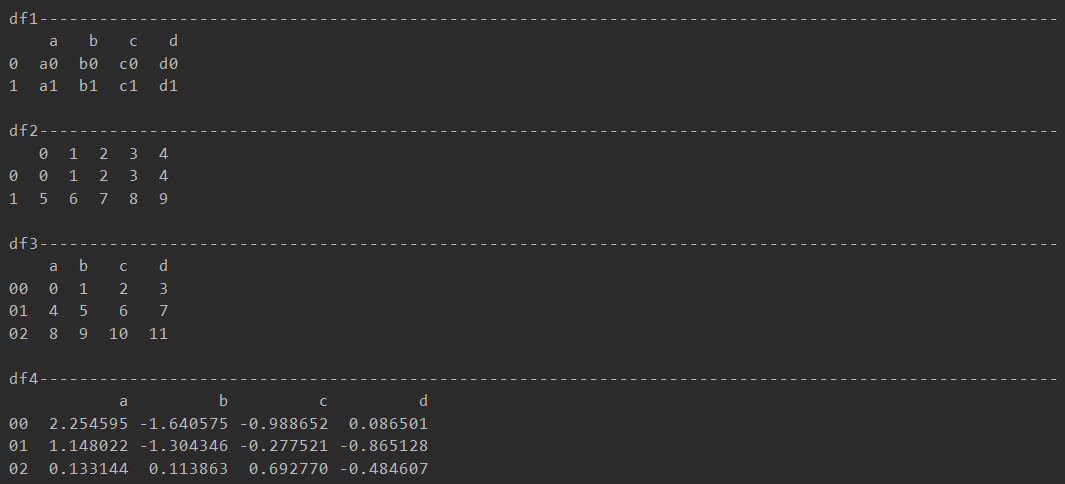
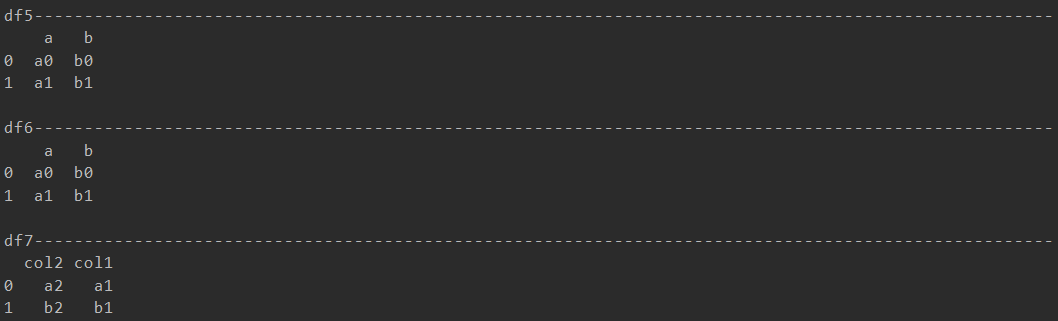
### 1.创建

1. # 直接创建
2. df1 = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
3. index=[0, 1],
4. columns=['a', 'b', 'c', 'd'])
5. # 二维数组转化
6. df2 = pd.DataFrame(pd.np.arange(10).reshape(2, 5))
7. df3 = pd.DataFrame(pd.np.arange(12).reshape(3, 4), index=['00', '01', '02'],
8. columns=['a', 'b', 'c', 'd'])
9. # 随机产生
10. df4 = pd.DataFrame(pd.np.random.randn(3, 4), index=['00', '01', '02'],
11. columns=['a', 'b', 'c', 'd'])

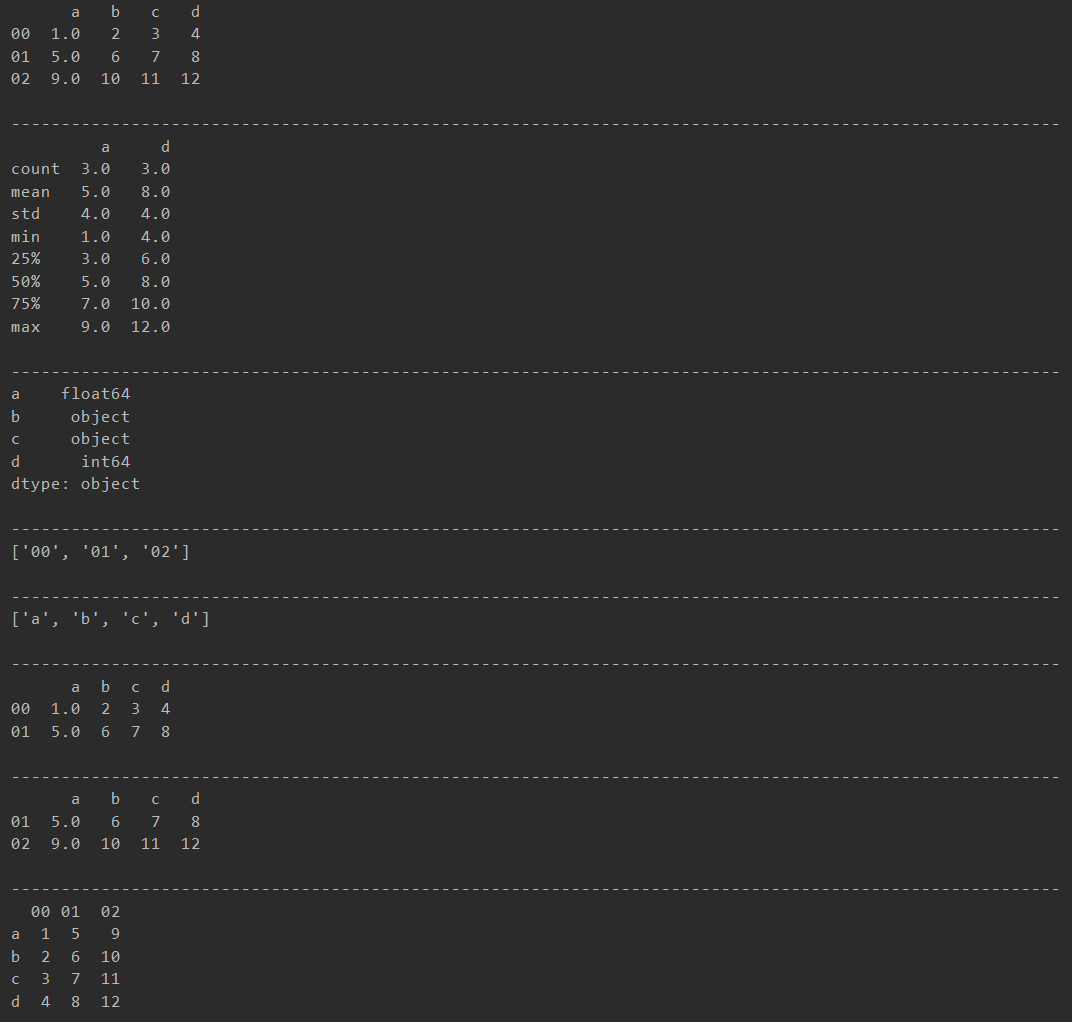


1. # 字典转化
2. # 嵌套列表为一列
3. \_dict = {'a': ['a0', 'a1'], 'b': ['b0', 'b1']}
4. df5 = pd.DataFrame(\_dict)
5. # 嵌套字典为一列
6. \_dict = {'a': {0: 'a0', 1: 'a1'}, 'b': {0: 'b0', 1: 'b1'}}
7. df6 = pd.DataFrame(\_dict)
8. # 指定列序
9. \_dict = {'col1': ['a1', 'b1'], 'col2': ['a2', 'b2']}
10. df7 = pd.DataFrame(\_dict, columns=['col2', 'col1'])

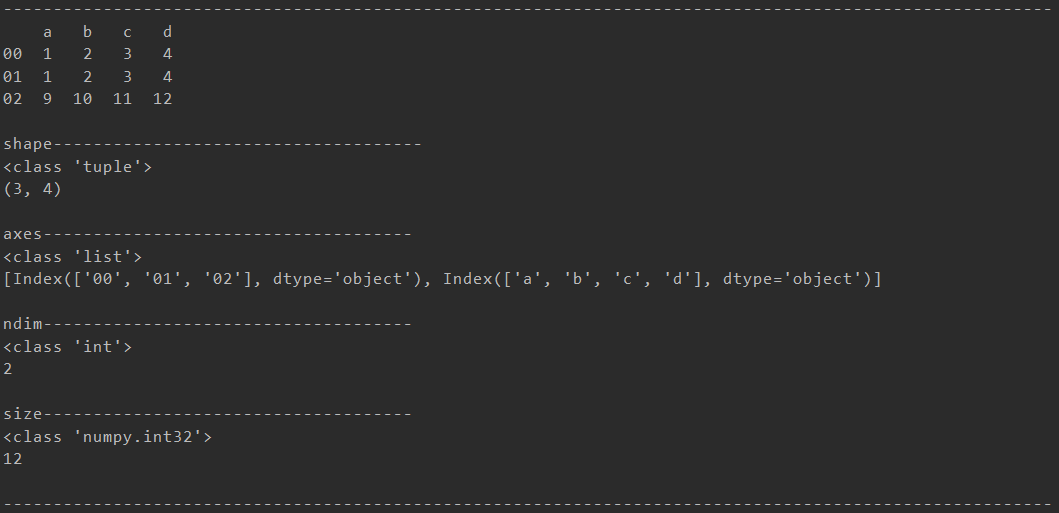


### 2.属性

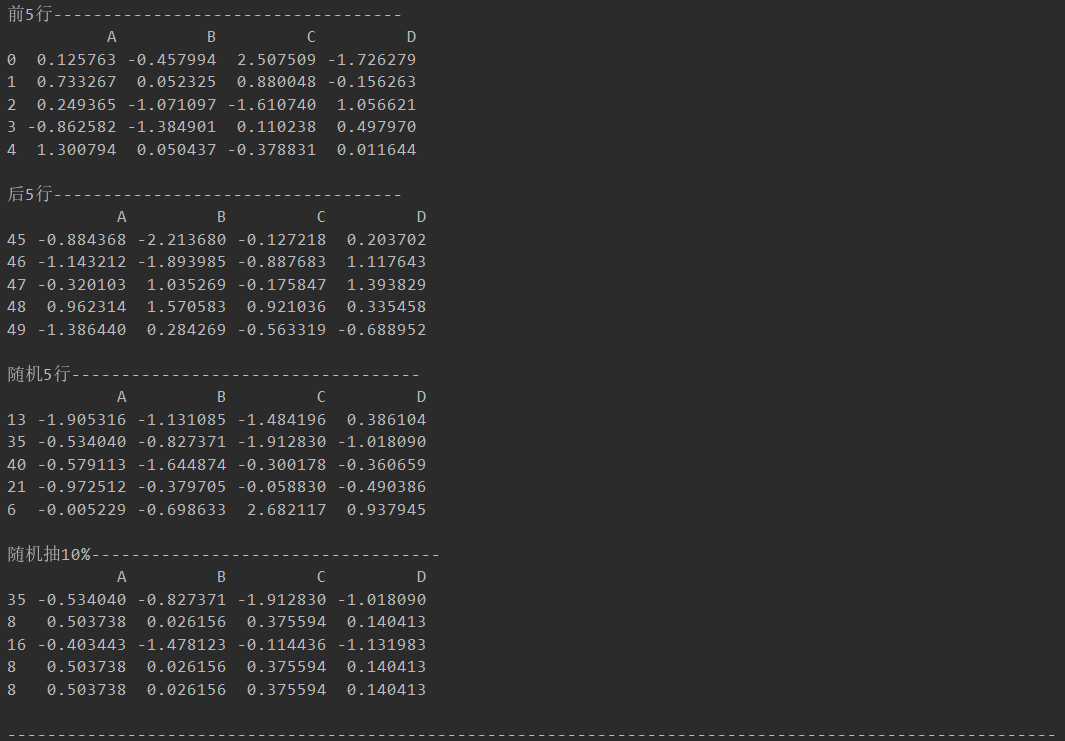
1. df = pd.DataFrame([[1, 2, '3', 4], [5, '6', '7', 8], [9.0, 10, '11', 12]],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\n---------------------------------------------------------------------------------------------------------')
6. # 描述
7. **print**(df.describe())
8. **print**('\n---------------------------------------------------------------------------------------------------------')
9. # 列类型
10. **print**(df.dtypes)
11. **print**('\n---------------------------------------------------------------------------------------------------------')
12. # 行列表
13. **print**(df.index.tolist())
14. **print**('\n---------------------------------------------------------------------------------------------------------')
15. # 列列表
16. **print**(df.columns.tolist())
17. **print**('\n---------------------------------------------------------------------------------------------------------')
18. # 查看前两行
19. **print**(df.head(2))
20. **print**('\n---------------------------------------------------------------------------------------------------------')
21. # 查看后两行
22. **print**(df.tail(2))
23. **print**('\n---------------------------------------------------------------------------------------------------------')
24. # 转置
25. **print**(df.T)



1. df = pd.DataFrame([[1, 2, 3, 4], [1, 2, 3, 4], [9, 10, 11, 12]],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\nshape-------------------------------------')
6. **print**(type(df.shape))
7. **print**(df.shape)
8. **print**('\naxes-------------------------------------')
9. **print**(type(df.axes))
10. **print**(df.axes)
11. **print**('\nndim-------------------------------------')
12. **print**(type(df.ndim))
13. **print**(df.ndim)
14. **print**('\nsize-------------------------------------')
15. **print**(type(df.size))
16. **print**(df.size)



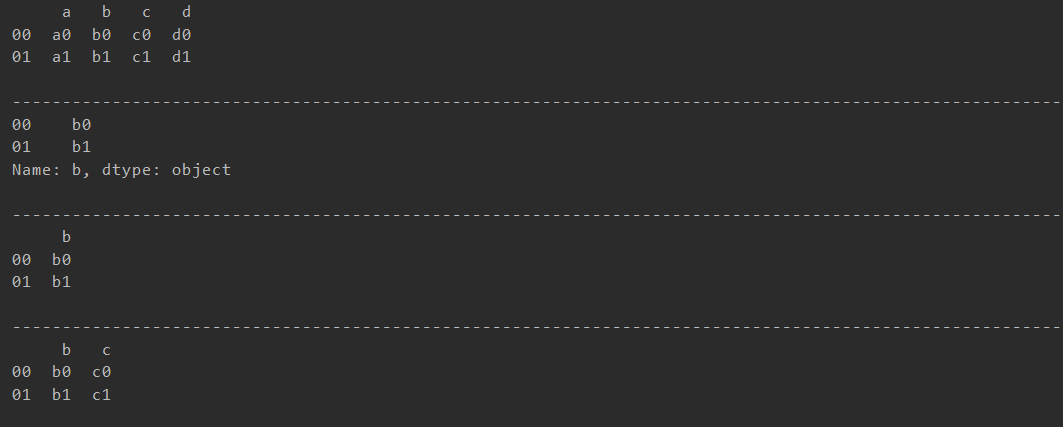
1. df = pd.DataFrame(pd.np.random.randn(50, 4), columns=list('ABCD'))
2. **print**(df)
3. **print**('\n前5行-----------------------------------')
4. **print**(df.head(5))
5. **print**('\n后5行-----------------------------------')
6. **print**(df.tail(5))
7. **print**('\n随机5行-----------------------------------')
8. **print**(df.sample(n=5))
9. **print**('\n随机抽10%-----------------------------------')
10. **print**(df.sample(frac=0.1, replace=True))



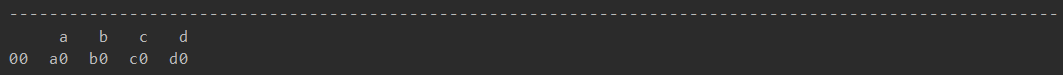
### 3.筛选

#### 切片

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=['00', '01'],
3. columns=['a', 'b', 'c', 'd'])
4. #切片筛选值
5. **print**(df)
6. **print**('\n---------------------------------------------------------------------------------------------------------')
7. # 取单列得Series
8. **print**(df['b'])
9. **print**('\n---------------------------------------------------------------------------------------------------------')
10. # 取单列得DataFrame
11. **print**(df[['b']])
12. **print**('\n---------------------------------------------------------------------------------------------------------')
13. # 取多列得DataFrame
14. **print**(df[['b', 'c']])

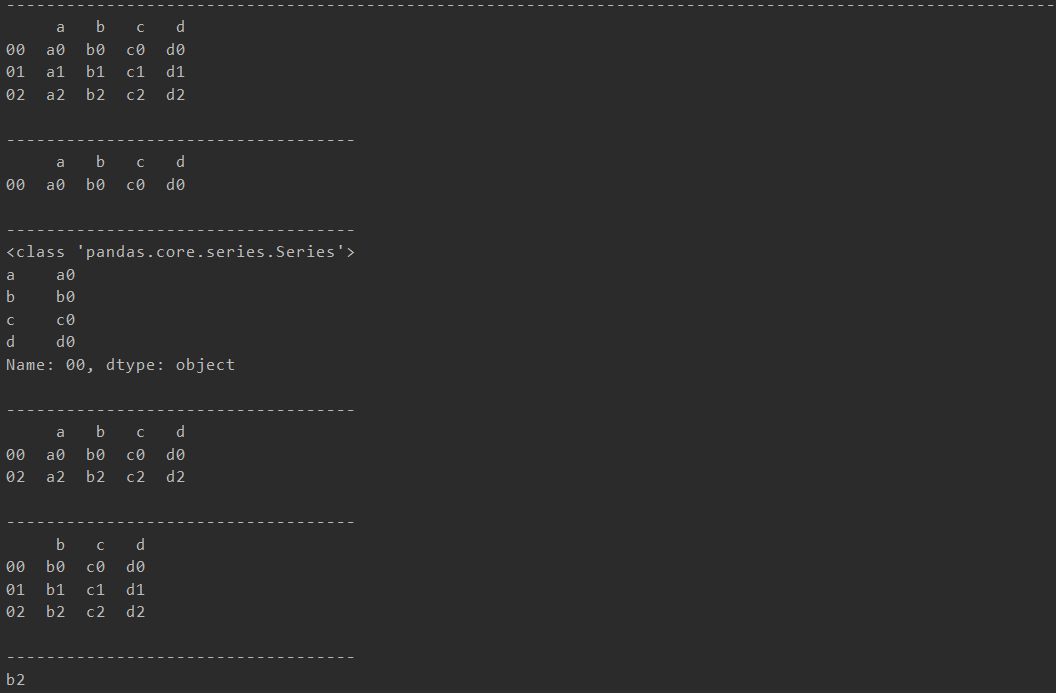


1. # 行选择
2. **print**(df[0:1])



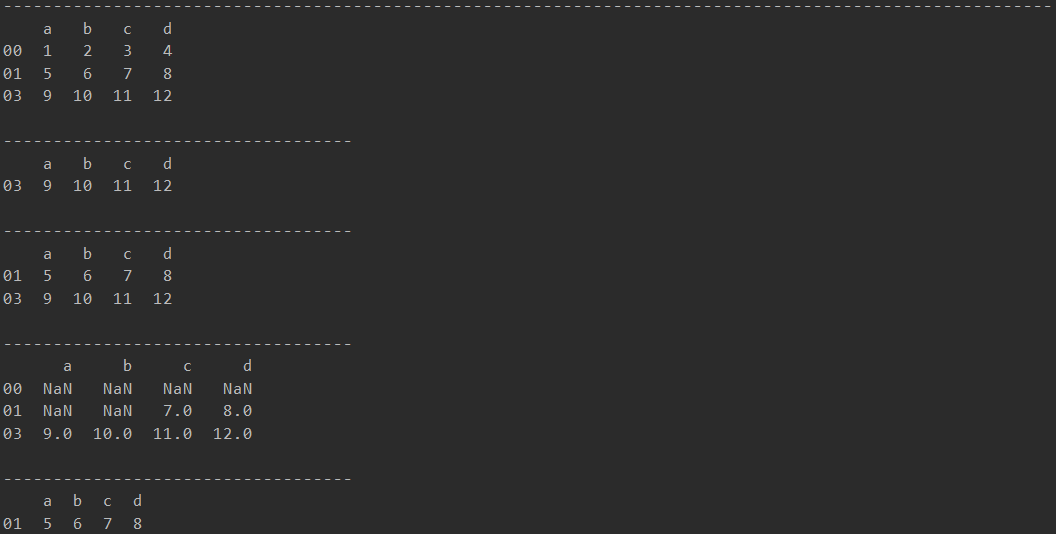
#### Loc

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1'], ['a2', 'b2', 'c2', 'd2']],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\n-----------------------------------')
6. # 选取行
7. **print**(df.loc[:'00', :])
8. **print**('\n-----------------------------------')
9. **print**(type(df.loc['00']))
10. **print**(df.loc['00'])
11. **print**('\n-----------------------------------')
12. **print**(df.loc[['00', '02']])
13. **print**('\n-----------------------------------')
14. # 选取列
15. **print**(df.loc[:, 'b':'d'])
17. **print**('\n-----------------------------------')
18. # 行列混合
19. **print**(df.loc['02', 'b'])



#### 值筛选

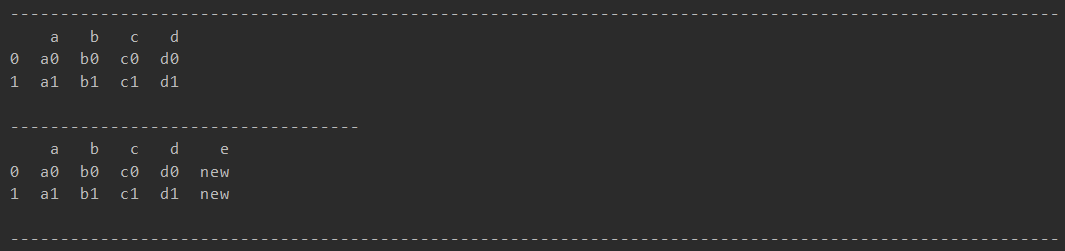
1. df = pd.DataFrame([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]],
2. index=['00', '01', '03'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\n-----------------------------------')
6. # 筛选某列大于某值的剩余行
7. **print**(df[df['b'] > 6])
8. **print**('\n-----------------------------------')
9. # 筛选某列值在某集合中的剩余行
10. **print**(df[df['b'].isin([6, 10])])
11. **print**('\n-----------------------------------')
12. # 所有大于6 的值
13. **print**(df[df > 6])
14. **print**('\n-----------------------------------')
15. # lambda
16. **print**(df[**lambda** df: df['b'] == 6])



### 4.增加

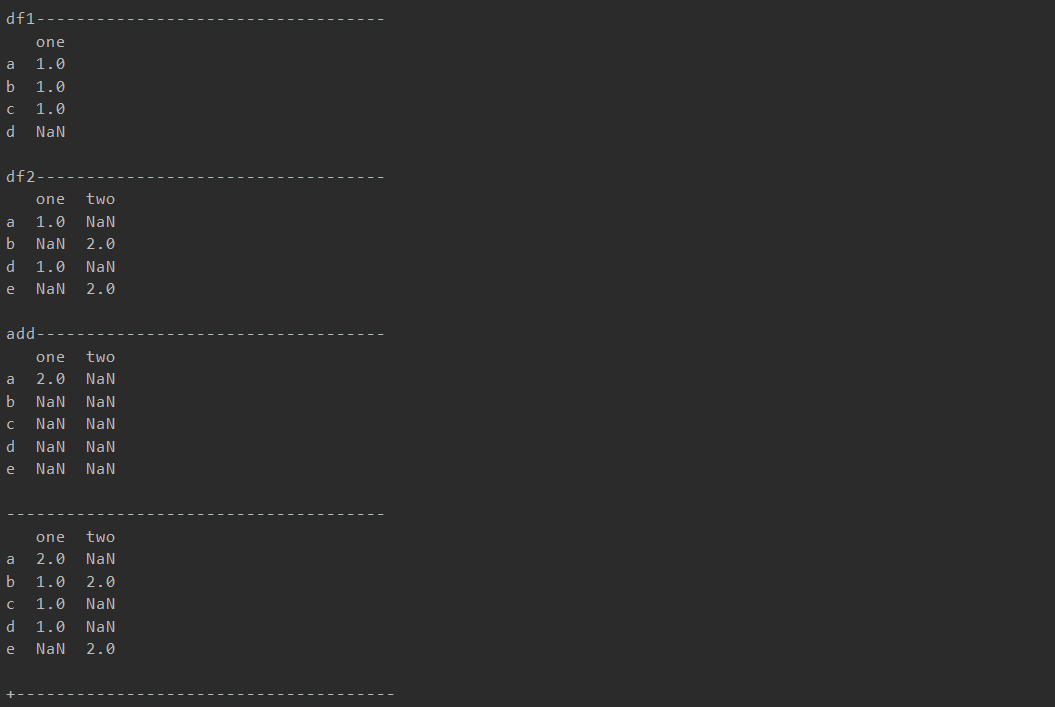
#### 增列

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\n-----------------------------------')
6. # 增加一列为相同值
7. df['e'] = 'new'



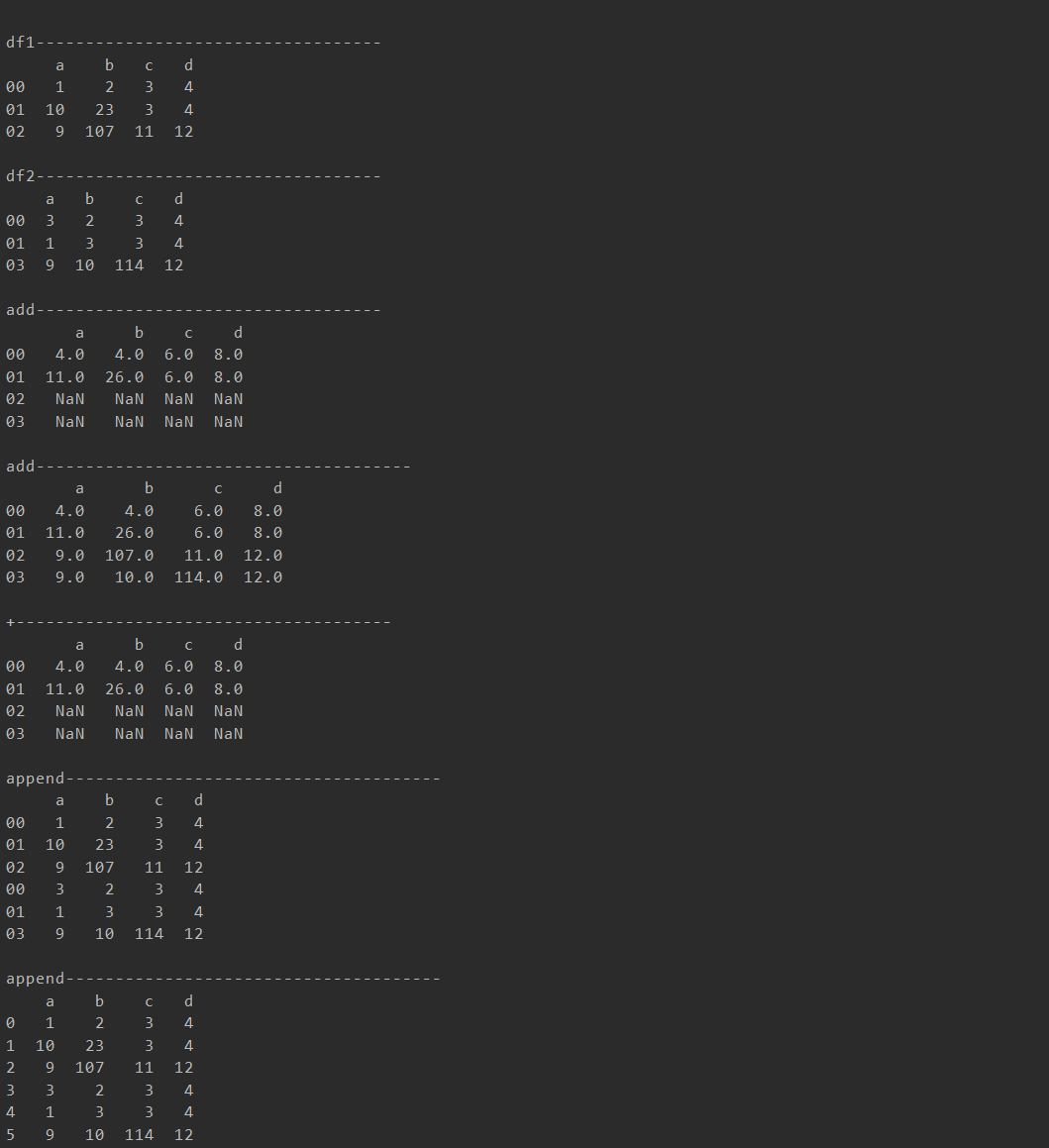
#### Add

1. df1 = pd.DataFrame([1, 1, 1, pd.np.nan], index=['a', 'b', 'c', 'd'],
2. columns=['one'])
3. df2 = pd.DataFrame(dict(one=[1, pd.np.nan, 1, pd.np.nan],
4. two=[pd.np.nan, 2, pd.np.nan, 2]),
5. index=['a', 'b', 'd', 'e'])
6. **print**('\ndf1-----------------------------------')
7. **print**(df1)
8. **print**('\ndf2-----------------------------------')
9. **print**(df2)
10. **print**('\nadd-----------------------------------')
11. **print**(df1.add(df2))
12. **print**('\n--------------------------------------')
13. **print**(df1.add(df2, fill\_value=0))



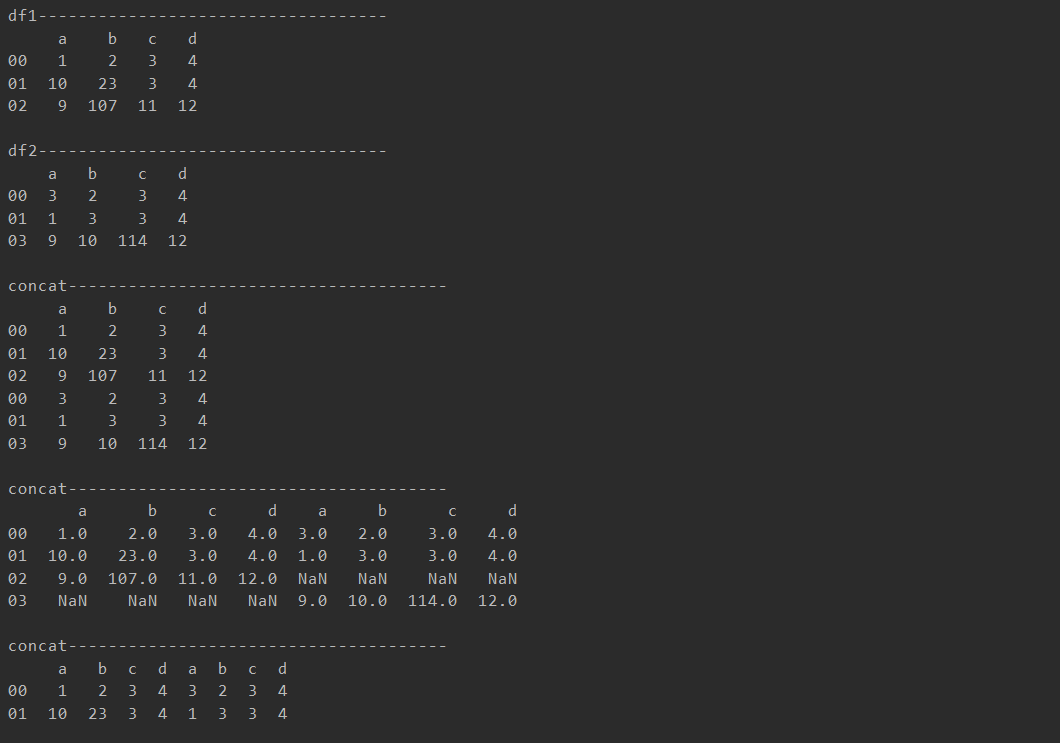
#### Append

1. df1 = pd.DataFrame([[1, 2, 3, 4], [10, 23, 3, 4], [9, 107, 11, 12]],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
5. df2 = pd.DataFrame([[3, 2, 3, 4], [1, 3, 3, 4], [9, 10, 114, 12]],
6. index=['00', '01', '03'],
7. columns=['a', 'b', 'c', 'd'])
8. **print**('\ndf1-----------------------------------')
9. **print**(df1)
10. **print**('\ndf2-----------------------------------')
11. **print**(df2)
12. **print**('\nadd-----------------------------------')
13. **print**(df1.add(df2))
14. **print**('\nadd--------------------------------------')
15. **print**(df1.add(df2, fill\_value=0))
16. **print**('\n+--------------------------------------')
17. **print**(df1+df2)
18. **print**('\nappend--------------------------------------')
19. **print**(df1.append(df2))
20. **print**('\nappend--------------------------------------')
21. **print**(df1.append(df2, ignore\_index=True))



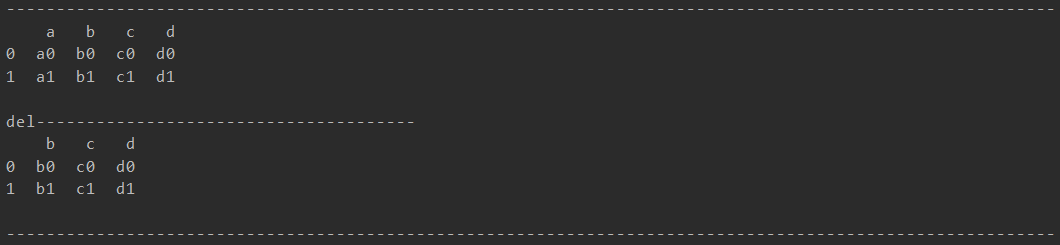
#### Concat

1. df1 = pd.DataFrame([[1, 2, 3, 4], [10, 23, 3, 4], [9, 107, 11, 12]],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
5. df2 = pd.DataFrame([[3, 2, 3, 4], [1, 3, 3, 4], [9, 10, 114, 12]],
6. index=['00', '01', '03'],
7. columns=['a', 'b', 'c', 'd'])
8. **print**('\ndf1-----------------------------------')
9. **print**(df1)
10. **print**('\ndf2-----------------------------------')
11. **print**(df2)
12. **print**('\nconcat--------------------------------------')
13. **print**(pd.concat([df1, df2]))
14. **print**('\nconcat--------------------------------------')
15. **print**(pd.concat([df1, df2], axis=1))
16. **print**('\nconcat--------------------------------------')
17. **print**(pd.concat([df1, df2], axis=1, join='inner'))



### 5.删除

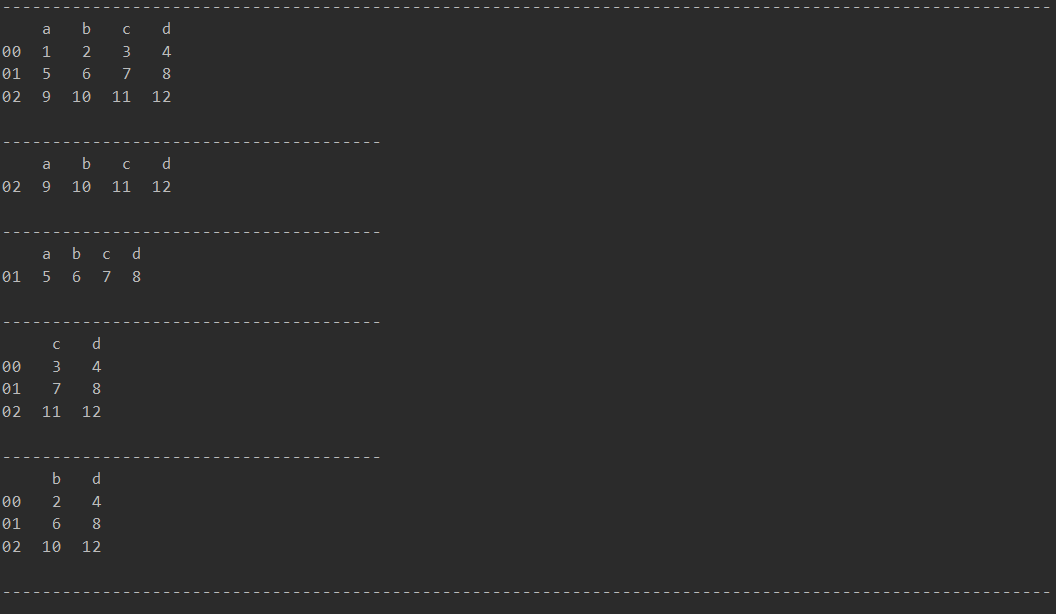
1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\ndel--------------------------------------')
6. # 真删除、无法多列删
7. **del** df['a']
8. **print**(df)



### 6.移除drop

**注**：使用drop删除时，删除的是视图！

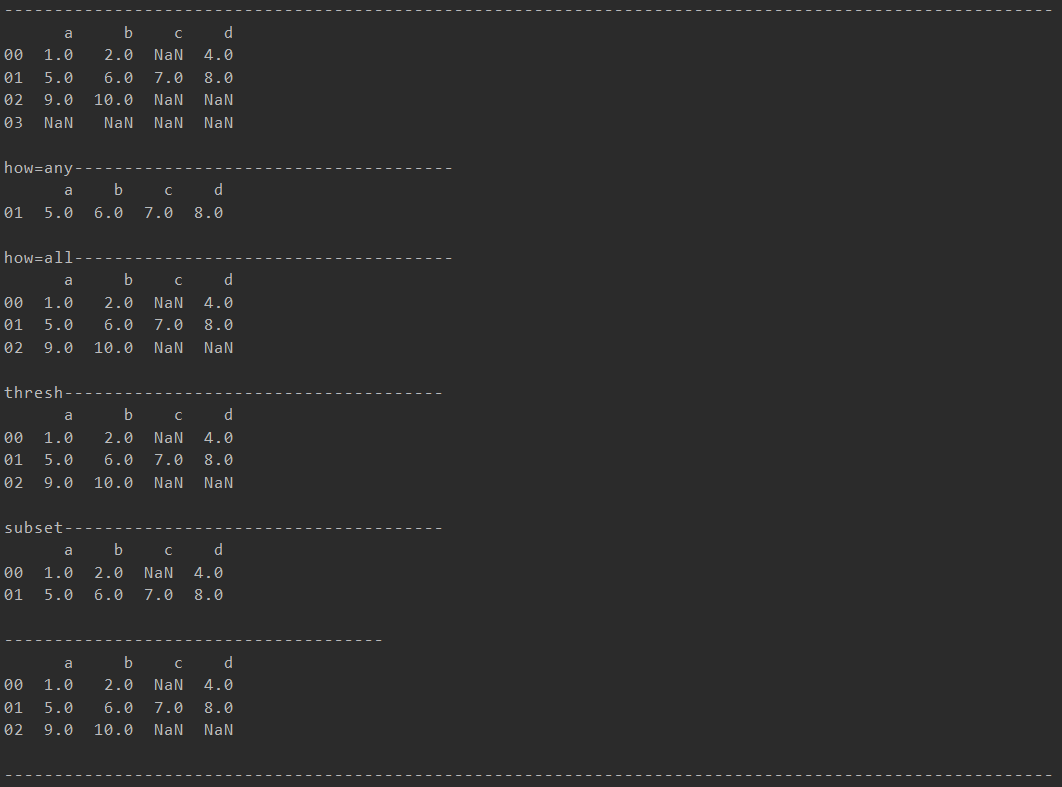
1. df = pd.DataFrame([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\n--------------------------------------')
6. # 移除行
7. **print**(df.drop(['00', '01']))
8. **print**('\n--------------------------------------')
9. **print**(df.drop(index=['00', '02']))
10. **print**('\n--------------------------------------')
12. # 移除列
13. **print**(df.drop(['a', 'b'], axis=1))
14. **print**('\n--------------------------------------')
15. df.drop(columns=['a', 'c'], inplace=True)
16. **print**(df)



### 7.去空dropna

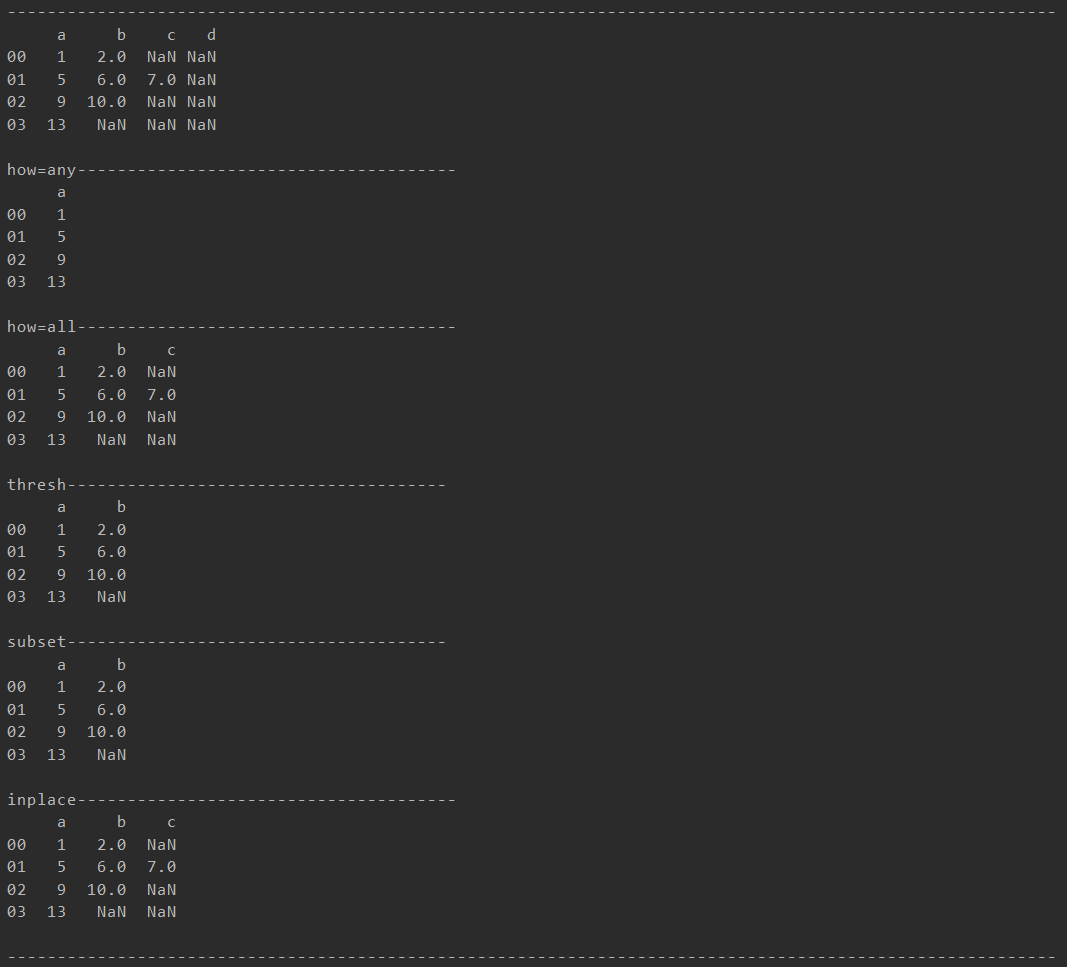
#### 行处理

1. df = pd.DataFrame([[1, 2, NaN, 4], [5, 6, 7, 8], [9, 10, NaN, NaN], [NaN, NaN, NaN, NaN]],
2. index=['00', '01', '02', '03'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\nhow=any--------------------------------------')
6. # 删除任意有丢失的行
7. **print**(df.dropna(how='any'))
8. **print**('\nhow=all--------------------------------------')
9. # 删除全丢失的行
10. **print**(df.dropna(how='all'))
11. **print**('\nthresh--------------------------------------')
12. # 保持一行最多有几个NaN
13. **print**(df.dropna(thresh=2))
14. **print**('\nsubset--------------------------------------')
15. # 删除某些列任意有NaN
16. **print**(df.dropna(subset=['a', 'd']))
17. **print**('\n--------------------------------------')
18. # 删除某些列全部为NaN
19. **print**(df.dropna(subset=['a', 'd'], how='all'))



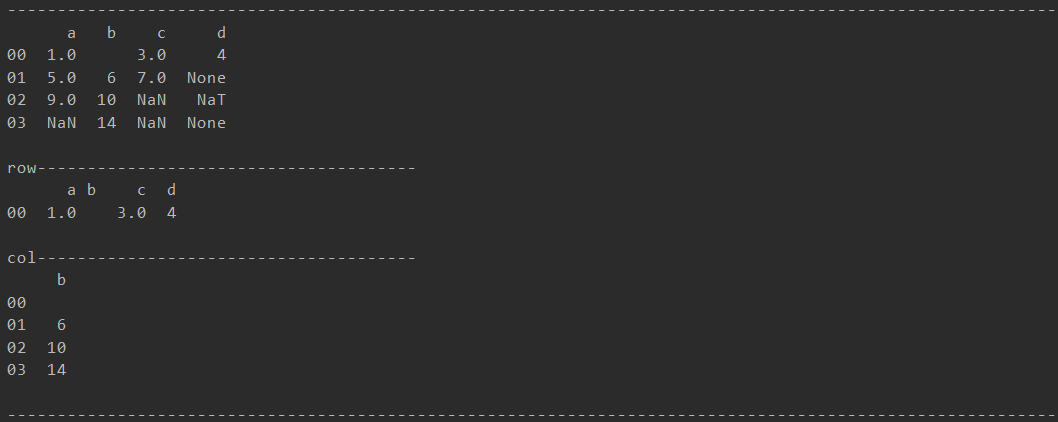
#### 列处理

1. df = pd.DataFrame([[1, 2, NaN, NaN], [5, 6, 7, NaN], [9, 10, NaN, NaN], [13, NaN, NaN, NaN]],
2. index=['00', '01', '02', '03'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\nhow=any--------------------------------------')
6. # 删除任意有丢失的列
7. **print**(df.dropna(how='any', axis=1))
8. **print**('\nhow=all--------------------------------------')
9. # 删除全丢失的列
10. **print**(df.dropna(how='all', axis=1))
11. **print**('\nthresh--------------------------------------')
12. # 保持一列最多有几个NaN
13. **print**(df.dropna(thresh=2, axis=1))
14. **print**('\nsubset--------------------------------------')
15. # 删除某些列任意有NaN
16. **print**(df.dropna(subset=['00', '01'], axis=1))
17. **print**('\ninplace--------------------------------------')
18. # # 删除某些列全部为NaN
19. df.dropna(subset=['00', '01'], how='all', axis=1, inplace=True)
20. **print**(df)



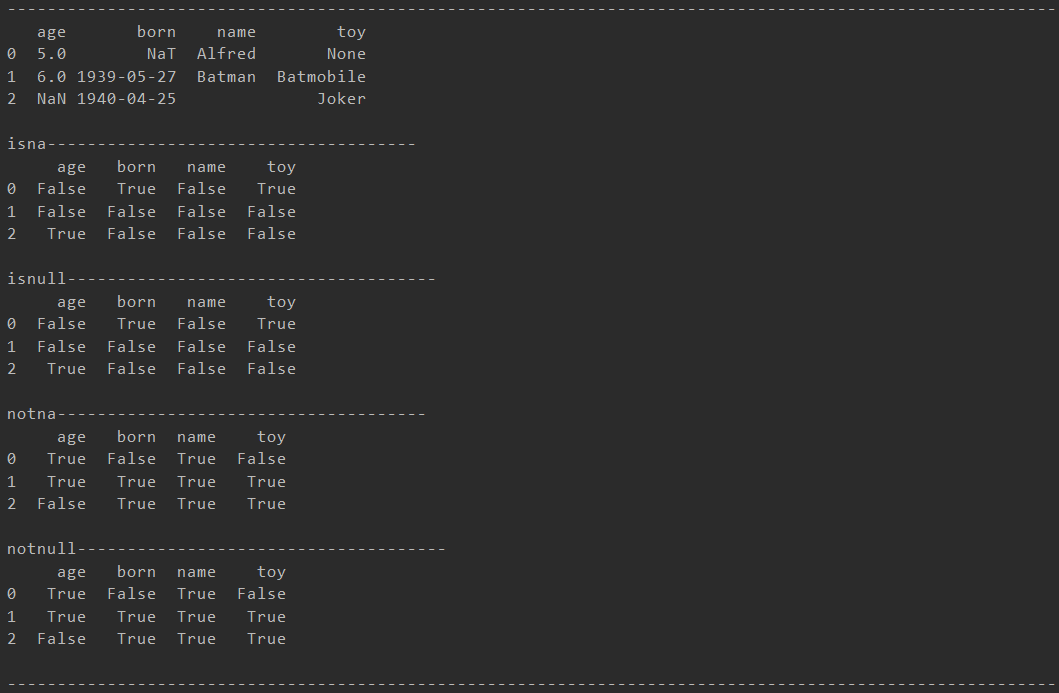
#### 不同的空

1. df = pd.DataFrame([[1, '', 3, 4], [5, 6, 7, None], [9, 10, pd.np.nan, pd.NaT], [None, 14, None, None]],
2. index=['00', '01', '02', '03'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\nrow--------------------------------------')
6. **print**(df.dropna())
7. **print**('\ncol--------------------------------------')
8. **print**(df.dropna(axis=1))



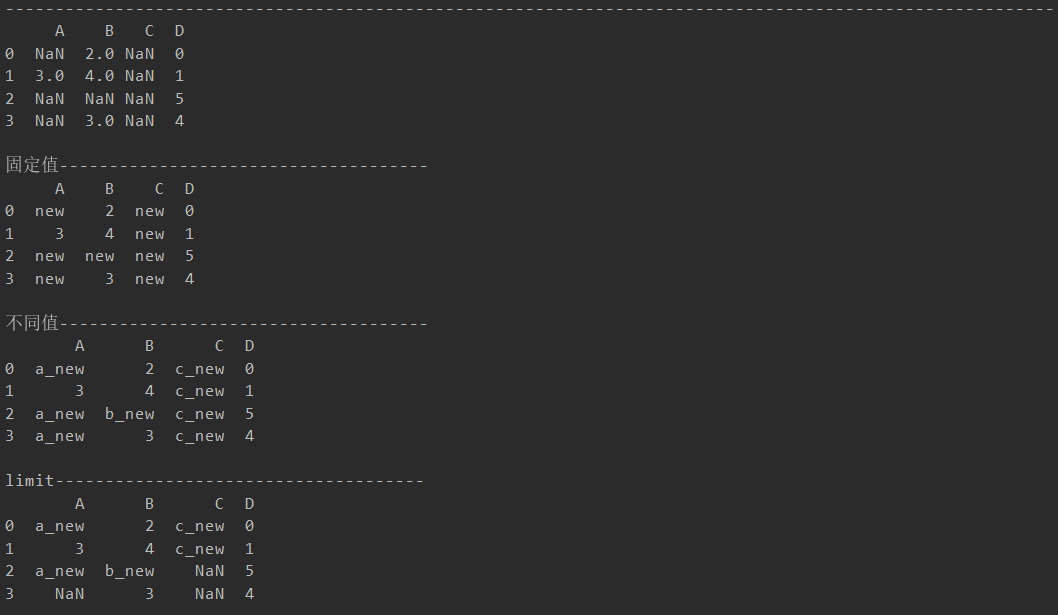
### 8.判空isna

1. df = pd.DataFrame({'age': [5, 6, NaN],
2. 'born': [pd.NaT, pd.Timestamp('1939-05-27'),
3. pd.Timestamp('1940-04-25')],
4. 'name': ['Alfred', 'Batman', ''],
5. 'toy': [None, 'Batmobile', 'Joker']})
6. **print**(df)
7. **print**('\nisna-------------------------------------')
8. # 判断是否为空
9. **print**(df.isna())
10. **print**('\nisnull-------------------------------------')
11. **print**(df.isnull())
12. **print**('\nnotna-------------------------------------')
13. **print**(df.notna())
14. **print**('\nnotnull-------------------------------------')
15. **print**(df.notnull())

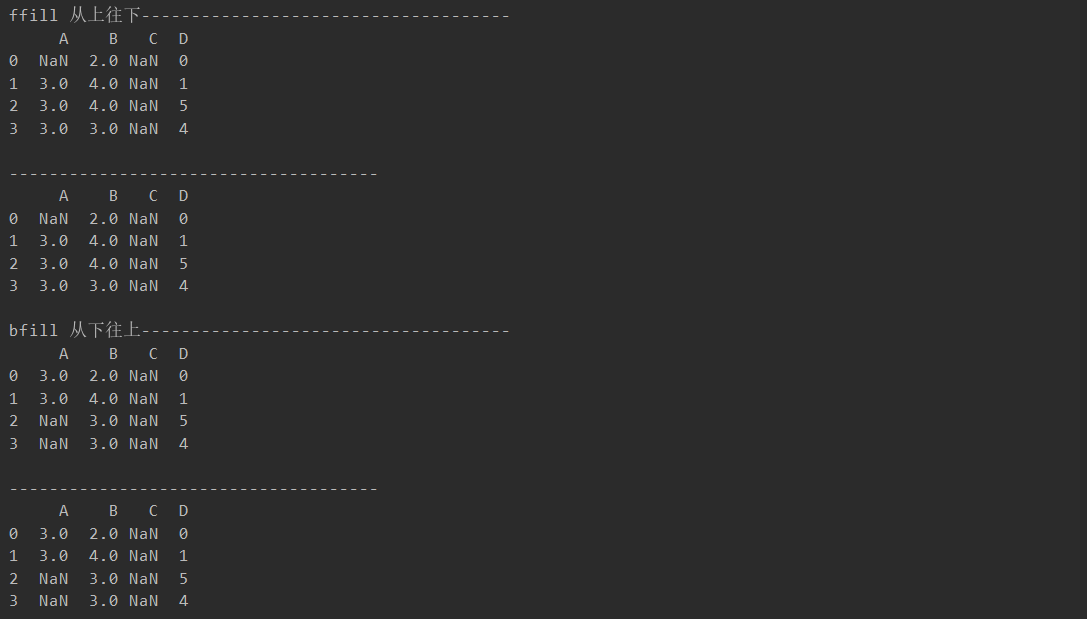


### 9.填空fillna

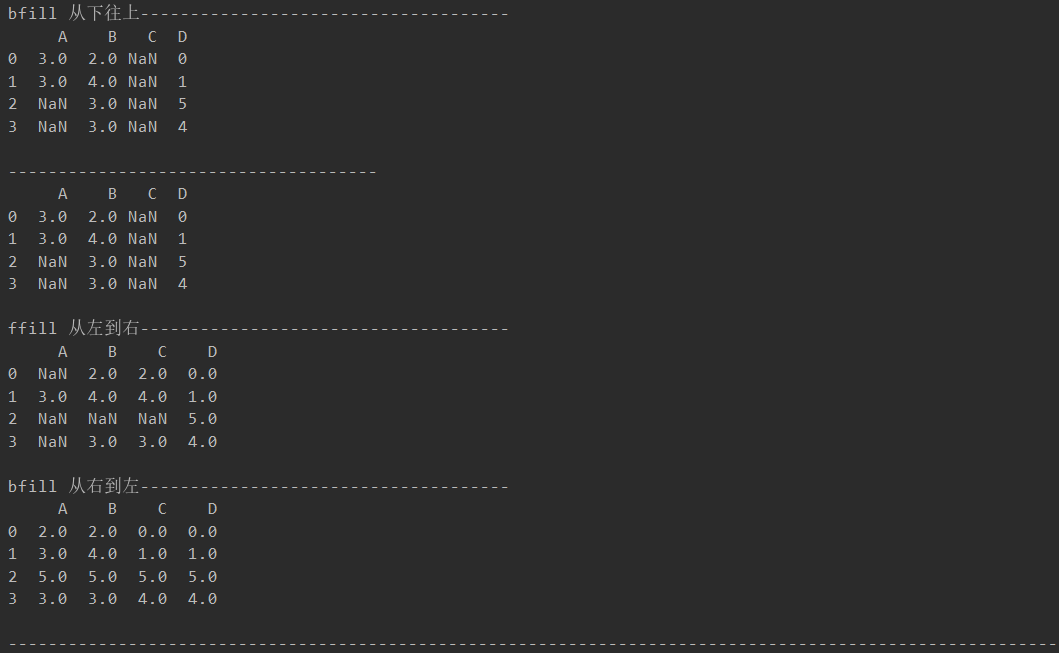
1. df = pd.DataFrame([[NaN, 2, NaN, 0],
2. [3, 4, NaN, 1],
3. [NaN, NaN, NaN, 5],
4. [NaN, 3, NaN, 4]], columns=list('ABCD'))
5. **print**(df)
6. **print**('\n固定值-------------------------------------')
7. # 将所有NaN替换为固定值
8. **print**(df.fillna('new'))
9. **print**('\n不同值-------------------------------------')
10. # 将NaN值不同行改为不同值
11. values = {'A': 'a\_new', 'B': 'b\_new', 'C': 'c\_new', 'D': 'd\_new'}
12. **print**(df.fillna(value=values))
13. **print**('\nlimit-------------------------------------')
14. # 修改固定数量
15. **print**(df.fillna(value=values, limit=2))



1. **print**('\nffill 从上往下-------------------------------------')
2. # 将NaN值改为上一个值 从上往下
3. **print**(df.fillna(method='ffill'))
4. **print**('\n-------------------------------------')
5. # 等价
6. **print**(df.ffill())
7. **print**('\nbfill 从下往上-------------------------------------')
8. # 将NaN值改为下一个值
9. **print**(df.fillna(method='bfill'))
10. **print**('\n-------------------------------------')
11. # 等价
12. **print**(df.bfill())

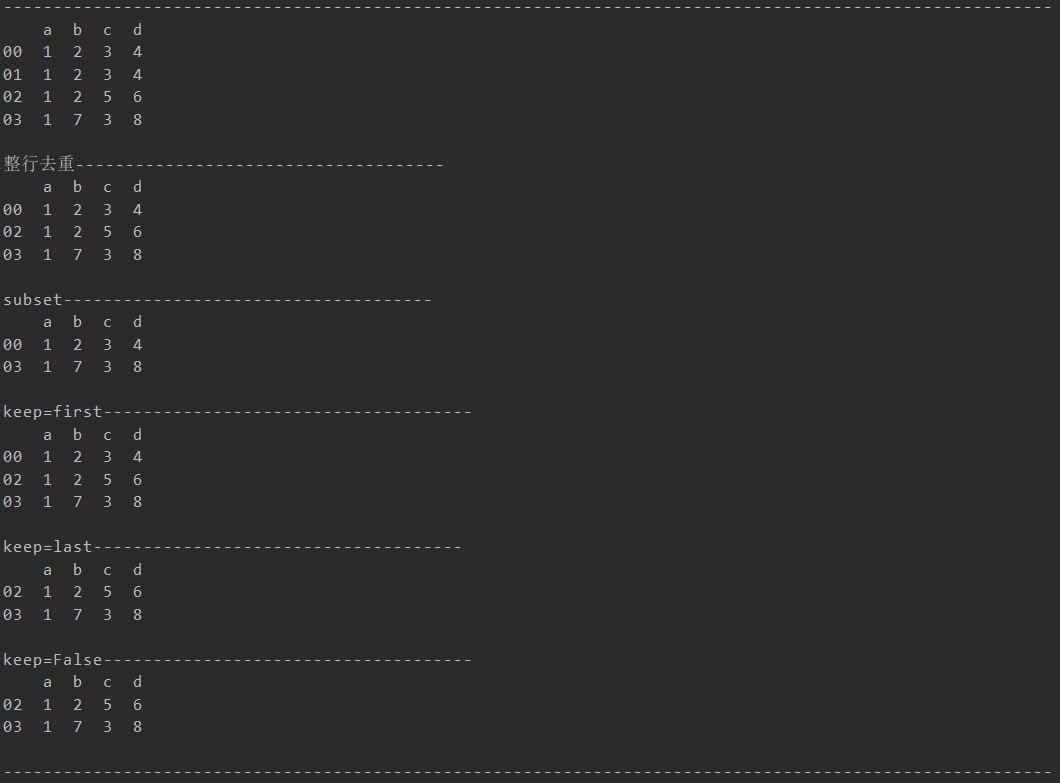


1. **print**('\nffill 从左到右-------------------------------------')
2. # 从左到右，替换NaN
3. **print**(df.ffill(axis=1))
4. **print**('\nbfill 从右到左-------------------------------------')
5. # 从右到左，替换NaN
6. **print**(df.bfill(axis=1))



### 10.去重drop\_duplicates

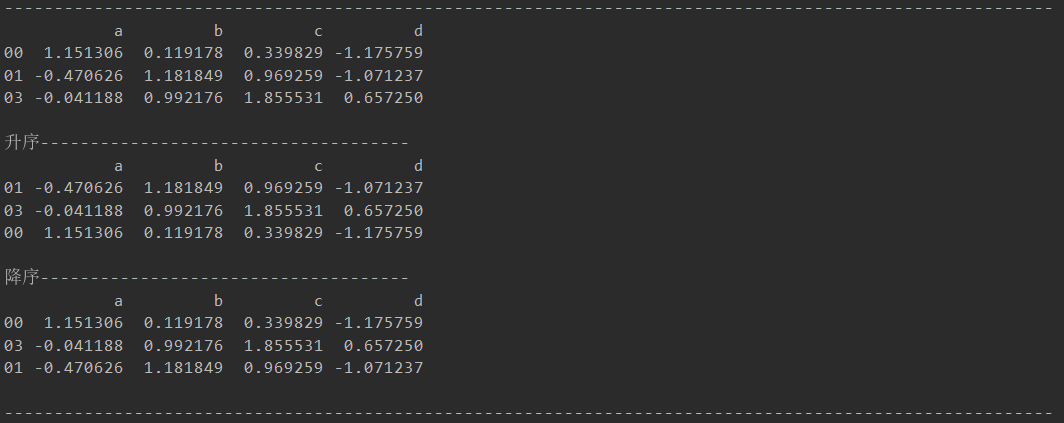
1. df = pd.DataFrame([[1, 2, 3, 4], [1, 2, 3, 4], [1, 2, 5, 6], [1, 7, 3, 8]],
2. index=['00', '01', '02', '03'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\n整行去重-------------------------------------')
6. # 整行去重
7. **print**(df.drop\_duplicates())
8. **print**('\nsubset-------------------------------------')
9. # 选某几行去重
10. **print**(df.drop\_duplicates(subset=['a', 'b']))
11. **print**('\nkeep=first-------------------------------------')
12. # 重复默认保持第一行
13. **print**(df.drop\_duplicates(keep='first'))
14. **print**('\nkeep=last-------------------------------------')
15. # 重复保持最后一行
16. **print**(df.drop\_duplicates(keep='last', subset=['a', 'b']))
17. **print**('\nkeep=False-------------------------------------')
18. # 删除所有重复项
19. df.drop\_duplicates(keep=False, inplace=True)
20. **print**(df)



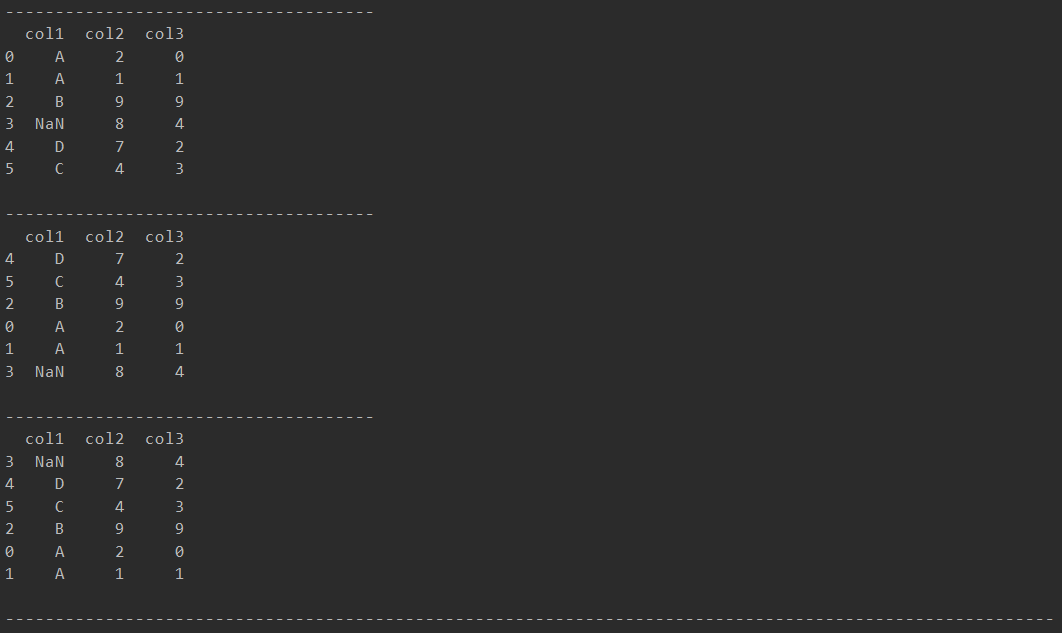
### 排序

#### 按值

1. df = pd.DataFrame(pd.np.random.randn(3, 4), index=['00', '01', '03'],
2. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\n升序-------------------------------------')
6. # 默认升序 先排a列，a列相同排b列
7. **print**(df.sort\_values(by=['a', 'b']))
8. **print**('\n降序-------------------------------------')
9. # 降序排
10. **print**(df.sort\_values(by='a', ascending=False))

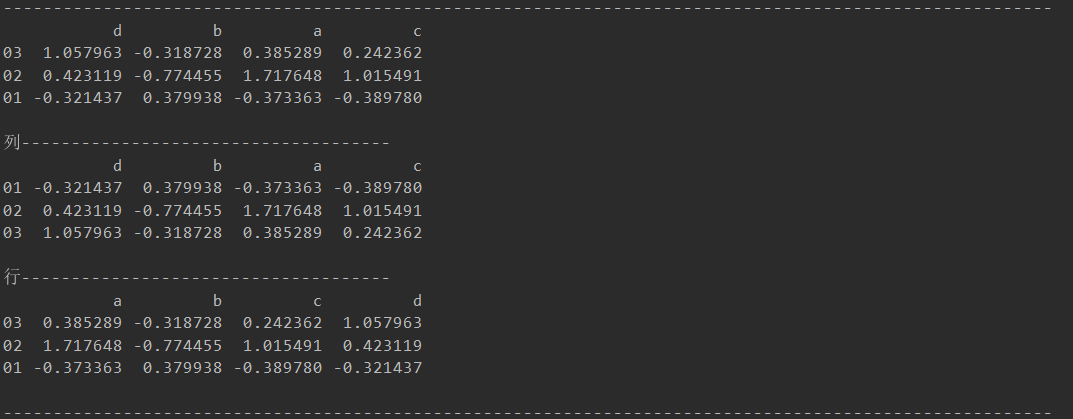


1. df = pd.DataFrame({
2. 'col1': ['A', 'A', 'B', pd.np.nan, 'D', 'C'],
3. 'col2': [2, 1, 9, 8, 7, 4],
4. 'col3': [0, 1, 9, 4, 2, 3],
5. })
6. # nan值默认位置
7. **print**(df)
8. **print**('\n-------------------------------------')
9. **print**(df.sort\_values(by='col1', ascending=False))
10. **print**('\n-------------------------------------')
11. **print**(df.sort\_values(by='col1', ascending=False, na\_position='first'))



#### 按轴

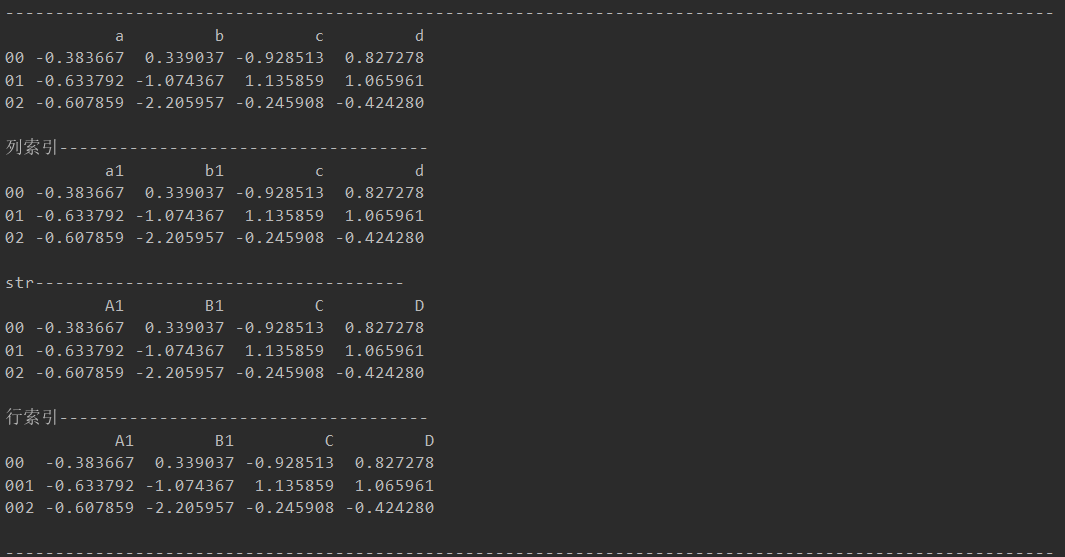
1. df = pd.DataFrame(pd.np.random.randn(3, 4), index=['03', '02', '01'],
2. columns=['d', 'b', 'a', 'c'])
3. **print**(df)
4. **print**('\n列索引-------------------------------------')
5. **print**(df.sort\_index())
6. **print**('\n行索引-------------------------------------')
7. **print**(df.sort\_index(axis=1))



### 换名

#### Rename

1. df = pd.DataFrame(pd.np.random.randn(3, 4), index=['00', '01', '02'],
2. columns=['a', 'b', 'c', 'd'])
3. **print**(df)
4. **print**('\n列索引-------------------------------------')
5. # 列换名
6. df.rename(columns={'a': 'a1', 'b': 'b1'}, inplace=True)
7. **print**(df)
8. **print**('\nstr-------------------------------------')
9. # 应用str
10. df.rename(str.upper, axis='columns', inplace=True)
11. **print**(df)
12. **print**('\n行索引-------------------------------------')
13. # 行换名
14. df.rename({'01': '001', '02': '002'}, axis='index', inplace=True)
15. **print**(df)



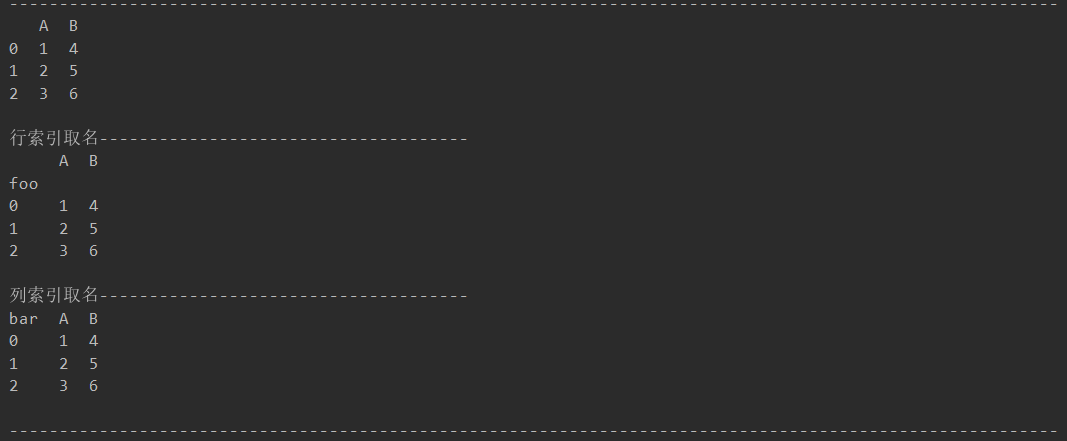
#### Set\_axis

1. df = pd.DataFrame([[1, 2, 3, 4], [1, 2, 3, 4], [9, 10, 11, 12]],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\n列索引-------------------------------------')
6. **print**(df.set\_axis(['I', 'II', 'III', 'IIII'], axis=1, inplace=False))
7. **print**('\n行索引-------------------------------------')
8. **print**(df.set\_axis(['I', 'II', 'III'], axis=0, inplace=False))



#### Rename\_axis

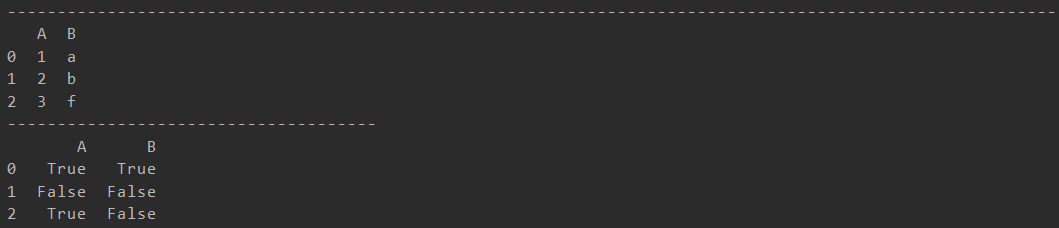
1. df = pd.DataFrame({"A": [1, 2, 3], "B": [4, 5, 6]})
2. **print**(df)
3. **print**('\n行索引取名-------------------------------------')
4. **print**(df.rename\_axis("foo"))
5. **print**('\n列索引取名-------------------------------------')
6. **print**(df.rename\_axis("bar", axis="columns"))



### 13.包含isin

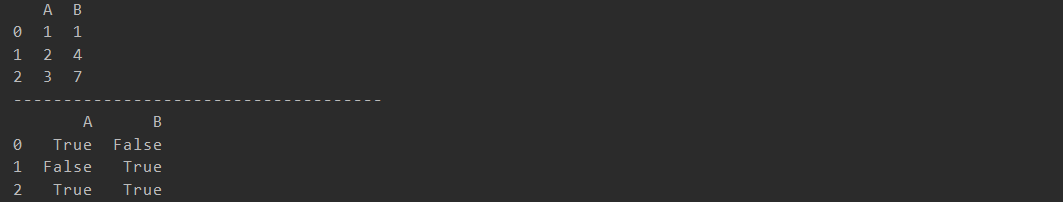
#### 列表

1. df = pd.DataFrame({'A': [1, 2, 3], 'B': ['a', 'b', 'f']})
2. **print**(df)
3. **print**('-------------------------------------')
4. **print**(df.isin([1, 3, 12, 'a']))



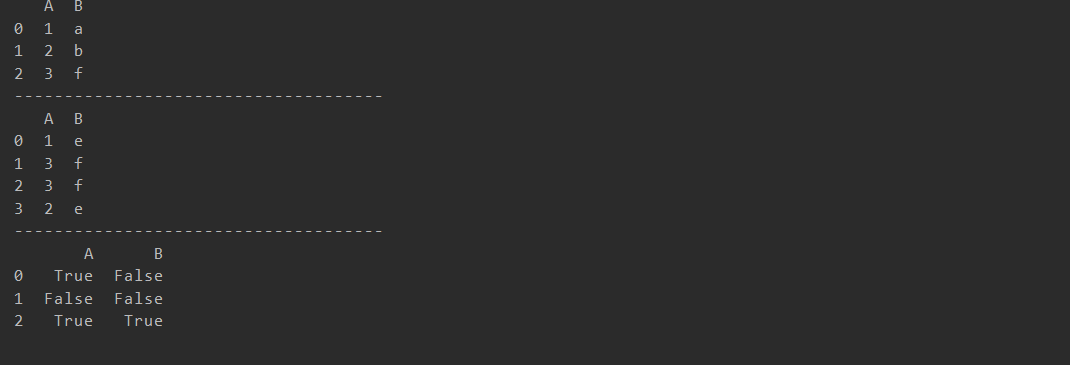
#### 字典

1. df = pd.DataFrame({'A': [1, 2, 3], 'B': [1, 4, 7]})
2. **print**(df)
3. **print**('-------------------------------------')
4. **print**(df.isin({'A': [1, 3], 'B': [4, 7, 12]}))



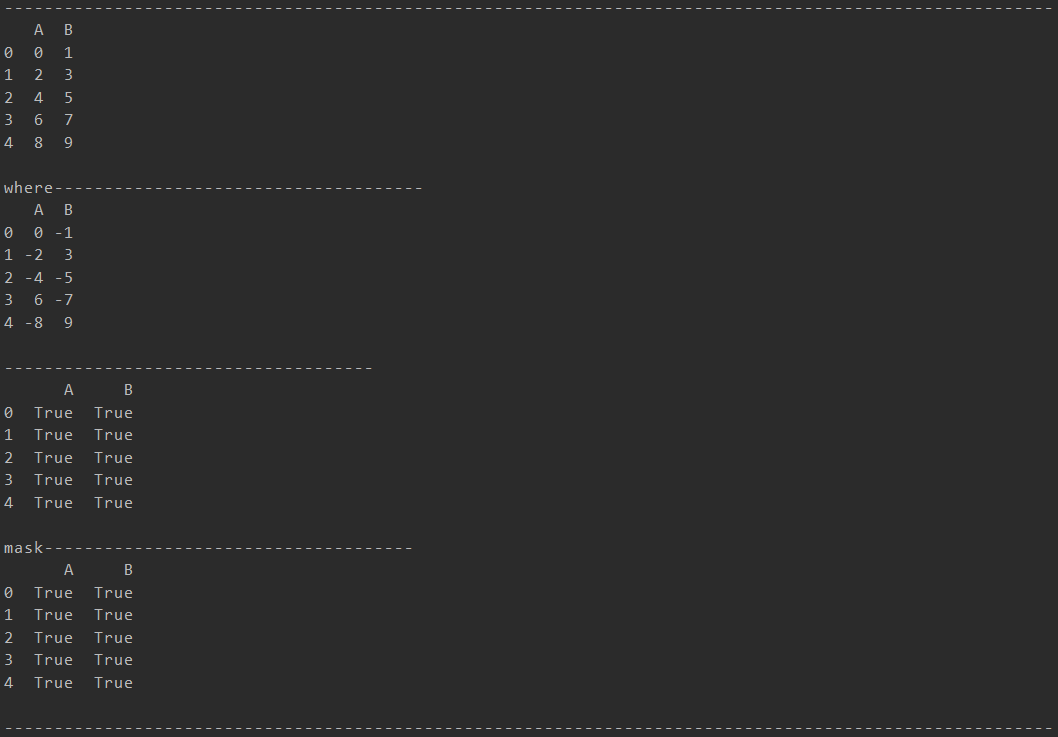
#### DataFrame

1. df = pd.DataFrame({'A': [1, 2, 3], 'B': ['a', 'b', 'f']})
2. **print**(df)
3. **print**('-------------------------------------')
4. other = pd.DataFrame({'A': [1, 3, 3, 2], 'B': ['e', 'f', 'f', 'e']})
5. **print**(other)
6. **print**('-------------------------------------')
7. **print**(df.isin(other))



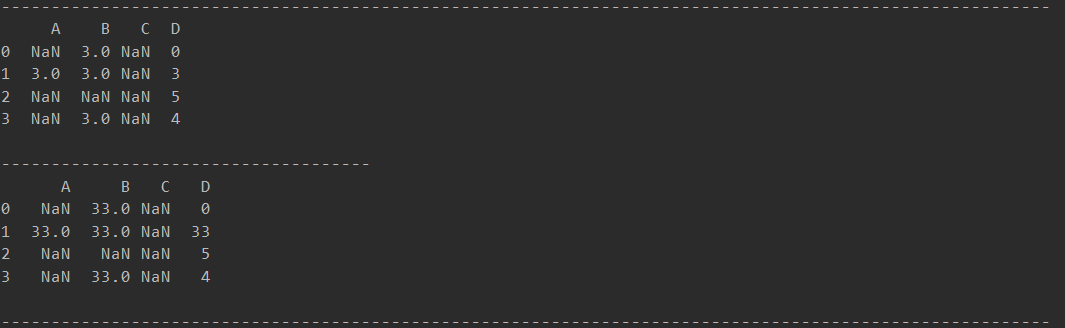
### 14.判断where mask

1. df = pd.DataFrame(pd.np.arange(10).reshape(-1, 2), columns=['A', 'B'])
2. m = df % 3 == 0
3. **print**(df)
4. **print**('\nwhere-------------------------------------')
5. # 条件成立不修改
6. **print**(df.where(m, -df))
7. **print**('\n-------------------------------------')
8. # 等价
9. **print**(df.where(m, -df) == pd.np.where(m, df, -df))
10. **print**('\nmask-------------------------------------')
11. # 取反
12. **print**(df.where(m, -df) == df.mask(~m, -df))



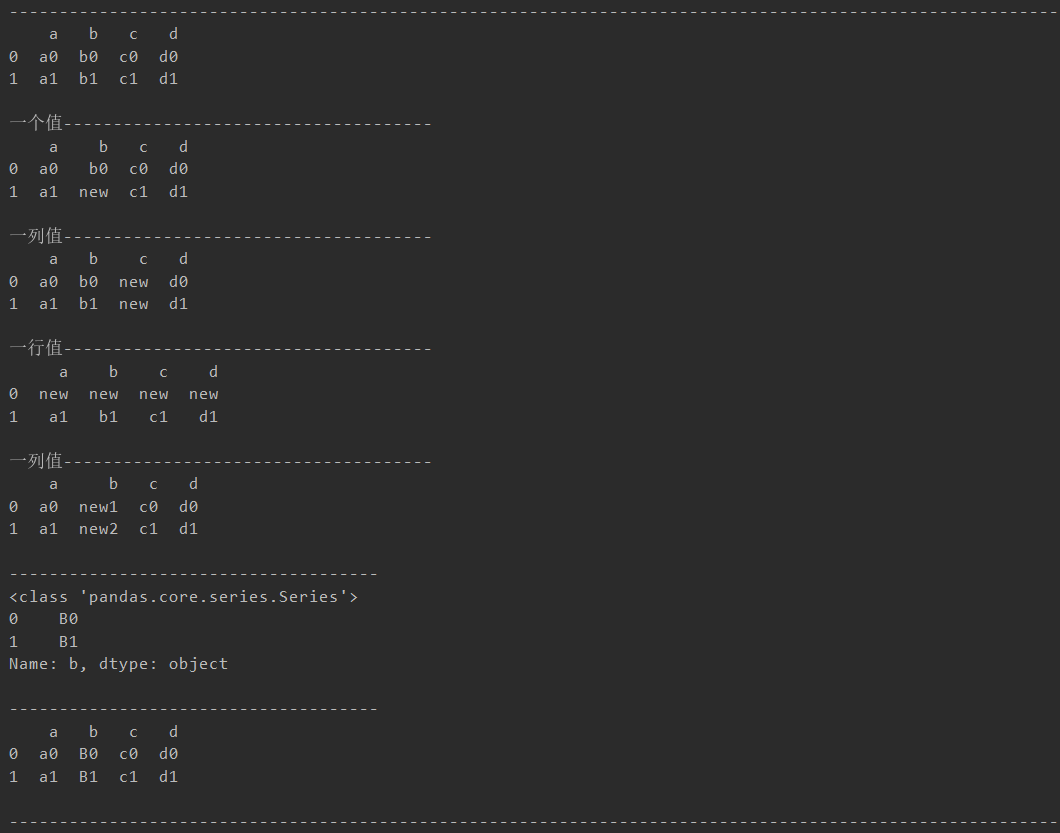
### 15.替换replace

1. df = pd.DataFrame([[NaN, 3, NaN, 0],
2. [3, 3, NaN, 3],
3. [NaN, NaN, NaN, 5],
4. [NaN, 3, NaN, 4]], columns=list('ABCD'))
5. **print**(df)
6. **print**('\n-------------------------------------')
7. # 旧值
8. **print**(df.replace(3, 33))



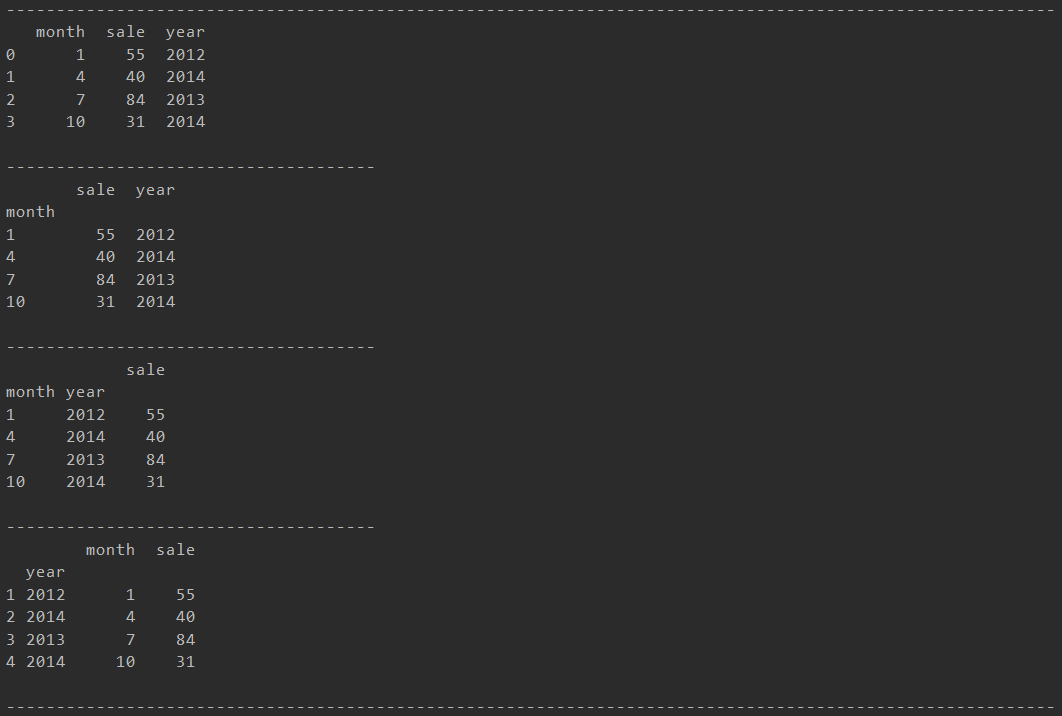
### 16更新

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\n一个值-------------------------------------')
6. # 具体值
7. df1 = df.copy()
8. df1['b'][1] = 'new'
9. **print**(df1)
11. **print**('\n一列值-------------------------------------')
12. # 修改一列为同一值
13. df2 = df.copy()
14. df2['c'] = 'new'
15. **print**(df2)
17. **print**('\n一行值-------------------------------------')
18. # 修改一行为同一值
19. df3 = df.copy()
20. df3[:1] = 'new'
21. **print**(df3)
23. **print**('\n一列值-------------------------------------')
24. # 修改一列为Series
25. df4 = df.copy()
26. df4['b'] = pd.Series(['new1', 'new2', 'new3'])
27. **print**(df4)
29. **print**('\n-------------------------------------')
30. df5 = df.copy()
31. df5 = df5['b'].str.upper()
32. **print**(type(df5))
33. **print**(df5)
35. **print**('\n-------------------------------------')
36. df['b'] = df['b'].str.upper()
37. **print**(df)



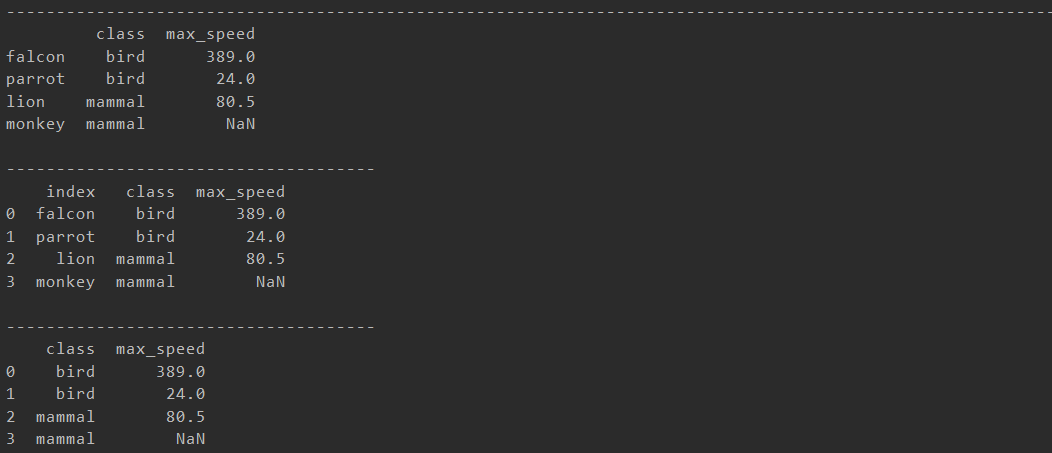
### 17.列转index

1. df = pd.DataFrame({'month': [1, 4, 7, 10],
2. 'year': [2012, 2014, 2013, 2014],
3. 'sale': [55, 40, 84, 31]})
4. **print**(df)
5. **print**('\n-------------------------------------')
6. # 将一列作为index
7. **print**(df.set\_index('month'))
8. **print**('\n-------------------------------------')
9. # 将几列组合作为index
10. **print**(df.set\_index(['month', 'year']))
11. **print**('\n-------------------------------------')
12. # 将index重新赋值
13. **print**(df.set\_index([[1, 2, 3, 4], 'year']))

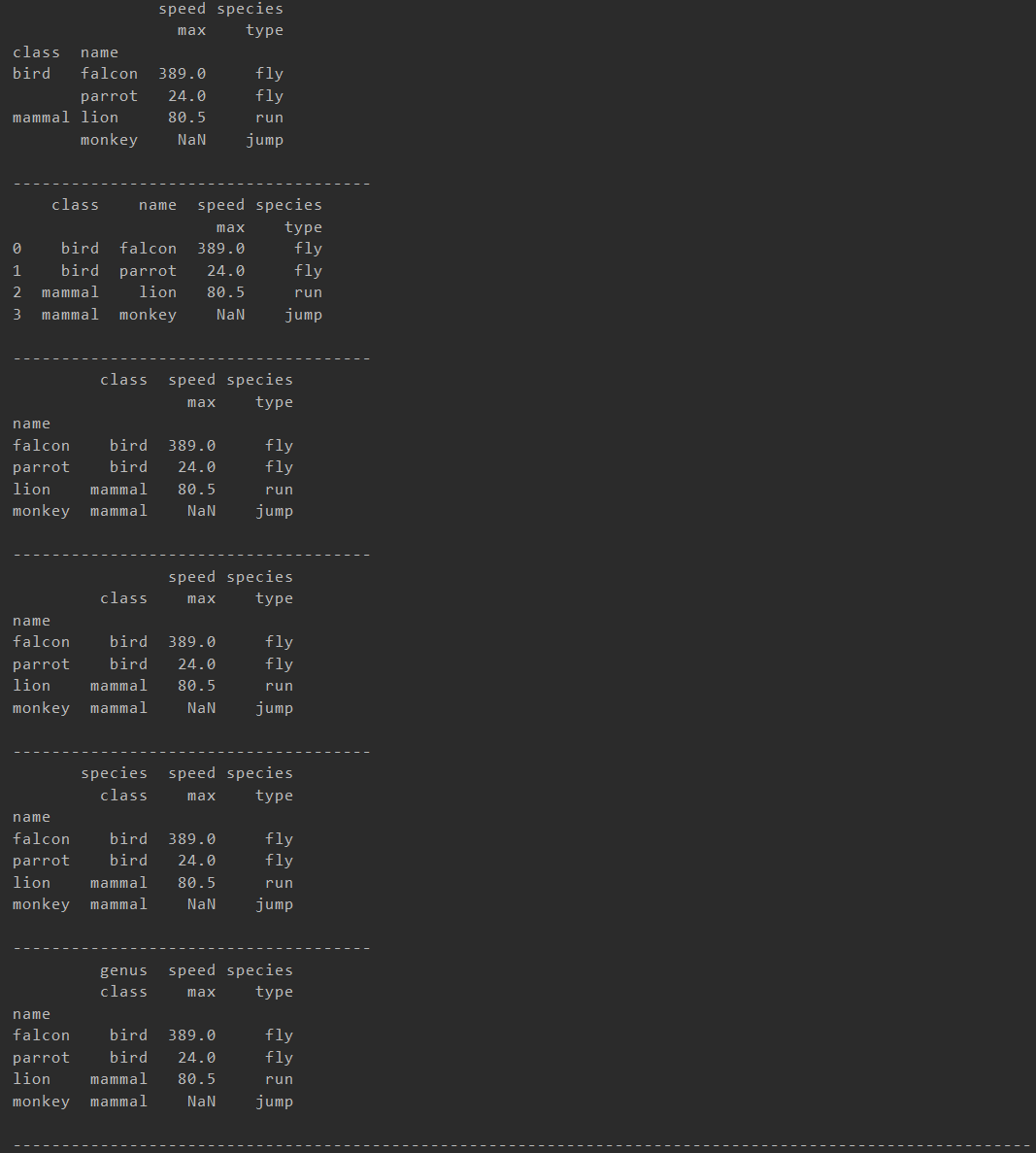


### 18.index转列

1. df = pd.DataFrame([('bird', 389.0),
2. ('bird', 24.0),
3. ('mammal', 80.5),
4. ('mammal', NaN)],
5. index=['falcon', 'parrot', 'lion', 'monkey'],
6. columns=('class', 'max\_speed'))
8. **print**(df)
9. **print**('\n-------------------------------------')
10. # 将index作为列名为index的一列
11. **print**(df.reset\_index())
12. **print**('\n-------------------------------------')
13. # 将index去掉并改为索引
14. **print**(df.reset\_index(drop=True))

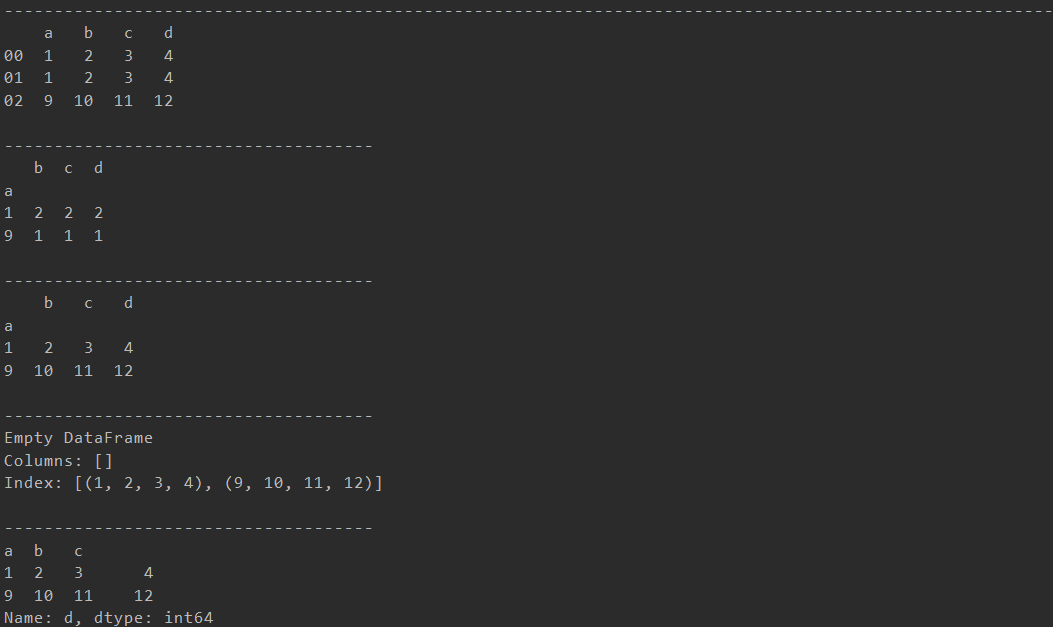


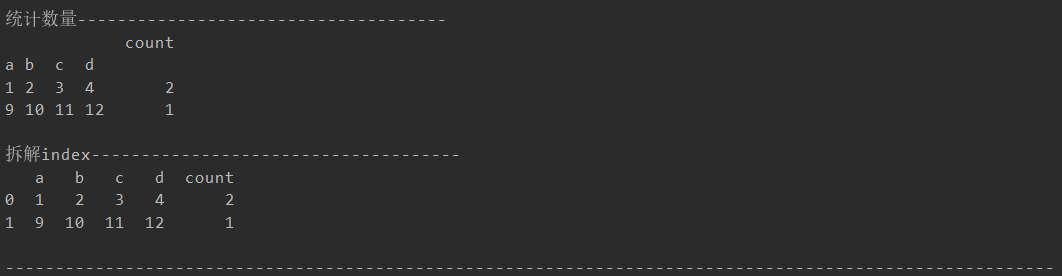
1. index = pd.MultiIndex.from\_tuples([('bird', 'falcon'),
2. ('bird', 'parrot'),
3. ('mammal', 'lion'),
4. ('mammal', 'monkey')],
5. names=['class', 'name'])
6. columns = pd.MultiIndex.from\_tuples([('speed', 'max'),
7. ('species', 'type')])
9. df = pd.DataFrame([(389.0, 'fly'),
10. (24.0, 'fly'),
11. (80.5, 'run'),
12. (NaN, 'jump')],
13. index=index,
14. columns=columns)
15. **print**(df)
16. **print**('\n-------------------------------------')
17. **print**(df.reset\_index())
18. **print**('\n-------------------------------------')
19. **print**(df.reset\_index(level='class'))
20. **print**('\n-------------------------------------')
21. **print**(df.reset\_index(level='class', col\_level=1))
22. **print**('\n-------------------------------------')
23. **print**(df.reset\_index(level='class', col\_level=1, col\_fill='species'))
24. **print**('\n-------------------------------------')
25. **print**(df.reset\_index(level='class', col\_level=1, col\_fill='genus'))



### 19.统计数量groupby

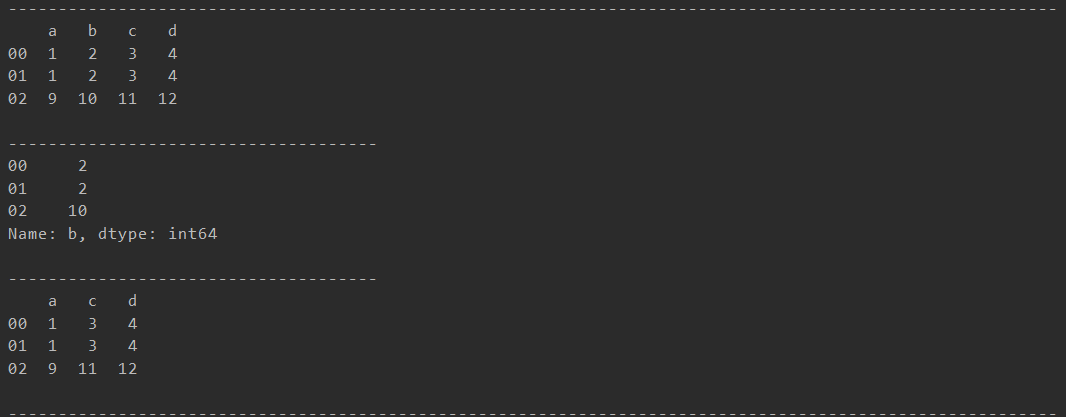
1. df = pd.DataFrame([[1, 2, 3, 4], [1, 2, 3, 4], [9, 10, 11, 12]],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\n-------------------------------------')
6. **print**(df.groupby(by='a').count())
7. **print**('\n-------------------------------------')
8. **print**(df.groupby(by='a').mean())
9. **print**('\n-------------------------------------')
10. **print**(df.groupby(by=['a', 'b', 'c', 'd']).count())
11. **print**('\n-------------------------------------')
12. **print**(df.groupby(by=['a', 'b', 'c'])['d'].mean())
13. **print**('\n统计数量-------------------------------------')
14. col = df.columns.tolist()
15. df['count'] = 1
16. df = df.groupby(col).count()
17. **print**(df)
18. **print**('\n拆解index-------------------------------------')
19. **print**(df.reset\_index())





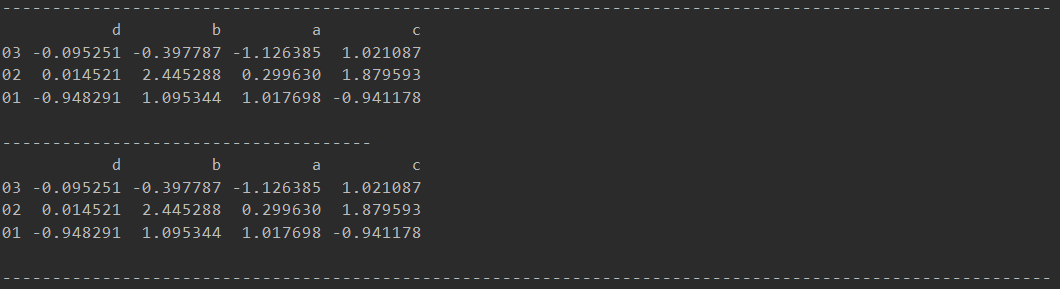
### 20.pop

1. df = pd.DataFrame([[1, 2, 3, 4], [1, 2, 3, 4], [9, 10, 11, 12]],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. **print**('\n-------------------------------------')
6. **print**(df.pop('b'))
7. **print**('\n-------------------------------------')
8. **print**(df)



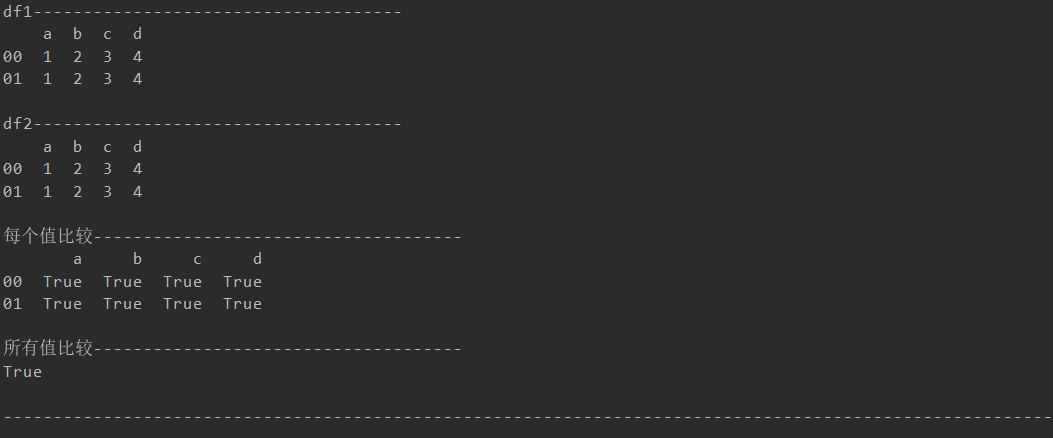
### 21.squeeze

1. df = pd.DataFrame(pd.np.random.randn(3, 4), index=['03', '02', '01'],
2. columns=['d', 'b', 'a', 'c'])
3. **print**(df)
4. **print**('\n-------------------------------------')
5. **print**(df.squeeze(axis=0))



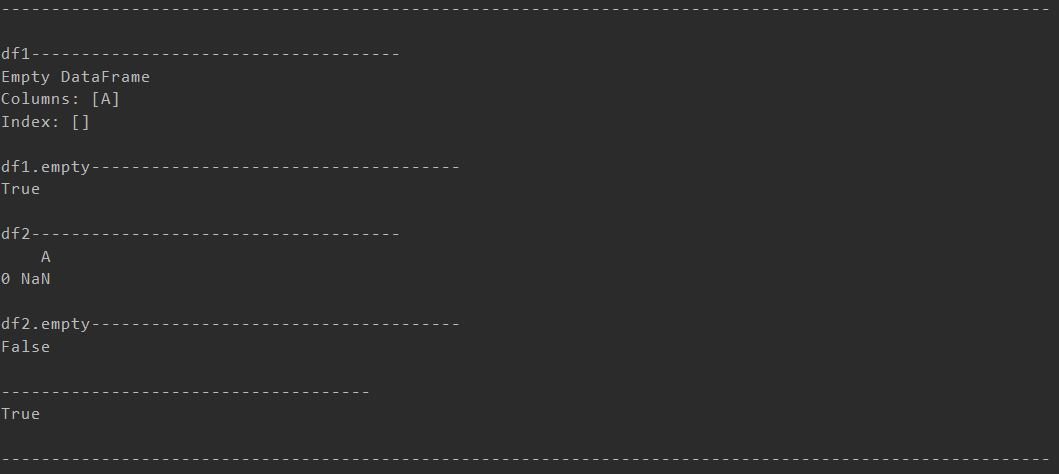
### 22.相等

1. df1 = pd.DataFrame([[1, 2, 3, 4], [1, 2, 3, 4]],
2. index=['00', '01'],
3. columns=['a', 'b', 'c', 'd'])
4. df2 = pd.DataFrame([[1, 2, 3, 4], [1, 2, 3, 4]],
5. index=['00', '01'],
6. columns=['a', 'b', 'c', 'd'])
7. **print**('\ndf1-------------------------------------')
8. **print**(df1)
9. **print**('\ndf2-------------------------------------')
10. **print**(df2)
11. **print**('\n每个值比较-------------------------------------')
12. **print**(df1 == df2)
13. **print**('\n所有值比较-------------------------------------')
14. **print**(df1.equals(df2))



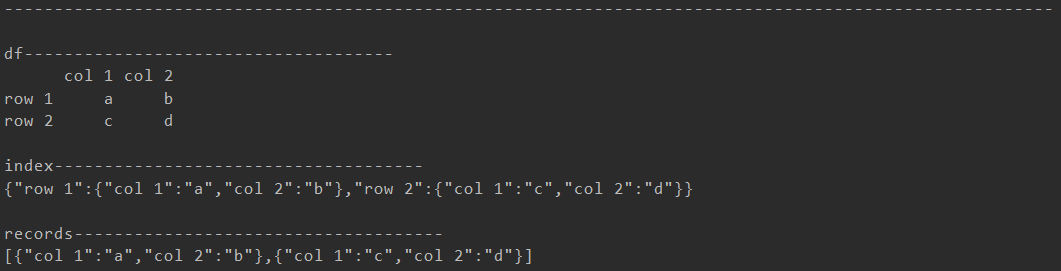
### 23.判空empty

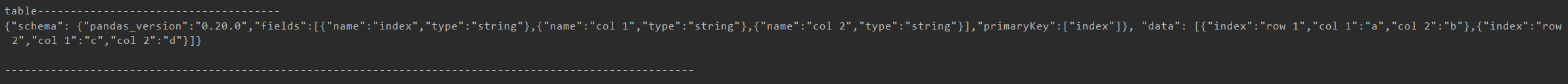
1. df1 = pd.DataFrame({'A': []})
2. **print**('\ndf1-------------------------------------')
3. **print**(df1)
4. **print**('\ndf1.empty-------------------------------------')
5. **print**(df1.empty)
7. **print**('\ndf2-------------------------------------')
8. df2 = pd.DataFrame({'A': [pd.np.nan]})
9. **print**(df2)
10. **print**('\ndf2.empty-------------------------------------')
11. **print**(df2.empty)
12. **print**('\n-------------------------------------')
13. **print**(df2.dropna().empty)



### 24转化json to\_json

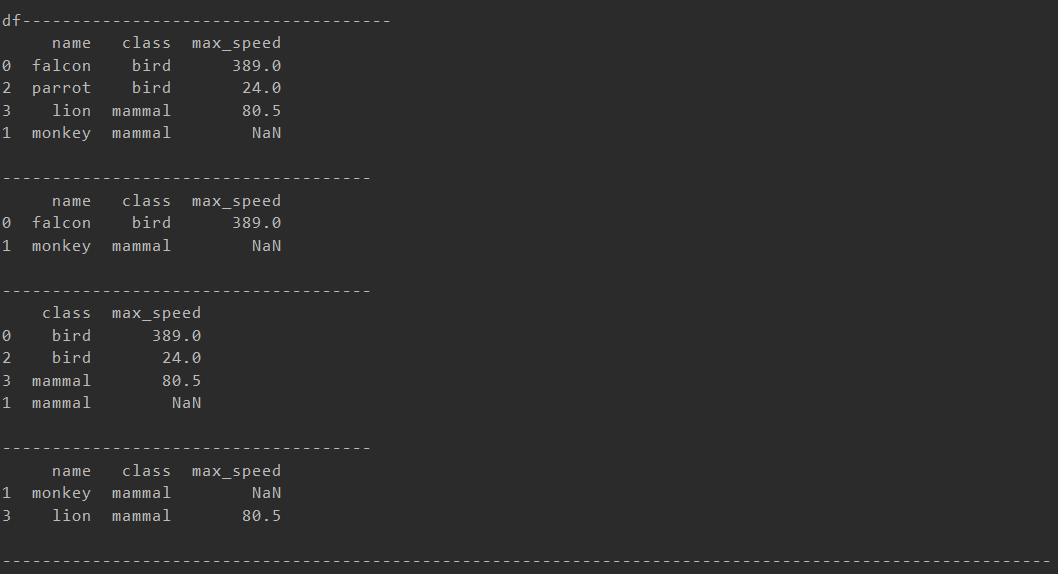
1. df = pd.DataFrame([['a', 'b'], ['c', 'd']], index=['row 1', 'row 2'], columns=['col 1', 'col 2'])
2. **print**('\ndf-------------------------------------')
3. **print**(df)
4. **print**('\nindex-------------------------------------')
5. **print**(df.to\_json(orient='index'))
6. **print**('\nrecords-------------------------------------')
7. **print**(df.to\_json(orient='records'))
8. **print**('\ntable-------------------------------------')
9. **print**(df.to\_json(orient='table'))





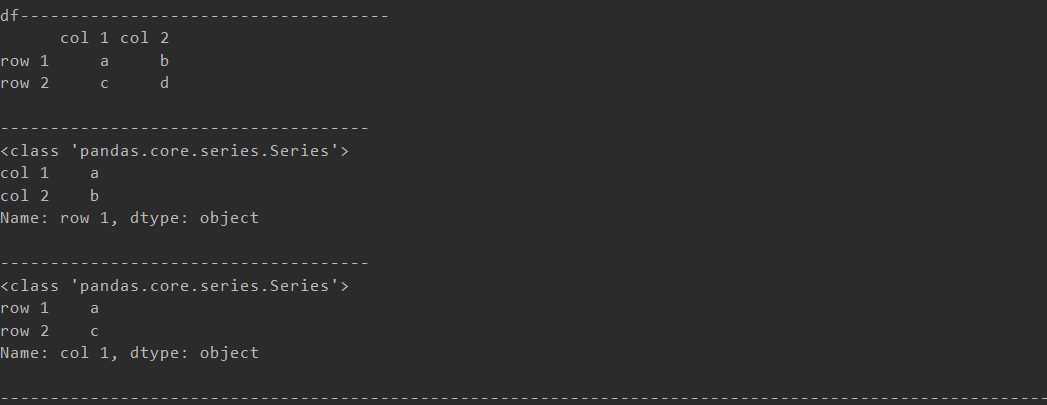
### 25.take

1. df = pd.DataFrame([('falcon', 'bird', 389.0),
2. ('parrot', 'bird', 24.0),
3. ('lion', 'mammal', 80.5),
4. ('monkey', 'mammal', pd.np.nan)],
5. columns=('name', 'class', 'max\_speed'),
6. index=[0, 2, 3, 1])
7. **print**('\ndf-------------------------------------')
8. **print**(df)
9. **print**('\n-------------------------------------')
10. **print**(df.take([0, 3]))
11. **print**('\n-------------------------------------')
12. **print**(df.take([1, 2], axis=1))
13. **print**('\n-------------------------------------')
14. **print**(df.take([-1, -2]))



### 26.xs

1. # 只能get不能set
2. df = pd.DataFrame([['a', 'b'], ['c', 'd']], index=['row 1', 'row 2'], columns=['col 1', 'col 2'])
3. **print**('\ndf-------------------------------------')
4. **print**(df)
5. **print**('\n-------------------------------------')
6. **print**(type(df.xs('row 1')))
7. **print**(df.xs('row 1'))
8. **print**('\n-------------------------------------')
9. **print**(type(df.xs('col 1', axis=1)))
10. **print**(df.xs('col 1', axis=1))



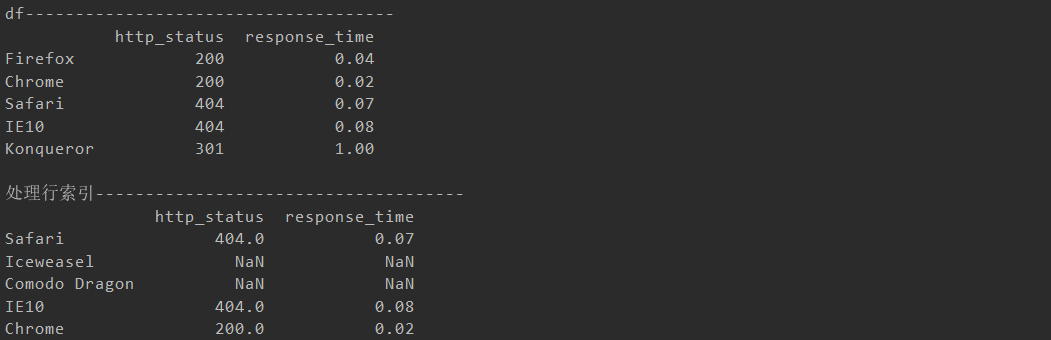
### 27.add\_prefix add\_suffix

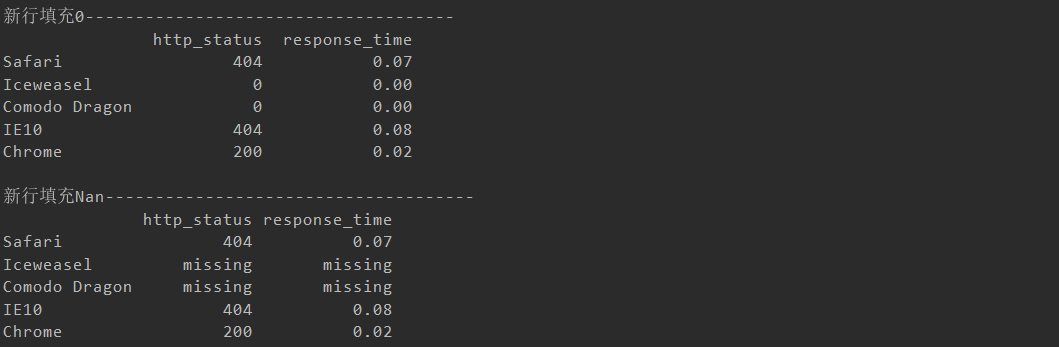
1. df = pd.DataFrame([['a', 'b'], ['c', 'd']], index=['row1', 'row2'], columns=['col1', 'col2'])
2. **print**('\ndf-------------------------------------')
3. **print**(df)
4. **print**('\n前缀-------------------------------------')
5. **print**(df.add\_prefix('new'))
6. **print**('\n后缀-------------------------------------')
7. **print**(df.add\_suffix('new'))



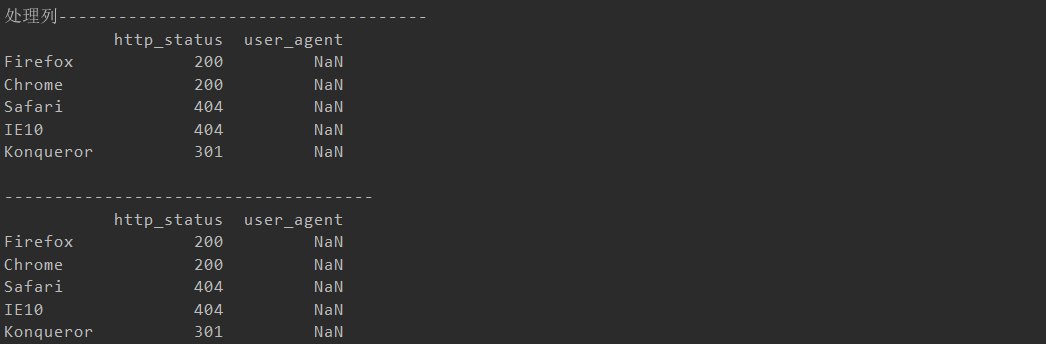
### 28.reindex

1. index = ['Firefox', 'Chrome', 'Safari', 'IE10', 'Konqueror']
2. df = pd.DataFrame({
3. 'http\_status': [200, 200, 404, 404, 301],
4. 'response\_time': [0.04, 0.02, 0.07, 0.08, 1.0]},
5. index=index)
6. **print**('\ndf-------------------------------------')
7. **print**(df)
8. **print**('\n处理行索引-------------------------------------')
9. new\_index = ['Safari', 'Iceweasel', 'Comodo Dragon', 'IE10',
10. 'Chrome']
11. **print**(df.reindex(new\_index))
12. **print**('\n新行填充0-------------------------------------')
13. **print**(df.reindex(new\_index, fill\_value=0))
14. **print**('\n新行填充Nan-------------------------------------')
15. **print**(df.reindex(new\_index, fill\_value='missing'))

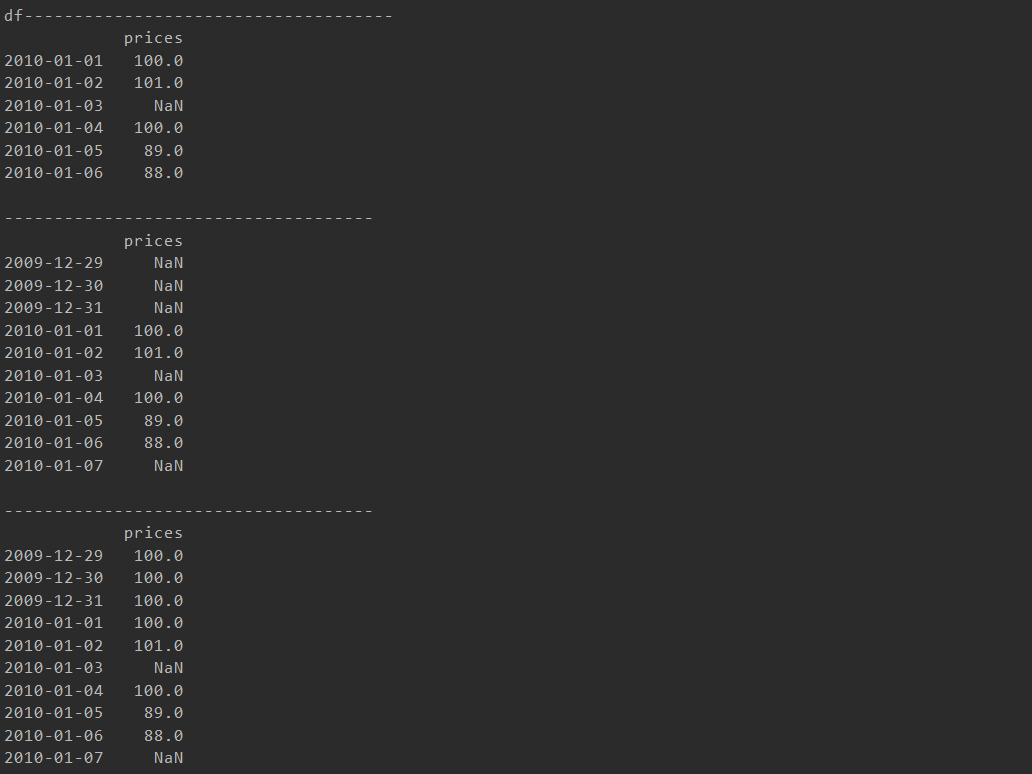




1. **print**('\n处理列-------------------------------------')
2. **print**(df.reindex(columns=['http\_status', 'user\_agent']))
3. **print**('\n-------------------------------------')
4. **print**(df.reindex(['http\_status', 'user\_agent'], axis="columns"))

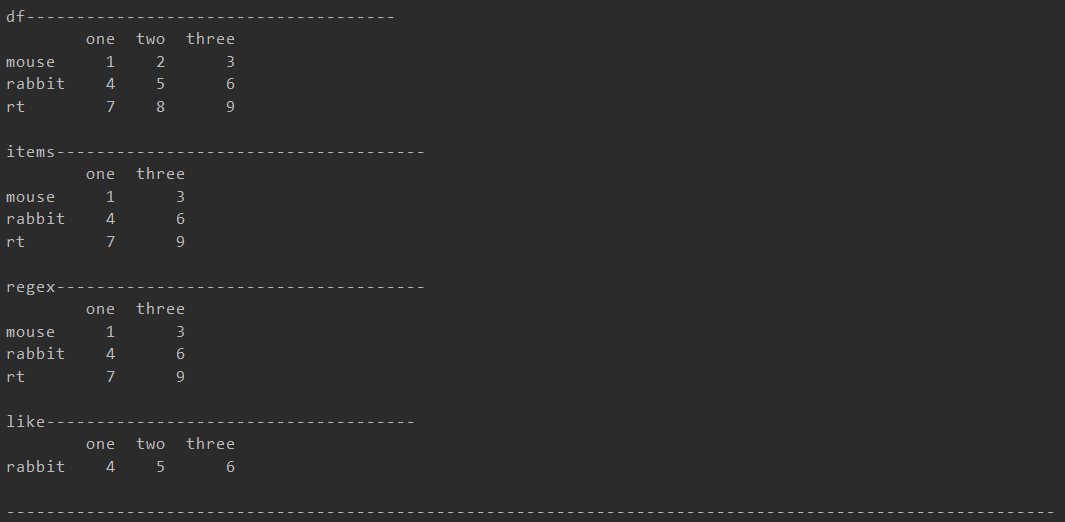


1. date\_index = pd.date\_range('1/1/2010', periods=6, freq='D')
2. df2 = pd.DataFrame({"prices": [100, 101, pd.np.nan, 100, 89, 88]},
3. index=date\_index)
4. **print**('\ndf-------------------------------------')
5. **print**(df2)
7. **print**('\n-------------------------------------')
8. date\_index2 = pd.date\_range('12/29/2009', periods=10, freq='D')
9. **print**(df2.reindex(date\_index2))
10. **print**('\n-------------------------------------')
11. **print**(df2.reindex(date\_index2, method='bfill'))



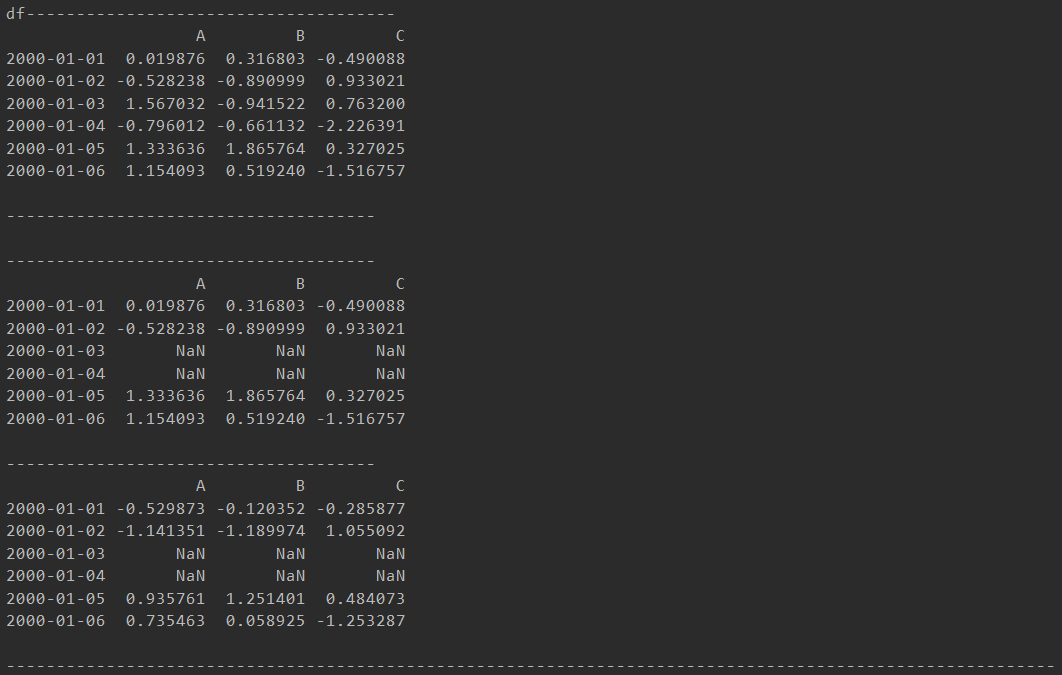
### 29.过滤filter

1. df = pd.DataFrame([[1, 2, 3], [4, 5, 6], [7, 8, 9]],
2. index=['mouse', 'rabbit', 'rt'],
3. columns=['one', 'two', 'three'])
4. **print**('\ndf-------------------------------------')
5. **print**(df)
6. **print**('\nitems-------------------------------------')
7. **print**(df.filter(items=['one', 'three']))
8. **print**('\nregex-------------------------------------')
9. **print**(df.filter(regex='e$', axis=1))
10. **print**('\nlike-------------------------------------')
11. **print**(df.filter(like='bbi', axis=0))



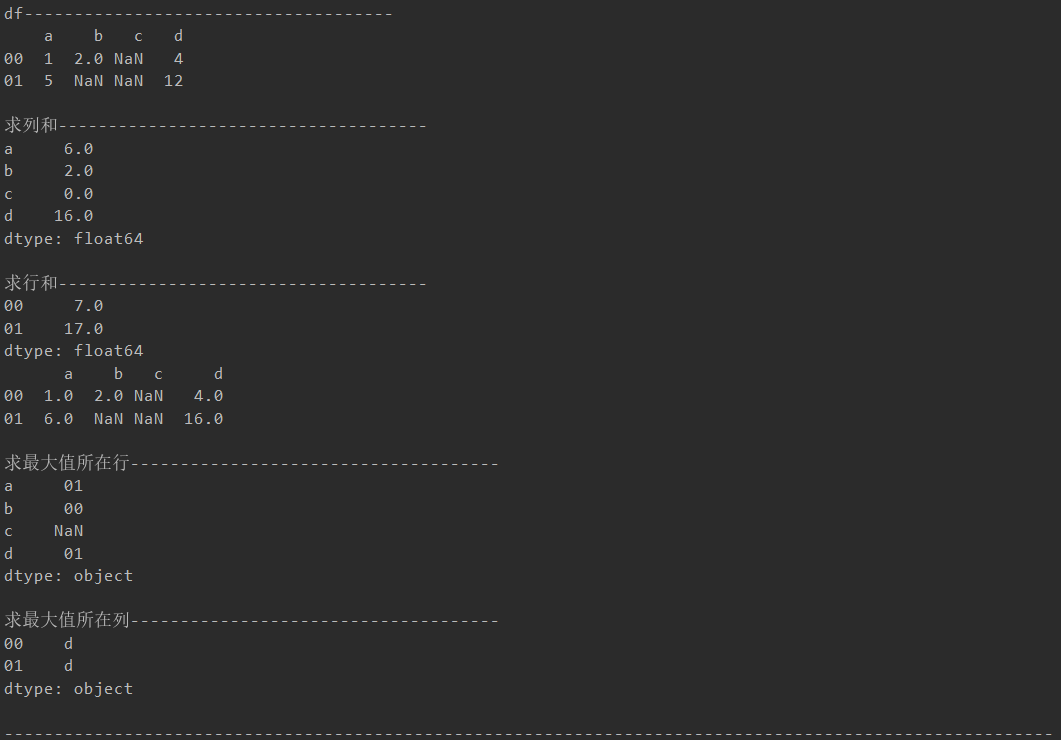
### 30.transform

1. df = pd.DataFrame(pd.np.random.randn(6, 3), columns=['A', 'B', 'C'], index=pd.date\_range('1/1/2000', periods=6))
2. **print**('\ndf-------------------------------------')
3. **print**(df)
4. **print**('\n-------------------------------------')
5. df.iloc[2:4] = pd.np.nan
6. **print**('\n-------------------------------------')
7. **print**(df)
8. **print**('\n-------------------------------------')
9. **print**(df.transform(**lambda** x: (x - x.mean()) / x.std()))

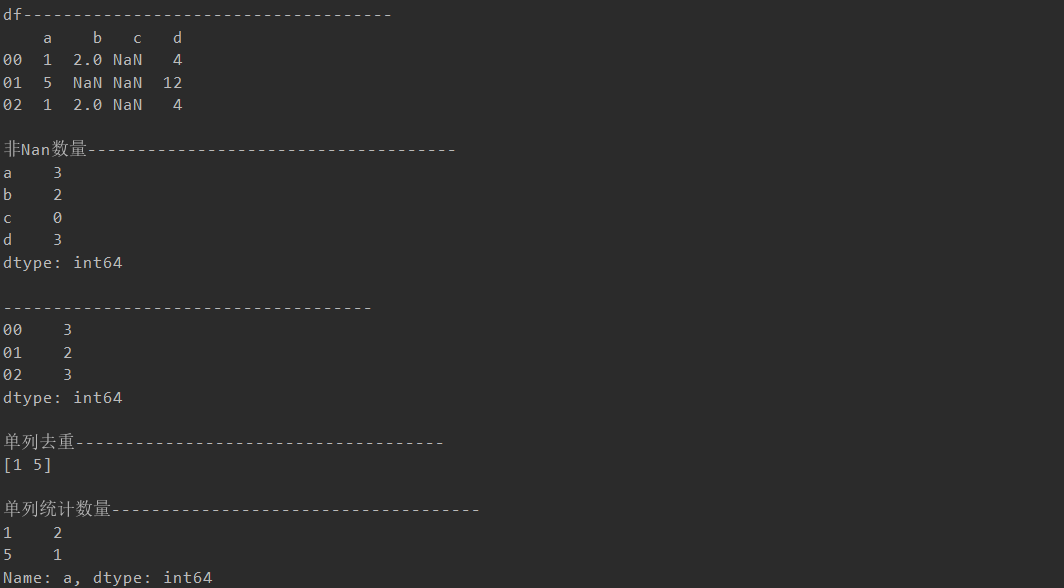


### 31.数学统计

1. df = pd.DataFrame([[1, 2, NaN, 4], [5, NaN, NaN, 12]],
2. index=['00', '01'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**('\ndf-------------------------------------')
5. **print**(df)
6. **print**('\n求列和-------------------------------------')
7. **print**(df.sum())
8. **print**('\n求行和-------------------------------------')
9. **print**(df.sum(axis=1))
11. **print**('\n求累加-------------------------------------')
12. **print**(df.cumsum())
14. **print**('\n求最大值所在行-------------------------------------')
15. **print**(df.idxmax())
16. **print**('\n求最大值所在列-------------------------------------')
17. **print**(df.idxmax(axis=1))

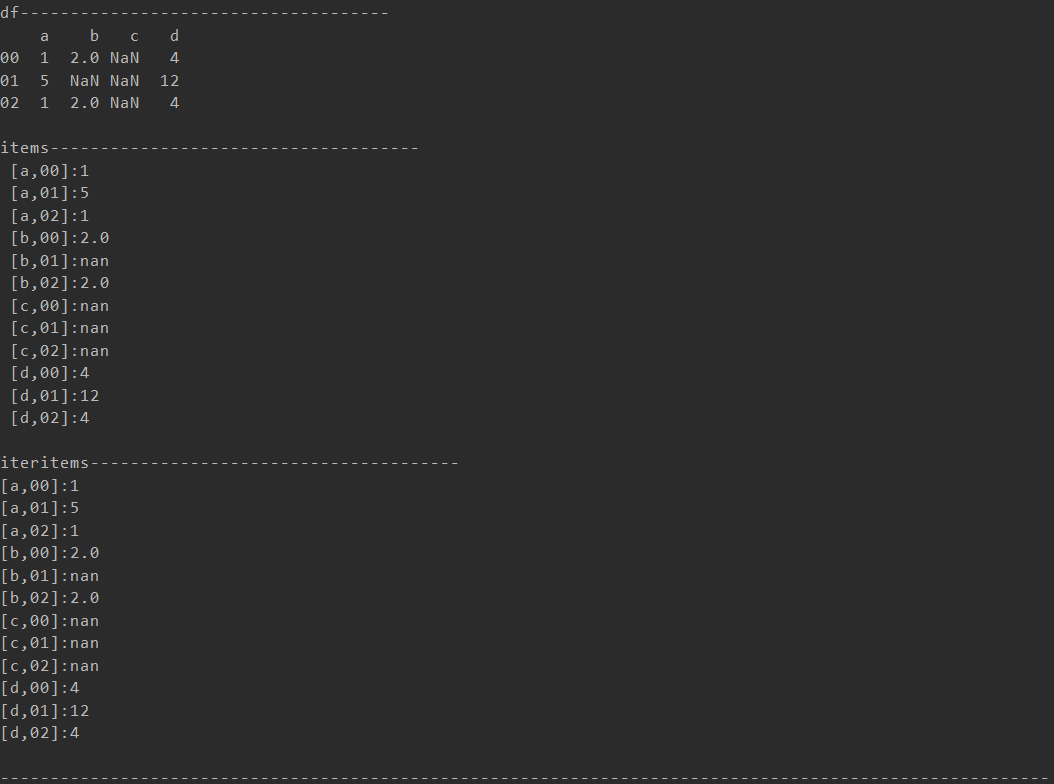


1. df = pd.DataFrame([[1, 2, NaN, 4], [5, NaN, NaN, 12], [1, 2, NaN, 4]],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**('\ndf-------------------------------------')
5. **print**(df)
6. **print**('\n非Nan数量-------------------------------------')
7. **print**(df.count())
8. **print**('\n-------------------------------------')
9. **print**(df.count(axis=1))
10. **print**('\n单列去重-------------------------------------')
11. **print**(df['a'].unique())
12. **print**('\n单列统计数量-------------------------------------')
13. **print**(df['a'].value\_counts())



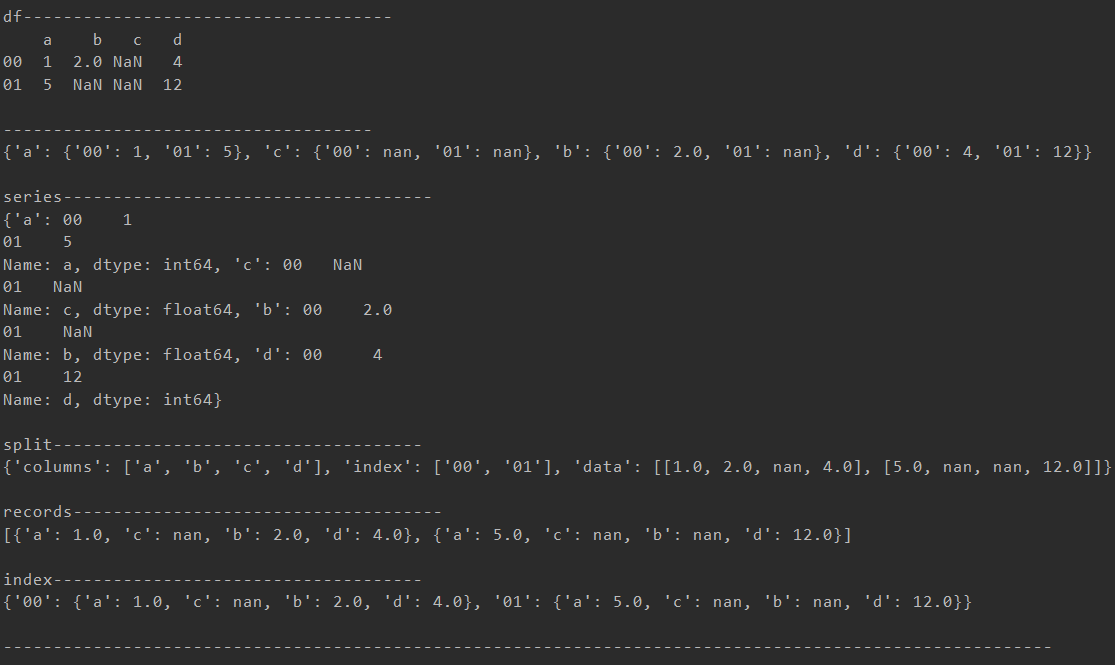
### 32.遍历items

1. df = pd.DataFrame([[1, 2, NaN, 4], [5, NaN, NaN, 12], [1, 2, NaN, 4]],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**('\ndf-------------------------------------')
5. **print**(df)
6. **print**('\nitems-------------------------------------')
7. **for** col\_key, col\_val **in** df.items():
8. **for** row\_key,row\_val **in** col\_val.items():
9. **print**(' [{0},{1}]:{2}'.format(col\_key, row\_key, row\_val))
11. **print**('\niteritems-------------------------------------')
12. **for** col\_key, col\_val **in** df.iteritems():
13. **for** row\_key,row\_val **in** col\_val.iteritems():
14. **print**('[{0},{1}]:{2}'.format(col\_key, row\_key, row\_val))



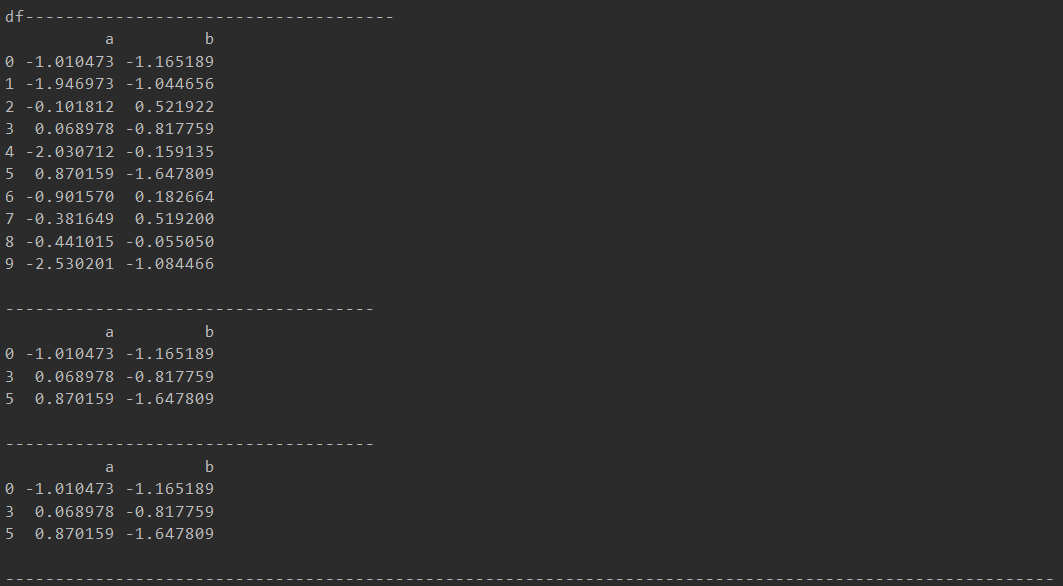
### 33.to\_dict

1. df = pd.DataFrame([[1, 2, NaN, 4], [5, NaN, NaN, 12]],
2. index=['00', '01'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**('\ndf-------------------------------------')
5. **print**(df)
6. **print**('\n-------------------------------------')
7. **print**(df.to\_dict())
8. **print**('\nseries-------------------------------------')
9. **print**(df.to\_dict('series'))
10. **print**('\nsplit-------------------------------------')
11. **print**(df.to\_dict('split'))
12. **print**('\nrecords-------------------------------------')
13. **print**(df.to\_dict('records'))
14. **print**('\nindex-------------------------------------')
15. **print**(df.to\_dict('index'))



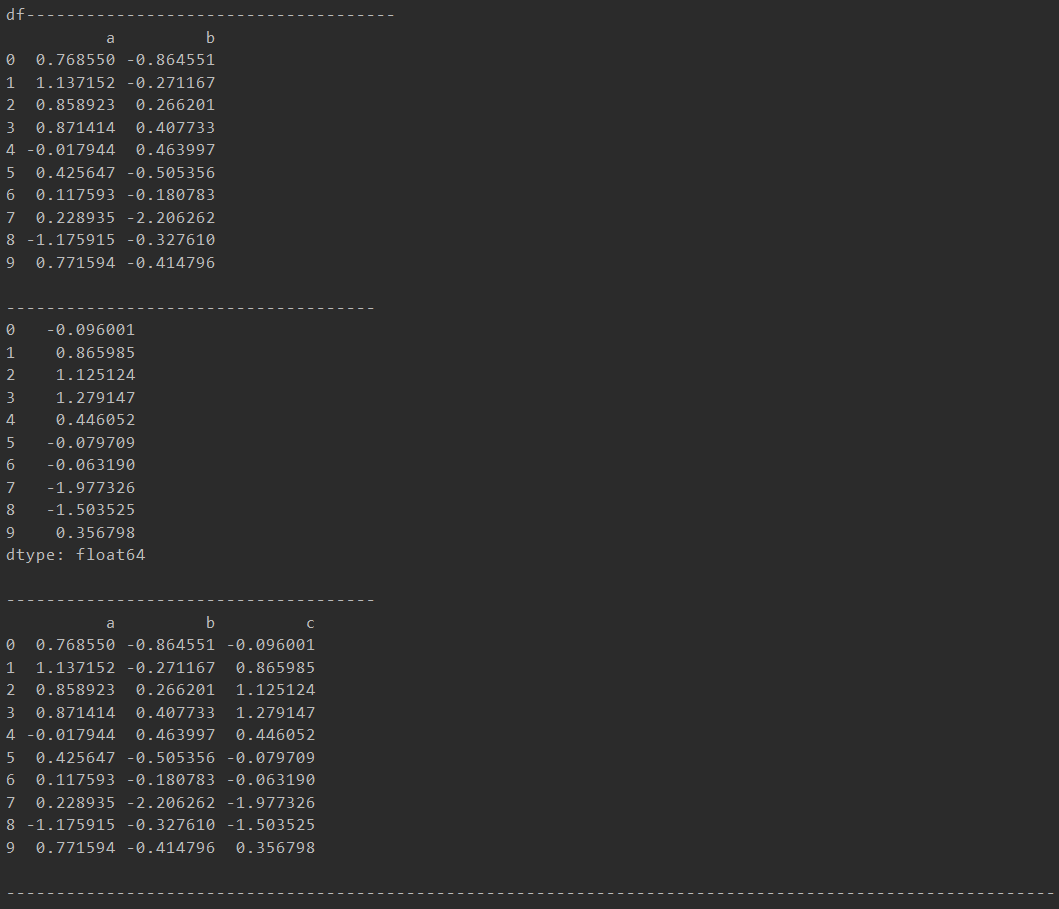
### 34.query

1. df = pd.DataFrame(randn(10, 2), columns=list('ab'))
2. **print**('\ndf-------------------------------------')
3. **print**(df)
4. **print**('\n-------------------------------------')
5. **print**(df.query('a > b'))
6. **print**('\n-------------------------------------')
7. **print**(df[df.a > df.b])



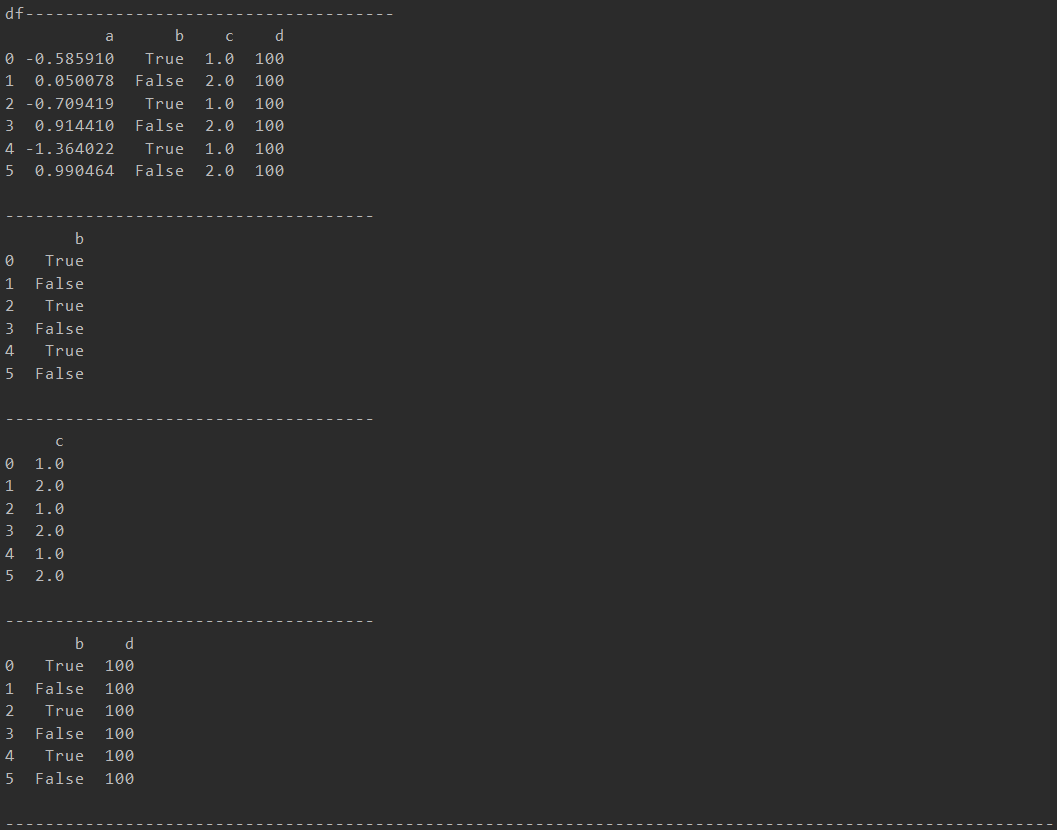
### 35evel

1. df = pd.DataFrame(randn(10, 2), columns=list('ab'))
2. **print**('\ndf-------------------------------------')
3. **print**(df)
4. **print**('\n-------------------------------------')
5. **print**(df.eval('a + b'))
6. **print**('\n-------------------------------------')
7. **print**(df.eval('c = a + b'))



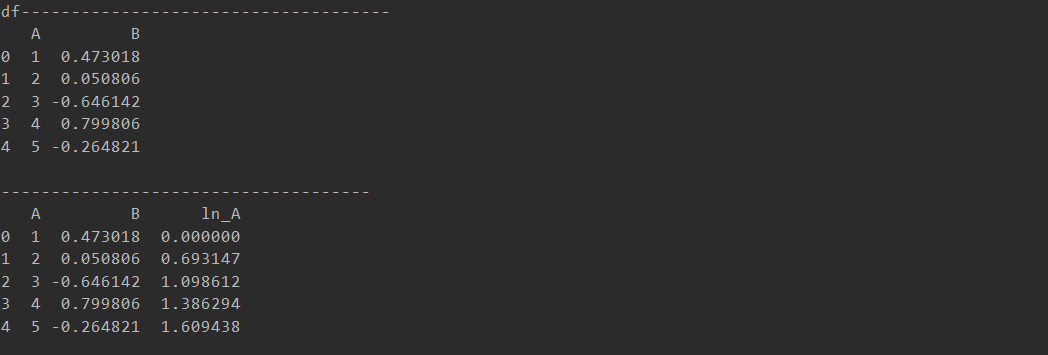
### 36.select\_dtypes

1. df = pd.DataFrame({'a': pd.np.random.randn(6).astype('f4'),
2. 'b': [True, False] \* 3,
3. 'c': [1.0, 2.0] \* 3,
4. 'd': [100] \* 6})
5. **print**('\ndf-------------------------------------')
6. **print**(df)
7. **print**('\n-------------------------------------')
8. **print**(df.select\_dtypes(include='bool'))
9. **print**('\n-------------------------------------')
10. **print**(df.select\_dtypes(include=['float64']))
11. **print**('\n-------------------------------------')
12. **print**(df.select\_dtypes(exclude=['floating']))



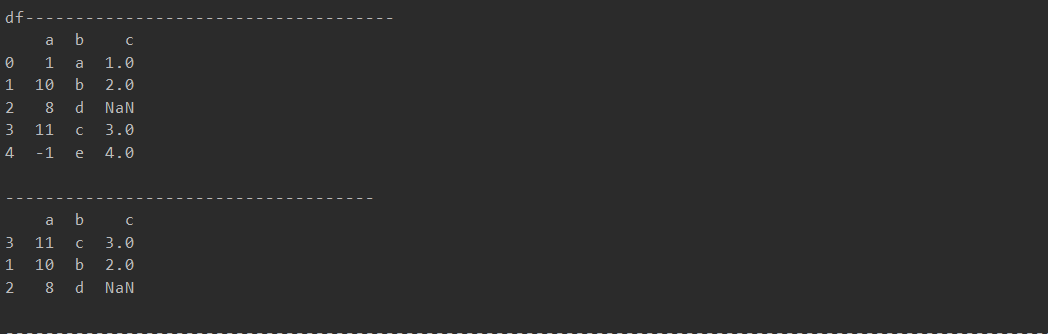
### 37.assign

1. df = pd.DataFrame({'A': range(1, 6), 'B': pd.np.random.randn(5)})
2. **print**('\ndf-------------------------------------')
3. **print**(df)
4. **print**('\n-------------------------------------')
5. **print**(df.assign(ln\_A=**lambda** x: pd.np.log(x.A)))



### 38.nlargest nsmallest

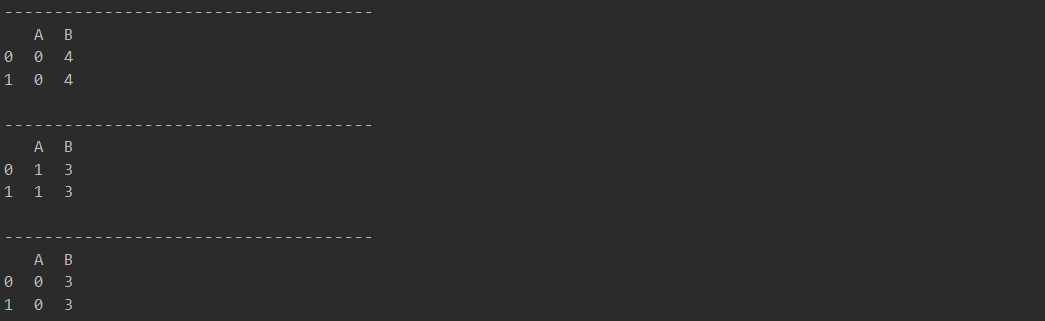
1. df = pd.DataFrame({'a': [1, 10, 8, 11, -1],
2. 'b': list('abdce'),
3. 'c': [1.0, 2.0, pd.np.nan, 3.0, 4.0]})
4. **print**('\ndf-------------------------------------')
5. **print**(df)
6. **print**('\n以某列值降序取前3个-------------------------------------')
7. **print**(df.nlargest(3, 'a'))
8. **print**(df.nsmallest(3, 'a'))





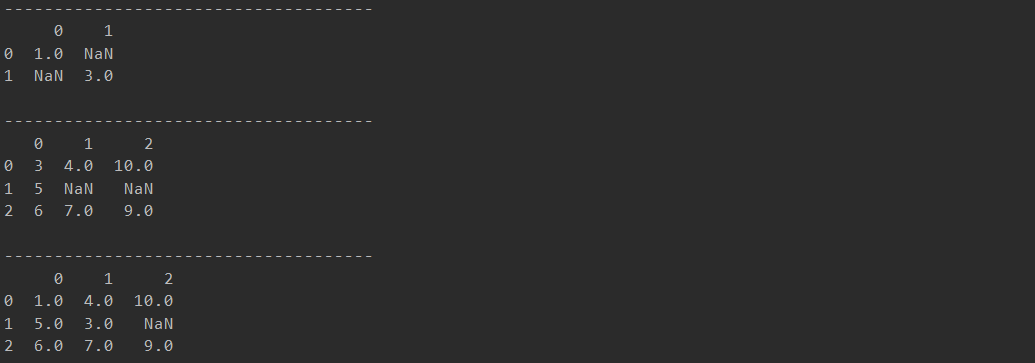
### 39.combine

1. df1 = pd.DataFrame({'A': [0, 0], 'B': [4, 4]})
2. df2 = pd.DataFrame({'A': [1, 1], 'B': [3, 3]})
3. **print**('\n-------------------------------------')
4. **print**(df1)
5. **print**('\n-------------------------------------')
6. **print**(df2)
7. **print**('\n-------------------------------------')
8. **print**(df1.combine(df2, **lambda** s1, s2: s1 **if** s1.sum() < s2.sum() **else** s2))



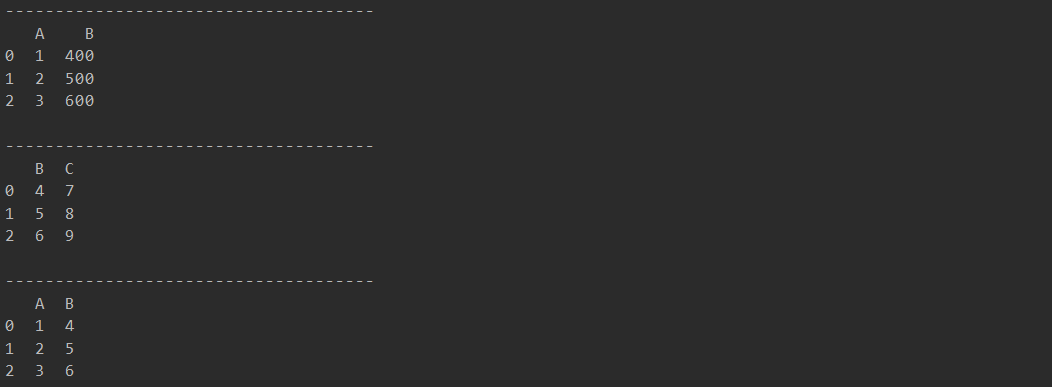
### 40.combine\_first

1. df1 = pd.DataFrame([[1, pd.np.nan], [NaN, 3]])
2. df2 = pd.DataFrame([[3, 4, 10], [5, NaN, NaN], [6, 7, 9]])
3. **print**('\n-------------------------------------')
4. **print**(df1)
5. **print**('\n-------------------------------------')
6. **print**(df2)
7. **print**('\n-------------------------------------')
8. **print**(df1.combine\_first(df2))

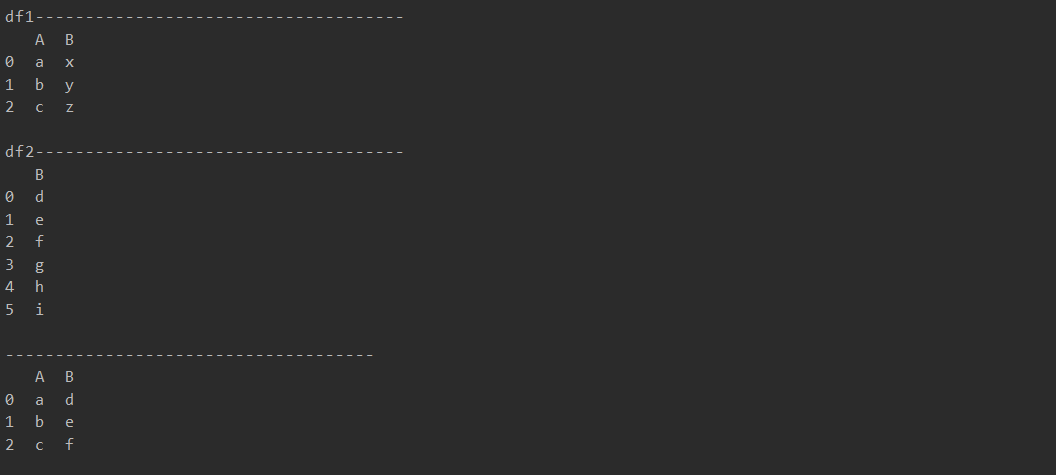


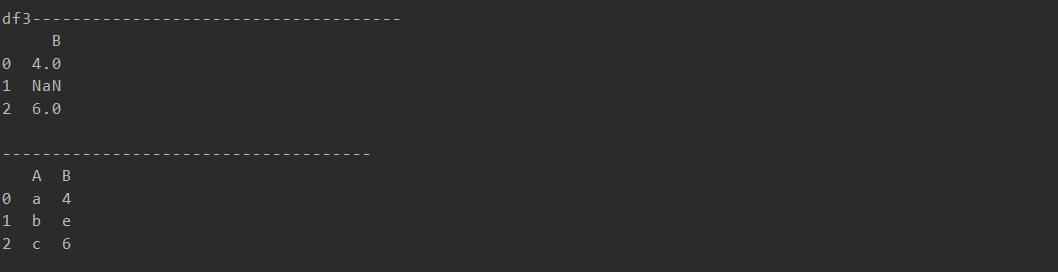
### 41.update

1. df1 = pd.DataFrame({'A': [1, 2, 3],
2. 'B': [400, 500, 600]})
3. df2 = pd.DataFrame({'B': [4, 5, 6],
4. 'C': [7, 8, 9]})
5. **print**('\n-------------------------------------')
6. **print**(df1)
7. **print**('\n-------------------------------------')
8. **print**(df2)
9. **print**('\n-------------------------------------')
10. df1.update(df2)
11. **print**(df1)



1. df1 = pd.DataFrame({'A': ['a', 'b', 'c'], 'B': ['x', 'y', 'z']})
2. df2 = pd.DataFrame({'B': ['d', 'e', 'f', 'g', 'h', 'i']})
3. df3 = pd.DataFrame({'B': [4, pd.np.nan, 6]})
4. df4 = pd.DataFrame({'B': ['d', 'e']}, index=[1, 2])
5. **print**('\ndf1-------------------------------------')
6. **print**(df1)
7. **print**('\ndf2-------------------------------------')
8. **print**(df2)
9. **print**('\n-------------------------------------')
10. df1.update(df2)
11. **print**(df1)
12. **print**('\ndf3-------------------------------------')
13. **print**(df3)
14. **print**('\n-------------------------------------')
15. df1.update(df3)
16. **print**(df1)
17. **print**('\ndf4-------------------------------------')
18. **print**(df4)
19. **print**('\n-------------------------------------')
20. df1.update(df4)
21. **print**(df1)

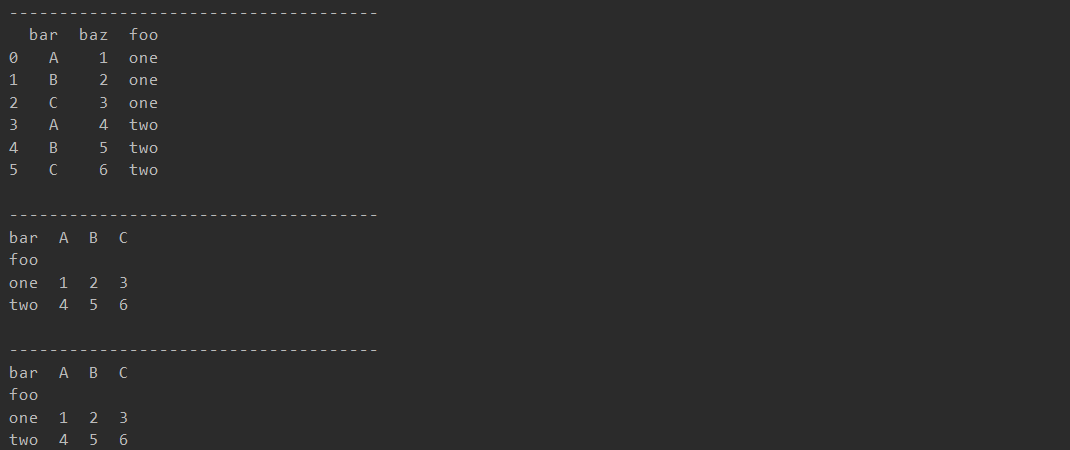






### 42.pivot

1. df = pd.DataFrame({'foo': ['one', 'one', 'one', 'two', 'two', 'two'],
2. 'bar': ['A', 'B', 'C', 'A', 'B', 'C'],
3. 'baz': [1, 2, 3, 4, 5, 6]})
4. **print**('\n-------------------------------------')
5. **print**(df)
6. **print**('\n-------------------------------------')
7. **print**(df.pivot(index='foo', columns='bar', values='baz'))
8. **print**('\n-------------------------------------')
9. **print**(df.pivot(index='foo', columns='bar')['baz'])



### 43.stack

1. df = pd.DataFrame([[1, 2], [5, NaN]],
2. index=['00', '01'],
3. columns=['a', 'b'])
4. **print**('\n-------------------------------------')
5. **print**(df)
6. **print**('\n-------------------------------------')
7. **print**(df.stack())



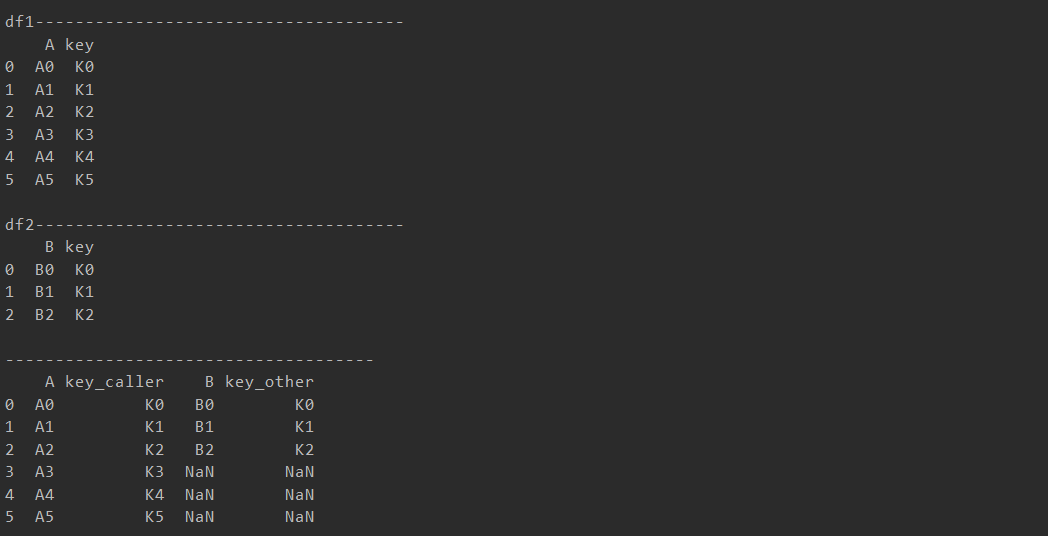
### 44.applymap

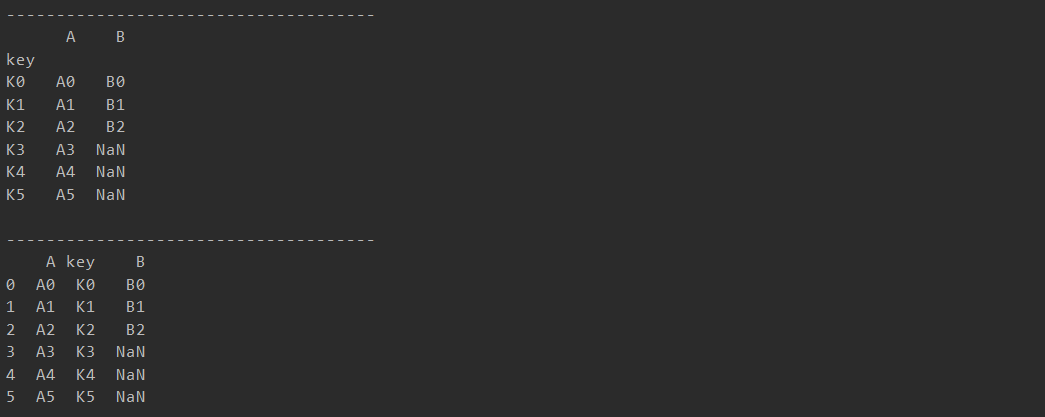
1. df = pd.DataFrame(pd.np.random.randn(3, 3))
2. **print**('\n-------------------------------------')
3. **print**(df)
4. **print**('\n-------------------------------------')
5. df = df.applymap(**lambda** x: '%.2f' % x)
6. **print**(df)



### 45.join

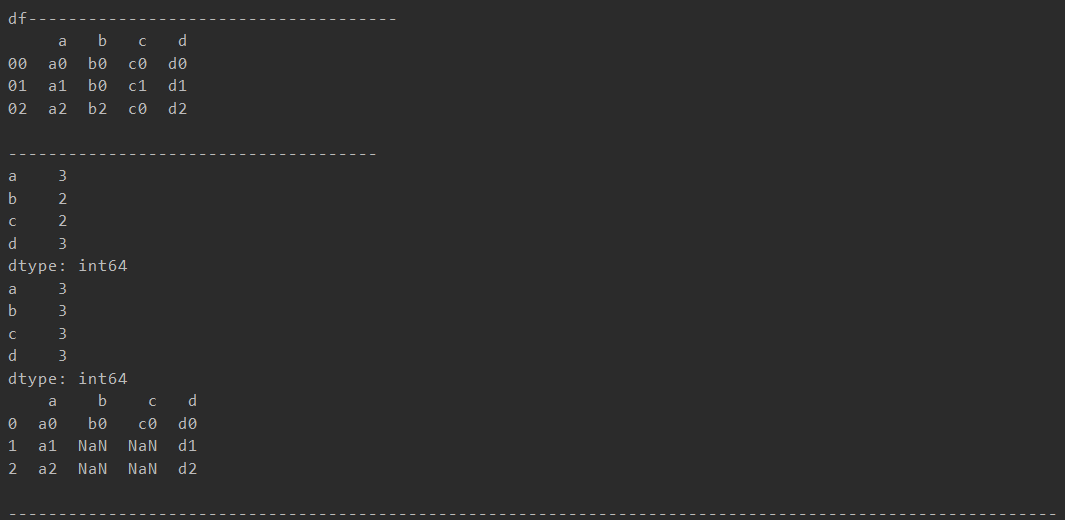
1. df1 = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3', 'K4', 'K5'],
2. 'A': ['A0', 'A1', 'A2', 'A3', 'A4', 'A5']})
3. df2 = pd.DataFrame({'key': ['K0', 'K1', 'K2'],
4. 'B': ['B0', 'B1', 'B2']})
5. **print**('\ndf1-------------------------------------')
6. **print**(df1)
7. **print**('\ndf2-------------------------------------')
8. **print**(df2)
9. **print**('\n-------------------------------------')
10. **print**(df1.join(df2, lsuffix='\_caller', rsuffix='\_other'))
11. **print**('\n-------------------------------------')
12. **print**(df1.set\_index('key').join(df2.set\_index('key')))
13. **print**('\n-------------------------------------')
14. **print**(df1.join(df2.set\_index('key'), on='key'))





### 46.nunique.

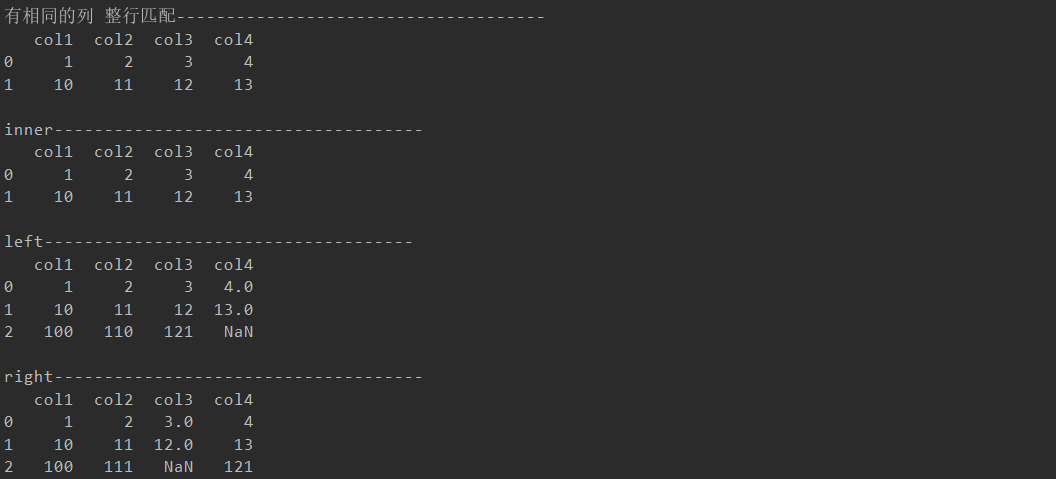
1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b0', 'c1', 'd1'], ['a2', 'b2', 'c0', 'd2']],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**('\ndf-------------------------------------')
5. **print**(df)
6. **print**('\nnunique-------------------------------------')
7. **print**(df.nunique())
8. **print**('\ncount-------------------------------------')
9. **print**(df.count())
10. **print**('\nmode-------------------------------------')
11. **print**(df.mode())

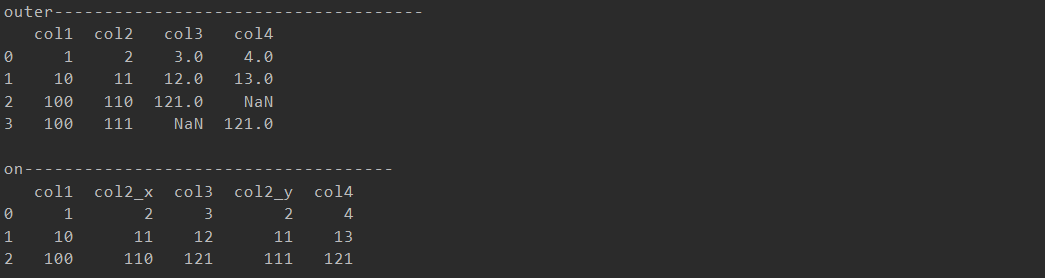


### 47.merge

1. df1 = pd.DataFrame([[1, 2, 3], [10, 11, 12], [100, 110, 121]], index=['row1', 'row2', 'row3'], columns=['col1', 'col2', 'col3'])
2. df2 = pd.DataFrame([[1, 2, 4], [10, 11, 13], [100, 111, 121]], index=['row1', 'row2', 'row4'], columns=['col1', 'col2', 'col4'])
3. **print**('\ndf1-------------------------------------')
4. **print**(df1)
5. **print**('\ndf2-------------------------------------')
6. **print**(df2)
7. **print**('\n有相同的列 整行匹配-------------------------------------')
8. **print**(pd.merge(df1, df2))
9. **print**('\ninner-------------------------------------')
10. **print**(pd.merge(df1, df2, how='inner'))
11. **print**('\nleft-------------------------------------')
12. **print**(pd.merge(df1, df2, how='left'))
13. **print**('\nright-------------------------------------')
14. **print**(pd.merge(df1, df2, how='right'))
15. **print**('\nouter-------------------------------------')
16. **print**(pd.merge(df1, df2, how='outer'))
17. **print**('\non-------------------------------------')
18. **print**(pd.merge(df1, df2, on=['col1']))



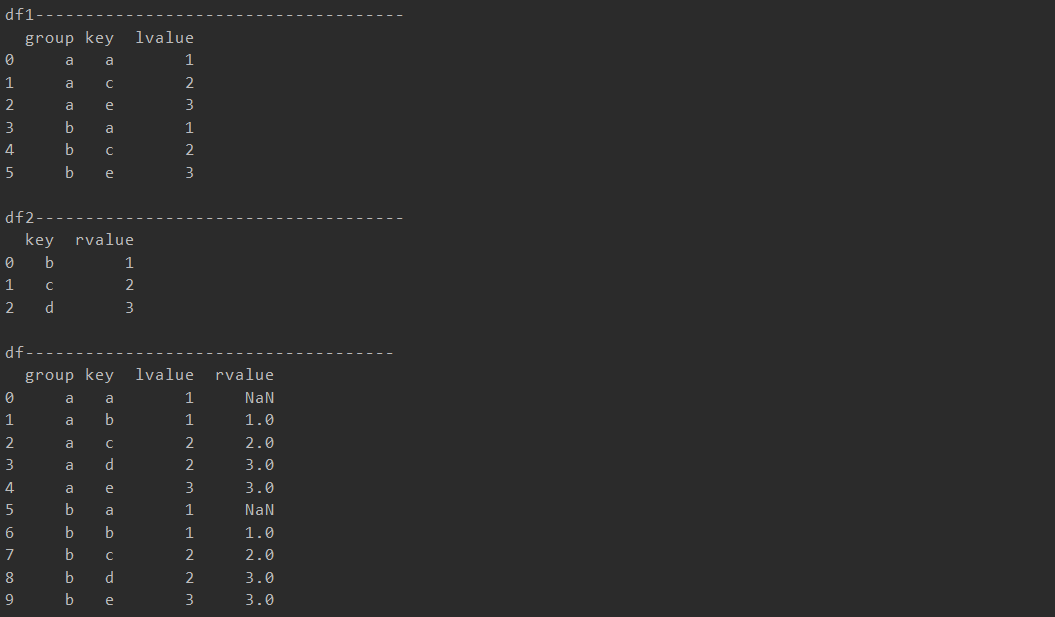




1. df1 = pd.DataFrame({'key': ['a', 'c', 'e', 'a', 'c', 'e'],
2. 'lvalue': [1, 2, 3, 1, 2, 3],
3. 'group' : ['a', 'a', 'a', 'b', 'b', 'b']})
4. df2 = pd.DataFrame({'key': ['b', 'c', 'd'],
5. 'rvalue': [1, 2, 3]})
6. **print**('\ndf1-------------------------------------')
7. **print**(df1)
8. **print**('\ndf2-------------------------------------')
9. **print**(df2)
10. **print**('\ndf-------------------------------------')
11. **print**(pd.ordered\_merge(df1, df2, fill\_method='ffill', left\_by='group'))

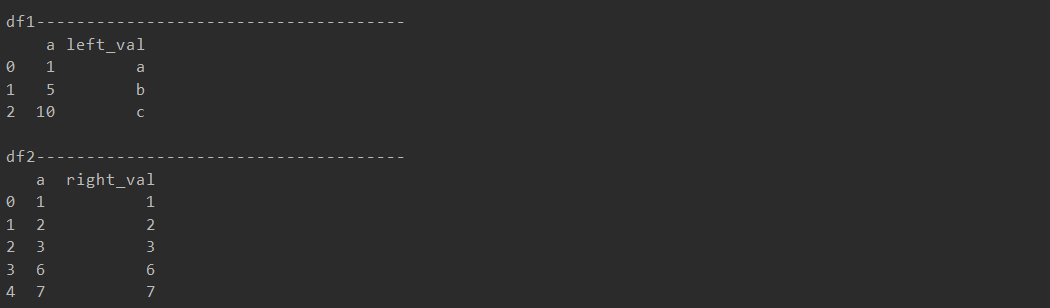
### 48.merge\_ordered

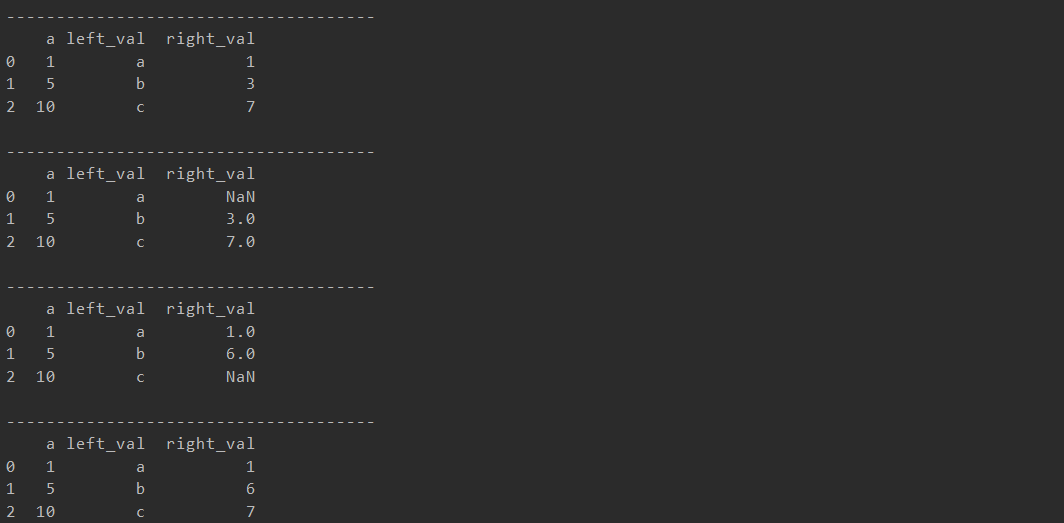
1. df1 = pd.DataFrame({'key': ['a', 'c', 'e', 'a', 'c', 'e'],
2. 'lvalue': [1, 2, 3, 1, 2, 3],
3. 'group' : ['a', 'a', 'a', 'b', 'b', 'b']})
4. df2 = pd.DataFrame({'key': ['b', 'c', 'd'],
5. 'rvalue': [1, 2, 3]})
6. **print**('\ndf1-------------------------------------')
7. **print**(df1)
8. **print**('\ndf2-------------------------------------')
9. **print**(df2)
10. **print**('\ndf-------------------------------------')
11. **print**(pd.merge\_ordered(df1, df2, fill\_method='ffill', left\_by='group'))



### 49.merge\_asof

1. df1 = pd.DataFrame({'a': [1, 5, 10], 'left\_val': ['a', 'b', 'c']})
2. df2 = pd.DataFrame({'a': [1, 2, 3, 6, 7], 'right\_val': [1, 2, 3, 6, 7]})
3. **print**('\ndf1-------------------------------------')
4. **print**(df1)
5. **print**('\ndf2-------------------------------------')
6. **print**(df2)
7. **print**('\n-------------------------------------')
8. **print**(pd.merge\_asof(df1, df2, on='a'))
9. **print**('\n-------------------------------------')
10. **print**(pd.merge\_asof(df1, df2, on='a', allow\_exact\_matches=False))
11. **print**('\n-------------------------------------')
12. **print**(pd.merge\_asof(df1, df2, on='a', direction='forward'))
13. **print**('\n-------------------------------------')
14. **print**(pd.merge\_asof(df1, df2, on='a', direction='nearest'))





1. left = pd.DataFrame({'left\_val': ['a', 'b', 'c']}, index=[1, 5, 10])
2. right = pd.DataFrame({'right\_val': [1, 2, 3, 6, 7]}, index=[1, 2, 3, 6, 7])
3. **print**('\nleft-------------------------------------')
4. **print**(left)
5. **print**('\nright-------------------------------------')
6. **print**(right)
7. **print**('\n-------------------------------------')
8. **print**(pd.merge\_asof(left, right, left\_index=True, right\_index=True))

