**pandas之数据结构**

[pandas之数据结构\_pandas.core.series.series-CSDN博客](https://blog.csdn.net/qq_45286306/article/details/129535169)

**1 pandas数据类型**

pandas包含两种数据类型：series和dataframe。  
series是一种一维数据结构，每一个元素都带有一个索引，与一维数组的含义相似，其中索引可以为数字或字符串。series结构名称：  
  A close up of a text

Description automatically generated with medium confidence

dataframe是一种二维数据结构，数据以表格形式（与excel类似）存储，有对应的行和列。dataframe结构名称：

A black and white text with red arrows

Description automatically generated

以上内容借鉴：

[非常全面的Pandas入门教程\_pandas教程\_Summer1Li的博客-CSDN博客](https://blog.csdn.net/weixin_44489066/article/details/89494395)

**2 创建数据**

**2.1 创建Series**

Series类似于一维数组的数据结构，能够保存任何类型的数据，如整数、字符串、浮点数等，主要由一组数据和与之相关的索引**两部分构成。**

|  |
| --- |
| 1. import pandas as pd 2. #不定义索引，默认创建从0-n的整数索引 3. s = pd.Series(['wang','23','1999']) 4. print(s) 5. #单独附加定义索引 6. s.index=['name','age','year'] 7. print(s) 8. print('------------分隔-----------') 10. #直接定义索引名称：index 参数 11. s2 = pd.Series(['wang','23','1999'],index=['name','age','year']) 12. print(s2) 13. print('------------分隔-----------') 15. #也可以使用字典键值对来创建,注意字典的格式{} 16. s3 = pd.Series({'name':'wang','age':23,'year':1999}) 17. print(s3) 19. #简单使用索引获取数据 20. print(s3.iloc[0]) 21. print(s3.loc['name']) |

**2.2 创建DataFrame**

DataFrame类似于二维数组或表格(如excel)的对象，既有行索引，又有列索引，常使用字典创建，键值对：键代表列名，值是列的内容。

|  |
| --- |
| 1. import pandas as pd 3. #不定义索引，默认0-n，常使用字典创建，键值对{} 4. message = pd.DataFrame({ 5. 'name':['wang','sun','zhang','li'], 6. 'age':['22','23','21','22'], 7. 'country':['china','china','nbc','english'], 8. 'score':['65','76','87','92'] 9. }) 10. print(message) 11. print('------------分隔-----------') 13. #使用index参数指定行索引，columns参数指定列顺序 14. message1 = pd.DataFrame( 15. data={ 16. 'age':['22','23','21','22'], 17. 'country':['china','china','nbc','english'], 18. 'score':['65','76','87','92'] 19. }, 20. index=['wang','sun','zhang','li'], 21. columns=['age','score','country'] 22. ) 23. print(message1)  26. ''' 27. name age country score 28. 0 wang 22 china 65 29. 1 sun 23 china 76 30. 2 zhang 21 nbc 87 31. 3 li 22 english 92 32. ------------分隔----------- 33. age score country 34. wang 22 65 china 35. sun 23 76 china 36. zhang 21 87 nbc 37. li 22 92 english 38. Process finished with exit code 0 39. ''' |

**3 Series**

**3.1 使用DataFrame 的loc属性获取数据的某一行，就会得到一个Series对象**

|  |
| --- |
| **index 输出’索引‘**  **index[0] 在方法之后加入方括号指定某一’索引‘**  **keys() 和 index 作用相同，查看’索引‘**  **keys()[0] 在方法之后加入方括号指定某一’索引‘**  **values 查看Series 值**  **size 查看 serise 中元素的数量** |

|  |
| --- |
| 1. import pandas as pd 2. #使用DataFrame 的loc属性获取数据的某一行，就会得到一个Series对象 3. #先创建一个DataFrame 4. message = pd.DataFrame( 5. data={ 6. 'age':['22','23','21','22'], 7. 'country':['china','china','nbc','english'], 8. 'score':['65','76','87','92'] 9. }, 10. index=['wang','sun','zhang','li'], 11. columns=['age','score','country'] 12. ) 13. print(message) 14. print('------------间隔-----------') 16. #使用行索引，获取某一行，查看属性发现是Series属性 17. #或者使用列索引，获取某一列，例如 all\_age = message['age'] 18. first\_people = message.loc['wang'] 19. print(first\_people) #此时输出两列，第一列’索引‘，第二列’值‘ 20. print(type(first\_people)) #type()属性 21. print('------------间隔-----------') 23. print(first\_people.index) #.index 输出’索引‘ 24. print(first\_people.index[1]) #.index[0] 在方法之后加入方括号指定某一’索引‘ 25. print(first\_people.keys()) #keys() 和 index 作用相同，查看’索引‘ 26. print(first\_people.keys()[1]) #keys()[0] 在方法之后加入方括号指定某一’索引‘ 27. print(first\_people.values) #values 查看Series 值 28. print(first\_people.size) #size 查看 serise 中元素的数量  31. ''' 32. age score country 33. wang 22 65 china 34. sun 23 76 china 35. zhang 21 87 nbc 36. li 22 92 english 37. ------------间隔----------- 38. age 22 39. score 65 40. country china 41. Name: wang, dtype: object 42. <class 'pandas.core.series.Series'> 43. ------------间隔----------- 44. Index(['age', 'score', 'country'], dtype='object') 45. score 46. Index(['age', 'score', 'country'], dtype='object') 47. score 48. ['22' '65' 'china'] 49. 3 50. ''' |

**3.2 Series的一些方法，类似于numpy.ndarray**

A white paper with black text

Description automatically generatedA close up of a document

Description automatically generated

下面使用mean()和std() 输出平均值、标准差会出错，因为开始使用字典创建age列时为'age':['22','23','21','22']  此时是object类型而不是int64整型，应改为'age':[22,23,21,22]

|  |
| --- |
| 1. import pandas as pd 2. #使用DataFrame 的loc属性获取数据的某一行，就会得到一个Series对象 3. #先创建一个DataFrame 4. message = pd.DataFrame( 5. data={ 6. 'age':['22','23','21','22'], 7. 'country':['china','china','nbc','english'], 8. 'score':['65','76','87','92'] 9. }, 10. index=['wang','sun','zhang','li'], 11. columns=['age','score','country'] 12. ) 13. print(message) 14. print('------------间隔-----------')  17. #使用列索引，获取某一列，查看属性是Series属性 18. all\_age = message['age'] 19. print(all\_age) #输出两列，第一列’索引‘，第二列’值‘ 20. print('------------间隔-----------') 22. print(all\_age.mean()) #平均值 #？？？输出结果错误？？？ 23. print(all\_age.min()) #最小值 24. print(all\_age.max()) #最大值 25. print(all\_age.sample()) #随机值 26. #print(all\_age.std()) #标准差  29. ''' 30. wang 22 31. sun 23 32. zhang 21 33. li 22 34. Name: age, dtype: object 35. ------------间隔----------- 36. 5558030.5 37. 21 38. 23 39. li 22 40. ''' |

**3.3 布尔子集**

|  |
| --- |
| 1. import pandas as pd 3. all\_age = pd.Series([22, 24, 13, 22, 21, 12]) 4. print(all\_age) 5. print('------------间隔-----------') 7. print(all\_age.describe()) #调用describe()方法获得多项统计数据 8. print(all\_age.mean()) #获得所有年龄的平均值 9. print(all\_age[all\_age > all\_age.mean()]) #方法一 获得所有大于平均值的年龄 10. print(all\_age > all\_age.mean()) #方法二 使用这条语句，返回一个Series，可以查看dtype为bool ，与上面语句输出结果比较一下区别 12. #手动提供布尔向量来获取某些数据 13. age\_values = [True,True,False,False,False,False] #获取索引0、1的年龄 14. print(all\_age[age\_values])  17. ''' 18. 0 22 19. 1 24 20. 2 13 21. 3 22 22. 4 21 23. 5 12 24. dtype: int64 25. ------------间隔----------- 26. count 6.000000 27. mean 19.000000 28. std 5.138093 29. min 12.000000 30. 25% 15.000000 31. 50% 21.500000 32. 75% 22.000000 33. max 24.000000 34. dtype: float64 35. 19.0 36. 0 22 37. 1 24 38. 3 22 39. 4 21 40. dtype: int64 41. 0 True 42. 1 True 43. 2 False 44. 3 True 45. 4 True 46. 5 False 47. dtype: bool 48. 0 22 49. 1 24 50. dtype: int64 51. ''' |

**3.4 操作向量运算**

|  |
| --- |
| 1. import pandas as pd 3. all\_age = pd.Series([22, 24, 13, 22, 21, 12]) 4. print(all\_age) 5. print('------------间隔-----------') 7. #同长度向量运算 8. print(all\_age+all\_age) 9. print(all\_age\*all\_age) 10. #向量与整数标量运算 11. print(all\_age+100) 12. print(all\_age\*2) 13. #不同长度向量运算：当运算向量只有前三项有值，其余元素会被填充为‘缺失’，用NaN表示，指‘非数值’，这种处理方式为‘广播’ 14. print(all\_age + pd.Series([1,10,3])) 16. #带有默认索引标签的向量，会依据索引标签进行自动对其 17. #sort\_index() 依据索引值排序，此时是正序，下面例子ascending=False表示降序 18. #sort\_values() 依据 值 排序，使用方法类似 19. #例如 1.先顺序输出 2.使用sort\_index()对索引号排序 3.将前两项相加，即在向量反向的情况下相加时根据索引标签自动对齐 #观察结果 20. print(all\_age) 21. rev\_ages = all\_age.sort\_index(ascending=False) #降序 22. print(rev\_ages) 23. print(all\_age+rev\_ages)  26. ''' 27. 0 22 28. 1 24 29. 2 13 30. 3 22 31. 4 21 32. 5 12 33. dtype: int64 34. ------------间隔----------- 35. 0 44 36. 1 48 37. 2 26 38. 3 44 39. 4 42 40. 5 24 41. dtype: int64 42. 0 484 43. 1 576 44. 2 169 45. 3 484 46. 4 441 47. 5 144 48. dtype: int64 49. 0 122 50. 1 124 51. 2 113 52. 3 122 53. 4 121 54. 5 112 55. dtype: int64 56. 0 44 57. 1 48 58. 2 26 59. 3 44 60. 4 42 61. 5 24 62. dtype: int64 63. 0 23.0 64. 1 34.0 65. 2 16.0 66. 3 NaN 67. 4 NaN 68. 5 NaN 69. dtype: float64 70. 0 22 71. 1 24 72. 2 13 73. 3 22 74. 4 21 75. 5 12 76. dtype: int64 77. 5 12 78. 4 21 79. 3 22 80. 2 13 81. 1 24 82. 0 22 83. dtype: int64 84. 0 44 85. 1 48 86. 2 26 87. 3 44 88. 4 42 89. 5 24 90. dtype: int64 91. ''' |

**4 DataFrame**

 布尔向量和操作自动对齐与[向量化](https://so.csdn.net/so/search?q=%E5%90%91%E9%87%8F%E5%8C%96&spm=1001.2101.3001.7020" \t "_blank)

|  |
| --- |
| 1. import pandas as pd 3. #先创建一个DataFrame 4. message = pd.DataFrame( 5. data={ 6. 'age':[22,23,21,22], 7. 'country':['china','china','nbc','english'], 8. 'score':['65','76','87','92'] 9. }, 10. #index=['wang','sun','zhang','li'], 11. columns=['age','score','country'] 12. ) 13. print(message) 14. print('------------间隔-----------')  17. #布尔向量获取部分数据行 18. #Series 与 DataFrame 注意区别 19. #print(all\_age[all\_age > all\_age.mean()]) #方法一 获得所有大于平均值的年龄 20. #print(all\_age > all\_age.mean()) #方法二 使用这条语句，返回一个Series，可以查看 21. print(message[message['age'] > message['age'].mean()]) 22. #返回0、1 两行 23. print(message.loc[[True,True,False,False]]) 24. print('------------间隔-----------') 26. #操作向量运算 27. first\_half = message[:3] #切片，获取0、1、2 28. second\_half = message[2:] #获取2、3 29. print(first\_half) 30. print(second\_half) 31. print(message+message) #每个元素会分别运算，数值会加倍，字符串会重复翻倍 |

**5 添加更改列 删除**

**5.1 添加**

|  |
| --- |
| 1. import pandas as pd 2. #先创建一个DataFrame 3. message = pd.DataFrame( 4. data={ 5. 'age':[22,23,21,22], 6. 'country':['china','china','nbc','english'], 7. 'score':['65','76','87','92'], 8. 'born':[1999,2000,2009,1998], 9. 'now':[2023,2023,2023,2023] 10. }, 11. #index=['wang','sun','zhang','li'], 12. columns=['age','score','country','born','now'] 13. ) 14. print(message) 15. print('------------间隔-----------')  18. #添加列 19. #将出生日期和现在日期现在的object类型 转换成datatime类型 ，利于下一步执行日期计算的操作 20. born\_time = pd.to\_datetime(message['born'], format='%Y') #日期年月日格式 '%Y-%m-%d' 21. print(born\_time) 22. now\_time = pd.to\_datetime(message['now'], format='%Y') 23. print(now\_time) 25. #转换成datatime类型的日期只是被赋值 ，并没有加入到message中，下面将born\_time,now\_time加入到message中添加列 26. message['born\_time\_1'],message['now\_time'] = (born\_time,now\_time) 27. print(message) 29. ''' 30. 0 1999-01-01 31. 1 2000-01-01 32. 2 2009-01-01 33. 3 1998-01-01 34. Name: born, dtype: datetime64[ns] 35. 0 2023-01-01 36. 1 2023-01-01 37. 2 2023-01-01 38. 3 2023-01-01 39. Name: now, dtype: datetime64[ns] 40. age score country born now born\_time\_1 now\_time 41. 0 22 65 china 1999 2023 1999-01-01 2023-01-01 42. 1 23 76 china 2000 2023 2000-01-01 2023-01-01 43. 2 21 87 nbc 2009 2023 2009-01-01 2023-01-01 44. 3 22 92 english 1998 2023 1998-01-01 2023-01-01 45. ''' |

**5.2 更改 astype()**

|  |
| --- |
| 1. #更改列 2. #通过now\_time born\_time 计算出差值，得出天数 3. message['age\_day'] = (message['now\_time'] - message['born\_time\_1']) 4. print(message) 5. #利用astype('timedelta64[Y]')方法将天数转换成年 6. message['age\_year'] = message['age\_day'].astype('timedelta64[Y]') 7. print(message)  10. ''' 11. age score country born now born\_time\_1 now\_time age\_day 12. 0 22 65 china 1999 2023 1999-01-01 2023-01-01 8766 days 13. 1 23 76 china 2000 2023 2000-01-01 2023-01-01 8401 days 14. 2 21 87 nbc 2009 2023 2009-01-01 2023-01-01 5113 days 15. 3 22 92 english 1998 2023 1998-01-01 2023-01-01 9131 days 16. age score country born now born\_time\_1 now\_time age\_day age\_year 17. 0 22 65 china 1999 2023 1999-01-01 2023-01-01 8766 days 24.0 18. 1 23 76 china 2000 2023 2000-01-01 2023-01-01 8401 days 23.0 19. 2 21 87 nbc 2009 2023 2009-01-01 2023-01-01 5113 days 13.0 20. 3 22 92 english 1998 2023 1998-01-01 2023-01-01 9131 days 24.0 21. Process finished with exit code 0 22. ''' |

**5.3 删除 drop**

|  |
| --- |
| 1. #删除值 2. #删除行 drop默认删除行，加axis=1 为列 3. message = message.drop([2]) 4. print(message) 5. #删除列，加axis=1 为列 6. message = message.drop(['age\_day'],axis=1) 7. print(message) 9. ''' 10. age score country born now born\_time\_1 now\_time age\_day age\_year 11. 0 22 65 china 1999 2023 1999-01-01 2023-01-01 8766 days 24.0 12. 1 23 76 china 2000 2023 2000-01-01 2023-01-01 8401 days 23.0 13. 3 22 92 english 1998 2023 1998-01-01 2023-01-01 9131 days 24.0 14. age score country born now born\_time\_1 now\_time age\_year 15. 0 22 65 china 1999 2023 1999-01-01 2023-01-01 24.0 16. 1 23 76 china 2000 2023 2000-01-01 2023-01-01 23.0 17. 3 22 92 english 1998 2023 1998-01-01 2023-01-01 24.0 18. ''' |

**6 导出导入数据**

**6.1 csv**

|  |
| --- |
| 1. #导入CSV文件 2. df = pandas.read\_csv('D://postgraduatestudies/py/mypythonstudy/gapdata0.csv',encoding="utf-8",sep=',') 4. #保存CSV文件 5. #index=False用于控制是否写行名 6. message.to\_csv('D://postgraduatestudies/py/mypythonstudy/message\_3.csv',index=False) |

有无index=False结果对比：

A screenshot of a table

Description automatically generatedA table of numbers with black text

Description automatically generated with medium confidence

**6.2 Excel**

|  |
| --- |
| 1. # 导出Excel文件 3. #Series结构不支持to\_excel,保存前需要将结构转换成单列的DataFrame :to\_frame() 4. #age\_1 = age.to\_frame() 6. #导出为xls文件 7. import xlwt #先安装pip install 8. message.to\_excel('D://postgraduatestudies/py/mypythonstudy/message\_4.xls',index=False) 10. #导出为xlsx文件 11. import openpyxl #先安装openpyxl 12. message.to\_excel('D://postgraduatestudies/py/mypythonstudy/message\_4.xlsl',index=False) |