DAY03

数据类型（2）

字符串 str

作用:

是用来记录文本信息(文字信息)，比如：记录学员姓名："张小黎"

字面值

说明:

在非注释中凡是用引号括起来的部分都是字符串

```
引号可以使用如下几种:
'       单引号
"       双引号
'''     三单引
"""     三双引
```

示例:

```
# 空字符串  (四项的id()相同)
''
""
''''''
""""""
# 非空字符串  (四项的id()相同)
'hello'
"hello"
'''hello'''
"""hello"""
```

注意事项:

- 单引号和双引号的区别

  单引号内的双引号不算结束符；双引号内的单引号不算结束符。

  ```
  # 示例
  print("I'm a teacher")
  print('I love "Beijing"')
  ```

- 三引号字符串的使用

  三引号内可以包含单引号和双引号

  三引号字符串中的换行会自动转换为换行符'\n'

```python
# 示例
print("welcome to beijing.\nI like python!\nI am studing!")
print("""第一行内容
第二行内容
第三行内容""")
```

- 字面值的隐式拼接

  多个字符串字面值连续书写，Python会自动将其拼接起来。

```python
# 示例
print("I'm teacher."  'I like "Python". ' "'''"   '""" is str')
# 输出结果
# "I'm teacher."  'I like "Python". ' "'''"   '""" is str'
```

- 用转义字符代表特殊字符

  本节课重点：观看视频 Python01阶段 DAY03 "03-字符串02" 时间：00:100:16 - 00::

  字符串字面值中用字符反斜杠\ 后跟一个或一些字符代表特殊的一个字符

转义字符	说明
\'	一个单引号
\"	一个双引号
\	一个斜杠
\n	换行符
\r	返回光标至行首
\f	换页
\t	水平制表符
\v	垂直制表符
\b	倒退
\a	响铃(已无效)
\0	字符串(编码值为0的字符)
\0oo	oo为两位八进制表示的字符
\xXX	xx为两位十六进制表示的字符
\uXXXX	Unicode 16 的十六进制表示的字符
\UXXXXXXXX	Unicode 32 的十六进制表示的字符

```
```

```
# 如何使用编码值

>>> print("?")
?
>>> print("\077")
?
>>> print("\x3F")
?
>>> print("\u003F")
?
>>> print("\U0000003F")
?

# 中文编码

>>> print("\u7115")
焕
>>> print("\u7119")
焙
>>> print("\u7120")
焠
>>> print("\u8120")
艇
>>> print("\u6120")
愠
>>> print("\u6320")
挠
>>> print("\u6325")
挥
>>> print("\u6328")
挨
```

- ASCII编码

  Linux查看ASCII编码：`$ man ascii`

  常用的ASCII 编码:

字符	十进制	十六进制
'0'	48	0x30
'A'	65	0x41
'a'	97	0x61

raw字符串

raw字符串，也称为：原始字符串

格式:

```
r'字符串内容'
r"字符串内容"
r'''字符串内容'''
r"""字符串内容"""
```

作用:

让转义字符 \ 无效

示例:

```
# 表示  C:\newfile\test.py
s1 = 'C:\newfile\test.py'
print(s1)
s2 = r'C:\newfile\test.py'
print(s2)
```

运算符操作

算术运算符

“+”加号运算符

加号运算符用于拼接字符串

```
x = "ABCD" + "1234"
print(x)   # ABCD1234
y = "1234" + "ABCD"
print(y)   # 1234ABCD
```

注意： 字符串本身不可修改，拼接为新的字符串。

```
>>> str1 = "ABCD"
>>> str2 = "EFGH"
>>> str3 = str1 + str2

>>> print(id(str1))
2283683235184
>>> print(id(str2))
2283683235408
>>> print(id(str3))
2283683241136
```

“*”乘号运算符

乘号运算符，用于生成重复的字符串

```
x = "ABC" * 3
print(x)    # ABCABCABC
```

注: 字符串只能和整数相乘， 不能使用浮点数

复合赋值运算符

“+=”加等运算符

加等运算符，用于拼接字符串,同时改变原字符串的变量的绑定的关系

```
x = "abcd"
y = "1234"
y += x   # 等同于 y = y + x  # 生成新的字符串
print(x)  # 不变
print(y)  # 1234abcd
```

“*=”加号运算符

乘等运算符，用于生成重复字符串,同时用原变量来绑定

```
x = "123"
x *= 4
print(x)  # 123123123123
```

同样注意： 字符串本身不可修改，用当前变量来绑定新的字符串。

```
>>> str1 *= 3
>>> str1
'123123123'
>>> id(str1)
2283683241200
```

练习

写一个程序,打印一个高度为4行的矩形方框 要求输入一个整数,此整数代表矩形的宽度,输出此矩形: 如: 请输入矩形的宽度: 10 打印如下:

```
##########
#        #
# #########
```

答案:

```python
width = int(input("请输入矩形宽度: "))
line1 = "#" * width
print(line1)

if width == 1:
    line2 = '#'
else:
    line2 = '#' + ' ' * (width - 2) + '#'

print(line2)
print(line2)
print(line1)
```

比较运算符

运算符:

" > >= < <= == != "

比较规则:

- 依次按编码值进行两两字符比较,一但不同则比较结果,返回
- 比较结果,当编码值与长度完全相同时,两个字符串相同

示例:

```python
'A'   < 'B'     # True
'ABC' > 'ABB'   # True
'ACB' < 'ABC'   # False
'AB'  <= 'ABC'  # True
'ABC' == 'abc'  # False
```

成员运算符

运算符:

"in" / "not in"

作用:

in 用于序列,字典,集合中,用于判断某个值是否存在于容器中。如果存在返回True, 否则返回False

格式:

字符串对象 in 序列

示例:

```
x = 'welcome to tarena!'
'to' in x  # True
'china' in x  # False
```

序列操作

本节课重点：观看视频 Python01阶段 DAY03 "04-字符串03" 时间：00:06:07 - 00:17:24

字符串就是序列，字面意思(有序排序)

Python 字符串是不可变的序列

不可变是指一旦创建将不可改变

序列是指有先后顺序关系有序排列

序列长度 len

求序列长度的函数 len(x)

对于字符串，len(str) 返回字符串的个数

```
# 示例:
s = 'abcd'
len(x)  # 返回4
```

练习

试读出如下字符串含有几个字符? '123abcd' '5\'4"' '\"A\x34\056' '\a\bc\td\n'

答案

```
print(len('123abcd')) # 7
print(len('5\'4"'))    # 4
print(len('\"A\x34\056')) # 4
print(len('\a\bc\td\n'))  # 6
```

小测试

```
>>> print(len("sdf//\\sdf\077\x89890\u4578"))
15
>>> print(("sdf//\\sdf\077\x89890\u4578"))
sdf//\sdf?•890藍
>>> print(len(r"sdf//\\sdf\077\x89890\u4578"))
27
>>> print((r"sdf//\\sdf\077\x89890\u4578"))
sdf//\\sdf\077\x89890\u4578
>>>
```

索引 index

本节课重点: 观看视频 Python01阶段 DAY03 "06-字符串索引01" 时间: 00:09:16 - 00:17:24

作用:

从一个字符串中获取其中的一个字符

语法:

字符串[整数表达式]

说明:

python 序列都可以用索引来访问序列中的对象

python序列的**正向**索引是**从0开始的,第二个索引为1**, ... **最后一个索引为len(x)-1**

python的序列的**反向**索引是**从-1开始的,-1代表最后一个**, -2代表最后一个,以此类推,**第一个是 -len(x)**

示例:

```
s = "ABCDE"
s[1]  # 'B'
s[-1]  # 'E'
s[-4]  # 'B'
s[-8]  # 越界错误 IndexError
```

练习:

输入一个字符串,打印如下内容?

1. 打印这个字符串的第一个字符
2. 打印这个字符串的最后一个字符
3. 如果这个字符串的长度是奇数,打印中间这个字符

答案:

```
s = input("请输入字符串: ")
print("第一个字符是:", s[0])
print("最后一个字符是:", s[-1])
if len(s) % 2 == 1:   # if len(s) % 2:
    # center = int((len(s) -1 ) / 2)
    # print(s[center])
    print('中间这个字符是:', s[len(s) // 2])
```

切片 slice

本节课重点: 观看视频 Python01阶段 DAY03 "08-字符串切片01" 时间: 00:00:25 - 00:23:15

作用:

从字符串序列中取出相应的元素,重新组成一个新的字符串

语法:

字符串[(开始索引b) : (结束索引e) (: (步长s))]

注: 小括号() 括起来的部分代表可省略

说明:

开始索引是切片切下的位置,0代表第一个元素,1代表第二个元素..... (与索引相同)，结束索引是切片的终止点(但不包含终止点)

步长是切片每次获取完当前元素后移动的方向和偏移量。没有步长, 相当于**步长为1(默认为1)**

当步长为正整数时，取**正向切片**: **开始索引默认为0**，**结束索引默认是最后一个元素的下一个位置**

当步长为负整数时，取**反向切片**: **开始索引默认为最后一个元素**，**结束索引默认是第一个元素的前一个位置**

示例:

```
s = 'ABCDE'
a = s[1:4]  # 'BCD'
a = s[1:]  # 'BCDE'
a = s[:4]  # 'ABCD'
a = s[:]  # 'ABCDE'  等同于a=s[0:5]
a = s[1:1]  # ''   空字符串
a = s[0::2]  # 'ACE'
a = s[::2]  # 等同于a = s[0:5:2]
a = s[4:0:-1]  # 'EDCB'
a = s[3:0:-2]  # 'DB'
a = s[4::-2]   # 'ECA'
a = s[::-2]   # 'ECA'

正向切片:
mail = 'mail4homework@yeah.net'
>>> mail[:]
'mail4homework@yeah.net'
>>> mail[::1]
'mail4homework@yeah.net'

>>> mail[0:]
'mail4homework@yeah.net'
>>> mail[0:22]
'mail4homework@yeah.net'
>>> mail[0::1]
'mail4homework@yeah.net'
>>> mail[0:22:1]
'mail4homework@yeah.net'


>>> mail[-22:]
'mail4homework@yeah.net'
>>> mail[-22:22]
'mail4homework@yeah.net'
```

```
>>> mail[-22::1]
'mail4homework@yeah.net'
>>> mail[-22:22:1]
'mail4homework@yeah.net'


反向切片:
>>> mail[::-1]
'ten.haey@krowemoh4liam'

>>> mail[21::-1]
'ten.haey@krowemoh4liam'
>>> mail[-1::-1]
'ten.haey@krowemoh4liam'

>>> mail[21:-23:-1]
'ten.haey@krowemoh4liam'
>>> mail[-1:-23:-1]
'ten.haey@krowemoh4liam'
```

练习:

1. 写一个程序,输入一个字符串,把字符串的第一个字符和最后一个字符去掉后, 打印出处理后的字符串
2. 输入任意一个字符串,判断这个字符串是否是回文，回文是指中心对称的文字 如: 上海自来水来自海上 ABCCBA
   随意输入一个字符串,判断是否为回文

答案:

1.

```
s = input('请输入一个比较长的字符串: ')
# s2 = s[1:-1]
s2 = s[1:len(s)-1]
print("结果是:", s2)
```

2.

```
#思路, 原字符串 反转 后等于源字符串,则是回文
s1 = input("请输入字符串: ")
s2 = s1[::-1]  # 反转后用s2绑定
if s1 == s2:
    print(s1, '是回文')
else:
    print(s1, '不是回文')
```

其他序列函数

函数	说明
len(x)	返回容器中数据的个数(长度)
max(x)	返回容器中的最大值元素
min(x)	返回容器中的最小值元素

示例:

```
s = "ABCD1234"
print(len(s))  # 8
print(max(s))  # D
print(min(s))  # 1
```

字符串相关函数

字符串编码转换函数

函数	说明
ord(c)	返回一个字符串的Unicode编码值
chr(i)	返回整数i这个值所对应的字符

示例:

```
print(ord('A'))  # 65
print(ord('中'))  # 20013
```

练习:

1. 写一个程序,输入一段字符串,如果字符串不为空,则把第一个字符的编码打印出来
2. 写一个程序,输入一个整数值(0~65535) 打印出这个数所对应的字符

答案:

1.

```
s = input("请输入字符串: ")
if s != '':
    print("第一个字的编码值是:", ord(s[0]))
```

2.

```
code = int(input("请输入整数(0~65535): "))

ch = chr(code)
print(code, '对应的字符是:', ch)
```

整数转换为字符串函数

函数	说明
bin(i)	将整数转为二进制字符串
oct(i)	将整数转为八进制字符串
hex(i)	将整数转为十六进制字符串

示例:

```
x = 1234
print(bin(x))
print(oct(x))
print(hex(x))
```

字符串的构造函数

函数	说明
str(obj='')	将对象转换为字符串

示例:

```
a = str(100)  # a = '100'
a = str(3.14)  # a = '3.14'
a = str(None)  # a = 'None'
```

查看函数的帮助

```
>>> help(函数名)
>>> help(类型名)
>>> help(对象)
```

常用的字符串方法

语法:

```
对象.方法名(方法传参)
```

示例:

```
'abc'.isalpha()  # 语法是对的
123.isalpha()   # 错的
```

函数说明:

常用的方法	说明
S.isdigit()	判断字符串中的字符是否全为数字
S.isalpha()	判断字符串是否全为英文字母
S.islower()	判断字符串所有字符是否全为小写英文字母
S.isupper()	判断字符串所有字符是否全为大写英文字母
S.isspace()	判断字符串是否全为空白字符
S.center(width[,fill])	将原字符串居中，左右默认填充空格
S.count(sub[, start[,end]])	获取一个字符串中子串的个数
S.find(sub[, start[,end]])	获取字符串中子串sub的索引,失败返回-1
S.strip([chars])	返回去掉左右char字符的字符串(默认char为空白字符)
S.lstrip([chars])	返回去掉左侧char字符的字符串(默认char为空白字符)
S.rstrip([chars])	返回去掉右侧char字符的字符串(默认char为空白字符)
S.upper()	生成将英文转换为大写的字符串
S.lower()	生成将英文转换为小写的字符串
S.replace(old, new[, count])	将原字符串的old用new代替，生成一个新的字符串
S.startswith(prefix[, start[, end]])	返回S是否是以prefix开头，如果以prefix开头返回True,否则返回False,
S.endswith(suffix[, start[, end]])	返回S是否是以suffix结尾，如果以suffix结尾返回True,否则返回False

不常用的方法	说明
S.title()	生成每个英文单词的首字母大写字符串
S.isnumeric()	判断字符串是否全为数字字符

空白符：是指空格,水平制表符(\t),换行符(\n)等不可见的字符。

讲师课后作业

作业

1. 用字符串 * 运算符打印三角形 输入一个整数,此整数代表此三角形左侧预留的字符数，当数字越大时,此三角形越靠右

```
   *
  ***
 *****
*******
```

2. 输入一个字符串,把输入的字符串中的空格全部去掉,打印出处理后的字符串的长度及内容
3. 输入三行文字,让这三行文字在一个方框内居中显示。(不要输入中文)

   如输入: hello! I'm studing python! I like python! 显示如下:

```
      +---------------------+
      |        hello!       |
      | I'm studing python! |
      |   I like python!    |
      +---------------------+
```

答案

1.

```
n = int(input("请输入左侧空格个数: "))

print(' ' * n + "   *")
print(' ' * n + "  ***")
print(' ' * n + " *****")
print(' ' * n + "*******")
```

2.

```
s = input("请输入任意一段字符串: ")

s2 = s.replace(' ', '')
print("去掉空格后的字符串是:", s2)
```

3.

```
s1 = input("请输入第1行: ")
s2 = input("请输入第2行: ")
s3 = input("请输入第3行: ")
max_length = max(len(s1), len(s2), len(s3))

first_line = '+-' + '-' * max_length + '-+'
print(first_line)

# 打印第 2, 3, 4行
print('| ' + s1.center(max_length) + ' |')
print('| ' + s2.center(max_length) + ' |')
print('| ' + s3.center(max_length) + ' |')

# 打印最后一行
print(first_line)
```

DAY04

数据类型（2续）

字符串格式化表达式

作用

生成一定格式的字符串

语法

`%[ (name) ][ flags ][ width ].[ precision ] typecode`

- (name) 可选，用于选择指定的key
- flags 可选，可供选择的值有:
  - "+" 右对齐；正数前加正好，负数前加负号；
  - "-" 左对齐；正数前无符号，负数前加负号；
  - " " 右对齐；正数前加空格，负数前加负号；
  - "0" 右对齐；正数前无符号，负数前加负号；用0填充空白处
- width 可选，占有宽度
- .precision 可选，小数点后保留的位数
- **typecode 必选**

简版： % [- + 0 宽度.精度] 类型码

选项	说明
-	左对齐(默认为右对齐)
+	显示正号
0	左侧空白位置补零
宽度	整个数据输入的宽度
精度	保留小数点后多少位(默认为6位))

示例

格式字符串 % 参数值

格式字符串 % (参数值1, 参数值2, 参数值3)

代码如下：

```python
fmt = "姓名: %s, 年龄: %d"
print(fmt % ('Tarena', 15))  # 姓名: Tarena, 年龄: 15
print(fmt % ('小张', 20))
print("这个学生的成绩是: %d" % 99)
```

```python
a = "%(name)s-----%(age)d "%{'name':'xx','age':20}
print(a) # xx-----20
```

常用类型码

占位符与类型码	说明
%s	转为字符串,使用str(x) 函数转换
%r	转为字符串,使用repr(x) 函数转换
%c	整数转为单个字符
%d	数字转为十进制整数
%o	整数转为八进制整数
%x	整数转为十六进制整数(字符a-f小写)
%X	整数转为十六进制整数(字符A-F大写)
%e	指数浮点数(e小写), 如:2.9e+10
%E	指数浮点数(E大写), 如:2.9E+10
%f, %F	转为十进制浮点数
%g, %G	十进制形式浮点数或指数浮占数自动转换
%%	等同于一个%字符

示例

```python
"%10d" % 123   # '       123'
'%-10d % 123   # '123       '
'%10s' % 'abc' # '       abc'
'%-5s' % 'abc' # 'abc  '
"%+d" % 123    # '+123'
'%+010d' % 123  # '+000000123'
'%f' % 3.1415926535
'%.10f' % 3.1415926535
'%7.2f' % 3.1415926535
```

练习

输入三行文字,让这三行文字依次以 20个字符的宽度右对齐输出

如: 请输入第1行: hello world! 请输入第2行: abcd 请输入第3行: a 输出结果为: hello world! abcd a 做完上面的题后再思考:能否以最长字符串的长度进行右对齐显示(左侧填充空格)

答案

```
s1 = input("请输入第1行: ")
s2 = input("请输入第2行: ")
s3 = input("请输入第3行: ")
# 方法1
# zuida = len(s1)
# if len(s2) > zuida:
#     zuida = len(s2)
# if len(s3) > zuida:
#     zuida = len(s3)
# 方法2
zuida = max(len(s1), len(s2), len(s3))
print("最长的字符串长度是:", zuida)

# 右对齐方法1
# print(' ' * (zuida-len(s1)) + s1)
# print(' ' * (zuida-len(s2)) + s2)
# print(' ' * (zuida-len(s3)) + s3)

# 右对齐方法2
fmt = "%" + str(zuida) + "s"

print(fmt % s1)
print(fmt % s2)
print(fmt % s3)

########################################

s1 = input("请输入第1行: ")
s2 = input("请输入第2行: ")
s3 = input("请输入第3行: ")

print("%20s" % s1)
print("%20s" % s2)
print("%20s" % s3)
```