# Python核心数据类型-字符串

一.字符串特性

```
1. 序列类型, 支持索引 分片 + * 等运算符操作
2. 不可变性, 字符串不可以在远处修改
```

In [2]:
```
x = 'apple'
y = 'banana'

print x[0]
print x[1:3]

print x + " " + y
print x * 4
```

```
a
pp
apple banana
appleappleappleapple
```

二. 字符串常用方法

In [2]:
```
print(dir('apple'))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__',
'__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_
ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswi
th', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'iss
pace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip',
'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

In [44]:
```
#1.strip() 删除字符串两边的指定字符, 默认删除空白符
help(x.strip)

x = ' apple  '
y = x.strip()
print y

x = '$$appl$e$$$'
y = x.strip('$')
print y
```

```
Help on built-in function strip:

strip(...)
    S.strip([chars]) -> string or unicode

    Return a copy of the string S with leading and trailing
    whitespace removed.
    If chars is given and not None, remove characters in chars instead.
    If chars is unicode, S will be converted to unicode before stripping

apple
appl$e
```

In [46]:
```
#2.lstrip() 删除字符串左边的指定字符, 默认删除空白符
help(x.lstrip)
x = ' apple   '
y = x.lstrip()
print y

x = '||a|pple|||'
y = x.lstrip('|')
print y
```

```
Help on built-in function lstrip:

lstrip(...)
    S.lstrip([chars]) -> string or unicode

    Return a copy of the string S with leading whitespace removed.
    If chars is given and not None, remove characters in chars instead.
    If chars is unicode, S will be converted to unicode before stripping

apple
a|pple|||
```

In [49]:
```
#3.rstrip()
help(x.rstrip)

x = ' apple   '
y = x.rstrip()
print y

x = '||a|pple|||'
y = x.rstrip('|')
print y
```

```
Help on built-in function rstrip:

rstrip(...)
    S.rstrip([chars]) -> string or unicode

    Return a copy of the string S with trailing whitespace removed.
    If chars is given and not None, remove characters in chars instead.
    If chars is unicode, S will be converted to unicode before stripping

 apple
||a|pple
```

In [53]:
```
#4.split() 以指定的字符从左向右切割字符串, 默认用空格切割, 如果指定了切割的最大长度, 则切割到指定的位置, 如未指定 则切割到字符串结尾, 返回一个列表
help(x.split)

x = 'a b c d e f g'
s = x.split()
print s

s = x.split(" ", 2)
print s

x = 'a|b|c|d|e'
s = x.split('|')
print s
s = x.split('|', 3)
print s
```

```
Help on built-in function split:

split(...)
    S.split([sep [,maxsplit]]) -> list of strings

    Return a list of the words in the string S, using sep as the
    delimiter string.  If maxsplit is given, at most maxsplit
    splits are done. If sep is not specified or is None, any
    whitespace string is a separator and empty strings are removed
    from the result.

['a', 'b', 'c', 'd', 'e', 'f', 'g']
['a', 'b', 'c d e f g']
['a', 'b', 'c', 'd', 'e']
['a', 'b', 'c', 'd|e']
```

In [54]:
```
#5.rsplit() 以指定的字符从左向右切割字符串, 默认用空格切割, 如果指定了切割的最大长度, 则切割到指定的位置, 如未指定 则切割到字符串结尾, 返回一个列表

help(x.rsplit)

x = 'a b c d e f g'
s = x.rsplit()
print s

s = x.rsplit(" ", 2)
print s

x = 'a|b|c|d|e'
s = x.rsplit('|')
print s
s = x.rsplit('|', 3)
print s
```

```
Help on built-in function rsplit:

rsplit(...)
    S.rsplit([sep [,maxsplit]]) -> list of strings

    Return a list of the words in the string S, using sep as the
    delimiter string, starting at the end of the string and working
    to the front.  If maxsplit is given, at most maxsplit splits are
    done. If sep is not specified or is None, any whitespace string
    is a separator.

['a', 'b', 'c', 'd', 'e', 'f', 'g']
['a b c d e', 'f', 'g']
['a', 'b', 'c', 'd', 'e']
['a|b', 'c', 'd', 'e']
```

In [56]:
```
#6.splitlines() 把字符串以换行符作为分割点, 进行切割
help("".splitlines)

x = 'a\nb\nc'
print x.splitlines()

#等价于

print x.split('\n')


#7.partition() 以指定的字符从左向右第一次出现的位置作为分割点, 把字符串分成三份[分割字符左边字符串,分割字符, 分割字符右边字符串]
help(x.partition)

x = 'apple'
print x.partition('p')
```

```
Help on built-in function partition:

partition(...)
    S.partition(sep) -> (head, sep, tail)

    Search for the separator sep in S, and return the part before it,
    the separator itself, and the part after it.  If the separator is not
    found, return S and two empty strings.

('a', 'p', 'ple')
```

In [57]:
```
#8.rpartition() 以指定的字符从右向左第一次出现的位置作为分割点, 把字符串分成三份[分割字符左边字符串,分割字符, 分割字符右边字符串]
help(x.rpartition)

x = 'apple'
print x.rpartition('p')
```

```
Help on built-in function rpartition:

rpartition(...)
    S.rpartition(sep) -> (head, sep, tail)

    Search for the separator sep in S, starting at the end of S, and return
    the part before it, the separator itself, and the part after it.  If the
    separator is not found, return two empty strings and S.

('ap', 'p', 'le')
```

In [61]:
```
#9.join() 将一个全部元素为字符串的可迭代对象, 用指定字符拼接成一个大字符串
help("".join)
x = ["a", 'b', 'c']
print '|'.join(x)

y = 'apple'
print '|'.join(y)
```

```
Help on built-in function join:

join(...)
    S.join(iterable) -> string

    Return a string which is the concatenation of the strings in the
    iterable.  The separator between elements is S.

a|b|c
a|p|p|l|e
```

In [75]:
```
#10.startswith() 判断字符串是否以某个字符或字符串开头
help(x.startswith)

x = 'apple'
print x.startswith('a')
print x.startswith('ap')
print x.startswith('ab')
```

```
Help on built-in function startswith:

startswith(...)
    S.startswith(prefix[, start[, end]]) -> bool

    Return True if S starts with the specified prefix, False otherwise.
    With optional start, test S beginning at that position.
    With optional end, stop comparing S at that position.
    prefix can also be a tuple of strings to try.

True
True
False
```

In [77]:
```
#11.endswith() 判断字符串是否以某个字符或字符串结尾
help(x.endswith)

x = 'apple'
print x.endswith('e')
print x.endswith('le')
print x.endswith('pe')
```

```
Help on built-in function endswith:

endswith(...)
    S.endswith(suffix[, start[, end]]) -> bool

    Return True if S ends with the specified suffix, False otherwise.
    With optional start, test S beginning at that position.
    With optional end, stop comparing S at that position.
    suffix can also be a tuple of strings to try.

True
True
False
```

In [85]:
```
#12.format() 拼接字符串
help(x.format)

x = 'the name is {name} and age is {age}'
y = x.format(age=20, name="lilei")
print y

#或

x = 'the name is %s and age is %s'
y = x %("liushuo", 18)
print y

x = 'the name is %s and age is %s'
y = x %('22', "liushuo")
print y

x = 'the name is %s and age is %s'
y = x %('22')
print y

#使用第二种形式拼接字符串, 要注意传入参数的位置, 以及参数的数量
```

```
Help on built-in function format:

format(...)
    S.format(*args, **kwargs) -> string

    Return a formatted version of S, using substitutions from args and kwargs.
    The substitutions are identified by braces ('{' and '}').

the name is lilei and age is 20
the name is liushuo and age is 18
the name is 22 and age is liushuo

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-85-47ac47bfb07f> in <module>()
     17 
     18 x = 'the name is %s and age is %s'
---> 19 y = x %('22')
     20 print y
     21 

TypeError: not enough arguments for format string
```

In [87]:
```
#13.lower() 将字符串中的所有字符变成小写
help(x.lower)

x = 'AppLe'
y = x.lower()
print y
```

```
Help on built-in function lower:

lower(...)
    S.lower() -> string

    Return a copy of the string S converted to lowercase.

apple
```

In [88]:
```
#14.islower() 判断是一个字符串中的所有字符，是否都为小写
help(x.islower)

x = 'AppLe'
print x.islower()
y = x.lower()
print y.islower()
```

```
Help on built-in function islower:

islower(...)
    S.islower() -> bool

    Return True if all cased characters in S are lowercase and there is
    at least one cased character in S, False otherwise.

False
True
```

In [89]:
```
#15.upper() 将字符串中的左右字符变成大写
x = 'aPple'
y = x.upper()
print y
```

```
APPLE
```

In [90]:
```
#16.isupper() 判断字符串中的所有字符是否都为大写
x = 'aPpLE'
print x.isupper()

y = x.upper()
print y.isupper()
```

```
False
True
```

In [91]:
```
#17.title() 将字符串转换为title样式
help(x.title)
x = 'my name is liushuo'
y = x.title()
print y
```

```
Help on built-in function title:

title(...)
    S.title() -> string

    Return a titlecased version of S, i.e. words start with uppercase
    characters, all remaining cased characters have lowercase.

My Name Is Liushuo
```

In [93]:
```
#18.istitle() 判断字符串是否是title样式

x = 'my Name is Liushuo'
print x.istitle()

y = x.title()
print y.istitle()
```

```
False
True
```

In [94]:
```
#19.capitalize() 将字符串的第一个单词的首字母变成大写

x = 'my name is liushuo'
y = x.capitalize()
print y
```

```
My name is liushuo
```

In [98]:
```
#20.isalnum() 判断字符串至少有一个字符，且每个字符都是字母或数字时返回true
help(x.isalnum)
x = 'apple1'
y = 'apple!'
z = ""
w = '123'
k = 'apple'
print x.isalnum()
print y.isalnum()
print z.isalnum()
print w.isalnum()
print k.isalnum()
```

```
Help on built-in function isalnum:

isalnum(...)
    S.isalnum() -> bool

    Return True if all characters in S are alphanumeric
    and there is at least one character in S, False otherwise.

True
False
False
True
True
```

In [96]:
```
#21.isdigit() 判断字符串中的所有字符是否都为数字
help(x.isdigit)

x = '123'
print x.isdigit()

x = '123abc'
print x.isdigit()
```

```
Help on built-in function isdigit:

isdigit(...)
    S.isdigit() -> bool

    Return True if all characters in S are digits
    and there is at least one character in S, False otherwise.

True
False
```

In [99]:
```
#22.isalpha() 判断字符串至少有一个字符且所有字符都是字母时，返回True

x = ""
y = 'apple1'
z = 'apple!'
w = 'apple'

print x.isalpha()
print y.isalpha()
print z.isalpha()
print w.isalpha()
```

```
False
False
False
True
```

In [104]:
```
#23.isspace() 判断字符串中是否都是空白字符，如果是则返回true
help(x.isspace)

x = ' \t\n'
print x.isspace()

x ='a  \t'
print x.isspace()
```

```
Help on built-in function isspace:

isspace(...)
    S.isspace() -> bool

    Return True if all characters in S are whitespace
    and there is at least one character in S, False otherwise.

True
False
```

In [103]:
```
#24.translate() 字符翻译
import string
inattr = 'aeiou'
outattr = '12345'
trans = string.maketrans(inattr, outattr)
test = 'apple'
print test.translate(trans)
```

```
1ppl2
```

In [106]:
```
#25.find() 在字符串中查找某个字符，如果存在返回它第一次出现时的从左到右的索引位置，如果不存在返回-1

x = 'apple'
print x.find('p')
print x.find('o')
```

```
1
-1
```

In [107]:
```
#26.rfind() 在字符串中从右向左查找某个字符，如果存在返回它第一次出现时的从左到右的索引位置，如果不存在返回-1
x = 'apple'
print x.rfind('p')
print x.rfind('z')
```

```
2
-1
```

In [108]:
```
#27.index() 在字符串中查找某个字符，如果存在返回它第一次出现时的从左到右的索引位置，如果不存在代码报错
x = 'apple'
print x.index('p')
print x.index('o')
```

```
1

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-108-b884211bbf13> in <module>()
      2 x = 'apple'
      3 print x.index('p')
----> 4 print x.index('o')

ValueError: substring not found
```

In [110]:
```
#28.rindex() 在字符串中从右向左查找某个字符，如果存在返回它第一次出现时的从左到右的索引位置，如果不存在代码报错
x = 'apple'
print x.rindex('p')
print x.rindex('o')
```

```
2

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-110-32b063f4a730> in <module>()
      2 x = 'apple'
      3 print x.rindex('p')
----> 4 print x.rindex('o')

ValueError: substring not found
```

In [116]:
```
#29.center() 指定一个字符串长度，将原字符居中，两侧用指定字符做填充，直至填充到你所指定的长度为止，默认用空格符填充
help(x.center)

x = 'apple'
x.center(9, '|')

x = 'apple'
x.center(9, "#")
```

```
Help on built-in function center:

center(...)
    S.center(width[, fillchar]) -> string

    Return S centered in a string of length width. Padding is
    done using the specified fill character (default is a space)
```

Out[116]:
```
'##apple##'
```

In [119]:
```
#30.ljust() 指定一个最终的字符串长度，原字符串左对齐，剩下的位置用指定的字符填充，默认用空格符填充
help(x.ljust)

x = 'apple'
y = x.ljust(10, '|')
print y
```

```
Help on built-in function ljust:

ljust(...)
    S.ljust(width[, fillchar]) -> string

    Return S left-justified in a string of length width. Padding is
    done using the specified fill character (default is a space).

apple|||||
```

In [121]:
```
#31.rjust() 指定一个最终的字符串长度，原字符串右对齐，剩下的位置用指定的字符填充，默认用空格符填充

x = 'apple'
y = x.rjust(10, '&')
print y
```

```
&&&&&apple
```

In [123]:
```
#32.swapcase() 将原字符串的大小写字母做调换
help(x.swapcase)

x = 'appLe'
y = x.swapcase()
print y
```

```
Help on built-in function swapcase:

swapcase(...)
    S.swapcase() -> string

    Return a copy of the string S with uppercase characters
    converted to lowercase and vice versa.

APPlE
```

In [125]:
```
#33.count() 返回一个字符在字符串中出现的次数

x = 'apple'
print x.count('p')
print x.count('a')
```

```
2
1
```

In [139]:
```
#34.expandtabs() 将Tab转换成空格，默认1个tab等于8个空格，1个tab等于几个空格可以手工指定
help(x.expandtabs)

x = '\tapple'
y = x.expandtabs()
print len(y), y

z = x.expandtabs(12)
print len(z), z
```

```
Help on built-in function expandtabs:

expandtabs(...)
    S.expandtabs([tabsize]) -> string

    Return a copy of S where all tab characters are expanded using spaces.
    If tabsize is not given, a tab size of 8 characters is assumed.

13          apple
17              apple
```

In [129]:
```
#35.replace() 将字符串中的一个字符替换为另一个字符，默认替换所有，可以指定替换的数量
help(x.replace)
x = 'apple'
y = x.replace('p', 'w')
print y

z = x.replace('p', 'w', 1)
print z
```

```
Help on built-in function replace:

replace(...)
    S.replace(old, new[, count]) -> string

    Return a copy of string S with all occurrences of substring
    old replaced by new.  If the optional argument count is
    given, only the first count occurrences are replaced.

awwle
awple
```

In [141]:
```
#36.encode()  指定一个字符串的编码方案，对字符串做编码，errors代表设置不同的错误处理方案
help(x.encode)

x = 'apple banana'
print x.encode("base64", "strict")
```

```
Help on built-in function encode:

encode(...)
    S.encode([encoding[,errors]]) -> object

    Encodes S using the codec registered for encoding. encoding defaults
    to the default encoding. errors may be given to set a different error
    handling scheme. Default is 'strict' meaning that encoding errors raise
    a UnicodeEncodeError. Other possible values are 'ignore', 'replace' and
    'xmlcharrefreplace' as well as any other name registered with
    codecs.register_error that is able to handle UnicodeEncodeErrors.

YXBwbGUgYmFuYW5h
```

In [154]:
```
#37.decode() 对字符串进行解码
help(x.decode)

x = 'apple'
y = x.encode('base64', 'strict')
print y

z = y.decode('base64', 'strict')
print z
```

```
Help on built-in function decode:

decode(...)
    S.decode([encoding[,errors]]) -> object

    Decodes S using the codec registered for encoding. encoding defaults
    to the default encoding. errors may be given to set a different error
    handling scheme. Default is 'strict' meaning that encoding errors raise
    a UnicodeDecodeError. Other possible values are 'ignore' and 'replace'
    as well as any other name registered with codecs.register_error that is
    able to handle UnicodeDecodeErrors.

YXBwbGU=

apple
```

In [1]:
```
#38.zfill()  指定一个长度，原字符右对齐，剩下的位置用0填充
help(x.zfill)

x = 'apple'
y = x.zfill(10)
print y
```

```
  File "<ipython-input-1-5626a9c170ea>", line 6
    print y
          ^
SyntaxError: Missing parentheses in call to 'print'
```

In [6]:
```
#以下方法为Python3中新增
#39.isidentifier 判断一个字符串是否可以成为变量名，已知变量名不能以数字开头
x = 'apple'
y = '1apple'
help(x.isidentifier)

print(x.isidentifier())
print(y.isidentifier())
```

```
Help on built-in function isidentifier:

isidentifier(...) method of builtins.str instance
    S.isidentifier() -> bool

    Return True if S is a valid identifier according
    to the language definition.

    Use keyword.iskeyword() to test for reserved identifiers
    such as "def" and "class".

True
False
```

In [8]:
```
#40.isdecimal 判断一个字符串中的内容是否是十进制的
x = '123'
y = '0x0001'
help(x.isdecimal)
print(x.isdecimal())
print(y.isdecimal())
```

```
Help on built-in function isdecimal:

isdecimal(...) method of builtins.str instance
    S.isdecimal() -> bool

    Return True if there are only decimal characters in S,
    False otherwise.

True
False
```

In [10]:
```
#41.isprintable 判断一个字符串是否可以原样输出
x = 'apple'
y = 'apple\n'
help(x.isprintable)
print(x.isprintable())
print(y.isprintable())
```

```
Help on built-in function isprintable:

isprintable(...) method of builtins.str instance
    S.isprintable() -> bool

    Return True if all characters in S are considered
    printable in repr() or S is empty, False otherwise.

True
False
```

In [12]:
```
#42.format_map 传入字典格式化字符串
x = '{name} {age}'
help(x.format_map)
x.format_map({"name": 'liu', 'age': 20})
'liu 20'
```

```
Help on built-in function format_map:

format_map(...) method of builtins.str instance
    S.format_map(mapping) -> str

    Return a formatted version of S, using substitutions from mapping.
    The substitutions are identified by braces ('{' and '}').
```

Out[12]:
```
'liu 20'
```

In [14]:
```
#43. casefold 与 lower功能类似，将字符转换成小写，支持多国文字

x = 'APPLE'
y = "ß"  #已知德文的小写是 ss

print(x.lower())
print(y.lower())

print(x.casefold())
print(y.casefold())
```

```
apple
ß
apple
ss
```

In [1]:
```
#44.isnumeric 与isdigit类似，判断字符串中的字符是否都是数字，支持多种语言 如罗马数字等.
x = '123'
y = '四'
print(x.isnumeric())
print(y.isnumeric())
```

```
True
True
```

In [1]:
```
#45.maketrans与translate python3

x = 'aeiou'
y = '12345'
t = x.maketrans(dict(zip(x, y)))
print(t)
```

```
{97: '1', 101: '2', 105: '3', 111: '4', 117: '5'}
```

In [5]:
```
y = 'apple'
print(y.translate(t))
```

```
1ppl2
```

In [ ]:
```
#46.maketrans与translate python2

x = 'aeiou'
y = '12345'
import string
t = string.maketrans(x, y)
z = 'apple'
z.translate(t)
'1ppl2'
```