day03 直播课

1. 面试题（新浪）

```
# 伪java,报错。
for i in range(10){

}
print(i)
```

```
for i in range(10):
    pass

print(i) # 9
```

```
def func():
    return 123

data_list = [ func   for i in range(10)   ]
print(data_list)
# A, 列表 [123,123,123,...]
# B, 列表 [fun,fun,fun,...]  正确

def func():
    return 123

data_list = [ func()   for i in range(10)   ]
print(data_list)
# A, 列表 [123,123,123,...] 正确
# B, 列表 [fun,fun,fun,...]
```

```
data_list = [ lambda:123 for i in range(10)]

print(data_list)
v1 = data_list[0]()
v2 = data_list[6]()
```

```
data_list = [ lambda:i for i in range(10)]
print(data_list)

v1 = data_list[0]()
v2 = data_list[3]()
v3 = data_list[9]()
```

更复杂的题目（自己实现）：

```
def num():
    return [lambda x: i * x for i in range(4)]

result = [m(2) for m in num()]
print(result)
```

2. 快递分拣（对比案例）

现有一部分快递的信息，如下：

```
db = [
    ['王*龙', '北京市海淀区苏州街大恒科技大厦南座4层'],
    ['庞*飞', '北京市昌平区汇德商厦四楼403'],
    ['顾*锐', '江苏省扬州市三垛镇工业集中区扬州市立华畜禽有限公司'],
    ['王*飞', '上海市徐汇区上海市徐汇区H88越虹广场B座5E'],
    ['华*升', '北京市海淀区杰睿大厦'],
    ['朱*锴', '上海市浦东新区川沙新镇华川家园33号楼503'],
    ...
]
```

请根据城市将快递进行分拣为如下格式的数据：

```
result = {
    "北京市": [
        ['王*龙', '北京市海淀区苏州街 大恒 科技 大厦 南座 4层'],
        ['庞*飞', '黑龙江省昌平区汇德商厦四楼403']
        ...
    ],
    "上海市":[
        ['王*飞', '上海市徐汇区上海市徐汇区H88越虹广场B座5E'],
        ['朱*锴', '上海市浦东新区川沙新镇华川家园33号楼503'],
        ...
    ]
}
```

- 实现思路1

```
dic = {'河南省':[],
        '北京市':[],
        '湖北省':[],
        '山东省':[],
        '上海市':[],
        '安徽省':[],
        '江苏省':[],
        '陕西省':[],
        '甘肃省':[],
        '浙江省':[],
        '辽宁省':[],
        '山西省':[],
        '海南省':[],
        '内蒙古':[]}
for k in dic:
    a = []
    for data in data_list:
        if k in data[1]:
            a.append(data)
        dic[k] = a
print(json.dumps(dic,ensure_ascii=False, indent=4))
```

- 实现思路2

```
pre_ls = [] # 存放所有省份
for i in var:
    prefix = i[1][0:3]
    if prefix not in pre_ls:
        pre_ls.append(prefix)

sum = {}

for k in var:
    for line in pre_ls:
        if k[1][0:3] == line:
            if k[1][0:3] in sum:
                sum[k[1][0:3]].append(k)
            else:
                sum[k[1][0:3]] = [k]


pprint.pprint(sum)
```

- 实现思路3

```
result = {}
for item in address_data:
    province = item[1][0:3]
    if province in result:
        result[province].append(item)
    else:
        result[province] = [item, ]

print(result)
```

- 实现思路4

```
special_province_dict = {
    "内蒙古": "内蒙古自治区",
    "广西壮": "广西壮族自治区",
    "西藏自": "西藏自治区",
    "宁夏回": "宁夏回族自治区",
    "新疆维": "新疆维吾尔自治区",
    "香港特": "香港特别行政区",
```

```
    "澳门特": "澳门特别行政区",
    "黑龙江": "黑龙江省",
}

result = {}
for item in db:
    # 获取前3个字符省份
    province = item[1][0:3]
    # 去检查是否是特殊省份（用一个字典去判断是否特殊省份，
startswith来判断和处理？）
    # 字典 > startswith
    special = special_province_dict.get(province)
    if special:
        province = special
    if province in result:
        result[province].append(item)
    else:
        result[province] = [item, ]

print(result)
```

扩展题

有如下的一个数据的结构，请将数据构造成为特定的格式。

```
发生的咖啡机破爱乌尔排放量科技实质性的魄力附近嫂子真好看，越看越好看。
真棒呀
    帮你妹
        说的太对了
    次品
        滚蛋
            你是金角大王八
真是垃圾
    说的太对了
    哈哈哈哈
湿了湿了
```

```
db = [
    {'id': 1, 'name': 'alex', 'text': '真棒呀', 'parent': None,
'child': []},
    {'id': 2, 'name': '李杰', 'text': '真是垃圾', 'parent': None,
'child': []},
    {'id': 3, 'name': '大飞机', 'text': '湿了湿了', 'parent':
None, 'child': []},
    {'id': 4, 'name': '朝阳群众', 'text': '帮你妹', 'parent': 1,
'child': []},
    {'id': 5, 'name': '海淀网友', 'text': '次品', 'parent': 1,
'child': []},
    {'id': 6, 'name': 'alex', 'text': '滚蛋', 'parent': 5,
'child': []},
    {'id': 7, 'name': '铁锤', 'text': '说的太对了', 'parent': 2,
'child': []},
    {'id': 8, 'name': '钢弹', 'text': '说的太对了', 'parent': 4,
'child': []},
    {'id': 9, 'name': '二狗', 'text': '哈哈哈哈', 'parent': 2,
'child': []},
    {'id': 10, 'name': '二狗', 'text': '你是金角大王八', 'parent':
6, 'child': []},
    ...
]
```

```
result = [
```

```
{
  "id": 1,
  "name": "alex",
  "text": "真棒呀",
  "parent": null,
  "child": [
    {
      "id": 4,
      "name": "朝阳群众",
      "text": "帮你妹",
      "parent": 1,
      "child": [
        {
          "id": 8,
          "name": "钢弹",
          "text": "说的太对了",
          "parent": 4,
          "child": []
        }
      ]
    },
    {
      "id": 5,
      "name": "海淀网友",
      "text": "次品",
      "parent": 1,
      "child": [
        {
          "id": 6,
          "name": "alex",
          "text": "滚蛋",
          "parent": 5,
          "child": [
            {
              "id": 10,
              "name": "二狗",
              "text": "你是金角大王八",
              "parent": 6,
              "child": []
```

```
                }
              ]
            }
          ]
        }
      ]
    },
    {
      "id": 2,
      "name": "李杰",
      "text": "真是垃圾",
      "parent": null,
      "child": [
        {
          "id": 7,
          "name": "铁锤",
          "text": "说的太对了",
          "parent": 2,
          "child": []
        },
        {
          "id": 9,
          "name": "二狗",
          "text": "哈哈哈哈",
          "parent": 2,
          "child": []
        }
      ]
    },
    {
      "id": 3,
      "name": "大飞机",
      "text": "湿了湿了",
      "parent": null,
      "child": []
    }
  ]
```

参考答案：

```
db_dict = {}
for item in db:
    key = item["id"]
    db_dict[key] = item

result = []
for item in db:
    pid = item['parent']
    if not pid:
        result.append(item)
    else:
        db_dict[pid]['child'].append(item)
```

3. 股票查询（对比案例）

开发程序对 stock_data.txt 进行以下操作：

1. 程序启动后，给用户提供查询接口，允许用户重复查股票行情信息(用到循环)
2. 允许用户通过模糊查询股票名，比如输入“啤酒”, 就把所有股票名称中包含“啤酒”的信息打印出来
3. 允许按 `当前价`、`涨跌幅`、`换手率` 这几列来筛选信息，比如输入“当前价>50”则把价格大于50的股票都打印，输入“市盈率<50“，则把市盈率小于50的股票都打印，不用判断等于。

思路提示：加载文件内容到内存，转成dict or list结构，然后对dict or list 进行查询等操作。 这样以后就不用每查一次就要打开一次文件了，效率会高。

程序启动后执行效果参考：

```
股票查询接口>>:换手率>25
['序号', '代码', '名称', '最新价', '涨跌幅', '涨跌额', '成交量(手)',
'成交额', '振幅', '最高', '最低', '今开', '昨收', '量比', '换手率',
'市盈率', '市净率']
```

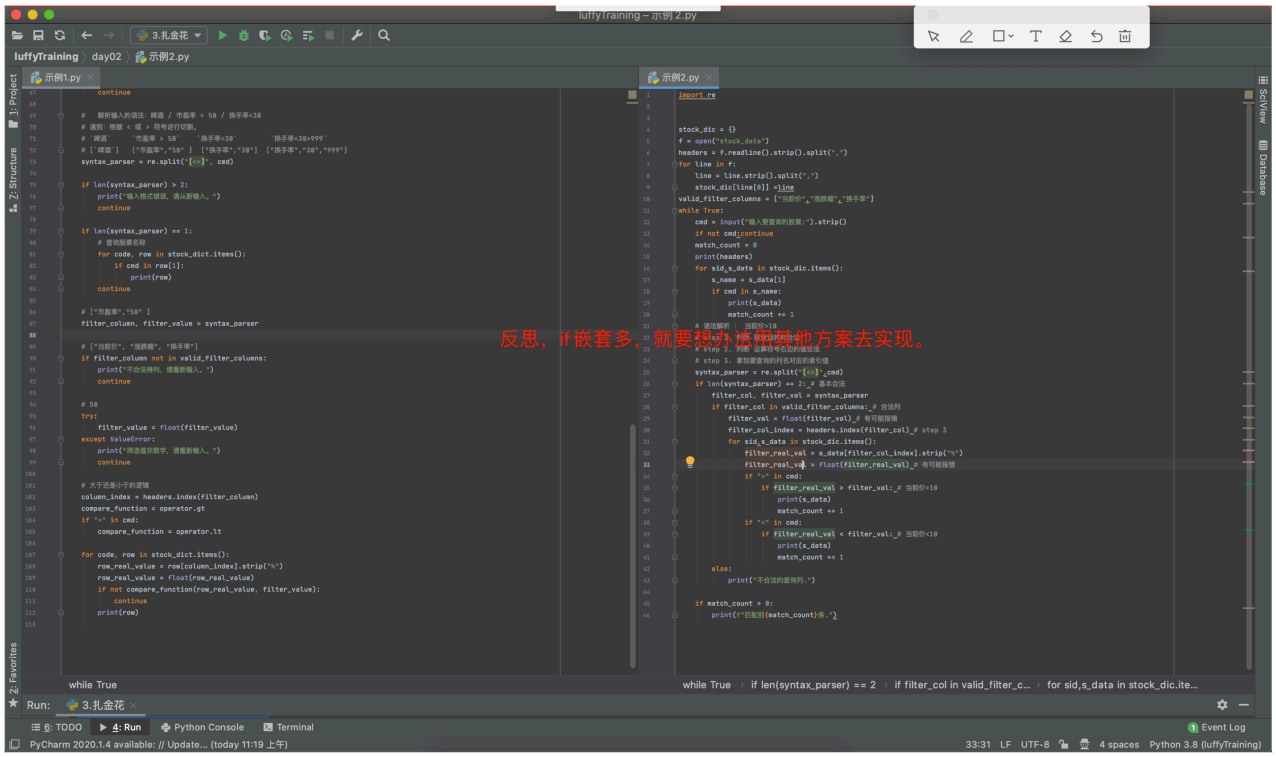
```
['18', '603697', '有友食品', '22.73', '10.02%', '2.07', '34.93
万', '7.68亿', '8.23%', '22.73', '21.03', '21.17', '20.66',
'1.4', '43.94%', '38.1', '4.66']
['23', '603956', '威派格', '22.52', '10.01%', '2.05', '18.33万',
'4.01亿', '10.60%', '22.52', '20.35', '20.35', '20.47', '2.16',
'43.02%', '-', '9.82']
['36', '300748', '金力永磁', '59.7', '10.01%', '5.43', '11.02万',
'6.38亿', '6.98%', '59.7', '55.91', '56.88', '54.27', '0.9',
'26.49%', '234.09', '23.54']
['37', '300767', '震安科技', '41.13', '10.00%', '3.74', '6.22万',
'2.49亿', '10.32%', '41.13', '37.27', '37.48', '37.39', '3.86',
'31.11%', '43.32', '3.68']
['38', '603045', '福达合金', '32', '10.00%', '2.91', '17.06万',
'5.31亿', '9.87%', '32', '29.13', '29.13', '29.09', '1.39',
'25.17%', '52.74', '4.02']
['39', '2952', '亚世光电', '58.98', '10.00%', '5.36', '4.18万',
'2.41亿', '7.42%', '58.98', '55', '55.91', '53.62', '3.04',
'27.44%', '53.09', '5.51']
找到6条
股票查询接口>>:最新价<5
['序号', '代码', '名称', '最新价', '涨跌幅', '涨跌额', '成交量(手)',
'成交额', '振幅', '最高', '最低', '今开', '昨收', '量比', '换手率',
'市盈率', '市净率']
['2', '2676', '顺威股份', '3.69', '10.15%', '0.34', '15.23万',
'5516万', '9.55%', '3.69', '3.37', '3.37', '3.35', '1.16',
'2.11%', '-', '2.58']
['3', '601619', '嘉泽新能', '4.91', '10.09%', '0.45', '16.55万',
'8006万', '8.52%', '4.91', '4.53', '4.54', '4.46', '1.82',
'3.28%', '52.26', '3.64']
找到2条
股票查询接口>>:食品
['18', '603697', '有友食品', '22.73', '10.02%', '2.07', '34.93
万', '7.68亿', '8.23%', '22.73', '21.03', '21.17', '20.66',
'1.4', '43.94%', '38.1', '4.66']
找到1条
股票查询接口>>:能源
['9', '2828', '贝肯能源', '14.25', '10.04%', '1.3', '17.83万',
'2.52亿', '4.71%', '14.25', '13.64', '13.8', '12.95', '3.45',
'18.03%', '-', '3.08']
```

找到1条

股票查询接口>>:科技

```
['12', '2866', '传艺科技', '13.81', '10.04%', '1.26', '13.59万', '1.83亿', '9.72%', '13.81', '12.59', '12.61', '12.55', '2.63', '16.86%', '33.37', '3.43']
['19', '300777', '中简科技', '24.92', '10.02%', '2.27', '5952', '1483万', '0.00%', '24.92', '24.92', '24.92', '22.65', '3.45', '1.49%', '102.24', '11.49']
['21', '300245', '天玑科技', '11.53', '10.02%', '1.05', '26.86万', '3.05亿', '9.64%', '11.53', '10.52', '10.52', '10.48', '1.06', '10.35%', '127.47', '2.57']
['26', '300391', '康跃科技', '7.8', '10.01%', '0.71', '3.9万', '3027万', '10.01%', '7.8', '7.09', '7.09', '7.09', '0.75', '1.94%', '27.35', '1.89']
['37', '300767', '震安科技', '41.13', '10.00%', '3.74', '6.22万', '2.49亿', '10.32%', '41.13', '37.27', '37.48', '37.39', '3.86', '31.11%', '43.32', '3.68']
['40', '603327', '福蓉科技', '21.56', '10.00%', '1.96', '3586', '773.1万', '0.00%', '21.56', '21.56', '21.56', '19.6', '2.81', '0.70%', '31.97', '8.05']
```

找到6条



实现思路

- 第一步：读取文件内容到内存中，字典。
- 第二步：用户输入搜索条件
  - 关键字搜索
  - 范围搜索

示例1：

```
import re
import operator

# ################### 1.读取文件到字典 ###################
# 保存所有的股票信息
stock_dict = {}

# 以后做范围查询，只能查这几项。
valid_filter_columns = ["当前价", "涨跌幅", "换手率"]

# 打开文件，文件内容读取到stock_dict中
with open("stock_data.txt") as file_object:
    # 股票代码,股票名称,当前价,涨跌额,涨跌幅,年初至今,成交量,成交额,换手率,市盈率(TTM),股息率,市值
    # 去读取一行，剔除两边空白
    # ["股票代码","股票名称","当前价"...]
    headers = file_object.readline().strip().split(",")

    for line in file_object:
        line = line.strip().split(",")
        code = line[0]
        stock_dict[code] = line

# ################### 2.用户输入内容然后搜索 ###################
# 关键字：啤酒
# 范围：换手率>9
while True:
```

```python
    # "集团"
    # 涨跌额<9
    cmd = input("请输入要查询的股票: ").strip()
    # 如果是空，则不再继续往下执行，而是重新输入。
    if not cmd:
        continue

    # 对用户的输入进行解析:  关键字 or 范围
    syntax_parser = re.split("[<>]", cmd)

    # ["涨跌额",3,"asdf"]
    if len(syntax_parser) > 2:
        print("输入格式错误，请从新输入。")
        continue

    # 2.1 关键字处理
    if len(syntax_parser) == 1:
        # 查询股票名称
        for code, row in stock_dict.items():
            if cmd in row[1]:
                print(row)
        continue

    # 2.2 处理返回
    # ["市盈率","50" ]
    filter_column, filter_value = syntax_parser

    # ["当前价", "涨跌幅", "换手率"]
    if filter_column not in valid_filter_columns:
        print("不合法得列，请重新输入。")
        continue
    # 50
    try:
        filter_value = float(filter_value)
    except ValueError:
        print("筛选值非数字，请重新输入。")
        continue

    # 大于还是小于的逻辑
```

```
    column_index = headers.index(filter_column)
    compare_function = operator.gt
    if "<" in cmd:
        compare_function = operator.lt

    for code, row in stock_dict.items():
        row_real_value = row[column_index].strip("%")
        row_real_value = float(row_real_value)
        if not compare_function(row_real_value, filter_value):
            continue
        print(row)
```

示例2：

```
import re

# ################### 1.读取文件到字典 ###################
stock_dic = {}
f = open("stock_data")
headers = f.readline().strip().split(",")
for line in f:
    line = line.strip().split(",")
    stock_dic[line[0]] = line
f.close()

valid_filter_columns = ["当前价", "涨跌幅", "换手率"]

# ################### 2.用户输入内容然后搜索 ###################
# 关键字：啤酒
# 范围：换手率>9
while True:
    cmd = input("输入要查询的股票:").strip()
    if not cmd:
        continue
    match_count = 0
    print(headers)
    # 2.1 查询关键字
    for sid, s_data in stock_dic.items():
        s_name = s_data[1]
```

```
        if cmd in s_name:
            print(s_data)
            match_count += 1

    # 语法解析 ：  当前价>10
    # step 1. 判断 要查询的列合法
    # step 2. 判断 运算符号右边的值合法
    # step 3. 拿到要查询的列名对应的索引值
    # 2.2 处理范围
    syntax_parser = re.split("[<>]", cmd)

    # 涨跌额>3
    # ["涨跌额",3]
    if len(syntax_parser) == 2:  # 基本合法
        filter_col, filter_val = syntax_parser
        if filter_col in valid_filter_columns:  # 合法列
            filter_val = float(filter_val)  # 有可能报错
            filter_col_index = headers.index(filter_col)  #
step 3
            for sid, s_data in stock_dic.items():
                filter_real_val =
s_data[filter_col_index].strip("%")
                filter_real_val = float(filter_real_val)  # 有可
能报错
                if ">" in cmd:
                    if filter_real_val > filter_val:  # 当前价
>10
                        print(s_data)
                        match_count += 1
                if "<" in cmd:
                    if filter_real_val < filter_val:  # 当前价
<10
                        print(s_data)
                        match_count += 1
        else:
            print("不合法的查询列.")

    if match_count > 0:
        print(f"匹配到{match_count}条.")
```

4.下载优酷VIP视频

一个好玩的小脚本，见代码。