Pandas基础

1、将字典创建为DataFrame

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

df = pd.DataFrame(data)
```

1、将字典创建为DataFrame

2、提取含有字符串串"Python"的行行行

```
#方方法一一
df[df['grammer']  ==  'Python']
#方方法二二
results = df['grammer'].str.contains("Python")
results.fillna(value=False,inplace = True)
df[results]
```

3、输出df的所有列列名

```
print(df.columns)
```

4、修改第二二列列列列名为'popularity'

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

5、统计grammer列列中每种编程语言言出现的次数

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

6、将空值用用上下值的平均值填充

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

7、提取popularity列列中值大大于3的行行行

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

8、按照grammer列列进行行行去除重复值

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

9、计算popularity列列平均值

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

10、将grammer列列转换为list

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

11、将DataFrame保存为EXCEL

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

12、查看数据行行行列列数

```
df.shape
```

13、提取popularity列列值大大于3小小于7的行行行

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

14、交换两列列位置

```
# 方方法一一
temp = df['popularity']
df.drop(labels=['popularity'], axis=1,inplace = True)
df.insert(0, 'popularity', temp)
# 方方法2
cols = df.columns[[1,0]]
df = df[cols]
```

15、提取popularity列列最大大值所在行行行

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

16、查看最后5行行行数据

```
df.tail()
```

17、删除最后一一行行行数据

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

18、添加一一行行行数据['Perl',6.6]

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

19、对数据按照"popularity"列列值的大大小小进行行行排序

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

20、统计grammer列列每个字符串串的长长度

```
df['grammer'] = df['grammer'].fillna('R')
df['len_str'] = df['grammer'].map(lambda x: len(x))
```

Pandas 数 据 处 理 理

21、读取本地EXCEL数据

```
import pandas as pd
df = pd.read_excel('pandas.xlsx')
```

22、查看df数据前5行行行

```
df.head()
```

23、将salary列列数据转换为最大大值与最小小值的平均值

```
#备注，在某些版本pandas中.ix方方法可能失效，可使用用.iloc，
#为什什么不不能直接使用用max，min函数，因为我们的数据中是20k-35k这种字符串串，所以需要先用用正则表达式提取数字
import re
# 方方法一一：apply + 自自定义函数
def func(df):
    lst = df['salary'].split('-')
    smin = int(lst[0].strip('k'))
```

```
    smax = int(lst[1].strip('k'))
    df['salary'] = int((smin + smax) / 2 * 1000)
    return df

df = df.apply(func,axis=1)
# 方方法二二：iterrows + 正则
import re
for index,row in df.iterrows():
    nums = re.findall('\d+',row[2])
    df.iloc[index,2] = int(eval(f'({nums[0]} + {nums[1]}) / 2 * 1000'))
```

24、将数据根据学历进行行行分组并计算平均薪资

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

25、将createTime列列时间转换为月月-日日

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

26、查看索引、数据类型和内存信息

```
df.info()
```

27、查看数值型列列的汇总统计

```
df.describe()
```

28、新增一一列列根据salary将数据分为三组

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

29、按照salary列列对数据降序排列列

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

30、取出第33行行行数据

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

31、计算salary列列的中位数

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

32、绘制薪资水水平频率分布直方方图

```
#执行行行两次
df.salary.plot(kind='hist')
```

33、绘制薪资水水平密度曲线

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

34、删除最后一一列列categories

```
del df['categories']
# 等价于
df.drop(columns=['categories'], inplace=True)
```

35、将df的第一一列列与第二二列列合并为新的一一列列

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

36、将education列列与salary列列合并为新的一一列列

```
#备注：salary为int类型，操作与35题有所不不同
df["test1"] = df["salary"].map(str) + df['education']
```

37、计算salary最大大值与最小小值之差

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

38、将第一一行行行与最后一一行行行拼接

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

39、将第8行行行数据添加至至末尾

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

40、查看每列列的数据类型

```
df.dtypes
```

41、将createTime列列设置为索引

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

42、生生成一一个和df长长度相同的随机数dataframe

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

43、将上一一题生生成的dataframe与df合并

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

44、生生成新的一一列列new为salary列列减去之前生生成随机数列列

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

45、检查数据中是否含有任何缺失值

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

46、将salary列列类型转换为浮点数

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

47、计算salary大大于10000的次数

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

48、查看每种学历出现的次数

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

49、查看education列列共有几几种学历

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

50、提取salary与new列列的和大大于60000的最后3行行行

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

金 金 金 融 数 据 处 理 理

51、使用用绝对路路径读取本地Excel数据

```
#请将下面面的路路径替换为你存储数据的路路径
data = pd.read_excel('/Users/Desktop/600000.SH.xls')
```

52、查看数据前三行行行

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

53、查看每列列数据缺失值情况

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

54、提取日日期列列含有空值的行行行

```
data[data['日日期'].isnull()]
```

55、输出每列列缺失值具体行行行数

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

56、删除所有存在缺失值的行行行

```
'''
备注
axis：0-行行行操作（默认），1-列列操作
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
'''
data.dropna(axis=0, how='any', inplace=True)
```

57、绘制收盘价的折线图

```
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid') # 设置画图的风风格
plt.rc('font',  size=6) #设置图中字体和大大小小
plt.rc('figure', figsize=(4,3), dpi=150) # 设置图的大大小小
data['收盘价(元)'].plot()

# 等价于
import matplotlib.pyplot as plt
plt.plot(df['收盘价(元)'])
```

58、同时绘制开盘价与收盘价

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

59、绘制涨跌幅的直方方图

```
plt.hist(df['涨跌幅(%)'])
# 等价于
df['涨跌幅(%)'].hist()
```

60、让直方方图更更细致

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

61、以data的列列名创建一一个dataframe

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

62、打印所有换手手率不不是数字的行行行

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

63、打印所有换手手率为--的行行行

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

64、重置data的行行行号

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

65、删除所有换手手率为非非数字的行行行

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

66、绘制换手手率的密度曲线

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

67、计算前一一天与后一一天收盘价的差值

```
data['收盘价(元)'].diff()
```

68、计算前一一天与后一一天收盘价变化率

```
data['收盘价(元)'].pct_change()
```

69、设置日日期为索引

```
data = data.set_index('日日期')
```

70、以5个数据作为一一个数据滑动窗口口，在这个5个数据上取均值(收盘价)

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

71、以5个数据作为一一个数据滑动窗口口，计算这五个数据总和(收盘价)

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

72、将收盘价5日日均线、20日日均线与原始数据绘制在同一一个图上

```
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、按周为采样规则，取一一周收盘价最大大值

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

74、绘制重采样数据与原始数据

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

75、将数据往后移动5天

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

76、将数据向前移动5天

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

77、使用用expending函数计算开盘价的移动窗口口均值

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

78、绘制上一一题的移动均值与原始数据折线图

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

79、计算布林林指标

```
data['former 30 days rolling Close mean']=data['收盘价(元)'].rolling(20).mean()
data['upper bound']=data['former 30 days rolling Close mean']+2*data['收盘价
(元)'].rolling(20).std()
#在这里里里我们取20天内的标准差
data['lower bound']=data['former 30 days rolling Close mean']-2*data['收盘价
(元)'].rolling(20).std()
```

80、计算布林林线并绘制

```
data[['收盘价(元)', 'former 30 days rolling Close mean','upper bound','lower bound'
]].plot(figsize=(16, 6))
```

当Pandas遇上NumPy

81、导入入并查看pandas与numpy版本

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

82、从NumPy数组创建DataFrame

```
#备注 使用用numpy生生成20个0-100随机数
tem = np.random.randint(1,100,20)
df1 = pd.DataFrame(tem)
```

83、从NumPy数组创建DataFrame

```
#备注 使用用numpy生生成20个0-100固定步长长的数
tem = np.arange(0,100,5)
df2 = pd.DataFrame(tem)
```

84、从NumPy数组创建DataFrame

```
#备注 使用用numpy生生成20个指定分布(如标准正态分布)的数
tem = np.random.normal(0, 1, 20)
df3 = pd.DataFrame(tem)
```

85、将df1，df2，df3按照行行行合并为新DataFrame

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

86、将df1，df2，df3按照列列合并为新DataFrame

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

87、查看df所有数据的最小小值、25%分位数、中位数、75%分位数、最大大值

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

88、修改列列名为col1,col2,col3

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

89、提取第一一列列中不不在第二二列列出现的数字

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

90、提取第一一列列和第二二列列出现频率最高高的三个数字

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

91、提取第一一列列中可以整除5的数字位置

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

92、计算第一一列列数字前一一个与后一一个的差值

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

93、将col1,col2,clo3三列列顺序颠倒

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

94、提取第一一列列位置在1,10,15的数字

```
df['col1'].take([1,10,15])
# 等价于
df.iloc[[1,10,15],0]
```

95、查找第一一列列的局部最大大值位置

```
#备注 即比比它前一一个与后一一个数字的都大大的数字
tem = np.diff(np.sign(np.diff(df['col1'])))
np.where(tem == -2)[0] + 1
```

96、按行行行计算df的每一一行行行均值

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

97、对第二二列列计算移动平均值

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

98、将数据按照第三列列值的大大小小升序排列列

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

99、将第一一列列大大于50的数字修改为'高高'

```
df.col1[df['col1'] > 50]= '高高'
```

100、计算第二二列列与第三列列之间的欧式距离

```
np.linalg.norm(df['col2']-df['col3'])
```

一一些补充

101、从CSV文文件中读取指定数据

```
#备注 从数据1中的前10行行行中读取positionName, salary两列列
df = pd.read_csv('数据1.csv',encoding='gbk', usecols=['positionName', 'salary'],nrows = 10)
```

102、从CSV文文件中读取指定数据

```
#备注 从数据2中读取数据并在读取数据时将薪资大大于10000的为改为高高
df = pd.read_csv('数据2.csv',converters={'薪资水水平': lambda x: '高高' if float(x) > 10000 else '低'} )
```

103、从上一一题数据中，对薪资水水平列列每隔20行行行进行行行一一次抽样

```
df.iloc[::20, :][['薪资水水平']]
```

104、将数据取消使用用科学计数法

```
#输入入
df = pd.DataFrame(np.random.random(10)**10, columns=['data'])
df.round(3)
```

105、将上一一题的数据转换为百分数

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

106、查找上一一题数据中第3大大值的行行行号

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

107、反转df的行行行

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

108、按照多列列对数据进行行行合并

```
#输入入
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']})

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

109、按照多列列对数据进行行行合并

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

110.再次读取数据1并显示所有的列列

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

111、查找secondType与thirdType值相等的行行行号

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

112、查找薪资大大于平均薪资的第三个数据

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

113、将上一一题数据的salary列列开根号

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

114、将上一一题数据的linestaion列列按_拆分

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

115、查看上一一题数据中一一共有多少列列

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

116、提取industryField列列以'数据'开头的行行行

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

117、按列列制作数据透视表

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

118、同时对salary、score两列列进行行行计算

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

119、对salary求平均，对score列列求和

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

120、计算并提取平均薪资最高高的区

```
df[['district','salary']].groupby(by='district').mean().sort_values('salary',ascending=False).head(1)
```