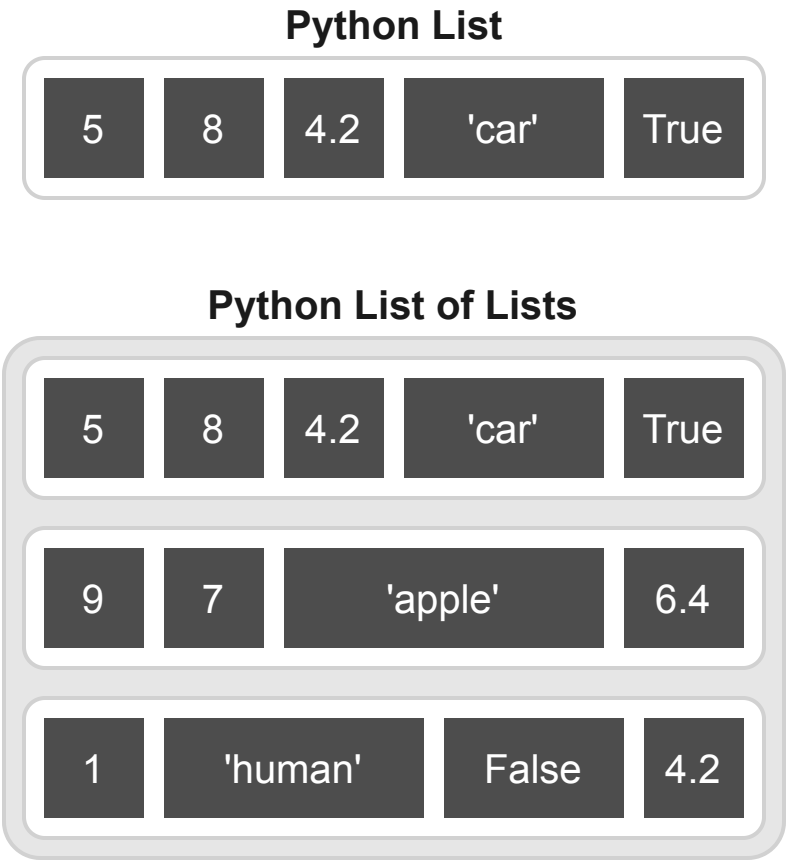
List 与 Tuple

Python 序列类型基础

课程信息

- **难度**: ⭐
- **时长**: 45 分钟
- **主题**: Python 序列类型 - List 和 Tuple

如何组织数据？


Python List
5
8
4.2
'car'
True
Python List of Lists
5
8
4.2
'car'
True
9
7
'apple'
6.4
1
'human'
False
4.2

序列的基本概念

- **序列** 是一种复合数据类型，由一组成员排序构成
- 示例：`[1,2,3,4]`，`'China'`
- 序列包含 List 和 Tuple 两种主要类型
- 还包含 str, bytes, bytearray

序列的索引特性

- 元素有编号（下标），如 `a[2]`
- 编号从 0 开始，`a[0]` 是首成员
- 可以用负数索引，如 `a[-1]`

练习：如果 `a=[1,2,3,4]`，那么：

- `a[1]` = ?
- `a[-1]` = ?

List 的特点

- 每个成员的类型可以不同
- 示例：`student = ['Tom', 42]`
- 通常列表中的所有成员类型相同（为了简单）

Tuple 和 List 的联系

联系：tuple 和 list 都是序列类型

```
edward1 = ['Edward Gumby', 42]  # List
edward2 = ('Edward Gumby', 42)  # Tuple
```

edward1 和 edward2 有何相同与不同之处？

Tuple 和 List 的区别

主要区别：

1. List 可以修改，Tuple 不可修改
2. 能用 Tuple 的地方就能用 List
3. List 几乎在所有情况下可代替 tuple
4. List 开销更大一些
5. Tuple 一般用来做函数返回值

单元素序列的表示

问题：在以下选项中，哪些能正确表示"只有一个元素的序列"？

- A. `[42,]`
- B. `(42,)`
- C. `[42]`
- D. `(42)`

注意：可带逗号结尾的语法有利于注释掉成员不影响语法

基础通用序列操作

1. 索引 `x[5]`
2. 切片 `s[2:4]`，`s[::]`，`s[::-1]`
3. 相加 `x+s`
4. 成员检查 `in`
5. 支持迭代 `for ... in`
6. 内置方法/函数

List 下标图解

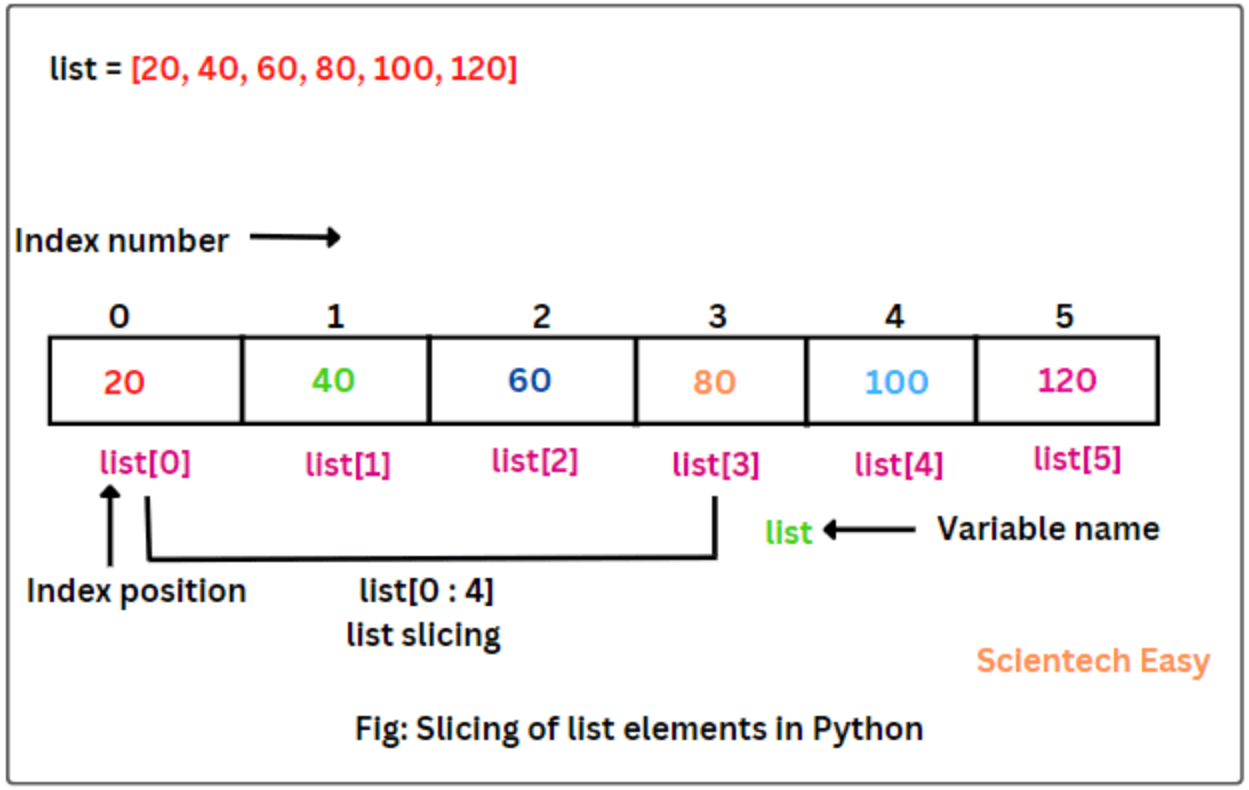


Fig: Slicing of list elements in Python

切片操作练习

如果 `x=[1,2,3,4,5]`，如何取出：

1. `[1,2,3]`
2. `[4,5]`
3. `[5,4,3,2,1]`
4. `[1,3,5]`
5. `[5,3,1]`

序列加法

执行以下代码，总结序列加法规则：

```
[1, 2, 3] + [4, 5, 6]
'Hello,' + 'world!'
[1, 2, 3] + 'world!'
[1, 2, 3] + (4, 5, 6)
(1, 2, 3) + (4, 5, 6)
[1, 2, 3] + ["4", "5", "6"]
```

序列乘法

执行以下代码，总结序列乘法规则：

```
[42] * 10
'python' * 5
[] + []
[] * 5
[None] * 10
```

成员包含检查

1. `[1,2,5,7]` 是否包含 `2`？
2. 文件属性 `Permissions = "rw"`，检查是否可写、可执行？
3. 检查输入用户名是否合法：`users = ['mlh', 'foo', 'bar']`

用户名和 PIN 码检查

练习：用户输入 `username='scuec', password='2345'`，检验是否为合法用户：

```
database = [['albert', '1234'],
            ['dilbert', '4242'],
            ['smith', '7524'],
            ['jones', '9843']]

['jones', '9843'] in database
```

内置函数：max, min, sort, len

执行以下代码，观察结果：

```
numbers = [100, 34, 678]
len(numbers)
max(numbers)
min(numbers)

week = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
max(week)
min(week)
week.sort()
```

List 赋值修改

如何修改列表的部分成员值？

- `a = [1,2,3,4,5,6]`
- 将 `a` 的第1个、最后一个元素置0
- 将 `a` 的最后一半置0
- 将 `a` 的偶数位置元素置0
- 将 `a` 的奇数位置元素的值 `*10`

删除列表成员

删除一个成员

删除 `a` 的第2个元素，哪个正确？

```
a = [1,2,3,4,5,6]

del a[2]          # 方法1
a[2:3] = []       # 方法2
a[2] = []         # 方法3
```

删除全部成员

删除 `a` 的全部元素，哪个正确？

```
a = [1,2,3,4,5,6]

a[:] = []      # 方法1
a = []         # 方法2
```

思考：两种方法有何区别？用 `id()` 探索

字符串与 List 相互转换

```
alist = list('Hello')
''.join(alist)
```

练习：将句子 `"He's a very naughty boy!"` 转成单词组成的 list

for...in 循环基础

打印每个成员

```
a = [1,2,3,4,5,6]
for each in a:
    print(each)
```

使用索引的循环

将偶数位置的成员值乘 10

```
a = [1,2,3,4,5,6]
index = 0
for each in a:
    if index % 2 == 0:
        a[index] = each * 10
    index += 1
```

enumerate() 用法

简化索引操作：

```
a = [1,2,3,4,5,6]
for index, each in enumerate(a):
    if index % 2 == 0:
        a[index] = each * 10
```

优点：自动处理索引初始化和自增

列表推导式 (List Comprehension)

传统写法：

```
# 传统方法
squares = []
for x in range(10):
    squares.append(x**2)
```

更简洁的列表创建方式：

```
# 列表推导式
squares = [x**2 for x in range(10)]
```

示例：将偶数位置的成员值乘 10

```
# 传统方法
a = [1,2,3,4,5,6]
for index, each in enumerate(a):
    if index % 2 == 0:
        a[index] = each * 10
```

简洁写法：

```
# 列表推导式
a = [x * 10 if i % 2 == 0 else x for i, x in enumerate(a)]
```

列表示推导的语法特点：

- 更简洁、可读性更好
- 性能通常优于传统循环
- 支持条件过滤：`[x for x in range(10) if x % 2 == 0]`
- 支持嵌套循环

zip() 用法

并行处理多个列表：

```
name = ['Tom', 'Alice', 'Bob']
age = [30, 42, 35]
for name, age in zip(name, age):
    print(name, age)
```

索引与切片赋值应用

问题1：将 `["p", "e", "r", "l"]` 修改成 `["p", "e", "a", "r"]`

```
# 方法1
var[-1] = "r"
var[-2] = "a"

# 方法2
var = var[:2] + ["r", "l"]

# 方法3
var[3:] = ["a", "r"]

# 方法4
var = list("".join(var).replace("rl", "ar"))
```

列表插入操作

问题2：将 `[1, 5]` 修改成 `[1, 2, 3, 4, 5]`

```
# 方法1
a[1] = [2,3,4]

# 方法2
a[1:1] = [2,3,4]
```

哪个方法可行？

列表赋值的副作用

```
send_signal = [1,2,3,4,5]
recv_signal = send_signal
recv_signal[2] = 10
print(send_signal)  # [1, 2, 10, 4, 5]
```

问题：为什么发送信号也发生了变化？

避免副作用的解決方案

方法1：使用 copy()

```
send_signal = [1,2,3,4,5]
recv_signal = send_signal.copy()
recv_signal[2] = 10
print(send_signal)  # [1, 2, 3, 4, 5]
```

避免副作用的解决方案

方法2：使用切片

```
send_signal = [1,2,3,4,5]
recv_signal = send_signal[:]
recv_signal[2] = 10
print(send_signal)  # [1, 2, 3, 4, 5]
```

探索：用 `id()` 检查对象关系

综合练习

已知学生数据列表，要求：

1. 根据姓名或学号返回完整记录
2. 根据姓名或学号排序

```
a = [('张三', 1001),
     ('李四', 1002),
     ('王五', 1003),
     ('陈六', 1004)]

# 输入：'王五'
# 输出：('王五', 1003)
```

总结

- List 和 Tuple 都是重要的序列类型
- List 可变，Tuple 不可变
- 掌握索引、切片、迭代操作
- 理解赋值和复制的区别
- 熟练使用内置函数和方法

练习：完成所有代码练习，加深理解