**Pandas**

1. **基本概念:**
2. **Pandas 是什么？**

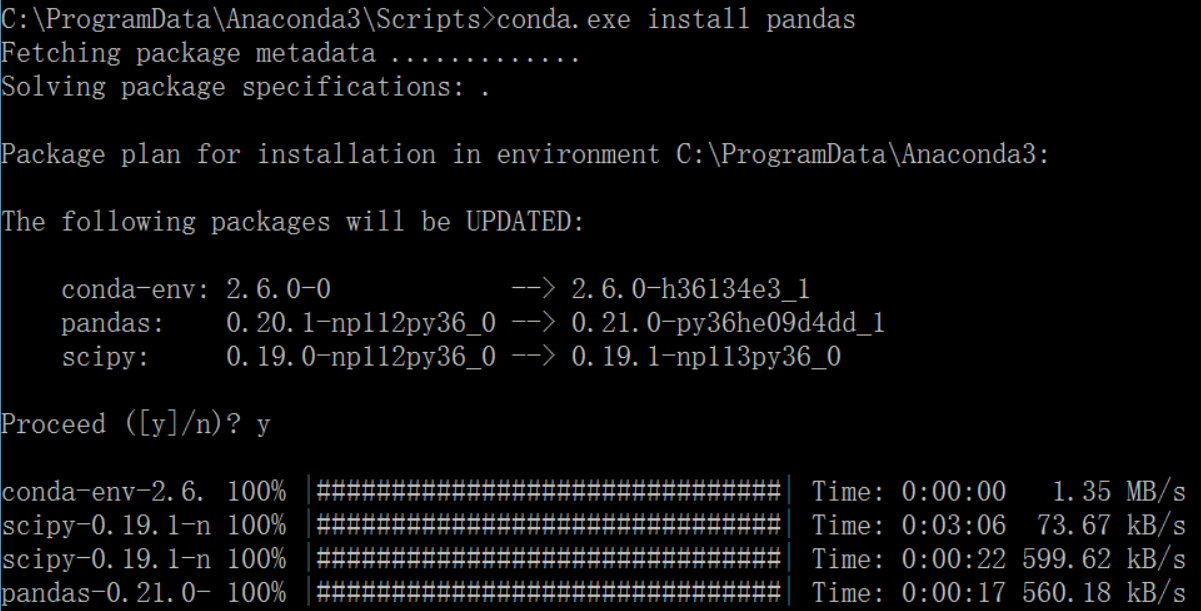
Python Data Analysis Library 或 pandas 是基于NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。Pandas 纳入了大量库和一些标准的数据模型，提供了高效地操作大型数据集所需的工具。pandas提供了大量能使我们快速便捷地处理数据的函数和方法。你很快就会发现，它是使Python成为强大而高效的数据分析环境的重要因素之一。

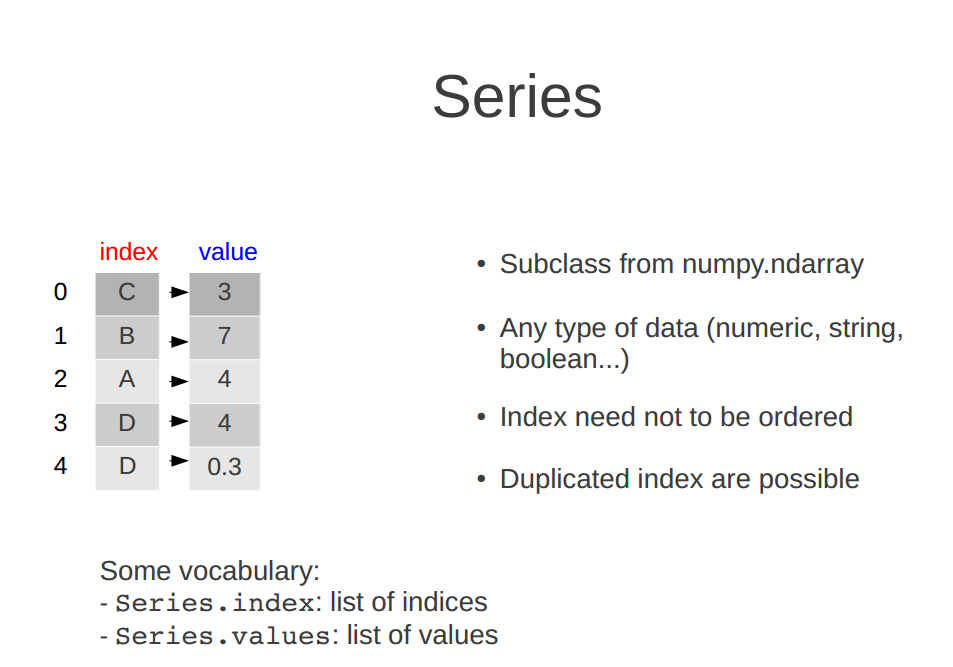
1. **Pandas 中的数据结构：**

* Series：一维数组，与Numpy中的一维array类似。二者与Python基本的数据结构List也很相近，其区别是：List中的元素可以是不同的数据类型，而Array和Series中则只允许存储相同的数据类型，这样可以更有效的使用内存，提高运算效率。
* DataFrame：二维的表格型数据结构。很多功能与R中的data.frame类似。可以将DataFrame理解为Series的容器。以下的内容主要以DataFrame为主。
* Panel ：三维的数组，可以理解为DataFrame的容器。

**3）Pandas 的安装：**

首先要安装过Numpy 1.6.1版本以上，Pandas安装类似Numpy的安装方式，有官网安装，pip安装等方法，推荐Anaconda安装方式。



1. **Series:**

* **创建Series:**

**(1)创建一个空的Series:**

**例1:**

import pandas as pd

s = pd.Series()

print(s)

**(2)通过Numpy的数组创建Series：**

**例2：**

import pandas as pd

import numpy as np

data = np.array([‘a’, ‘b’, ‘c’, ‘d’])

s = pd.Series(data)

print(s)

s2 = pd.Series(data, index= [10, 11, 12, 13])

print(s2)

**(3)通过字典创建Series:**

**例3：**

import numpy as np

import pandas as pd

data = {'a' : 0., 'b' : 1., 'c' : 2.}

s= pd.Series(data)

print(s) # 字典的键作为索引。

s2 = pd.Series(data, index=['b', 'c', 'd', 'a'])

print(s2)

**(4) 通过标量创建Series：**

**例4：**

import numpy as np

import pandas as pd

s = pd.Series(5, index=[0, 1, 2, 3])  
print (s)

* **Series的索引：**

**例5：**

import pandas as pd

s=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])

print(s[0])

print(s[:3])

print(s[-3:])

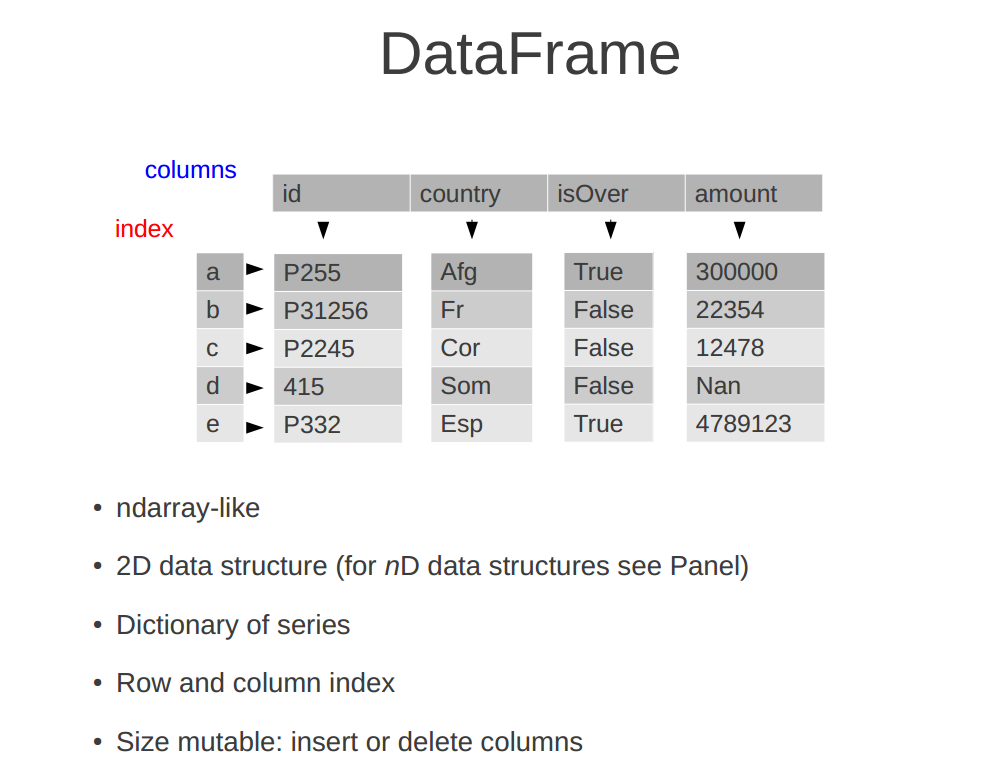
print(s[‘a’])

print(s[‘a’])

print(s[[‘a’, ‘b’, ‘c’]])

print(s[[‘a’, ‘j’, ‘k’]])

print(s[[‘o’, ‘j’, ‘k’]])

1. **DataFrame:**

* **创建DataFrame:**

**（1）创建一个空的DataFrame:**

**例6：**

import pandas as pd

a = pd.DataFrame()

print(a)

**(2) 通过列表创建DataFrame:**

**例7：**

import pandas as pd

data = [1,2,3,4,5]  
 a = pd.DataFrame(data)

print(a)

data2 = [['Alex',10],['Bob',12],['Clarke',13]]

b = pd.DataFrame(data,columns=['Name','Age'])

# b = pd.DataFrame(data,columns=['Name','Age'],dtype=float)

print(b)

**(3) 通过字典创建 DataFrame:**

**例8：**

import pandas as pd

data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],

'Age':[28,34,29,42]}

a = pd.DataFrame(data)

print(a)

b = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])

print(b)

**(4) 通过字典组合的列表创建 DataFrame：**

**复习：python两个列表组合生成对应的字典**

a = [('a','b', 'c') ,('d', 'e', 'f')]

b = ['A','B','C']

* [{'A': 'a', 'B': 'b', 'C': 'c'}, {'A': 'd', 'B': 'e', 'C': 'f'}] ?

**例9：**

import pandas as pd

data = [{'a': 1, 'b': 2},

{'a': 5, 'b': 10, 'c': 20}]

a = pd.DataFrame(data)  
 # a = pd.DataFrame(data, index=['first', 'second'])

# a = pd.DataFrame(data, index=[‘first’, ‘second’], columns=['a', 'b'])

# a = pd.DataFrame(data, index=[‘first’, ‘second’], columns=['a', 'b1'])

**(5) 通过Series组合的字典创建 DataFrame：**

**例10：**

import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),

'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

a = pd.DataFrame(d)

print(a)

# 选择列

print(a [‘one’])

# 增加列

a [‘three’] = pd.Series([10,20,30], index=[‘a’, ‘b’, ‘c’])

print(a)

a[‘four’] = a[‘one’] + a[‘three’]

print(a)

# 删除列

1. 通过del函数删除

del a[‘one’]

(2) 通过pop函数删除

a.pop(‘two’)

# 行的选择，添加和删除

a.loc[‘b’]

# 行分片

a[2:3]

# 增加行

a = pd.DataFrame ([[1,2], [3, 4]], columns = [‘a’, ‘b’])

b = pd.DataFrame ([[5, 6], [7,8]], columns = [‘a’, ‘b’])

a = a.append(b)

# 删除行

a = a.drop(0)

1. **Panel:**

Panel是3维数据容器。在Panel中，有三个重要的参数：

* items: 0轴， 每个项目对应其中的一个DataFrame
* major\_axis: 1轴，它是每个DataFrame的index
* minor\_axis: 2轴，它是每个DataFrame的column
* **创建Panel：**

1. **通过三维数组创建：**

**例11：**

import numpy as np

import pandas as pd

a = np.random.rand(2,4,5)

b = pd.Panel(a)

**（2） 通过DataFrame组合的字典创建：**

**例12：**

import numpy as np

import pandas as pd

a = {‘Item1’ : pd.DataFrame(np.random.randn(4,3)),

‘Item2’ : pd.DataFrame(np.random.randn(4,2))}

b = pd.Panel(a)

**（3） 创建一个空Panel：**

**例13:**

import pandas as pd

a = pd.Panel()

* **从Panel中选取数据：**

**（1）通过Items 去选取：**

**例14**

import pandas as pd

import numpy as np

a = {'Item1' : pd.DataFrame(np.random.randn(4, 3)),  
 'Item2' : pd.DataFrame(np.random.randn(4, 2))}

b = pd.Panel(a)

b[‘Item1’]

**(2) 通过major\_axis去选取：**

**例15**

import pandas as pd

import numpy as np

a = {'Item1' : pd.DataFrame(np.random.randn(4, 3)),

'Item2' : pd.DataFrame(np.random.randn(4, 2))}

b = pd.Panel(a)

b.major\_xs(1)

**（3）通过minor\_axis 去选取：**

b.minor\_xs(1)

1. **用Pandas常用函数查看和操作数据**

**例16**

import pandas as pd

import numpy as np

dates = pd.date\_range('20170926', periods=6)

a = pd.DataFrame(np.random.randn(6,4), index=dates, col  
umns=list('ABCD'))

#查看DataFrame中头部和尾部的行：

a.head()

a.tail(3)

#显示索引、列和底层的 numpy 数据：

a.index

a.columns

a.values

# describe() 函数对于数据的快速统计汇总：

a.describe() #复习四分位数

# 对数据的转置：

a.T

# 按轴进行排序：

a.sort\_index(axis=1)

a.sort.values(‘b’)

# 选择数据

# 选择一个单独的列

a[‘B’]

# 对行进行切片

a[:3]

a[‘20170926’ :’20170928’]

# loc按照索引来进行行列选择。

# 选择一列,两列

a.loc[:,’A’]， a.loc[:,[‘A’, ’B’]]

# 选择一行

b.loc['20170930',:]

# 比较b[1:3] 和 b.loc[1;3]

# iloc按照索引的位置来进行选取, iloc不关心索引的具体值是多少，只关心位置是多少，所以使用iloc时方括号中只能使用数值。

b.iloc[3]

虽然标准的 Python/Numpy 的选择和设置表达式都能够直接派上用场，但是作为工程使用的代码，我们推荐使用经过优化的 pandas 数据访问方式： .at , .iat ,.loc , .iloc 和 .ix 。