Pandas进阶修炼120题

刘早起早起 早起python 今天

『**Pandas进阶修炼120题**』系列现已完结，我们对Pandas中常用的操作以习题的形式发布。从读取数据到高级操作全部包含，希望可以通过刷题的方式来完整学习pandas中数据处理的各种方法，当然如果你是高手，也欢迎尝试给出与答案不同的解法。

1 创建DataFrame

题目：将下面的字典创建为DataFrame

```
data = {"grammer":["Python","C","Java","GO",np.nan,"SQL","PHP","Python"]
        "score":[1,2,np.nan,4,5,6,7,10]}
```

难度：☆

期望结果

	grammer	score
0	Python	1.0
1	C	2.0
2	Java	NaN
3	GO	4.0
4	R	5.0
5	SQL	6.0
6	PHP	7.0
7	Python	10.0

答案：

```
df = pd.DataFrame(data)
```

本期所有题目均基于该数据框给出

2 数据提取

题目：提取含有字符串"Python"的行

难度：☆☆

期望结果

```
  grammer  score
0  Python    1.0
7  Python   10.0
```

答案：

```
result=df[df['grammer'].str.contains("Python")]
```

3 提取列名

题目：输出df的所有列名

难度：☆

期望结果

```
Index(['grammer', 'score'], dtype='object')
```

答案

```
df.columns
```

4 修改列名

题目：修改第二列列名为'popularity'

难度：☆☆

答案

```
df.rename(columns={'score':'popularity'}, inplace = True)
```

5 字符统计

题目：统计grammer列中每种编程语言出现的次数

难度：☆☆

答案

```
df['grammer'].value_counts()
```

6 缺失值处理

题目：将空值用上下值的平均值填充

难度：☆☆☆

答案

```
df['popularity'] = df['popularity'].fillna(df['popularity'].interpolate(
```

7 数据提取

题目：提取popularity列中值大于3的行

难度：☆☆

答案

```
df[df['popularity'] > 3]
```

8 数据去重

题目：按照grammer列进行去重

难度：☆☆

答案

```
df.drop_duplicates(['grammer'])
```

9 数据计算

题目：计算popularity列平均值

难度：☆☆

答案

```
df['popularity'].mean()
```

10 格式转换

题目：将grammer列转换为list

难度：☆☆

答案

```
df['grammer'].to_list()
```

11 数据保存

题目：将DataFrame保存为EXCEL

难度：☆☆

答案

```
df.to_excel('filename.xlsx')
```

12 数据查看

题目：查看数据行列数

难度：☆

答案

```
df.shape
```

13 数据提取

题目：提取popularity列值大于3小于7的行

难度：☆☆

答案

```
df[(df['popularity'] > 3) & (df['popularity'] < 7)]
```

14 位置处理

题目：交换两列位置

难度：☆☆☆

答案

```
'''
方法1
'''
temp = df['popularity']
df.drop(labels=['popularity'], axis=1,inplace = True)
df.insert(0, 'popularity', temp)
df
'''
方法2
cols = df.columns[[1,0]]
df = df[cols]
df
'''
```

15 数据提取

题目：提取popularity列最大值所在行

难度：☆☆

答案

```
df[df['popularity'] == df['popularity'].max()]
```

16 数据查看

题目：查看最后5行数据

难度：☆

答案

```
df.tail()
```

17 数据修改

题目：删除最后一行数据

难度：☆

答案

```
df.drop([len(df)-1],inplace=True)
```

18 数据修改

题目：添加一行数据['Perl',6.6]

难度：☆☆

答案

```
row={'grammer':'Perl','popularity':6.6}
df = df.append(row,ignore_index=True)
```

19 数据整理

题目：对数据按照"popularity"列值的大小进行排序

难度：☆☆

答案

```
df.sort_values("popularity",inplace=True)
```

20 字符统计

题目：统计grammer列每个字符串的长度

难度：☆☆☆

答案

```
df['grammer'].map(lambda x: len(x))
```

第二期：数据处理基础

21 数据读取

题目：读取本地EXCEL数据

难度：☆

答案

```
df = pd.read_excel('pandas120.xlsx')
```

本期部分习题与该数据相关

22 数据查看

题目：查看df数据前5行

难度：☆

期望输出

	createTime	education	salary
0	2020-03-16 11:30:18	本科	20k-35k
1	2020-03-16 10:58:48	本科	20k-40k
2	2020-03-16 10:46:39	不限	20k-35k
3	2020-03-16 10:45:44	本科	13k-20k
4	2020-03-16 10:20:41	本科	10k-20k

答案

```
df.head()
```

23 数据计算

题目：将salary列数据转换为最大值与最小值的平均值

难度：☆☆☆☆

期望输出

	createTime	education	salary
0	2020-03-16 11:30:18	本科	27500
1	2020-03-16 10:58:48	本科	30000
2	2020-03-16 10:46:39	不限	27500
3	2020-03-16 10:45:44	本科	16500
4	2020-03-16 10:20:41	本科	15000

答案

```
#备注，在某些版本pandas中.ix方法可能失效，可使用.iloc，参考https://mp.weixin.qq.
#为什么不能直接使用max，min函数，因为我们的数据中是20k-35k这种字符串，所以需要先用正则
import re
for i in range(len(df)):
    str1 = df.ix[i,2]
    k = re.findall(r"\d+\.?\d*",str1)
    salary = ((int(k[0]) + int(k[1]))/2)*1000
    df.ix[i,2] = salary
df
```

24 数据分组

题目：将数据根据学历进行分组并计算平均薪资

难度：☆☆☆

期望输出

```
education salary
不限 19600.000000
大专 10000.000000
本科 19361.344538
```

```
硕士 20642.857143
```

答案

```
df.groupby('education').mean()
```

25 时间转换

题目：将`createTime`列时间转换为`月-日`

难度：☆☆☆

期望输出

	createTime	education	salary
0	03-16	本科	27500
1	03-16	本科	30000
2	03-16	不限	27500
3	03-16	本科	16500
4	03-16	本科	15000

答案

```
#备注，在某些版本pandas中.ix方法可能失效，可使用.iloc，参考https://mp.weixin.qq.
for i in range(len(df)):
    df.ix[i,0] = df.ix[i,0].to_pydatetime().strftime("%m-%d")
df.head()
```

26 数据查看

题目：查看索引、数据类型和内存信息

难度：☆

期望输出

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 135 entries, 0 to 134
Data columns (total 4 columns):
createTime 135 non-null object
```

```
education 135 non-null object
salary 135 non-null int64
categories 135 non-null category
dtypes: category(1), int64(1), object(2)
memory usage: 3.5+ KB
```

答案

```
df.info()
```

27 数据查看

题目：查看数值型列的汇总统计

难度：☆

答案

```
df.describe()
```

28 数据整理

题目：新增一列根据salary将数据分为三组

难度：☆☆☆☆

输入

期望输出

	createTime	education	salary	categories
0	03-16	本科	27500	高
1	03-16	本科	30000	高
2	03-16	不限	27500	高
3	03-16	本科	16500	中
4	03-16	本科	15000	中

答案

```
bins = [0,5000, 20000, 50000]
group_names = ['低', '中', '高']
df['categories'] = pd.cut(df['salary'], bins, labels=group_names)
```

29 数据整理

题目：按照salary列对数据降序排列

难度：☆☆

答案

```
df.sort_values('salary', ascending=False)
```

30 数据提取

题目：取出第33行数据

难度：☆☆

答案

```
df.loc[32]
```

31 数据计算

题目：计算salary列的中位数

难度：☆☆

答案

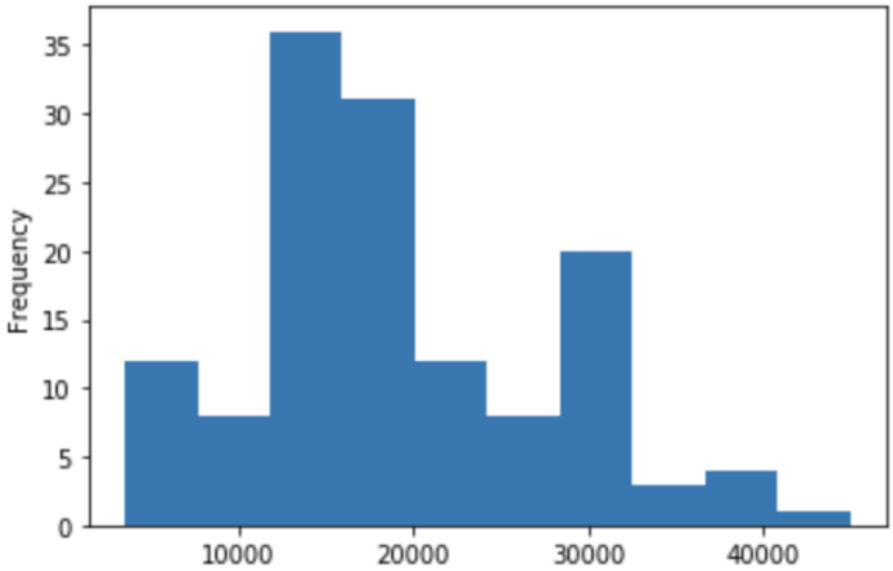
```
np.median(df['salary'])
```

32 数据可视化

题目：绘制薪资水平频率分布直方图

难度：☆☆☆

期望输出



答案

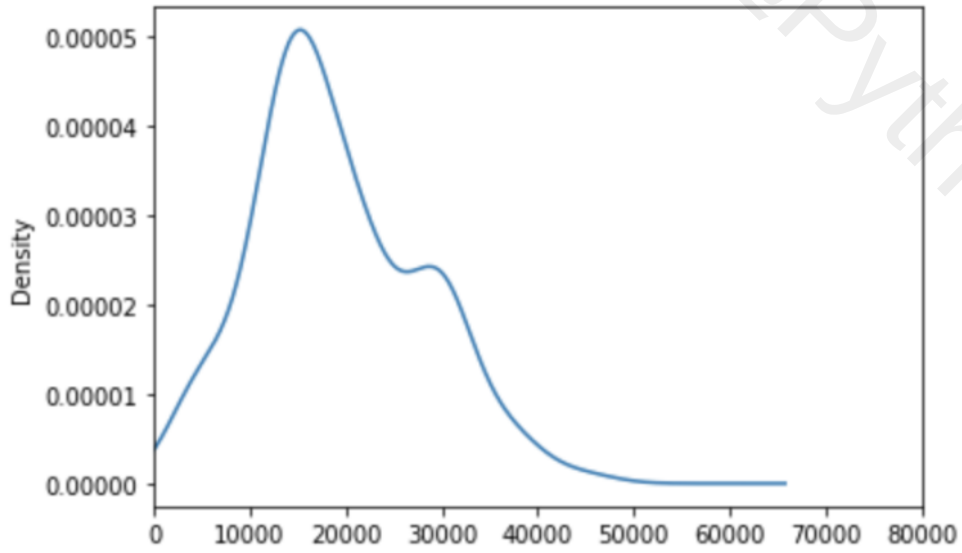
```
df.salary.plot(kind='hist')
```

33 数据可视化

题目：绘制薪资水平密度曲线

难度：☆☆☆

期望输出



答案

```
df.salary.plot(kind='kde',xlim=(0,80000))
```

34 数据删除

题目：删除最后一列categories

难度：☆

答案

```
del df['categories']
```

35 数据处理

题目：将df的第一列与第二列合并为新的一列

难度：☆☆

答案

```
df['test'] = df['education']+df['createTime']
```

36 数据处理

题目：将education列与salary列合并为新的一列

难度：☆☆☆

备注：salary为int类型，操作与35题有所不同

答案

```
df["test1"] = df["salary"].map(str) + df['education']
```

37 数据计算

题目：计算salary最大值与最小值之差

难度：☆☆☆

答案

```
df[['salary']].apply(lambda x: x.max() - x.min())
```

38 数据处理

题目：将第一行与最后一行拼接

难度：☆☆

答案

```
pd.concat([df[:1], df[-2:-1]])
```

39 数据处理

题目：将第8行数据添加至末尾

难度：☆☆

答案

```
df.append(df.iloc[7])
```

40 数据查看

题目：查看每列的数据类型

难度：☆

期望结果

```
createTime object
education object
salary int64
test object
test1 object
dtype: object
```

答案

```
df.dtypes
```

41 数据处理

题目：将createTime列设置为索引

难度：☆☆

答案

```
df.set_index("createTime")
```

42 数据创建

题目：生成一个和df长度相同的随机数dataframe

难度：☆☆

答案

```
df1 = pd.DataFrame(pd.Series(np.random.randint(1, 10, 135)))
```

43 数据处理

题目：将上一题生成的dataframe与df合并

难度：☆☆

答案

```
df= pd.concat([df,df1],axis=1)
```

44 数据计算

题目：生成新的一列new为salary列减去之前生成随机数列

难度：☆☆

答案

```
df["new"] = df["salary"] - df[0]
```

45 缺失值处理

题目：检查数据中是否含有任何缺失值

难度：☆☆☆

答案

```
df.isnull().values.any()
```

46 数据转换

题目：将salary列类型转换为浮点数

难度：☆☆☆

答案

```
df['salary'].astype(np.float64)
```

47 数据计算

题目：计算salary大于10000的次数

难度：☆☆

答案

```
len(df[df['salary']>10000])
```

48 数据统计

题目：查看每种学历出现的次数

难度：☆☆☆

期望输出

```
本科 119
硕士 7
不限 5
大专 4
Name: education, dtype: int64
```

答案

```
df.education.value_counts()
```

49 数据查看

题目： 查看education列共有几种学历

难度： ☆☆

答案

```
df['education'].nunique()
```

50 数据提取

题目： 提取salary与new列的和大于60000的最后3行

难度： ☆☆☆☆

期望输出

	createTime	education	salary	test	test1	0	new
92	03-16	本科	35000	本科03-16	35000本科	6	34994
101	03-16	本科	37500	本科03-16	37500本科	5	37495
131	03-16	硕士	37500	硕士03-16	37500硕士	6	37494

答案

```
df1 = df[['salary','new']]
rowsums = df1.apply(np.sum, axis=1)
res = df.iloc[np.where(rowsums > 60000)[0][-3:], :]
```


第三期：金融数据处理

51 数据读取

题目： 使用绝对路径读取本地Excel数据

难度： ☆

答案

```
data = pd.read_excel('/Users/Desktop/600000.SH.xls')
```

备注

请将答案中路径替换为自己机器存储数据的绝对路径，本期相关习题与该数据有关

52 数据查看

题目： 查看数据前三行

难度： ☆

期望结果

	代码	简称	日期	前收盘价(元)	开盘价(元)	最高价(元)	最低价(元)	收盘价(元)	成交量(股)	成交金额(元)	涨跌(元)	涨跌幅(%)	均价(元)	换手率(%)	A股流通市值(元)	总市值(元)
0	600000.SH	浦发银行	2016-01-04	16.1356	16.1444	16.1444	15.4997	15.7205	42240610	754425783	-0.4151	-2.5725	17.8602	0.2264	3.320318e+11	3.320318e+11
1	600000.SH	浦发银行	2016-01-05	15.7205	15.4644	15.9501	15.3672	15.8618	58054793	1034181474	0.1413	0.8989	17.8139	0.3112	3.350163e+11	3.350163e+11
2	600000.SH	浦发银行	2016-01-06	15.8618	15.8088	16.0208	15.6234	15.9855	46772653	838667398	0.1236	0.7795	17.9307	0.2507	3.376278e+11	3.376278e+11

答案

```
data.head(3)
```

53 缺失值处理

题目： 查看每列数据缺失值情况

难度： ☆☆

期望结果

代码 **1**
简称 **2**
日期 **2**
前收盘价(元) **2**
开盘价(元) **2**
最高价(元) **2**

最低价(元) 2
收盘价(元) 2
成交量(股) 2
成交金额(元) 2
.....

答案

```
data.isnull().sum()
```

54 缺失值处理

题目：提取日期列含有空值的行

难度：☆☆

期望结果

	代码	简称	日期	前收盘价(元)	开盘价(元)	最高价(元)	最低价(元)	收盘价(元)	成交量(股)	成交金额(元)	涨跌(元)	涨跌幅(%)	均价(元)	换手率(%)	A股流通市值(元)	总市值(元)	A股流通股本(股)	市盈率
327	NaN	NaN	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
328	数据来源：Wind资讯	NaN	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

答案

```
data[data['日期'].isnull()]
```

55 缺失值处理

题目：输出每列缺失值具体行数

难度：☆☆☆

期望结果

列名："代码"，第[327]行位置有缺失值
列名："简称"，第[327，328]行位置有缺失值
列名："日期"，第[327，328]行位置有缺失值
列名："前收盘价(元)"，第[327，328]行位置有缺失值
列名："开盘价(元)"，第[327，328]行位置有缺失值
列名："最高价(元)"，第[327，328]行位置有缺失值
列名："最低价(元)"，第[327，328]行位置有缺失值
列名："收盘价(元)"，第[327，328]行位置有缺失值
.....

答案

```
for columnname in data.columns:
    if data[columname].count() != len(data):
        loc = data[columname][data[columname].isnull().values==True].ind
        print('列名："{}", 第{}行位置有缺失值'.format(columname,loc))
```

56 缺失值处理

题目：删除所有存在缺失值的行

难度：☆☆

答案

```
data.dropna(axis=0, how='any', inplace=True)
```

备注

axis：0-行操作（默认），1-列操作
how：any-只要有空值就删除（默认），all-全部为空值才删除
inplace：**False**-返回新的数据集（默认），**True**-在原数据集上操作

57 数据可视化

题目：绘制收盘价的折线图

难度：☆☆

期望结果

答案

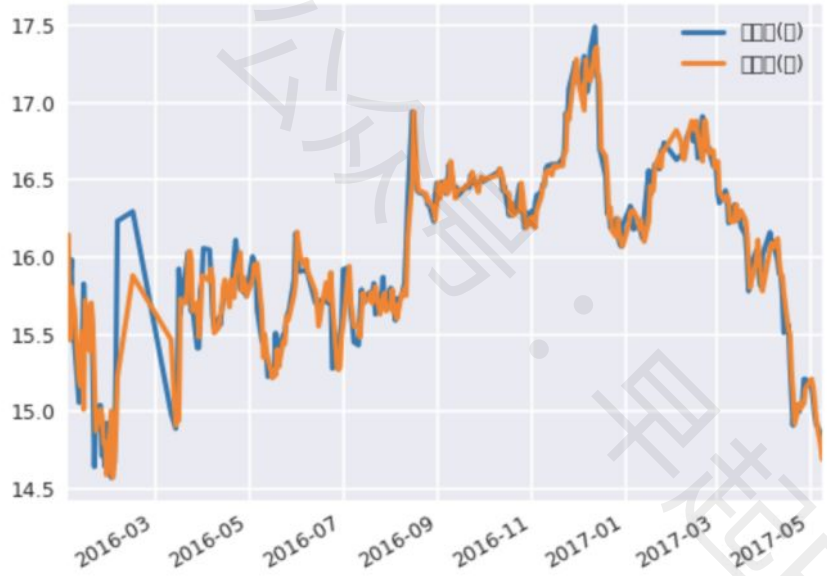
```
data['收盘价(元)'].plot()
```

58 数据可视化

题目：同时绘制开盘价与收盘价

难度：☆☆☆

期望结果



答案

```
data[['收盘价(元)','开盘价(元)']].plot()
```

备注

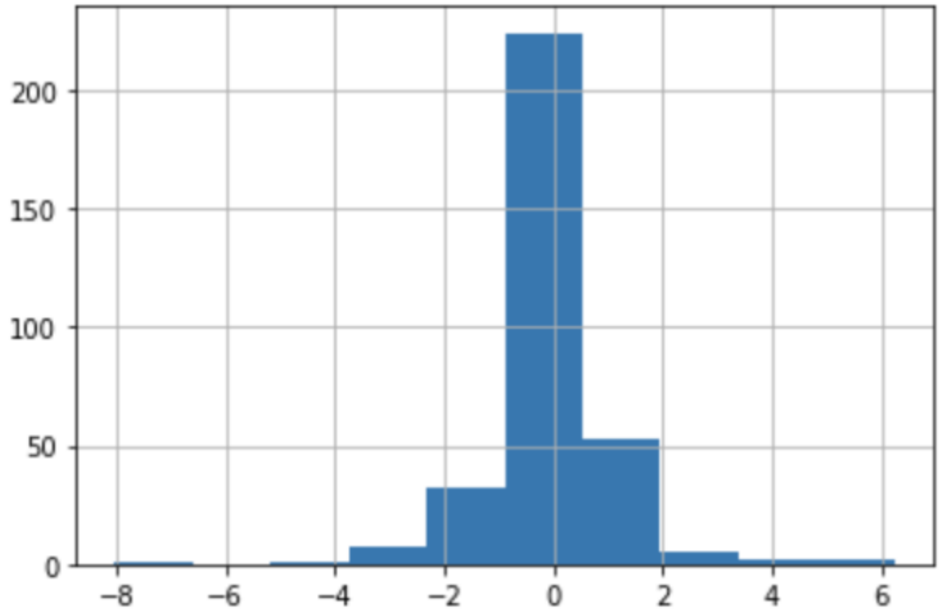
中文显示请自己设置，我的字体乱了

59 数据可视化

题目：绘制涨跌幅的直方图

难度：☆☆

期望结果



答案

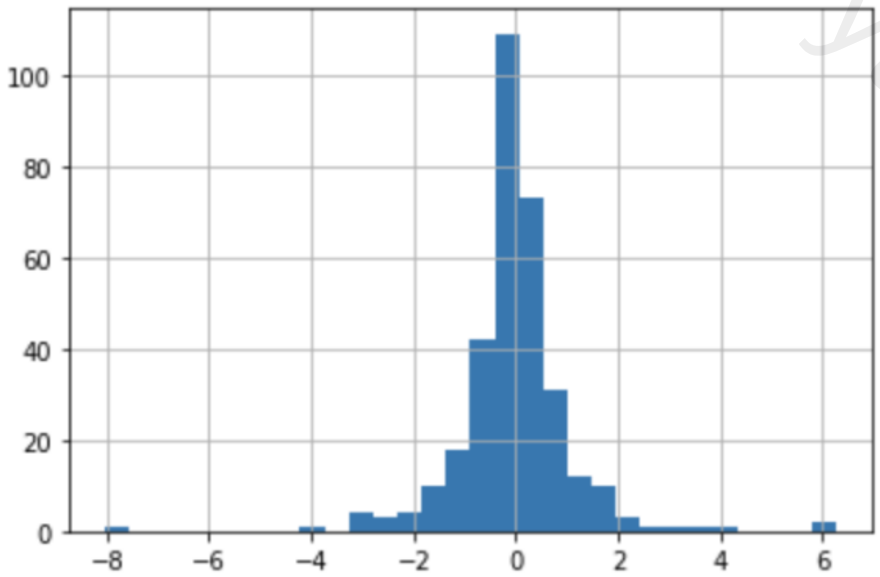
```
data['涨跌幅(%)'].hist()
```

60 数据可视化

题目：让直方图更细致

难度：☆☆

期望结果



答案

```
data['涨跌幅(%)'].hist(bins = 30)
```

61 数据创建

题目： 以data的列名创建一个dataframe

难度： ☆☆

答案

```
temp = pd.DataFrame(columns = data.columns.to_list())
```

62 异常值处理

题目： 打印所有换手率不是数字的行

难度： ☆☆☆

期望结果

	代码	简称	日期	前收盘价(元)	开盘价(元)	最高价(元)	最低价(元)	收盘价(元)	成交量(股)	成交金额(元)	涨跌(元)	涨跌幅(%)	均价(元)	换手率(%)	A股流通市值(元)	总市值(元)	A股流通股本(股)	市盈率
26	600000.SH	浦发银行	2016-02-16	16.2946	16.2946	16.2946	16.2946	16.2946	--	--	0.0	0.0	--	--	3.441565e+11	3.441565e+11	1.865347e+10	6.801
27	600000.SH	浦发银行	2016-02-17	16.2946	16.2946	16.2946	16.2946	16.2946	--	--	0.0	0.0	--	--	3.441565e+11	3.441565e+11	1.865347e+10	6.801
28	600000.SH	浦发银行	2016-02-18	16.2946	16.2946	16.2946	16.2946	16.2946	--	--	0.0	0.0	--	--	3.441565e+11	3.441565e+11	1.865347e+10	6.801

答案

```
for i in range(len(data)):
    if type(data.iloc[i,13]) != float:
        temp = temp.append(data.loc[i])

temp
```

63 异常值处理

题目： 打印所有换手率为--的行

难度： ☆☆☆

答案

```
data[data['换手率(%)'].isin(['--'])]
```

备注

通过上一题我们发现换手率的异常值只有--

64 数据处理

题目： 重置data的行号

难度： ☆

答案

```
data = data.reset_index()
```

备注

有时我们修改数据会导致索引混乱

65 异常值处理

题目： 删除所有换手率为非数字的行

难度： ☆☆☆

答案

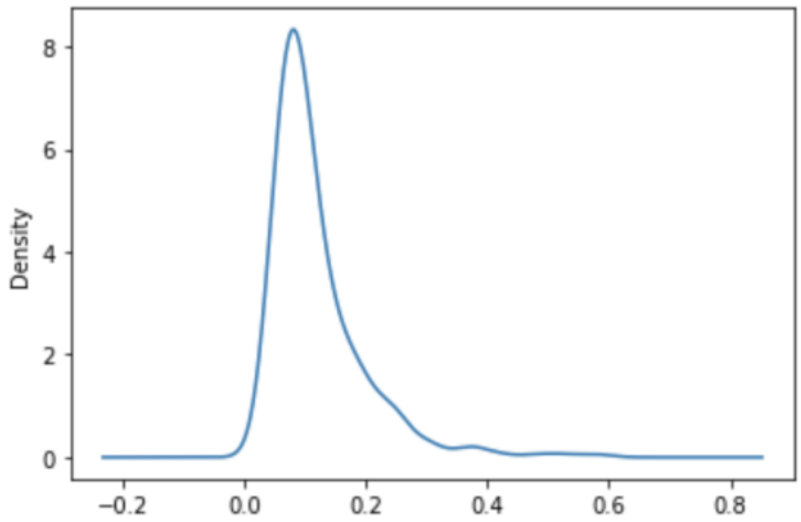
```
k =[]
for i in range(len(data)):
    if type(data.iloc[i,13]) != float:
        k.append(i)
data.drop(labels=k,inplace=True)
```

66 数据可视化

题目： 绘制换手率的密度曲线

难度： ☆☆☆

期望结果



答案

```
data['换手率(%)'].plot(kind='kde')
```

67 数据计算

题目： 计算前一天与后一天收盘价的差值

难度： ☆☆

答案

```
data['收盘价(元)'].diff()
```

68 数据计算

题目： 计算前一天与后一天收盘价变化率

难度： ☆☆

答案

```
data['收盘价(元)'].pct_change()
```

69 数据处理

题目： 设置日期为索引

难度： ☆

答案

```
data.set_index('日期')
```

70 指标计算

题目： 以5个数据作为一个数据滑动窗口，在这个5个数据上取均值(收盘价)

难度： ☆☆☆

答案

```
data['收盘价(元)'].rolling(5).mean()
```

71 指标计算

题目： 以5个数据作为一个数据滑动窗口，计算这五个数据总和(收盘价)

难度： ☆☆☆

答案

```
data['收盘价(元)'].rolling(5).sum()
```

72 数据可视化

题目： 将收盘价5日均线、20日均线与原始数据绘制在同一个图上

难度： ☆☆☆

期望结果

答案

```
data['收盘价(元)'].plot()
data['收盘价(元)'].rolling(5).mean().plot()
data['收盘价(元)'].rolling(20).mean().plot()
```

73 数据重采样

题目： 按周为采样规则，取一周收盘价最大值

难度： ☆☆☆

答案

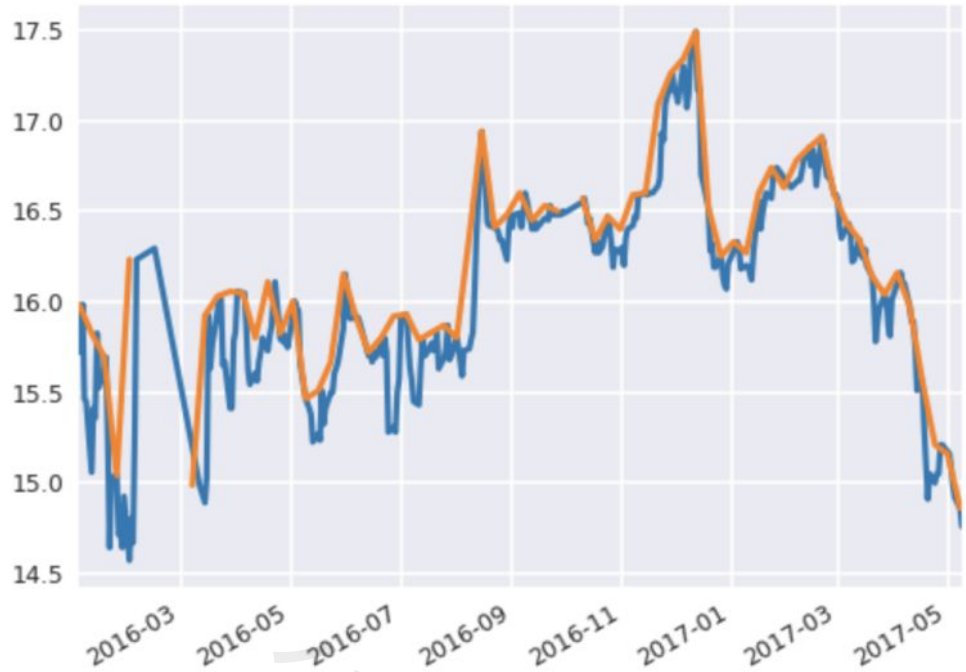
```
data['收盘价(元)'].resample('W').max()
```

74 Spyder——Python编程的“热带雨林”

题目： 绘制重采样数据与原始数据

难度： ☆☆☆

期望结果



答案

```
data['收盘价(元)'].plot()
data['收盘价(元)'].resample('7D').max().plot()
```

75 数据处理

题目：将数据往后移动5天

难度：☆☆

答案

```
data.shift(5)
```

76 数据处理

题目：将数据向前移动5天

难度：☆☆

答案

```
data.shift(-5)
```

77 数据计算

题目：使用expending函数计算开盘价的移动窗口均值

难度：☆☆

答案

```
data['开盘价(元)'].expanding(min_periods=1).mean()
```

78 数据可视化

题目：绘制上一题的移动均值与原始数据折线图

难度：☆☆☆

期望结果

答案

```
data[' expanding Open mean']=data['开盘价(元)'].expanding(min_periods=1).
data[['开盘价(元)', 'expanding Open mean']].plot(figsize=(16, 6))
```

79 数据计算

题目：计算布林指标

难度：☆☆☆☆

答案

```
data['former 30 days rolling Close mean']=data['收盘价(元)'].rolling(20).
data['upper bound']=data['former 30 days rolling Close mean']+2*data['收
```

```
data['lower bound']=data['former 30 days rolling Close mean']-2*data['收
```

80 数据可视化

题目： 计算布林线并绘制

难度： ☆☆☆

期望结果

答案

```
data[['收盘价(元)', 'former 30 days rolling Close mean','upper bound','lo
```

第四期：当Pandas遇上NumPy

81 数据查看

题目： 导入并查看pandas与numpy版本

难度： ☆

答案

```
import pandas as pd
import numpy as np
print(np.__version__)
print(pd.__version__)
```

82 数据创建

题目：从NumPy数组创建DataFrame

难度：☆

备注

使用numpy生成20个0-100**随机数**

答案

```
tem = np.random.randint(1,100,20)
df1 = pd.DataFrame(tem)
```

83 数据创建

题目：从NumPy数组创建DataFrame

难度：☆

备注

使用numpy生成20个0-100**固定步长**的数

答案

```
tem = np.arange(0,100,5)
df2 = pd.DataFrame(tem)
```

84 数据创建

题目：从NumPy数组创建DataFrame

难度：☆

备注

使用numpy生成20个**指定分布(如标准正态分布)**的数

答案

```
tem = np.random.normal(0, 1, 20)
df3 = pd.DataFrame(tem)
```

85 数据创建

题目：将df1，df2，df3按照行合并为新DataFrame

难度：☆☆

答案

```
df = pd.concat([df1,df2,df3],axis=0,ignore_index=True)
```

86 数据创建

题目：将df1，df2，df3按照列合并为新DataFrame

难度：☆☆

期望结果

```
0 1 2
0 95 0 0.022492
1 22 5 -1.209494
2 3 10 0.876127
3 21 15 -0.162149
4 51 20 -0.815424
5 30 25 -0.303792
..............
```

答案

```
df = pd.concat([df1,df2,df3],axis=1,ignore_index=True)
df
```

87 数据查看

题目：查看df所有数据的最小值、25%分位数、中位数、75%分位数、最大值

难度：☆☆

答案

```
print(np.percentile(df, q=[0, 25, 50, 75, 100]))
```

88 数据修改

题目：修改列名为col1,col2,col3

难度：☆

答案

```
df.columns = ['col1','col2','col3']
```

89 数据提取

题目：提取第一列中不在第二列出现的数字

难度：☆☆☆

答案

```
df['col1'][~df['col1'].isin(df['col2'])]
```

90 数据提取

题目：提取第一列和第二列出现频率最高的三个数字

难度：☆☆☆

答案

```
temp = df['col1'].append(df['col2'])
temp.value_counts().index[:3]
```

91 数据提取

题目：提取第一列中可以整除5的数字位置

难度：☆☆☆

答案

```
np.argwhere(df['col1'] % 5==0)
```

92 数据计算

题目：计算第一列数字前一个与后一个的差值

难度：☆☆

答案

```
df['col1'].diff().tolist()
```

93 数据处理

题目：将col1,col2,clo3三列顺序颠倒

难度：☆☆

答案

```
df.ix[:, ::-1]
```

94 数据提取

题目：提取第一列位置在1,10,15的数字

难度：☆☆

答案

```
df['col1'].take([1,10,15])
```

95 数据查找

题目：查找第一列的局部最大值位置

难度：☆☆☆☆

备注

即比它前一个与后一个数字的都大的数字

答案

```
tem = np.diff(np.sign(np.diff(df['col1'])))
```

```
np.where(tem == -2)[0] + 1
```

96 数据计算

题目：按行计算df的每一行均值

难度：☆☆

答案

```
df[['col1','col2','col3']].mean(axis=1)
```

97 数据计算

题目：对第二列计算移动平均值

难度：☆☆☆

备注

每次移动三个位置，不可以使用自定义函数

答案

```
np.convolve(df['col2'], np.ones(3)/3, mode='valid')
```

98 数据修改

题目：将数据按照第三列值的大小升序排列

难度：☆☆

答案

```
df.sort_values("col3",inplace=True)
```

99 数据修改

题目：将第一列大于50的数字修改为'高'

难度：☆☆

答案

```
df.col1[df['col1'] > 50]= '高'
```

100 数据计算

题目：计算第一列与第二列之间的欧式距离

难度：☆☆☆

备注

不可以使用自定义函数

答案

```
np.linalg.norm(df['col1']-df['col2'])
```

第五期：一些补充

101 数据读取

题目：从CSV文件中读取指定数据

难度：☆☆

备注

从数据1中的前10行中读取positionName, salary两列

答案

```
df = pd.read_csv('数据1.csv',encoding='gbk', usecols=['positionName', 'sa
```

102 数据读取

题目：从CSV文件中读取指定数据

难度：☆☆

备注

从数据2中读取数据并在读取数据时将薪资大于10000的为改为高

答案

```
df = pd.read_csv('数据2.csv',converters={'薪资水平': lambda x: '高' if flo
```

103 数据计算

题目：从dataframe提取数据

难度：☆☆☆

备注

从上一题数据中，对薪资水平列每隔20行进行一次抽样

期望结果

	薪资水平
0	高
20	高
40	高
60	高
80	高
100	高
120	高
140	高
160	高
180	高

答案

```
df.iloc[::20, :][['薪资水平']]
```

104 数据处理

题目：将数据取消使用科学计数法

难度：☆☆

输入

```
df = pd.DataFrame(np.random.random(10)**10, columns=['data'])
```

期望结果

	data
0	0.078
1	0.029
2	0.002
3	0.000
4	0.000
5	0.000
6	0.007
7	0.000
8	0.000
9	0.004

答案

```
df.round(3)
```

105 数据处理

题目：将上一题的数据转换为百分数

难度：☆☆☆

期望结果

	data
0	7.75%
1	2.94%
2	0.22%
3	0.00%
4	0.00%
5	0.00%
6	0.65%
7	0.01%
8	0.00%
9	0.38%

答案

```
df.style.format({'data': '{0:.2%}'.format})
```

106 数据查找

题目：查找上一题数据中第3大值的行号

难度：☆☆☆

答案

```
df['data'].argsort()[::-1][7]
```

107 数据处理

题目：反转df的行

难度：☆☆

答案

```
df.iloc[::-1, :]
```

108 数据重塑

题目：按照多列对数据进行合并

难度：☆☆

输入

```
df1= pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
'key2': ['K0', 'K1', 'K0', 'K1'],
'A': ['A0', 'A1', 'A2', 'A3'],
'B': ['B0', 'B1', 'B2', 'B3']})

df2= pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
'key2': ['K0', 'K0', 'K0', 'K0'],
'C': ['C0', 'C1', 'C2', 'C3'],
'D': ['D0', 'D1', 'D2', 'D3']})
```

答案

```
pd.merge(df1, df2, on=['key1', 'key2'])
```

109 数据重塑

题目：按照多列对数据进行合并

难度：☆☆

备注

只保存df1的数据

答案

```
pd.merge(df1, df2, how='left', on=['key1', 'key2'])
```

110 数据处理

题目：再次读取数据1并显示所有的列

难度：☆☆

备注

数据中由于列数较多中间列不显示

答案

```
df = pd.read_csv('数据1.csv',encoding='gbk')
pd.set_option("display.max.columns", None)
df
```

111 数据查找

题目：查找secondType与thirdType值相等的行号

难度：☆☆

答案

```
np.where(df.secondType == df.thirdType)
```

112 数据查找

题目：查找薪资大于平均薪资的第三个数据

难度：☆☆☆

答案

```
np.argwhere(df['salary'] > df['salary'].mean())[2]
```

113 数据计算

题目：将上一题数据的salary列开根号

难度：☆☆

答案

```
df[['salary']].apply(np.sqrt)
```

114 数据处理

题目：将上一题数据的linestaion列按_拆分

难度：☆☆

答案

```
df['split'] = df['linestaion'].str.split('_')
```

115 数据查看

题目：查看上一题数据中一共有多少列

难度：☆

答案

```
df.shape[1]
```

116 数据提取

题目：提取industryField列以'数据'开头的行

难度：☆☆

答案

```
df[df['industryField'].str.startswith('数据')]
```

117 数据计算

题目：按列制作数据透视表

难度：☆☆☆

答案

```
pd.pivot_table(df,values=["salary","score"],index="positionId")
```

118 数据计算

题目：同时对salary、score两列进行计算

难度：☆☆☆

答案

```
df[["salary","score"]].agg([np.sum,np.mean,np.min])
```

119 数据计算

题目：对不同列执行不同的计算

难度：☆☆☆

备注

对salary求平均，对score列求和

答案

```
df.agg({"salary":np.sum,"score":np.mean})
```

120 数据计算

题目：计算并提取平均薪资最高的区

难度：☆☆☆☆

答案

```
df[['district','salary']].groupby(by='district').mean().sort_values('sal
```

以上就是Pandas进阶修炼120题全部内容，如果能坚持走到这里，我想你已经掌握了处理数据的常用操作，并且在之后的数据分析中碰到相关问题，希望武装了Pandas的你能够从容的解决！我们下个专题见，拜拜～

点击给我留言