In [2]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

1 字符串序列操作

- 大多数 Pandas 的字符串方法借鉴了 Python 字符串内建函数的内容

In [3]:
```
import pandas as pd
import numpy as np
s = pd.Series(['Lower', 'CAPITALS', 'this is A sentence', 'SwApCaSe'])
s
```

Out[3]:
```
0               Lower
1            CAPITALS
2    this is A sentence
3            SwApCaSe
dtype: object
```

In [4]:
```
s1='aVD'
s1.lower()  #单个字符串
```

Out[4]:
```
'avd'
```

In [5]:
```
s.map(lambda x:x.lower()) #方法1
```

Out[5]:
```
0               lower
1            capitals
2    this is a sentence
3            swapcase
dtype: object
```

In [7]:
```
1 s
```

Out[7]:
```
0                Lower
1            CAPITALS
2    this is A sentence
3            SwApCaSe
dtype: object
```

In [6]:
```
1 s.str.lower()        #方法2
```

Out[6]:
```
0                lower
1            capitals
2    this is a sentence
3            swapcase
dtype: object
```

1.1 计算长度

In [9]:
```
1 import pandas as pd
2 import numpy as np
3 s = pd.Series(['lower', 'CAPITALS', 'this is a sentence', 'SwApCaSe'])
4 s
```

Out[9]:
```
0                lower
1            CAPITALS
2    this is a sentence
3            SwApCaSe
dtype: object
```

In [10]:
```
1 s.str.len() #字符串长度,向量化操作，优先考虑这种
```

Out[10]:
```
0     5
1     8
2    18
3     8
dtype: int64
```

In [11]:
```
1 s.map(lambda x:len(x))
```

Out[11]:
```
0     5
1     8
2    18
3     8
dtype: int64
```

In [ ]:
```
1
```

1.2 大小写转换

- .str.lower()和.str.upper()
- .str.title()和.str.capitalize()
- .str.swapcase()

In [13]:
```
1 s = pd.Series(['lower', 'CAPITALS', 'this is a sentence', 'SwApCaSe'])
2 s
```

Out[13]:
```
0                 lower
1             CAPITALS
2    this is a sentence
3             SwApCaSe
dtype: object
```

In [14]:
```
1 s.str.lower()  #转换为小写字母
```

Out[14]:
```
0                 lower
1             capitals
2    this is a sentence
3             swapcase
dtype: object
```

In [15]:
```
1 s.str.upper() #转换为大写字母
```

Out[15]:
```
0                LOWER
1              CAPITALS
2    THIS IS A SENTENCE
3              SWAPCASE
dtype: object
```

In [16]:
```
1 s
```

Out[16]:
```
0                lower
1              CAPITALS
2    this is a sentence
3              SwApCaSe
dtype: object
```

In [17]:
```
1 s.str.title() #将每个单词的首字母大写，把其他的小写
```

Out[17]:
```
0                Lower
1              Capitals
2    This Is A Sentence
3              Swapcase
dtype: object
```

In [ ]:
```
1
```

In [18]:
```
1 s.str.capitalize() #将字符串的首字母大写，其他小写
```

Out[18]:
```
0                Lower
1              Capitals
2    This is a sentence
3              Swapcase
dtype: object
```

In [19]:
```
1 s
```

Out[19]:
```
0                lower
1            CAPITALS
2    this is a sentence
3            SwApCaSe
dtype: object
```

In [20]:
```
1 s.str.swapcase()  #大小写互换
```

Out[20]:
```
0                LOWER
1            capitals
2    THIS IS A SENTENCE
3            sWaPcAsE
dtype: object
```

In [21]:
```
1 s
```

Out[21]:
```
0                lower
1            CAPITALS
2    this is a sentence
3            SwApCaSe
dtype: object
```

In [22]:
```
1 s.str.title().str.swapcase() #方法可以级联
```

Out[22]:
```
0                lOWER
1            cAPITALS
2    tHIS iS a sENTENCE
3            sWAPCASE
dtype: object
```

1.3 字符检索

1.3.1 get

In [23]:
```
#获取字符串中单个元素
s = pd.Series(['a_b_c','c_ed_e',np.nan,'f_g_h'])
s
```

Out[23]:
```
0    a_b_c
1    c_ed_e
2       NaN
3    f_g_h
dtype: object
```

In [24]:
```
s.str.get(0)          # get() 获取指定位置的字符串,只能获取单个    缺失值自动忽略
```

Out[24]:
```
0      a
1      c
2    NaN
3      f
dtype: object
```

In [25]:
```
s.str.get(-1)#     get() 获取指定位置的字符串
```

Out[25]:
```
0      c
1      e
2    NaN
3      h
dtype: object
```

1.3.2 slice

In [26]:
```
s = pd.Series(['abc','cede','abcd','fgh'])
s
```

Out[26]:
```
0    abc
1    cede
2    abcd
3    fgh
dtype: object
```

In [27]:
```
1 s.str.slice(0,2)     #切片
```

Out[27]:
```
0    ab
1    ce
2    ab
3    fg
dtype: object
```

In [28]:
```
1 s.map(lambda x:x[0:2])  #函数的形式
```

Out[28]:
```
0    ab
1    ce
2    ab
3    fg
dtype: object
```

In [29]:
```
1 s.str.slice(-2,)   #获取最后两个元素
```

Out[29]:
```
0    bc
1    de
2    cd
3    gh
dtype: object
```

In [ ]:
```
1
```

1.3.3 字符的提取：findall，extract

In [30]:
```
s = pd.Series(['a4783579数据','ce4爬虫',np.nan,'fg57清洗85h'])
s
```

Out[30]:
```
0    a4783579数据
1         ce4爬虫
2             NaN
3     fg57清洗85h
dtype: object
```

In [31]:
```
s.str.findall("[a-z]+")#找所有的，返回的是列表形式    正则表达式
#找每个元素中所有的a到z的字母  中间有被隔开的,就用逗号分开写,形成的是一个组合的形式
```

Out[31]:
```
0        [a]
1       [ce]
2        NaN
3    [fg, h]
dtype: object
```

In [32]:
```
s
```

Out[32]:
```
0    a4783579数据
1         ce4爬虫
2             NaN
3     fg57清洗85h
dtype: object
```

In [33]:
```
s.str.findall("[0-9]+")#提取数字
```

Out[33]:
```
0    [4783579]
1          [4]
2          NaN
3     [57, 85]
dtype: object
```

In [34]:
```
1 s.str.findall("\d+")#提取数字
```

Out[34]:
```
0    [4783579]
1          [4]
2          NaN
3     [57, 85]
dtype: object
```

In [35]:
```
1 s = pd.Series(['a47ff张三835798','c_e46李四765d_e',np.nan,'f_g5王老五785_h'])
2 s
```

Out[35]:
```
0    a47ff张三835798
1    c_e46李四765d_e
2                NaN
3     f_g5王老五785_h
dtype: object
```

In [36]:
```
1 s.str.findall('[\u4e00-\u9fa5]+').str.get(0)  #中文匹配
```

Out[36]:
```
0     张三
1     李四
2    NaN
3     王老五
dtype: object
```

1.4 索引操作

1.4.1 find和index

In [37]:
```
s = pd.Series(['abc','cecde','abcd','fgh'])
s
```

Out[37]:
```
0     abc
1    cecde
2     abcd
3      fgh
dtype: object
```

In [38]:
```
s.str.find('c')      #从左往右找
#返回要找的字母在该字符串里的对应的索引值,
#如果字符串中不含这个字母,就是找不到返回-1,如果字符串中的这个字母有重复,则默认从左到右找首次出现的并返回其索引值
```

Out[38]:
```
0    2
1    0
2    2
3   -1
dtype: int64
```

In [39]:
```
s
```

Out[39]:
```
0     abc
1    cecde
2     abcd
3      fgh
dtype: object
```

In [40]:
```
s.str.rfind('c') #从右边往左边开始找
```

Out[40]:
```
0    2
1    2
2    2
3   -1
dtype: int64
```

In [41]:
```
1 s
```

Out[41]:
```
0      abc
1    cecde
2     abcd
3      fgh
dtype: object
```

In [42]:
```
1 s.str.index('c') #s.str.index  返回的是索引,找不到报错
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_45072/2786631781.py in <module>
----> 1 s.str.index('c') #s.str.index  返回的是索引,找不到报错

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\strings\accessor.py in wrapper(self, *args, **kwargs)
    114                 )
    115                 raise TypeError(msg)
--> 116             return func(self, *args, **kwargs)
    117
    118         wrapper.__name__ = func_name

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\strings\accessor.py in index(self, sub, start, end)
   2643             raise TypeError(msg)
   2644
-> 2645         result = self._data.array._str_index(sub, start=start, end=end)
   2646         return self._wrap_result(result, returns_string=False)
   2647

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\strings\object_array.py in _str_index(self, sub, start, end)
    260         else:
    261             f = lambda x: x.index(sub, start, end)
--> 262         return self._str_map(f, dtype="int64")
    263
    264     def _str_rindex(self, sub, start=0, end=None):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\strings\object_array.py in _str_map(self, f, na_value, dtype,
convert)
     67         map_convert = convert and not np.all(mask)
     68         try:
---> 69             result = lib.map_infer_mask(arr, f, mask.view(np.uint8), map_convert)
     70         except (TypeError, AttributeError) as e:
     71             # Reraise the exception if callable `f` got wrong number of args.

C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\lib.pyx in pandas._libs.lib.map_infer_mask()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\strings\object_array.py in <lambda>(x)
    259             f = lambda x: x.index(sub, start, end)
    260         else:
```

```
--> 261             f = lambda x: x.index(sub, start, end)
    262         return self._str_map(f, dtype="int64")
    263

ValueError: substring not found
```

In [ ]:
```
1
```

1.5 字符类型判断

- .str.islower()和.str.isupper()
- .str.isnumeric()、.str.isalnum()、.str.isdecimal()、.str.isalpha()、.str.isdigit()
- .str.isspace()
- .str.istitle()

In [43]:
```
1 s = pd.Series(['lower', 'CAPITALS', 'this is a sentence', 'SwApCaSe','21',' ','12a',])
2 s
```

Out[43]:
```
0              lower
1            CAPITALS
2  this is a sentence
3            SwApCaSe
4                  21
5
6                 12a
dtype: object
```

In [44]:
```
1 s.str.islower() #是不是都是小写
```

Out[44]:
```
0     True
1    False
2     True
3    False
4    False
5    False
6     True
dtype: bool
```

In [45]:
```
1 s.str.isupper()#是不是都是大写
```

Out[45]:
```
0    False
1     True
2    False
3    False
4    False
5    False
6    False
dtype: bool
```

In [46]:
```
1 s.str.isspace()  #是不是空格
```

Out[46]:
```
0    False
1    False
2    False
3    False
4    False
5     True
6    False
dtype: bool
```

In [47]:
```
s.str.isnumeric() #是不是数字
```

Out[47]:
```
0    False
1    False
2    False
3    False
4     True
5    False
6    False
dtype: bool
```

1.6 字符判断

1.6.1 startswith，endswith

In [48]:
```
s = pd.Series(['bat', 'Bear', 'cat', np.nan])
s.str.startswith('b')  #以什么开头
```

Out[48]:
```
0     True
1    False
2    False
3      NaN
dtype: object
```

In [49]:
```
s.str.endswith('t')  #以什么结尾
```

Out[49]:
```
0     True
1    False
2     True
3      NaN
dtype: object
```

1.6.2 contains

In [50]:
```
1 s
```

Out[50]:
```
0     bat
1    Bear
2     cat
3     NaN
dtype: object
```

In [51]:
```
1 s.str.contains('t')  #contains() 是否包含表达式
```

Out[51]:
```
0     True
1    False
2     True
3      NaN
dtype: object
```

In [52]:
```
1 s = pd.Series(['a47ff张三835798','c_e46lisi765d_e',np.nan,'f_g5王老五785_h'])
2 s
```

Out[52]:
```
0      a47ff张三835798
1    c_e46lisi765d_e
2                NaN
3       f_g5王老五785_h
dtype: object
```

In [53]:
```
1 s.str.contains('[\u4e00-\u9fa5]+')  #是不是有中文字符
```

Out[53]:
```
0     True
1    False
2      NaN
3     True
dtype: object
```

In [ ]:
```
1
```

1.7 字符对齐与填充

- .str.center()
- .str.ljust()和.str.rjust()

In [54]:
```
s = pd.Series(['bat', 'Bear', 'cat', np.nan])
s.str.ljust(10,"*")
```

Out[54]:
```
0    bat*******
1    Bear******
2    cat*******
3           NaN
dtype: object
```

In [55]:
```
s.str.center(10,"*")
```

Out[55]:
```
0    ***bat****
1    ***Bear***
2    ***cat****
3           NaN
dtype: object
```

In [57]:
```
s.str.rjust(10,"*")
```

Out[57]:
```
0    *******bat
1    ******Bear
2    *******cat
3           NaN
dtype: object
```

In [56]:
```
s = pd.Series(['bat', 'Bear', 'cat', np.nan])
s.str.pad(10, side="left",fillchar="*")#pad() 左补齐  相当于右对齐
```

Out[56]:
```
0    *******bat
1    ******Bear
2    *******cat
3           NaN
dtype: object
```

In [58]:
```
s.str.pad(10, side="right", fillchar="*")#右补齐   相当于左对齐
```

Out[58]:
```
0    bat*******
1    Bear******
2    cat*******
3           NaN
dtype: object
```

In [59]:
```
s
```

Out[59]:
```
0     bat
1    Bear
2     cat
3     NaN
dtype: object
```

In [60]:
```
s.str.zfill(10) #zfill() 左边补0
```

Out[60]:
```
0    0000000bat
1    000000Bear
2    0000000cat
3           NaN
dtype: object
```

In [ ]:

In [ ]:

1.8 字符整理

strip

In [61]:
```
s = pd.Series([' bat ', ' Be*ar * ', 'cat ** ', np.nan])
s
```

Out[61]:
```
0         bat
1     Be*ar *
2      cat **
3         NaN
dtype: object
```

In [66]:
```
s.str.strip()  #括号里不写参数默认去掉空格
```

Out[66]:
```
0        bat
1    Be*ar *
2     cat **
3        NaN
dtype: object
```

In [62]:
```
s.str.strip(' *') #首尾字符交替去除     去掉空格和*
```

Out[62]:
```
0      bat
1    Be*ar
2      cat
3      NaN
dtype: object
```

In [ ]:
```

```

1.9 替换

replace

In [67]:
```
s=pd.Series(['foo', 'fuz', 'f.z',np.nan])
s.replace('f', 'b') #只能实现整体替换，不能实现单个字符的替换
```

Out[67]:
```
0    foo
1    fuz
2    f.z
3    NaN
dtype: object
```

In [68]:
```
s.str.replace('f', 'b') #可以实现单个字符串的替换
```

Out[68]:
```
0    boo
1    buz
2    b.z
3    NaN
dtype: object
```

In [69]:
```
s
```

Out[69]:
```
0    foo
1    fuz
2    f.z
3    NaN
dtype: object
```

In [70]:
```
s.str.replace('f.', 'ba', regex=False) #默认是False,意思是不开启正则表达式模式
```

Out[70]:
```
0    foo
1    fuz
2    baz
3    NaN
dtype: object
```

In [71]:
```
s.str.replace('f.', 'ba', regex=True) #开启正则表达式，正则表达式中的 点 表示任意字符
```

Out[71]:
```
0    bao
1    baz
2    baz
3    NaN
dtype: object
```

In [72]:
```
s=pd.Series(['fo844o', 'f012uz', 'f4895z','tuitu000'])
s   #把字符串序列中的数字去掉
```

Out[72]:
```
0      fo844o
1      f012uz
2      f4895z
3    tuitu000
dtype: object
```

In [73]:
```
s.str.replace('[0-9]+', '', regex=True)
```

Out[73]:
```
0      foo
1      fuz
2       fz
3    tuitu
dtype: object
```

1.10 字符分割

1.10.1 split

In [74]:
```
1 s = pd.Series(['a_b_c_d','c_d_e',np.nan,'f_g_h'])
2 s
```

Out[74]:
```
0    a_b_c_d
1      c_d_e
2        NaN
3      f_g_h
dtype: object
```

In [75]:
```
1 s.str.split('_') #split() 切分字符串
```

Out[75]:
```
0    [a, b, c, d]
1       [c, d, e]
2             NaN
3       [f, g, h]
dtype: object
```

In [76]:
```
1 s
```

Out[76]:
```
0    a_b_c_d
1      c_d_e
2        NaN
3      f_g_h
dtype: object
```

In [77]:
```
1 s.str.split('_',2) #用几个空格分,分几次
```

Out[77]:
```
0    [a, b, c_d]
1      [c, d, e]
2            NaN
3      [f, g, h]
dtype: object
```

1.10.2 partition

In [78]:
```
s = pd.Series(['Linda van-der Berg', 'George Pitt-Rivers'])
s.str.partition('-')  #partition() 切割为三部分，分隔符前，分隔符，分隔符后
```

Out[78]:

	0	1	2
0	Linda van	-	der Berg
1	George Pitt	-	Rivers

1.11 拼接

In [96]:
```
s = pd.Series(['a', 'b', np.nan, 'd'])
s
```

Out[96]:
```
0      a
1      b
2    NaN
3      d
dtype: object
```

In [80]:
```
s.str.cat(sep=' ')
```

Out[80]:
```
'a b d'
```

In [81]:
```
s1 = pd.Series(['a', 'b', np.nan, 'd'])
s2 = pd.Series(['A', 'B', 'C', 'D'])
```

In [82]:
```
s1.str.cat(s2, sep=',')  #cat() 拼接字符串     对应位置拼接,用逗号分隔开
```

Out[82]:
```
0    a,A
1    b,B
2    NaN
3    d,D
dtype: object
```

In [83]:
```
s1.str.cat(s2, sep=',',na_rep='-')  #cat() 拼接字符串     如果是空值和其它拼接用-代表空,逗号隔开
```

Out[83]:
```
0    a,A
1    b,B
2    -,C
3    d,D
dtype: object
```

In [84]:
```
s
```

Out[84]:
```
0      a
1      b
2    NaN
3      d
dtype: object
```

In [97]:
```
t = pd.Series(['d', 'a', 'e', 'c'], index=[3, 0, 4, 2])
s.str.cat(t, join='left', na_rep='-')     #相当于作了一个左连接
```

Out[97]:
```
0    aa
1    b-
2    -c
3    dd
dtype: object
```

In [98]:
```
1 s
```

Out[98]:
```
0      a
1      b
2    NaN
3      d
dtype: object
```

In [99]:
```
1 s.str.cat(t, join='inner', na_rep='-')
```

Out[99]:
```
0    aa
2    -c
3    dd
dtype: object
```

1.12 重复

In [88]:
```
1 s = pd.Series(['a', 'b', 'c'])
2 s
```

Out[88]:
```
0    a
1    b
2    c
dtype: object
```

In [89]:
```
1 s.str.repeat(repeats=2) #repeat() 重复
```

Out[89]:
```
0    aa
1    bb
2    cc
dtype: object
```

In [90]:
```
s.str.repeat(repeats=[1, 2, 3]) #设置不同重复次数
```

Out[90]:
```
0      a
1     bb
2    ccc
dtype: object
```

1.13 统计

In [91]:
```
s = pd.Series(['lower', 'CAPITALS', 'this is a sentence', 'SwApCaaSe','21',' ','12a',])
s
```

Out[91]:
```
0                 lower
1               CAPITALS
2    this is a sentence
3              SwApCaaSe
4                     21
5
6                    12a
dtype: object
```

In [92]:
```
s.str.count("a") #count() 计算给定字符出现的次数
```

Out[92]:
```
0    0
1    0
2    1
3    2
4    0
5    0
6    1
dtype: int64
```

In [ ]:
```

```