1 第八章：字符串和文本数据

数据小鱼Rexa
CSDN：https://blog.csdn.net/qq_38395376?spm=1011.2124.3001.5343
Blibli：https://space.bilibili.com/283181288
Github：https://github.com/Rexa-Yu

1.1 字符串基本操作

In [7]:

```
# 对2个容器进行赋值
word="grail"
sent="a scratch"
# python 对于str类型的字符串也有list的切片操作
print(word[0])
print(word[1:3])
print(word[-1])
print(word[:-1])
```

```
g
ra
l
grai
```

In [10]:

```
# 如果要取sent的除去最后一位的字符的内容
sent[:len(sent)-1]
# 其中len () 取得是字符长度
```

Out[10]:

```
'a scratc'
```

1.2 join函数

In [14]:

```
# 定义一些字符串
d1="24.56"
d2="abcd"
d3="^^%%111111&&"
d4="AedfFFF"
# 使用空格将其连接起来
coords=' '.join([d1,d2,d3,d4])
print(coords)
strr=','.join([d1,d2,d3,d4])
print(strr)
```

```
24.56 abcd ^^%%111111&& AedfFFF
24.56,abcd,^^%%111111&&,AedfFFF
```

1.3 split函数

In [23]:

```
coords_split=coords.split(' ')
coords_split
# 直接按照空格分割
# 如果分割行的话，可以使用splitlines()
```

Out[23]:

```
['24.56', 'abcd', '^^%%111111&&', 'AedfFFF']
```

1.4 格式化字符串

In [24]:

```
# 通过format进行关键词的格式化
var="flesh world"
s="it's just a {}!"
print(s.format(var))
```

```
it's just a flesh world!
```

1.5 格式化数字

In [25]:

```
# 普通款
print("there are {} potatoes!".format("55"))
```

```
there are 55 potatoes!
```

In [30]:

```
# 小数百分数款
# 特别的，大括号要和format（）参数对应起来，否则报错!
print("i have got {:.4} and {:.4%} rate to win! ".format(0.12548987,5/7))
```

```
i have got 0.1255 and 71.4286% rate to win!
```

1.6 Python 3.6 新特性 f-string

In [34]:

```
a="hello"
b="world"
print(f'Hi, {a} {b}!')
# 添加f新特性更好用
```

```
Hi, hello world!
```

1.7 查找模式

In [36]:

```
# 可以使用re包的findall ()
import re
s="12,555,asdawafa g a w t ,55123,,,213"
p='\d+'
result=re.findall(pattern=p,string=s)
result
```

Out[36]:

```
['12', '555', '55123', '213']
```

2 第九章：应用函数apply

2.1 Series 的apply函数

In [38]:

```
# 定义一个函数，比如平方
import pandas as pd
def my_sq(x):
    return x*x
# 设置一个Series
s=pd.Series([1,2,3,4,5,6,7,8])
s.apply(my_sq)
```

Out[38]:

```
0     1
1     4
2     9
3    16
4    25
5    36
6    49
7    64
dtype: int64
```

2.2 DataFrame 的apply函数

In [43]:

```
# 定义一个dataframe数据
dt=pd.DataFrame({'a':[100,200,300],'b':[1,2,3]})
dt.apply(my_sq,axis=1)
# axis=0 传入的是整列，axis=1传入的是整行!
```

Out[43]:

	a	b
0	10000	1
1	40000	4
2	90000	9

2.3 apply的高级用法

In [44]:

```
# 加载数据集
titanic=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\seaborn-data-mas
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    object  
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    object  
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 92.4+ KB
```

In [50]:

```
# 计算缺失数目
import numpy as np
def count_missing(vec):
    null_vec=pd.isnull(vec)
    null_count=np.sum(null_vec)
    return null_count
def prop_missing(vec):
    num=count_missing(vec)
    dem=vec.size
    return num/dem
def prop_complete(vec):
    return 1-prop_missing(vec)
# 用apply去套各个函数和数
cmis_col=titanic.apply(count_missing)
pmis_col=titanic.apply(prop_missing)
pcom_col=titanic.apply(prop_complete)
```

In [51]:

```
cmis_col
```

Out[51]:

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [52]:

```
pmis_col
```

Out[52]:

```
survived       0.000000
pclass         0.000000
sex            0.000000
age            0.198653
sibsp          0.000000
parch          0.000000
fare           0.000000
embarked       0.002245
class          0.000000
who            0.000000
adult_male     0.000000
deck           0.772166
embark_town    0.002245
alive          0.000000
alone          0.000000
dtype: float64
```

In [53]:

```
pcom_col
```

Out[53]:

```
survived       1.000000
pclass         1.000000
sex            1.000000
age            0.801347
sibsp          1.000000
parch          1.000000
fare           1.000000
embarked       0.997755
class          1.000000
who            1.000000
adult_male     1.000000
deck           0.227834
embark_town    0.997755
alive          1.000000
alone          1.000000
dtype: float64
```

做缺失值的查找，目的在于看这些数据是否可用，比如embarked列缺失较多，因此在分析的过程中要去除，比如age列缺失值较少，因此可以分析一下为什么会有确实，是不是因为人为因素导致的，可以给予分析人员一定的信息

In [57]:

```
# 按行运用apply可以查看行内的缺失值
cmis_row=titanic.apply(count_missing,axis=1)
pmis_row=titanic.apply(prop_missing,axis=1)
pcom_row=titanic.apply(prop_complete,axis=1)
```

In [58]:

```
cmis_row
```

Out[58]:

```
0      1
1      0
2      1
3      0
4      1
      ..
886    1
887    0
888    2
889    0
890    1
Length: 891, dtype: int64
```

In [59]:

```
pmis_row
```

Out[59]:

```
0      0.066667
1      0.000000
2      0.066667
3      0.000000
4      0.066667
         ...
886    0.066667
887    0.000000
888    0.133333
889    0.000000
890    0.066667
Length: 891, dtype: float64
```

In [60]:

```
pcom_row
```

Out[60]:

```
0      0.933333
1      1.000000
2      0.933333
3      1.000000
4      0.933333
         ...
886    0.933333
887    1.000000
888    0.866667
889    1.000000
890    0.933333
Length: 891, dtype: float64
```

In [61]:

```
# 查看多少行包括缺失值
print(cmis_row.value_counts())
```

```
1    549
0    182
2    160
dtype: int64
```

In [62]:

```
# 根据apply函数可以增加缺失列给titanic
titanic["num_missing"]=titanic.apply(count_missing,axis=1)
titanic
```

Out[62]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_ma
0	0	3	male	22.0	1	0	7.2500	S	Third	man	Tı
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fa
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fa
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fa
4	0	3	male	35.0	0	0	8.0500	S	Third	man	Tı
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.0	0	0	13.0000	S	Second	man	Tı
887	1	1	female	19.0	0	0	30.0000	S	First	woman	Fa
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	Fa
889	1	1	male	26.0	0	0	30.0000	C	First	man	Tı
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	Tı

891 rows × 16 columns

In [63]:

```
# 寻找具有缺失值的titanic的行
print(titanic.loc[titanic.num_missing>1,:])
```

```
     survived  pclass     sex  age  sibsp  parch     fare embarked   c
lass  \
5           0       3    male  NaN      0      0   8.4583        Q   T
hird
17          1       2    male  NaN      0      0  13.0000        S  Se
cond
19          1       3  female  NaN      0      0   7.2250        C   T
hird
26          0       3    male  NaN      0      0   7.2250        C   T
hird
28          1       3  female  NaN      0      0   7.8792        Q   T
hird
..        ...     ...     ...  ...    ...    ...      ...      ...
...
859         0       3    male  NaN      0      0   7.2292        C   T
hird
863         0       3  female  NaN      8      2  69.5500        S   T
hird
868         0       3    male  NaN      0      0   9.5000        S   T
hird
878         0       3    male  NaN      0      0   7.8958        S   T
hird
888         0       3  female  NaN      1      2  23.4500        S   T
hird

       who  adult_male deck  embark_town alive  alone  num_missing
5      man        True  NaN   Queenstown    no   True            2
17     man        True  NaN  Southampton   yes   True            2
19   woman       False  NaN    Cherbourg   yes   True            2
26     man        True  NaN    Cherbourg    no   True            2
28   woman       False  NaN   Queenstown   yes   True            2
..     ...         ...  ...          ...   ...    ...          ...
859    man        True  NaN    Cherbourg    no   True            2
863  woman       False  NaN  Southampton    no  False            2
868    man        True  NaN  Southampton    no   True            2
878    man        True  NaN  Southampton    no   True            2
888  woman       False  NaN  Southampton    no  False            2

[160 rows x 16 columns]
```

2.4 lambda函数

In [65]:

```
# lambda函数只要就方便编程，其中可以代替def
a=pd.Series([1,2,3,4,5])
d=a.apply(lambda x:x*x)
d
```

Out[65]:

```
0     1
1     4
2     9
3    16
4    25
dtype: int64
```