4)
```

```
x, y = point
```

```
print(f"x = {x}, y = {y}") # 输出: x = 3, y = 4
```

交换变量值

```
a, b = 10, 20
```

```
a, b = b, a
```

```
print(f"a = {a}, b = {b}") # 输出: a = 20, b = 10
```

3.6.2 元组的操作

遍历元组

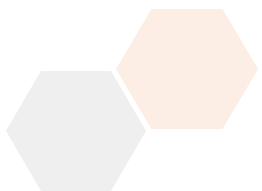
元组的遍历方式与列表完全相同：

```
colors = ("红", "绿", "蓝")
```

```
for color in colors:
```

```
    print(color)
```

```
## 输出：红 绿 蓝
```



3.6.2 元组的操作

元组的方法：元组只有两个内置方法count()和index()：

```
numbers = (1, 2, 3, 2, 4, 2, 5)
```

```
## count() - 统计元素出现次数
```

```
print(numbers.count(2)) # 输出： 3
```

```
## index() - 查找元素第一次出现的索引
```

```
print(numbers.index(4)) # 输出： 4
```

```
## 查找指定范围内的索引
```

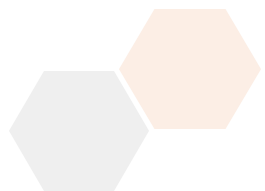
```
print(numbers.index(2, 2)) # 从索引2开始查找， 输出： 3
```

3.6.3 Ask AI：元组与列表的深入对比

列表和元组都是重要的序列类型，它们有相似之处也有重要区别。

当需要深入了解它们的性能差异、使用场景选择时，可以向AI助手询问：

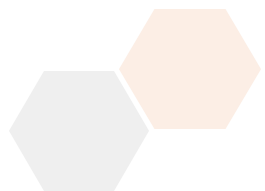
- "列表和元组在性能上有什么具体差异？"
- "什么情况下应该选择元组而不是列表？"
- "元组的不可变性在实际编程中有什么优势？"



3.6.4 元组的实际应用

元组在实际编程中有很多重要用途：

- 坐标和几何：表示2D/3D坐标点、矩形顶点等几何数据
- 函数返回多个值：函数可以返回元组来传递多个相关的返回值
- 配置和常量：存储数据库配置、RGB颜色值等不变的配置信息
- 作为字典的键：由于元组是immutable（不可变），可以用作字典的键来建立映射关系



实践练习

练习 3.6.1：员工信息管理

某公司有一批员工信息需要进行统计分析。每个员工的信息包含姓名、部门、工龄（年）和月薪资（元）四个字段。请创建一个元组列表来存储这些员工信息，其中每个员工的信息用一个元组表示，多个员工组成一个列表。

示例数据如下：

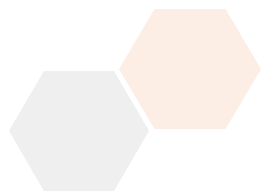
```
employees = [  
    ('张三', '技术部', 3, 8000),  
    ('李四', '销售部', 5, 9000),  
    ('王五', '技术部', 2, 7000),  
    ('赵六', '人事部', 4, 7500),  
    ('钱七', '销售部', 6, 9500),  
    ('孙八', '技术部', 3, 8200)  
]
```

实践练习

练习 3.6.1：员工信息管理

请完成以下任务：

1. 遍历所有员工信息，计算并输出所有员工的平均薪资（保留2位小数）
2. 找出工龄最长的员工，输出该员工的完整信息（姓名、部门、工龄、薪资）
3. 统计各部门的员工数量，以"部门名称: 员工人数"的格式输出每个部门的统计结果

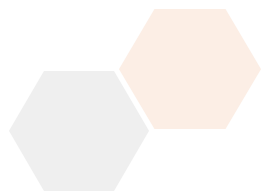


实践练习

练习 3.6.2: 几何计算

定义几个2D点的坐标（使用元组），编写函数：

1. 计算任意两点间的距离
2. 判断三个点是否能构成三角形
3. 计算三角形的周长和面积



实践练习

练习 3.6.3：数据解析

给定一个包含CSV格式数据的字符串，将其解析为元组列表：

```
data = "张三,25,工程师,50000\n李四,30,设计师,45000\n王五,28,产品经理,55000"
```

要求：

1. 将每行数据转换为元组
2. 计算平均工资
3. 找出工资最高的员工信息

