六、字符串

目标:

- 认识字符串
- 下标
- 切片
- 常用操作方法

6.1 认识字符串

字符串是 Python中最常用的数据类型。我们一般使用引号来创建字符串。创建字符串很简单，只要为变量分配一个值即可。

```
a = 'hello' \
     'world'
b = 'abcdefg'
print(type(a)) # str
print(type(b)) # str

# 三引号
e = '''i am tom'''
print(type(e)) # str
f = """i am tom"""
print(type(f)) # str

# 转义字符问题
# I'm Tom
c = " I'm Tom "
print(c) #  I'm Tom
print(type(c)) # str

# d = ' I'm Tom '
d = ' I\'m Tom '
print(d) #  I'm Tom
print(type(d)) # str

```

6.2 字符串输出

```
print('hello wolrd')

name = 'Tom'
print(f'我的名字是%s' % name)
print(f'我们的名字是{name}')

```

6.3 字符串输入

在python中，使用 `input()` 接收用户输入

```python
name = input('请输入名字：')
print(f'您输入的名字是{name}')
print(type(name)) # str

password = input('请输入您的密码：')
print(f'您输入的密码是{password}')
print(type(password)) # str
# 总结：无论是字符串还是数字都是str类型
```

6.4 下标

下标又叫索引，就是编号。比如火车座位号，座位号的作用：按照编号快速找到对应的座位。同理，下标的作用就是通过下标快速找到对应的数据。

```python
str1 = 'abcdefg'
print(str1)

# 想得到数据a字符，得到数据b字符 -- 使用字符串中某个特定的数据
# 这些字符数据从0开始顺序分配一个编号 -- 使用这个编号精确找到某个字符数据-- 下标或者索引或索引值
print(str1[0]) # a
print(str1[1]) # b

```

6.5 切片

6.5.1 切片简介

```python
str1 = 'abcdefg'
print(str1) # 获取整个

# 下标得到的是下标为某个数字的数据
print(str1[2]) # c
# 得到abc这三个数据该怎么办?
```

切片是指对操作的对象截取其中一部分的操作。**字符串、列表、元组**都支持切片操作。

语法：

```
序列[开始位置下标 ：结束位置下标 ：步长]
```

> 注意

1、不包含结束位置下标对应的数据，正负整数均可.

2、步长是选取间隔，正负整数均可，默认步长为1.

6.5.2 切片体验

```python
name = 'abcdefg'
```

```
print(name[2:5:1]) # cde
print(name[2:5]) # cde
print(name[:5]) # abcde  -- 如果不写开始，默认从0开始选取
print(name[2:]) # cdefg  -- 如果不写结束，表示选取到最后
print(name[:]) # abcdefg -- 如果不写开始和结束，表示选取所有

# 负数测试
print(name[::-1])  #gfedcba 如果步长为负数，表示倒序选取
print(anme[-4:-1]) # def 下标-1表示最后一个数据，依次向前类推

#终极测试
print(name[-4:-1:1]) # def
print(name[-4:-1:-1]) # 不能选取数据，从-4开始到-1结束，选取方向为从在到右，但是-1步长，从右向左选取
# ***** 如果选取方向(下标开始到结束的方向) 和 步长的方向冲突，则无法选取数据
print(name[-1:-4:-1]) # 要方向一致，才能选取数据
```

6.6 字符串常用方法

字符串的常用操作方法有**查找、修改和判断**三大类。

6.6.1 查找find()和index()

所谓字符串查找方法即是查找子串在字符串中的位置或出现的次数。

- find()：检测某个子串是否包含在这个字符串中，如果在返回这个子串开始的位置下标，否则则返回-1。

1、语法

```
字符串序列.find(子串，开始位置下标，结束位置下标)
```

> 注意：开始和解书位置下标可以省略，表示在整个字符串序列中查找。

2、快速体验

```
mystr = "hello world and itcast and iteima and python"

print(mystr.find('and')) # 12
print(mystr.find('and', 15, 30)) # 23
print(mystr.find('ands')) # -1
```

- index()：检测某个子串是否包含在这个字符串中，如果在返回这个子串开始的位置下标，否则则报异常

1、语法

```
字符串序列.index(子串，开始位置下标，结束位置下标)
```

> 注意：开始和结束位置下标可以省略，表示在整个字符串徐柳中查找。

2、快速体验

```
mystr = "hello world and itcast and iteima and python"

print(mystr.index('and')) # 12
print(mystr.index('and', 15, 30)) # 23
print(mystr.index('ands')) # 如果index查找子串不存在，会报错
```

- count ()

1、语法

```
字符串序列.count(子串，开始位置下标，结束位置下标)
```

2、快速体验

```
mystr = "hello world and itcast and iteima and python"

print(mystr.count('and',15,30)) # 1
print(mystr.count('and')) #  3
print(mystr.count('ands')) #  3
```

- rfind()：和find()功能相同，但查找方向从右侧开始
- rindex()：和index()功能相同，但查找方向为右侧开始
- count()：返回某个子串在字符串中出现的次数

```
mystr = "hello world and itcast and iteima and python"
print(mystr.rfind('and')) # 1
```

6.6.2 修改

所谓修改字符串，指的就是通过函数的形式修改字符串中的数据。

- replace()： 替换

1、语法

```
字符串序列.replace(旧字符串，新子串，替换次数)
```

> 注意：替换次数如果查出子串出现的数据，则替换次数为该子串出现的次数

2、快速体验

```python
mystr = "hello world and itcast and iteima and python"
print(mystr.replace('and', 'he')) # hello world he itcast he iteima he python

print(mystr.replace('and', 'he', 10)) # 替换次数如果超出字串出现的次数，表示替换所有这个字串
print(mystr)

#***** 调用了replace函数后，发现原有字符串的数据并没有做到修改，修改后的数据是replace函数的返回值
#--- 说明 字符串是不可变数据类型
#数据是否可以改变划分为 可变类型 和  不可变类型
```

> 注意：数据按照是否能直接修改分为可变类型和不可变类型两种。字符串类型的数据修改的时候
>
> 不能改变原有字符串，属于不能直接修改数据的类型即是不可变类型。

- split()：按照指定字符分割字符串 --- 分割， 返回一个列表， 丢失分割字符

1、语法

```
字符串序列.split(分割字符，num)
```

> 注意：num表示的是分割字符出现的次数，即将来返回数据个数为num+1个。

2、快速体验

```python
mystr = "hello world and itcast and iteima and python"

list1 = mystr.split('and')
# ['hello world ', ' itcast ', ' iteima ', ' python']
print(list1)
```

- join()： 用一个字符或子串合并字符串，即是将多个字符串合并为一个新的字符串

1、语法

```
字符或者子串.join(多字符串组成的序列)
```

2、快速体验

```python
list1 = ['chuan', 'zhi', 'bo', 'ke']
t1 = ('aa','b','cc','ddd')
#chuan...zhi...bo...ke
#<class 'str'>
new_list1 = '...'.join(list1)
print(new_list1)

```

6.6.3 修改之大小写转换

- capitalize()： 将字符串第一个字符转换成大写

```python
mystr = "hello world and itcast and iteima and python"
# Hello world and itcast and iteima and python
print(mystr.capitalize())

```

> 注意：capitalize()函数转换后，字符串第一个字符大写，其他的字符全是小写

- title()：将字符串每个单词首字母转换成大写。

```python
mystr = "hello world and itcast and iteima and python"
# Hello World And Itcast And Iteima And Python
print(mystr.title())
```

- lower()：将字符串中大写转换成小写

```python
mystr = "hello World and itcast and iteima and python"

print(mystr.lower())

```

- upper()：将字符串中小写转换成大写

```python
mystr = "hello World and itcast and iteima and python"

print(mystr.upper())
```

6.6.4 修改之删除空白字符

- lstrip()：删除字符串左侧空白字符
- rstrip()：删除字符串右侧空白字符
- strip()：删除字符串两侧空白字符（这里要注意和split函数注意区分）

```
mystr = "     hello World and itcast and iteima and python     "
new_str = mystr.lstrip()
print(new_str)
```

6.6.5 修改之字符串对齐

- ljust()：返回一个原字符串左对齐，并使用指定字符（默认空格）填充至对应长度的新字符串。

1、语法

```
字符串序列.ljust(长度，填充字符)
```

2、快速检测

```
mystr = 'hello'
print(mystr.ljust(10,'.')) # hello.....

```

- rjust()：返回一个原字符串右对齐并使用指定字符（默认空格）填充至对应长度的新字符串，语法和ljust()相同
- center()：返回一个原字符串居中对齐并使用指定字符（默认空）填充至对应长度的新字符串，语法和ljust()相同。

6.6.6 判断开头或结尾

所谓判断即是判断真假，返回的结果是布尔型数据类型：True或 False

- startswith()： 检查字符串是否是以指定子串开头，是则返回True，否则返回 False。如果设置开始和结束位置下标，则在指定范围内检查。

1、语法

```
字符串序列.startswitch(字串，开始位置下标，结束位置下标)
```

2、快速体验

```
mystr = "hello World and itcast and iteima and python"
print(mysrt.startswith('hello')) # True

```

- endswith（）：与startswitch函数类似

```
# endswitch()
print(mystr.endswitch('pythons')) # False
```

6.6.7 判断

- isalpha()：如果字符串至少有一个字符并且所有字符都是字母则返回True，否则返回False

```
mystr1 = 'hello'
mystr2 = 'hello1234'

print(mystr1.isalpha()) # True

print(mystr2.isalpha()) # False

```

- isdigit()：如果字符串只包含数字则返回True否则返回False

```
mystr1 = 'aaa23232'
mystr2 = '12121'
print(mystr1.isdigit()) # False
print(mystr2.isdigit()) # True
```

- isalnum：如果字符串至少有一个字符并且所有字符都是字母或者数字则返回True，否则返回Flase

```
mystr1 = 'aaa2332'
mystr2 = '33434-'

print(mystr1.isalnum()) # True
print(mystr1.isalnum()) # False
```

- isspace()：都是空白时，返回True

```
mystr1 = ‘1 2 3 4’
print(mystr1.isspace()) # False
```