# 字符串

## 课程目标

* 能够说出如何定义一个字符串
* 能够使用切片对一个字符串进行局部字符串内容的获取
* 能够使用find 方法完成在一个字符串中查找子串出现的位置
* 能够完成找出真凶案例

## 课程内容

* 字符串
* 字符串常用操作

### 字符串

#### 1.1字符串概念

字符串是一个容器，包含若干个字符并按照一定的顺序组织成一个整体。字符串支持索引操作。

#### 1.2创建字符串基本语法

变量名 = "字符串信息"

变量名 = '字符串信息'

变量名 = """字符串信息"""

变量名 = '''字符串信息'''

#### 1.3注意事项

字符串定义过程中如出现**"**匹配问题，使用**\"**进行转义

### 字符串切片

#### 2.1字符串切片概念

字符串切片指获取字符串中的某一部分，形成新的字符串对象。

#### 2.2字符串切片基本语法

字符串[开始索引:结束索引]

字符串[开始索引:结束索引:步长]

### 字符串操作

#### 3.1基本操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 操作格式/函数名称 | 功能 | 参数 | 返回值 | 范例 |
| 变量名[idx]  关键词： 获取单个字符 | 获取字符串中指定索引位置的字符 | idx:要获取的索引位置 | 结果是包含了单个字符的字符串 | info = **"itcast"** s = info[3] 结果：**"a"** |
| str1 in str2  关键词：包含 | 判断一个字符串str2是否包含另一个字符串str1 | str1:比对字符串对象  str2:原始字符串对象 | 结果是True或False | str1 = **"cast"** str2 = **"itcast"** str2 **in** str1 结果：**False** str1 **in** str2 结果：**True** |
| str1 not in str2  关键词：不包含 | 判断一个字符串str2是否不包含另一个字符串str1 | str1:比对字符串对象  str2:原始字符串对象 | 结果是True或False | str1 = **"cast"** str2 = **"itcast"** str2 **not** **in** str1 结果：**True** str1 **not in** str2 结果：**False** |
| len(str)  关键词：长度 | 本操作是一个函数，无须用字符串对象调用，用于获取一个字符串中所包含的字符数量 | str:字符串对象 | 结果是一个int整数 | info = **"itcast"** len(info) 结果：6 |
| max(str)  关键词：最大值 | 本操作是一个函数，无须用字符串对象调用，用于获取字符串中排序最大的字符 | str:字符串对象 | 结果是一个包含了单个字符的字符串 | info = **"itcast"** max(info) 结果：**"t"** |
| min(str)  关键词：最小值 | 本操作是一个函数，无须用字符串对象调用，用于获取字符串中排序最小的字符 | str:字符串对象 | 结果是一个包含了单个字符的字符串 | info = **"itcast"** max(info) 结果：**"a"** |

#### 3.2状态获取操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法名 | 功能 | 参数 | 返回值 | 范例 |
| islower()  关键词：是否全小写 | 判断字符串是否是全小写字母组成 | 无 | 结果是True或False | info1 = **"itcast"** info1.islower() 结果：**True**  info2 = **"ITHEIMA"** info2.islower() 结果：**False** |
| isupper()  关键词：是否全大写 | 判断字符串是否是全大写字母组成 | 无 | 结果是True或False | info1 = **"itcast"** info1.issupper() 结果：**False**  info2 = **"ITHEIMA"** info2.issupper() 结果：**True** |
| isdigit()  关键词：是否全数字 | 判断字符串是否是由纯数字组成 | 无 | 结果是True或False | info1 = **"666"** info1.isdigit() 结果：**True**  info2 = **"itcast666"** info2.isdigit() 结果：**False** |
| isalpha()  关键词：是否全字母 | 判断字符串是否是由纯字母组成 | 无 | 结果是True或False | info1 = **"itcast"** info1.isalpha() 结果：**True**  info2 = **"itcast666"** info2.isalpha() 结果：**False** |
| isalnum()  关键词：是否全数字字母 | 判断字符串是否是由纯数字和字母组成 | 无 | 结果是True或False | info1 = **"itcast666"** info1.isalnum() 结果：**True**  info2 = **"itcast\_666"** info2.isalnum() 结果：**False** |
| istilte()  关键词：是否单词首字母大写 | 判断字符串是否是满足单词首字母大写格式 | 无 | 结果是True或False | info1 = **"It Cast"** info1.istitle() 结果：**True**  info2 = **"ItCast"** info2.istitle() 结果：**False** |
| startswith(str)  关键词：判定前缀 | 判断字符串是否以指定字符串开始 | str:用于作为判定条件的字符串对象 | 结果是True或False | info = **"logo.png"** info.startswith(**"lo"**) 结果：**True** |
| endswith(str)  关键词：判定后缀 | 判断字符串是否以指定字符串结束 | str:用于作为判定条件的字符串对象 | 结果是True或False | info = **"logo.png"** info.endswith(**".png"**) 结果：**True** |

#### 3.3字符转换操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法名 | 功能 | 参数 | 返回值 | 范例 |
| lower()  关键词：转小写 | 字符串中所有字母转小写字母，支持英文字母 | 无 | 结果是一个字符串 | str1 = **"itcast"** str1.lower() 结果：**"itcast"** str2 = **"ITCAST666"** str2.lower() 结果：**"itcast666"** |
| casefold()  关键词：转小写 | 字符串中所有字母转小写字母，支持各种语言 | 无 | 结果是一个字符串 | str1 = **"itcast"** str1.lower() 结果：**"itcast"** str2 = **"ITCAST666"** str2.lower() 结果：**"itcast666"** |
| upper()  关键词：转大写 | 字符串中所有字母转大写字母 | 无 | 结果是一个字符串 | str1 = **"itcast"** str1.upper() 结果：**"ITCAST"** str2 = **"ITCAST666"** str2.upper() 结果：**"itcast666"** |
| swapcase()  关键词：大小写互换 | 字符串中字母大写转小写，小写转大写 | 无 | 结果是一个字符串 | str1 = **"IT heima"** str1.swapcase() 结果：**"is HEIMA"** str2 = **"Itcast666"** str2.swapcase() 结果：**"iTCAST666"** |
| title()  关键词：单词首字母大写其他小写 | 字符串中每个单词首字母大写，其余字母小写（区分单词以空格区分） | 无 | 结果是一个字符串 | str1 = **"itcast"** str1.title() 结果：**"Itcast"** str2 = **"it hei ma"** str2.title() 结果：**"It Hei Ma"** |
| capitalize()  关键词：字符串首字母大写其他小写 | 字符串首个字母大写，其余字母小写 | 无 | 结果是一个字符串 | str1 = **"itcast"** str1.capitalize() 结果：**"Itcast"** str2 = **"it hei ma"** str2.capitalize() 结果：**"It hei ma"** |

#### 3.4格式转换操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法名 | 功能 | 参数 | 返回值 | 范例 |
| strip (str)  关键词：左右去字符 | 去掉字符串左右两侧在参数字符串中包含的所有字符 | str:包含了若干个字符的字符串 | 结果是一个字符串 | str1 = **" itcast "** str1.strip(**" "**) 结果：**"itcast"** str2 = **"itcast"** str2.strip(**"it"**) 结果：**"cas"** str3 = **"itcast"** str3.strip(**"sat"**) 结果：**"itc"** |
| lstrip (str)  关键词：左侧去字符 | 去掉字符串左侧在参数字符串中包含的所有字符 | str:包含了若干个字符的字符串 | 结果是一个字符串 | str1 = **" itcast "** str1.strip(**" "**) 结果：**"itcast "** str2 = **"itcast"** str2.strip(**"it"**) 结果：**"cast"** str3 = **"itcast"** str3.strip(**"sat"**) 结果：**"itcast"** |
| ljust (len,str)  关键词：右侧补占位符 | 使用指定字符在原始字符串右侧补充到长度为指定值 | len:补充字符后的字符串长度  str:补充的字符，如果使用多个字符组成的字符串将报错 | 结果是一个字符串 | str1 = **"itcast"** str1.ljust(10,**"6"**) 结果:**"itcast6666"** str1.ljust(5,**"6"**) 结果:**"itcast"** str1.ljust(10,**"66"**) 结果:报错 |
| rjust (len,str)  关键词：左侧补占位符 | 使用指定字符在原始字符串左侧补充到长度为指定值 | len:补充字符后的字符串长度  str:补充的字符，如果使用多个字符组成的字符串将报错 | 结果是一个字符串 | str1 = **"itcast"** str1.rjust(10,**"6"**) 结果:**"6666itcast"** str1.rjust(5,**"6"**) 结果:**"itcast"** str1.rjust(10,**"66"**) 结果:报错 |
| center (len,str)  关键词：左右两侧补占位符 | 使用指定字符在原始字符串两侧补充到长度为指定值，左侧补充数量≥右侧补充数量 | len:补充字符后的字符串长度  str:补充的字符，如果使用多个字符组成的字符串将报错 | 结果是一个字符串 | str1 = **"itcast"** str1.center(10,**"6"**) 结果:**"66itcast66"** str1.center(11,**"6"**) 结果:**"666itcast66"** str1.center(5,**"6"**) 结果:**"itcast"** str1.center(10,**"66"**) 结果:报错 |
| zfill(len)  关键词：左侧补0 | 使用0在原始字符串左侧补充到长度为指定值，小数点占1位 | len:补充字符后的字符串长度 | 结果是一个字符串 | str1 = **"3.14"** str1.zfill(6) 结果：**"003.14"** str2 = **"314"** str2.zfill(6) 结果：**"000314"** str3 = **"3.14"** str3.zfill(3) 结果：**"3.14"** |

#### 3.5拆分操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法名 | 功能 | 参数 | 返回值 | 范例 |
| partition(str)  关键词：左侧查询3段拆分 | 从字符串左侧查找到参数后，将参数左侧、参数、参数右侧的三个字符串组成元组并返回 | str:比对字符串 | 包含三个字符串的元组对象 | str1 = **"itcast"** str1.partition(**"t"**) 结果：(**"i"**, **"t"**, **"cast"**) str1.partition(**"o"**) 结果：(**"itcast"**, **""**, **""**) |
| rpartition(str)  关键词：右侧查询3段拆分 | 从字符串右侧查找到参数后，将参数左侧、参数、参数右侧的三个字符串组成元组并返回 | str:比对字符串 | 包含三个字符串的元组对象 | str1 = **"hello python"** str1.rpartition(**"t"**) 结果： (**"hello py"**, **"t"**, **"hon"**) str1.rpartition(**"a"**) 结果： (**""**, **""**, **"hello python"**) |
| split(str)  关键词：多段拆分 | 使用参数作为分割线将原始字符串拆分成若干个字符串并组织成列表返回 | str:比对字符串 | 包含多个字符串的列表对象 | str1 = **"hello python"** str1.split(**"o"**) 结果：[**"hell"**, **" pyth"**, **"n"**] str1.split(**"oo"**) 结果：[**"hello python"**] |
| splitlines ()  关键词： | 使用换行符作为分割线将原始字符串拆分成若干个字符串并组织成列表返回 | 无 | 包含多个字符串的列表对象 | str1 = **"hello\nitcast\npython"** str1.splitlines() 结果：[**"hello"**, **"itcast"**, **"python"**] str2 = **"hello itcast"** str2.splitlines() 结果：[**"hello itcast"**] |

#### 3.6连接操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法名 | 功能 | 参数 | 返回值 | 范例 |
| join(str)  关键词：占位连接 | 将原始字符串填充到参数的每个字符之间组成新的字符串返回 | str:待填充的字符串 | 结果是一个字符串 | **"6"**.join(**"itcast"**) 结果：**"i6t6c6a6s6t" "66"**.join(**"itcast"**) 结果：**"i66t66c66a66s66t"** |
| str1 + str2  关键词：字符串连接 | 将两个字符串按照顺序拼接成一个新的字符串返回 | 无 | 结果是一个字符串 | **"it"**+**"heima"** 结果：**"itheima" "hello "**+**"python"** 结果：**"hello python"** |

#### 3.7查询操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法名 | 功能 | 参数 | 返回值 | 范例 |
| find(str,begin,end)  关键词：左侧查找 | 从左侧查找字符串从指定开始位置到指定结束位置间第一次出现的索引位置 | str:要查找的字符串  begin:开始索引，整数,该值要小于end，否则结果为-1  end:结束索引，整数,改制要小于begin，否则结果为-1 | 结果是一个int整数，如没有查找到返回-1 | str1 = **"itcast"** str1.find(**"t"**) 结果：1  str1.find(**"t"**,3,6) 结果：5  str1.find(**"t"**,6,3) 结果：-1 str1.find(**"cast"**) 结果：2 str1.find(**"m"**) 结果：-1 |
| rfind(str,begin,end)  关键词：右侧查找 | 从右侧查找字符串从指定开始位置到指定结束位置间第一次出现的索引位置 | str:要查找的字符串  begin:开始索引，整数,该值要小于end，否则结果为-1  end:结束索引，整数,改制要小于begin，否则结果为-1 | 结果是一个int整数，如没有查找到返回-1 | str1 = **"itcast"** str1.rfind(**"t"**) 结果：5  str1.rfind(**"t"**,0,4) 结果：1  str1.rfind(**"t"**,4,0) 结果：-1  str1.rfind(**"cast"**) 结果：2 str1.rfind(**"m"**) 结果：-1 |
| rindex(str,begin,end)  关键词：左侧查找 | 从左侧查找字符串从指定开始位置到指定结束位置间第一次出现的索引位置 | str:要查找的字符串  begin:开始索引，整数,该值要小于end，否则结果报错  end:结束索引，整数,改制要小于begin，否则结果报错 | 结果是一个int整数，如没有查找到报错 | str1 = **"itcast"** str1.index(**"t"**) 结果：1  str1.index(**"t"**,3,6) 结果：5  str1.index(**"t"**,6,3) 结果：报错 str1.index(**"cast"**) 结果：2 str1.index(**"m"**) 结果：报错 |
| rindex(str,begin,end)  关键词：右侧查找 | 从右侧查找字符串从指定开始位置到指定结束位置间第一次出现的索引位置 | str:要查找的字符串  begin:开始索引，整数,该值要小于end，否则结果报错  end:结束索引，整数,改制要小于begin，否则结果报错 | 结果是一个int整数，如没有查找到报错 | str1 = **"itcast"** str1.rindex(**"t"**) 结果：5  str1.rindex(**"t"**,0,4) 结果：1  str1.rindex(**"t"**,4,0) 结果：报错 str1.rindex(**"cast"**) 结果：2 str1.rindex(**"m"**) 结果：报错 |
| count(str)  关键词：字符串数量 | 查询指定字符串在原始字符串中出现的次数 | str:被查询的字符串 | 结果是一个int整数 | info = **"itcast"** info.count(**"t"**) 结果：2  info.count(**"tc"**) 结果：1 |

#### 3.8替换操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法名 | 功能 | 参数 | 返回值 | 范例 |
| replace(old\_str,new\_str,num)  关键词：替换 | 使用新字符串替换原始字符串中的指定字符串信息 | old\_str:被替换的字符串  new\_str:新字符串  num：替换数量 | 结果是一个字符串 | str1 = **"itcast"** str1.replace(**"t"**,**"T"**) 结果：**"iTcasT"** str1.replace(**"t"**,**"T"**,1) 结果：**"iTcast"** str1.replace(**"T"**,**"t"**) 结果：**"itcast"** |
| expandtabs()  关键词：\t换空格 | 使用空格替换原始字符串中的制表位\t | 无 | 结果是一个字符串 | str1 = **"66"** str2 = **"itca\tst"** str1+str2 **"66itca st" （空白是一个\t）** str1+str2.expandtabs() **"66itca st"（空白是4个空格）** |

#### 3.9其他操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法名 | 功能 | 参数 | 返回值 | 范例 |
| maketrans(str1,str2)  关键词：生成字典 | 使用两个等长的字符串中的每个对应位置的字符生成一个字典对象 | str1:用于生成字典中key的字符对应的字符串，长度必须与str2相同  str2:用于生成字典中value的字符对应的字符串，长度必须与str1相同 | 包含字符串对应位置字符组成的字典对象 | str.maketrans(**"123"**,**"abc"**) 结果： {49: 97, 50: 98, 51: 99} |
| translate (dict)  关键词：使用字典关系转换 | 使用字典对字符串信息进行转换 | dict:包含有转换关系的字典对象 | 结果是一个字符串 | dict1 = {49: 97, 50: 98, 51: 99} **"123456"**.translate(dict1) 结果：**"abc456"** |

### 案例：找出真凶

"""  
案例找出真凶  
分析：  
1.姓 name startswith  
2.名 name nick\_name find  
3.性别 gender ==  
4.血型 blood ==  
5.籍贯 native find  
6.出生日期 idcard 切片，比较运算  
"""  
db\_infos = [{"name": "张三丰","gender": 1, "nick\_name": "三爷", "idcard": "110101153808081017", "blood": "b", "native":"湖北省丹江口市武当山玉虚宫"},{"name": "张大彪","gender": 1, "nick\_name": "斌仔", "idcard": "130323197711111011", "blood": "b", "native":"河北省秦皇岛市武山海关区鞋拔子路胶水胡同103"}]  
# 1. 循环列表，取出个人信息  
for person in db\_infos:  
 # 2.取出个人信息  
 name = person["name"]  
 nick\_name = person["nick\_name"]  
 gender = person["gender"]  
 blood = person["blood"]  
 native = person["native"]  
 idcard = person["idcard"]  
 # 3.判断是否满足条件  
 # 3.1 所有条件都满足，打印个人信息  
 # 3.2 只要有一个条件不满足，检查下一个人  
 if not name.startswith("张") :  
 continue  
 if name.find("斌") == -1 and nick\_name.find("斌") == -1 :  
 continue  
 if gender != 1:  
 continue  
 if blood.lower() != "b":  
 continue  
 if native.find("河北") == -1:  
 continue  
 # 身份证号： XXXXXX XXXX XX XX XXX X  
 if int(idcard[6:10]) < 1975 or int(idcard[6:10]) > 1978 :  
 continue  
  
 # 设计字典，做对应关系（映射）  
 genders = {1:"男",0:"女",-1:"不详"}  
  
 # 到达此处说明所有条件都已经通过，打印个人信息  
 print("姓名:"+name+",曾用名:"+nick\_name+",性别:"+genders[gender]+",血型:"+blood+",籍贯:"+native+",身份证号:"+idcard)