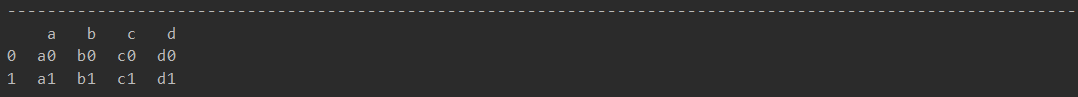
1.创建

1.1直接创建

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])



二维数组

1. df = pd.DataFrame(pd.np.arange(10).reshape(2, 5))
2. df = pd.DataFrame(pd.np.arange(12).reshape(3, 4), index=['00', '01', '03'],
3. columns=['a', 'b', 'c', 'd'])





随机产生

1. df = pd.DataFrame(pd.np.random.randn(3, 4), index=['00', '01', '03'],
2. columns=['a', 'b', 'c', 'd'])



1.2字典转化

嵌套列表为一列

1. \_dict = {'a': ['a0', 'a1'], 'b': ['b0', 'b1']}
2. df = pd.DataFrame(\_dict)



嵌套字典为一列

1. \_dict = {'a': {0: 'a0', 1: 'a1'}, 'b': {0: 'b0', 1: 'b1'}}
2. df = pd.DataFrame(\_dict)



指定列顺序

1. \_dict = {'col1': ['a1', 'b1'], 'col2': ['a2', 'b2']}
2. df = 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', '03'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. # 列类型
6. **print**(df.dtypes)
7. # 行列表
8. **print**(df.index.tolist())
9. # 列列表
10. **print**(df.columns.tolist())
11. # 查看前两行
12. **print**(df.head(2))
13. # 查看后两行
14. **print**(df.tail(2))
15. # 转置
16. **print**(df.T)



3.筛选

3.1行列筛选

标签选取

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])
4. # 取单列得Series
5. **print**(df['b'])
6. # 取单列得DataFrame
7. **print**(df[['b']])
8. # 取多列得DataFrame
9. **print**(df[['b', 'c']])



行选择

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=['00', '01'],
3. columns=['a', 'b', 'c', 'd'])
4. # 行选择
5. **print**(df[0:1])



Loc选择

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])
5. # loc选取行
6. **print**(df.loc[:0,:])
7. # loc选取列
8. **print**(df.loc[:,'b':'d'])



3.2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. # 筛选某列大于某值的剩余行
6. **print**(df[df['b'] > 6])
7. # 筛选某列值在某集合中的剩余行
8. **print**(df[df['b'].isin([6, 10])])
9. # 所有大于6 的值
10. **print**(df[df > 6])



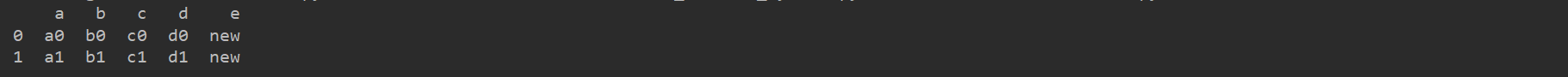
# 4.增删

4.1增加

增加一列

每行均为同一值

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])
4. df['e'] = 'new'



4.2删除

删除一列

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])
4. **del** df['a']



**注**：使用drop删除时，删除的是视图！

4.3修改

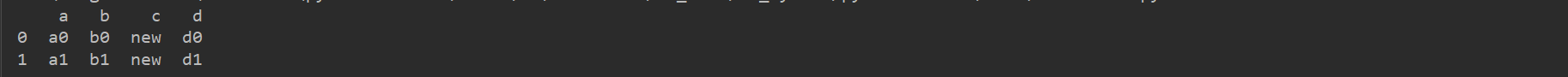
修改具体一个值

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])
4. df['b'][1] = 'new'



修改一列为同一值

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])
4. df['c'] = 'new'



修改一行为同一值

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])
4. df[:1] = 'new'



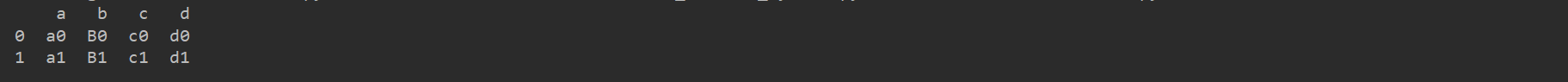
修改一列为Series

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])
5. df['b'] = pd.Series(['new1', 'new2', 'new3'])



修改一列同一str处理

1. df = pd.DataFrame([['a0', 'b0', 'c0', 'd0'], ['a1', 'b1', 'c1', 'd1']],
2. index=[0, 1],
3. columns=['a', 'b', 'c', 'd'])
5. df['b'] = df['b'].str.upper()



# 5.清洗

5.1排序

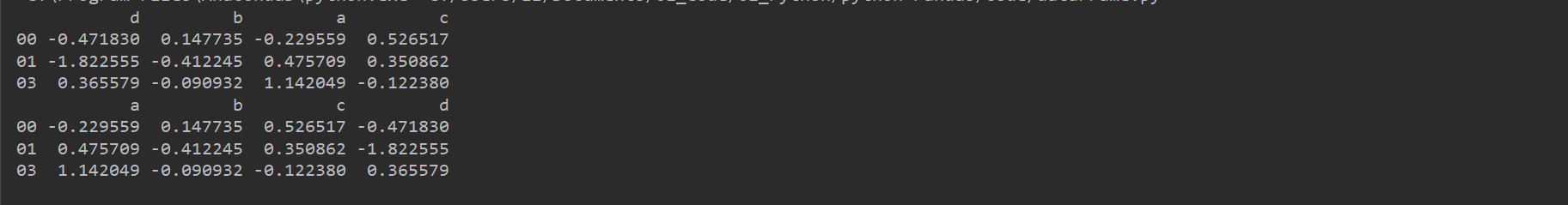
按值排序

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



按轴排序

1. df = pd.DataFrame(pd.np.random.randn(3, 4), index=['00', '01', '03'],
2. columns=['d', 'b', 'a', 'c'])
3. **print**(df)
4. **print**(df.sort\_index(axis=1))



5.2换名

1. df = pd.DataFrame(pd.np.random.randn(3, 4), index=['00', '01', '02'],
2. columns=['a', 'b', 'c', 'd'])
3. **print**(df)
4. # 列换名
5. df.rename(columns={'a': 'a1', 'b': 'b1'}, inplace=True)
6. **print**(df)
7. # 应用str
8. df.rename(str.upper, axis='columns', inplace=True)
9. **print**(df)

# 行换名

1. df.rename({'01': '001', '02': '002'}, axis='index', inplace=True)
2. **print**(df)



5.3移除

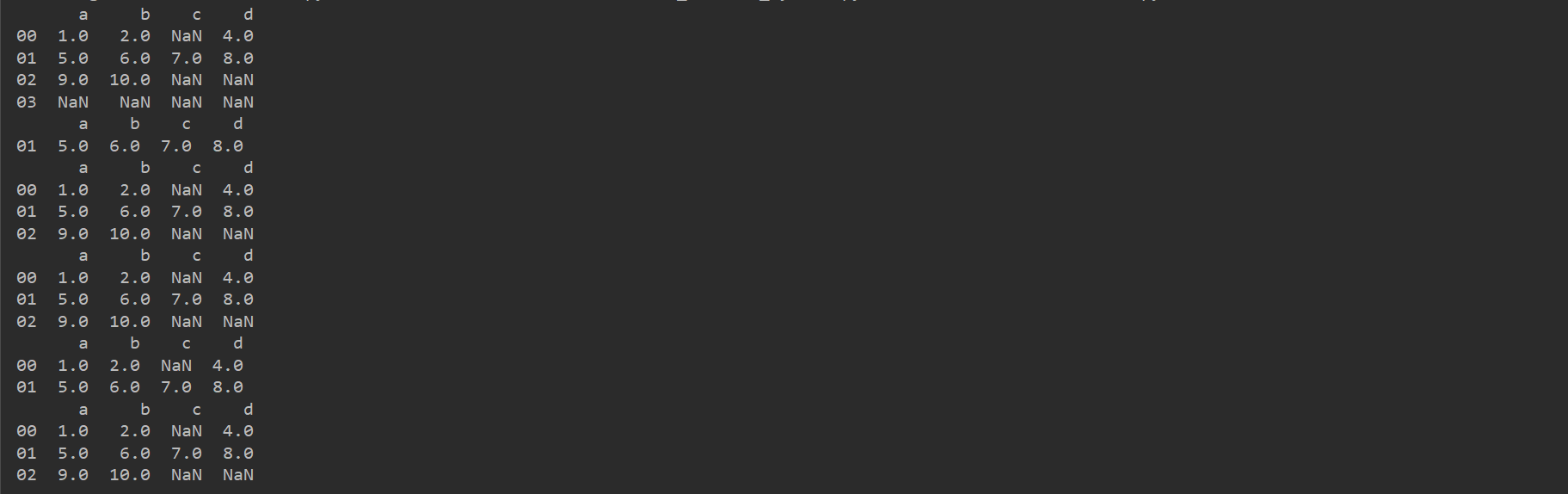
1. df = pd.DataFrame([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]],
2. index=['00', '01', '02'],
3. columns=['a', 'b', 'c', 'd'])
4. **print**(df)
5. # 移除行
6. **print**(df.drop(['00', '01']))
7. **print**(df.drop(index=['00', '02']))
9. # 移除列
10. **print**(df.drop(['a', 'b'], axis=1))
11. **print**(df.drop(columns=['a',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



5.4去空

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. # 删除任意有丢失的行
6. **print**(df.dropna(how='any'))
7. # 删除全丢失的行
8. **print**(df.dropna(how='all'))
9. df.dropna(how='all', inplace=True)
10. # 保持一行最多有几个NaN
11. **print**(df.dropna(thresh=2))
12. # 删除某些列任意有NaN
13. **print**(df.dropna(subset=['a', 'd']))
14. # 删除某些列全部为NaN
15. **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. # 删除任意有丢失的列
6. **print**(df.dropna(how='any', axis=1))
7. # 删除全丢失的列
8. **print**(df.dropna(how='all', axis=1))
9. df.dropna(how='all', axis=1, inplace=True)
10. # 保持一列最多有几个NaN
11. **print**(df.dropna(thresh=2, axis=1))
12. # 删除某些列任意有NaN
13. **print**(df.dropna(subset=['00', '01'], axis=1))
14. # # 删除某些列全部为NaN
15. df.dropna(subset=['00', '01'], how='all', axis=1, inplace=True)
16. **print**(df)

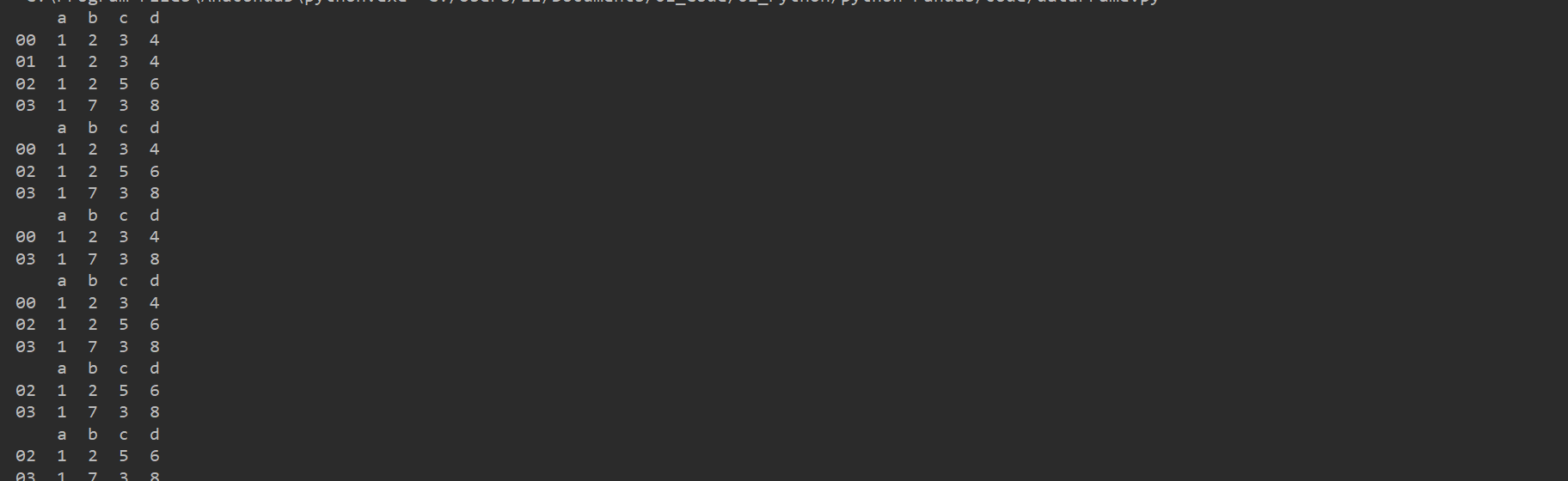


1. # None的效果与NaN等同
2. df = pd.DataFrame([[1, '', 3, 4], [5, 6, 7, None], [9, 10, None, None], [None, None, None, None]],
3. index=['00', '01', '02', '03'],
4. columns=['a', 'b', 'c', 'd'])
5. **print**(df)
6. **print**(df.dropna())
7. **print**(df.dropna(axis=1))



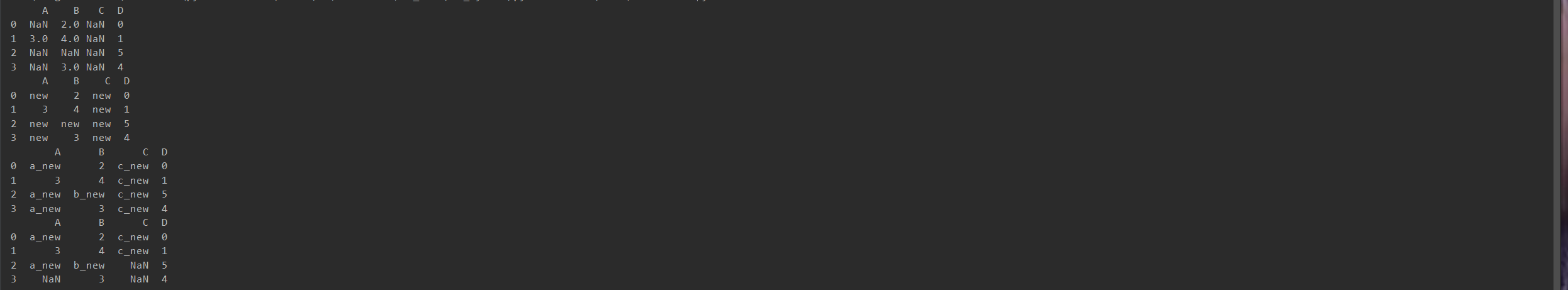
5.5去重

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. # 整行去重
6. **print**(df.drop\_duplicates())
7. # 选某几行去重
8. **print**(df.drop\_duplicates(subset=['a', 'b']))
9. # 重复默认保持第一行
10. **print**(df.drop\_duplicates(keep='first'))
11. # 重复保持最后一行
12. **print**(df.drop\_duplicates(keep='last', subset=['a', 'b']))
13. # 删除所有重复项
14. df.drop\_duplicates(keep=False, inplace=True)
15. **print**(df)



5.6填空

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. # 将所有NaN替换为固定值
7. **print**(df.fillna('new'))
8. # 将NaN值不同行改为不同值
9. values = {'A': 'a\_new', 'B': 'b\_new', 'C': 'c\_new', 'D': 'd\_new'}
10. **print**(df.fillna(value=values))
11. # 修改固定数量
12. **print**(df.fillna(value=values, limit=2))



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)
7. # 将NaN值改为前一个值
8. **print**(df.fillna(method='ffill'))
9. # 等价
10. **print**(df.ffill())
11. # 将NaN值改为后一个值
12. **print**(df.fillna(method='bfill'))
13. # 等价
14. **print**(df.bfill())



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)
7. # 从左到右，替换NaN
8. **print**(df.ffill(axis=1))
9. # 从右到左，替换NaN
10. **print**(df.bfill(axis=1))



5.7替换

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. # 旧值
7. **print**(df.replace(3, 33))



## 5.8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. # 将一列作为index
6. **print**(df.set\_index('month'))
7. # 将几列组合作为index
8. **print**(df.set\_index(['month', 'year']))
9. # 将index重新赋值
10. **print**(df.set\_index([[1, 2, 3, 4], 'year']))



## 5.9index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. # 将index作为列名为index的一列
10. **print**(df.reset\_index())
11. # 将index去掉并改为索引
12. **print**(df.reset\_index(drop=True))

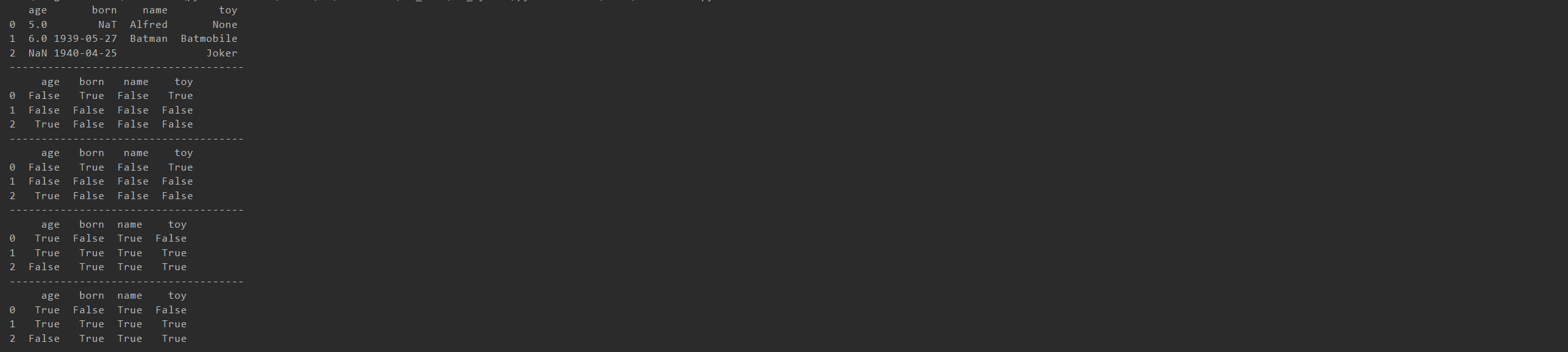


1. index = pd.MultiIndex.from\_tuples([('bird', 'falcon'),
2. ('bird', 'parrot'),
3. ('mammal', 'lion'),
4. ('mammal', 'monkey')],
5. names=['class', 'name'])
6. columns = pd.MultiIndex.from\_tuples([('speed', 'max'),
7. ('species', 'type')])
9. df = pd.DataFrame([(389.0, 'fly'),
10. (24.0, 'fly'),
11.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



5.10 判空

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**('-------------------------------------')
8. # 判断是否为空
9. **print**(df.isna())
10. **print**('-------------------------------------')
11. **print**(df.isnull())
12. **print**('-------------------------------------')
13. **print**(df.notna())
14. **print**('-------------------------------------')
15. **print**(df.notnull())



5.10包含

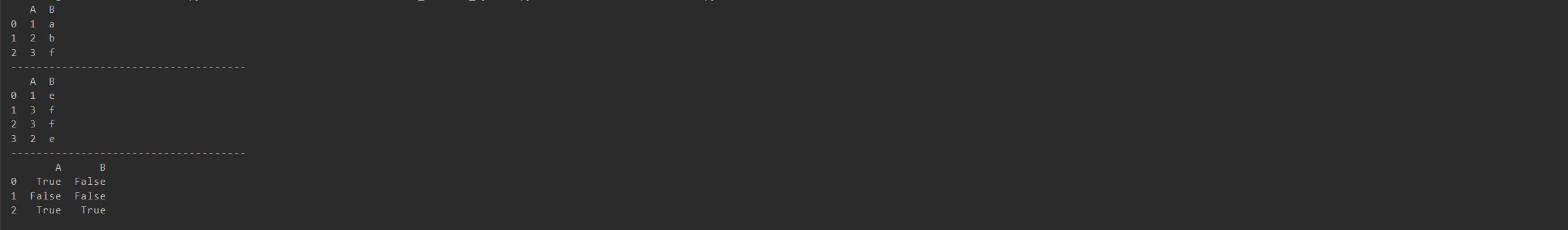
1. df = pd.DataFrame({'A': [1, 2, 3], 'B': ['a', 'b', 'f']})
2. **print**(df)
3. **print**('-------------------------------------')
4. **print**(df.isin([1, 3, 12, 'a']))



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]}))



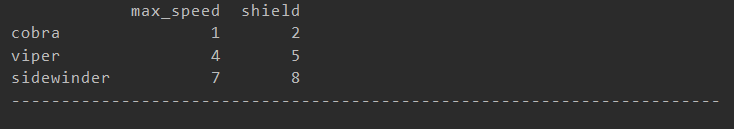
1. df = pd.DataFrame({'A': [1, 2, 3], 'B': ['a', 'b', 'f']})
2. **print**(df)
3. **print**('-------------------------------------')
4. other = pd.DataFrame({'A': [1, 3, 3, 2], 'B': ['e', 'f', 'f', 'e']})
5. **print**(other)
6. **print**('-------------------------------------')
7. **print**(df.isin(other))



5.11判断

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





1. **print**('-------------------------------------')
2. # 等价
3. **print**(df.where(m, -df) == pd.np.where(m, df, -df))
4. **print**('-------------------------------------')
5. # 取反
6. **print**(df.where(m, -df) == df.mask(~m, -df))

