数据序列

软件测试Python课程V5.0

黑马程序员 www.itheima.com 传智教育旗下 高端IT教育品牌

学习目标

Learning Objectives

1. 掌握字符串常用操作
2. 掌握列表的作用和操作方法
3. 掌握元组的作用和操作方法
4. 掌握字典的作用和操作方法

目录

Contents

01

字符串

- 字符串的定义
- 字符串常用方法

学习目标

1. 掌握如何定义字符串
2. 掌握字符串的常用操作方法

字符串的定义

特殊字符串的处理

说明: 根据需求的不同, 有时需要处理一些特殊的字符串数据

```
# 字符串中包含引号
str1 = "I'm tom"   # 单双引号配合使用
str2 = 'I\'m tom'  # 使用转义字符进行转义



# 处理转义字符
file_path1 = "C:\\Desktop\\test"  # 转义
file_path2 = r"C:\Desktop\test"   # 忽略转义字符
```

r 是单词 raw 的缩写，表示原始的、未加工的

查找 find()

字符串.find(被查找字符)

说明	被查找字符是否存在于当前字符串中, 如果存在则返加开始下标, 不存在则返回 -1

案例

字符串查找

需求:

1. 现有字符串数据: '黑马程序员'
2. 请设计程序, 实现判断"黑马"和"白马"是否存在于数据中
3. 要求如果数据存在, 则输出数据所在位置

```
data = "黑马程序员"

index = data.find("黑马")
print(f"index={index}")

index2 = data.find("白马")
print(f"index2={index2}")
```

替换 replace()

字符串.replace(原字符串, 新子字符串)

说明	使用新的子字符串, 按规则替换旧的字符串内容

注意：字符串属于不可变数据类型，所以修改并不会影响原来的内容

案例

字符串替换

需求:

1. 现有字符串数据: '金三胖同志被称之为世界最成功的80后，金三胖真牛！'

2. 请设计程序, 实现将'金三胖'替换为'马赛克'

```
data = "金三胖同志被称之为世界最成功的80后，金三胖真牛！"
new_data = data.replace("金三胖", "马赛克")
print(data)
print(new_data)
```

拆分 split()

字符串.split(分割符)

说明	按照指定字符来分割字符串

注意：
1. 方法执行完成后返回的数据类型为列表(list)
2. 不传入分割符时，默认以空格进行拆分

案例

字符串拆分

需求:

1. 现有字符串数据: 'hello Python and itcast and itheima'

2. 请设计程序, 使用 and 拆分字符串

```
data = "hello Python and itcast and itheima"
print(data.split("and"))  # ['hello Python ', ' itcast ', ' itheima']
```

连接 join()

字符串.join(一般为列表)

说明	一般用于将列表按指定子字符合并为字符串

案例

字符串连接

需求:

1. 现有列表数据: ['Python', 'Java', 'PHP']

2. 请设计程序, 实现将数据整理成: 'Python and Java and PHP'

```
data = ['Python', 'Java', 'PHP']
print(" and ".join(data))
```

1. 如何定义字符串数据?

2. 字符串常用方法有哪些?

目录

Contents

02

列表

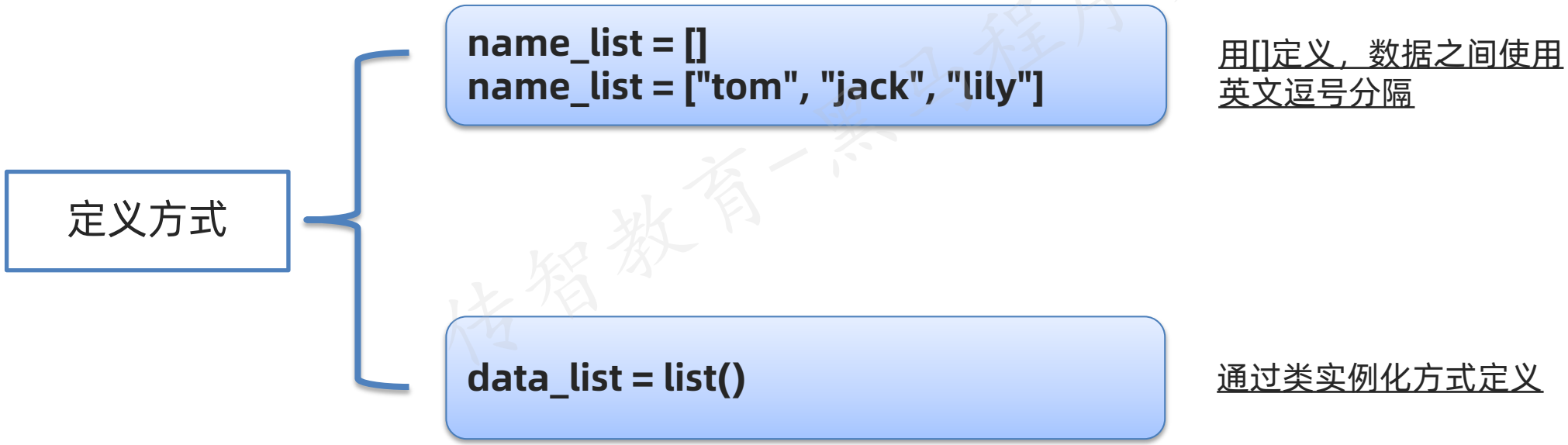
- 列表的定义
- 列表常用方法

学习目标

1. 掌握如何定义列表
2. 掌握列表的常用操作方法

列表的定义

说明	列表(list)是 Python 中使用最频繁的数据类型, 在其他语言中通常叫做数组, 专门用来存储一组数据



列表查询方法：索引

注意：使用不存在的索引，代码执行会报错！

item = 列表[索引]

说明	索引就是数据在列表中的位置编号，索引又被称为下标，默认从0开始

```
name_list = ['张三', '李四']
print(name_list[0])  # 张三
print(name_list[1])  # 李四
```

列表查询方法：count()

注意：如果目标数据不存在则返回 0

num = 列表.count(目标数据)

说明	统计被测试值出现的次数

```
data_list = ['python', 'java', 'python', 'php']


print(data_list.count("python"))  # 2
```

列表

列表增加方法：append()

注意：
1.方法执行是对原数据进行的修改，固列表是可变数据类型
2.如果增加一个列表，则此列表会被当做一个值添加到末尾

列表.append(新增数据)

说明	在列表的末尾添加数据

```
val_list = ["Web自动化", "UI自动化", "接口自动化"]

val_list.append("APP自动化")

print(val_list)  # ['Web自动化', 'UI自动化', '接口自动化', 'APP自动化']
```

列表删除方法：pop()

注意：使用方法时如果不传入索引值，默认删除列表中最后一个数据

del_data = 列表.pop(索引)

说明	删除指定索引对应的数据

```
val_list = ["Web自动化", "UI自动化", "接口自动化"]

val = val_list.pop(0)

print(val, val_list)  # web自动化，['UI自动化', '接口自动化']
```

列表修改方法：索引

注意：使用不存在的索引，代码执行会报错！

列表[索引] = 新数据

说明	通过指定索引修改对应数据

```
val_list = ["Web自动化", "UI自动化", "接口自动化", "Web自动化"]

val_list[1] = "黑马程序员"

print(val_list)  # ['Web自动化', '黑马程序员', '接口自动化', 'Web自动化']
```

列表修改方法：reverse()

注意：方法执行是对原数据进行的修改

列表.reverse()

说明	反转列表，将列表中的元素倒序

```
num_list = [1, 2, 3, 4]
num_list.reverse()
print(num_list)  # [4, 3, 2, 1]
```

列表修改方法：sort()

注意：reverse 表示排序规则，默认是False表示升序，设置为True表示降序

列表.sort(reverse=False)

说明	将列表按指定规则进行数据排序，默认为升序

```
val_list = [8, 100, 30, 10, 40, 2]

val_list.sort(reverse=True)

print(val_list)  # [100, 40, 30, 10, 8, 2]
```

列表其他方法：嵌套

说明: 列表数据可以多层嵌套

注意: 无论嵌套多少层, 都可以通过索引获取目标数据

```
student_list = [["张三", "18", "功能测试"], ["李四", "20", "自动化测试"]]


print(student_list[0][1])  # 18
```

1. 如何定义列表数据?

2. 列表数据的常用方法?

目录

Contents

03 元组

- 元组的定义
- 元组常用方法

学习目标

1. 掌握如何定义元组
2. 掌握元组的常用操作方法

元组

元组的定义

说明	元组和列表一样，都可用于存储多个数据，不同之处在于**元组的元素不能修改**

定义方式

```
user_info = ()
user_info = ("zhangsan", 18, 1.75)
```

用()定义，数据之间使用英文逗号分隔

```
info_tuple = tuple()
```

通过类实例化方式定义

```
# 注意：元组中只包含一个元素时，需要在元素后面添加逗号
data = (1,)
```

元组查询方法：索引

注意：使用不存在的索引，代码执行会报错！

item = 元组[索引]

```
tuple1 = (1, 2, 3)
print(tuple1[1])  # 2
```

元组查询方法: count()

注意：如果目标数据不存在则返回 0

num = 元组.count(目标数据)

说明	统计被测试值出现的次数

```
tuple1 = (1, 2, 3)
print(tuple1.count(3))  # 1
```

元组的特殊用法：交换两个变量的值【面试题】

说明：

- 在 Python 中可以使用对应数据个数的变量，获取对应元组数据中的每一个元素
- 在 Python 中定义多个元素的元组数据时，小括号可以省略
- 借助以上两个特性，可以通过元组快速实现交换两个变量的值

```
num1 = 100
num2 = 200
# 交换两个变量的值
num2, num1 = num1, num2
print(num1)  # 200
print(num2)  # 100
```

1. 如何定义元组数据?

2. 元组数据常用方法?

目录

Contents

04

字典

- 字典的定义
- 字典常用方法

学习目标

1. 掌握如何定义字典数据
2. 掌握字典的常用操作方法

应用场景

思考：一个学生包含学号、姓名、年龄等信息，在程序中如何更好的保存这些信息呢？

```
{
   "name": "xiaoming",
   "age": 28,
   "gender": "男"
}
```

字典不仅可以保存多个数据，同时还能给不同数据“起名字”

字典

字典的定义

提示：大括号内结构为：
键名：键值【俗称键值对】

定义方式

```
data = {}
data = {"name": "admin", "pwd": "123"}
```

用{}定义，数据之间使用英文逗号分隔

```
data = dict()
```

通过类实例化方式定义

注意 字典中的键一般都使用字符串，并且键名不能重复（如果重复原数据会被覆盖）

字典增加和修改方法

提示：如果键不存在，则增加键值对；如果键存在，则修改键对应的值

字典['键'] = 值

说明	在字典中增加键值对或修改已有键对应的值

```
info = {
    "name": "tom",
    "age": 18
}
info["salary"] = 100000

print(info)  # {'name': 'tom', 'age': 18,  'salary': 100000}
```

字典删除方法

提示：如果给出的键在字典中不存在，代码执行会报错！

字典.pop(键)

说明	在字典数据中根据给出的键删除对应值

```
info = {
    "name": "tom",
    "age": 18,
    "gender": "男"
}
info.pop("gender")
print(info)  # {'name': 'tom', 'age': 18}
```

字典查询方法： get()

注意：如果查询的键在字典中不存在，代码执行会返回 None(空值)！

value = 字典.get(键)

说明	通过键名查询对应值

```
info = {
    "name": "tom",
    "age": 18,
    "gender": "男"
}
print(info.get("name"))  # tom
print(info.get("abc"))   # None
```

遍历字典的Key

for key in 字典.keys():

说明	循环拿到字典中的每个键名

```
info = {
    "name": "tom",
    "age": 18,
    "gender": "男"
}
for key in info.keys():
    print(key)
```

遍历字典的Value

for value in 字典.values():

说明	循环拿到每个键对应的值

```
info = {
    "name": "tom",
    "age": 18,
    "gender": "男"
}
for value in info.values():
    print(value)
```

遍历字典的Key和Value

提示：调用字典.items()方法获取字典的键和值，并自动赋值给不同的变量

for k, v in 字典.items():

说明	循环拿到每个键和值

```
info = {
    "name": "tom",
    "age": 18,
    "gender": "男"
}
for k, v in info.items():
    print(f"key={k} value={v}")
```

1. 如何定义字典数据?

2. 字典数据的常用方法?

切片操作

适用类型: 字符串(str)/列表(list)/元组(tuple)

数据[起始索引:结束索引:步长]

说明	通过切片操作, 可以获取数据中指定部分的内容

注意:

- 结束索引对应的数据不会被截取到
- 支持正向索引和逆向索引
- 步长用于设置截取数据的间隔数量; 默认步长为 1, 可以省略不写

切片操作-示例代码

说明: 此处以字符串数据为例, 列表和元组数据操作方式一致

```
name = "abcdefg"
print(name[2:5:1])  # cde
print(name[2:5])    # cde
print(name[:5])     # abcde
print(name[1:])     # bcdefg
print(name[:])      # abcdefg
print(name[::2])    # aceg
print(name[:-1])    # abcdef, 负1表示倒数第一个数据print(name[-4:-1])  # def
print(name[::-1])   # gfedcba
```

通用方法

len()

适用类型: 字符串、列表、元组、字典

num = len(数据)

说明	获取数据的元素个数(数据长度)

```
str_data = "hello python"
print(len(str_data))  # 12: 字符中字符个数（包含空格）
list_data = ["python", "java"]
print(len(list_data))  # 2: 列表中元素个数
tuple_data = ("admin", 123456, 8888)
print(len(tuple_data))  # 3: 元组中元素个数
dict_data = {"name": "tom", "age": 18, "gender": "男"}
print(len(dict_data))  # 3: 字典中键值对的个数
```

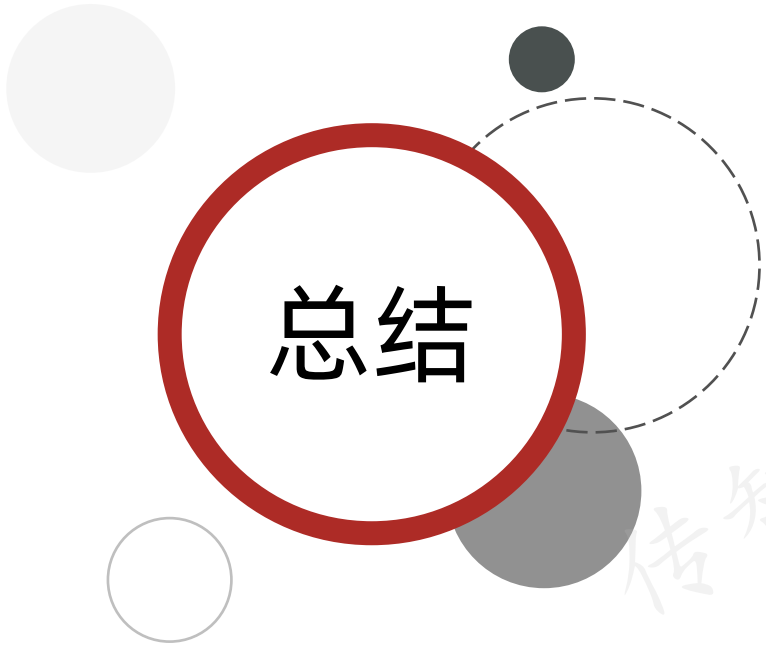
1．字符串、列表、元组和字典的通用方法?

案例

记录登录功能测试用例数据

需求：

1. 参考TPshop项目的登录功能（登录时需要输入用户名、密码、验证码），至少设计3条测试用例
2. 要求1：定义变量保存测试数据（包括不同测试数据对应的测试结果）
3. 要求2：至少写出3种以上不同的数据格式
4. 要求3：遍历测试数据并打印到控制台，数据格式“用户名：xxx 密码：xxx 验证码：xxx 期望结果：xxx”



1. 掌握字符串常用操作
2. 掌握列表的作用和操作方法
3. 掌握元组的作用和操作方法
4. 掌握字典的作用和操作方法

传智教育旗下高端IT教育品牌

黑马程序员-软件测试