Data Analysis

Velen Kong

摘要

基于Python的数据分析入门笔记，希望能自学掌握基本的数据分析能力，毕竟是统计学的基础之一。

内容安排主要参考

(美) Wes McKinney 著. Python for Data Analysis:2nd Edition [M] USA: O'Reilly Media 2017

该书作者同时也是pandas库的作者，同时借用其在GitHub上的数据资料。

第三方库与Python入门

IPython & Jupyter

用到的库：ipython，jupyter，numpy，matplotlib，pandas，scipy，scikit-learn，statsmodels。

jupyter确实好用，扩展了$Tab, ?, *$的功能。

还有一些快捷键仅限IPython使用。

另外就是一些常用的Magic Command，比如

```
%run
%load
%debug
%xmode Plain # less detailed
%xmode Verbose # more detailed
%matplotlib inline # draw in Jupyter
import numpy as np
import matplotlib.pyplot as plt
plt.plot(np.random.rand(50).cumsum())
```

```
# ----- Day 0 -----
```

Python基础

#注释。

```
# this is comment
```

所有变量都是对象(object)。

基本的函数调用，利用函数批量处理。

```
def append_element(some_list, element):
    some_list.append(element)
    return

data = [1, 2, 3]
append_element(data, 4)
data
# Out: [1, 2, 3, 4]
```

变量赋值类似引用(reference)。

动态类型，利用type()和isinstance()进行类型检查。后者可以传入tuple代替逻辑或操作。

```
a = 1.5
isinstance(a, (int, float))
# Out: True
```

print配合format输出。

```
a = 1; b = 1.5
print('a is {1}, b is {0}'.format(type(b),
type(a)))
# Out: a is <class 'int'>, b is <class 'float'>
```

Python中的object拥有各自的属性(attributes)和方法(methods)，可通过getattr, hasattr, setattr操作。其中getattr可以直接使用返回对象，setattr不改变原class。

```
class A(object):
    def set(self, a, b):
        x = a
        a = b
        b = x
        print a, b

a = A()
c = getattr(a, 'set')
c(a='1', b='2')
# Out: 2 1
```

可以利用try()检查容器是否是顺序，同时生成list。

```
def isiterable(obj):
    try:
        iter(obj)
        return True
    except TypeError:
```

```
        return False
    return

x = 'string'
isiterable(x)
# Out: True
isiterable(5)
# Out: False

if not isinstance(x, list) and isiterable(x):
    x = list(x)
x
# Out: ['s', 't', 'r', 'i', 'n', 'g']
```

import和as的使用。

is和is not的使用。

```
a = [1, 2, 3]
b = a
c = list(a)
d = None
print(a is b)
# Out: True
print(a is not c)
# Out: True
print(a == c)
# Out: True
print(d is None)
# Out: True
```

大部分Python的容器都是可变的(mutable)，而strings和tuples不可变(immutable)。

```
a = ['foo', 2, [4, 5]]
a[2] = (3, 4)
a
# Out: ['foo', 2, (3, 4)]
```

```
# ----- Day 1 -----
```

标量类型(scalar types)一般有$None, str, bytes, float, bool, int$。

多行字符串用三个引号。

```
c = """
This is a longer string that
spans multiple lines
"""
c.count('\n')
# Out: 3
```

str()方法可以生成字符串，+可以直接拼接字符串。而replace()方法可以替换字符串内容，但不改变原串。

```
a = 'this is a string'
b = a.replace('string', 'longer string')
print(b)
# Out: this is a longer string
print(a)
# Out: this is a string
```

Python采用$Unicode$编码，支持中文。对特殊字符可以采用\或者r进行转义。

```
s = '12\\34'
print(s)
# Out: 12\34
s = r'this\has\no\special\characters'
s
# Out: 'this\\has\\no\\special\\characters'
```

format和{}可以进行格式化输出。

```
template = '{0:.2f} {1:s} are worth US${2:d}'
template.format(4.5560, 'Argentine Pesos', 1)
# Out: '4.56 Argentine Pesos are worth US$1'
```

利用encode()和decode()函数可以对字符进行加解码。利用b生成二进制串。

```
val = "español"
val_utf8 = val.encode('utf-8')
val_utf8
# Out: b'espa\xc3\xb1ol'
type(val_utf8)
# Out: bytes
val_utf8.decode('utf-8')
# Out: 'español'
bytes_val = b'this is bytes'
bytes_val
# Out: b'this is bytes'
decoded = bytes_val.decode('utf8')
decoded
# Out: 'this is bytes'
```

None是一个单独的类型，常见于函数的默认参数中。

```
def add_and_maybe_multiply(a, b, c=None):
    result = a + b
    if c is not None:
        result = result * c
    return result

type(None)
# Out: NoneType
```

and和or运算。

Python自带时间与日期库。datetime是不可变类型，所以所有的操作都会产生新的对象而不改变原对象。

```
from datetime import datetime, date, time
dt = datetime(2011, 10, 29, 20, 30, 21)
dt.day
# Out: 29
dt.date()
# Out: datetime.date(2011, 10, 29)
dt.strftime('%m/%d/%Y %H:%M')
# Out: '10/29/2011 20:30'
datetime.strptime('20091031', '%Y%m%d')
# Out: datetime.datetime(2009, 10, 31, 0, 0)
dt.replace(minute=0, second=0)
# Out: datetime.datetime(2011, 10, 29, 20, 0)
```

时间差用timedelta表示，单位是天和秒。

```
dt2 = datetime(2011, 11, 15, 22, 30)
delta = dt2 - dt
delta
# Out: datetime.timedelta(17, 7179)
type(delta)
# Out: datetime.timedelta
dt + delta
# Out: datetime.datetime(2011, 11, 15, 22, 30)
```

时间的格式符号(format specification)如下

```
%Y #4位年数
%y #2位年数
%m #2位月数
%d #2位天数
%H #24小时制(2位)
%I #12小时制(2位)
%M #2位分钟数
%S #2位秒数，由于闰秒存在，范围为[00, 61]
```

```
%w #星期，范围为[0, 6]
%U #一年的第几个星期，以周日为标准，第一个周日之前的算00
%W #同上，以周一为标准
%z #用+HHMM或-HHMM表示相对于格林威治的时区偏移
%Z #时区名称，默认为空
%F #等价于%Y-%m-%d
%D #等价于%m/%d/%y
```

if, elif, else语句。单行if。

```
a = [1,2,3]
b = a if len(a) != 0 else ""
print(b)
# Out: [1, 2, 3]
```

for in语句。多变量for循环。

```
starts = [0,1,2,3,4]
ends = [5,6,7,8,9]
for start, end in zip(starts, ends):
    print((start, end))
'''
Out:
(0, 5)
(1, 6)
(2, 7)
(3, 8)
(4, 9)
'''
```

while语句。

range()函数。

```
# ----- Day 2 -----
```

数据结构

Numpy库