数据筛选和数据匹配的介绍

在今天的课程当中，我们重点通过两个案例，学习了如何解决Excel常见场景中的数据筛选问题和数据匹配问题

在这个过程中，会接触了两个新的名词——数据筛选和数据匹配

数据筛选要求在表中筛选出符合条件的数据。

数据匹配需要在多个表之间匹配相关的数据。

抽象一点来说，数据筛选就是从已有的大量数据中通过设置条件把符合条件的数据剥离出来，而数据匹配则是在多个独立但是又具有相关性的数据中把这些具有相关性的数据匹配出来。

它们俩的功能就有点像excel当中的if函数和vlookup函数，if函数可以对数值和公式进行条件检测，而vlookup函数可以用来核对数据，实现在多个表格之间快速导入数据的功能。

现在还是以课程中的两个案例代码来理解这两个概念吧～

首先是数据筛选的逻辑代码:

```
from openpyxl import load_workbook

# 设置工作簿路径
path = '工作簿路径'
# 打开工作簿
wb = load_workbook(path)
# 获取活动工作表
ws = wb.active

# 按行获取数据
for row in ws.iter_rows(min_row=2, values_only=True):
    # 获取数据a
    a = row[索引1]
    # 获取数据b
    b = row[索引2]

    # 构造筛选条件，如a满足条件1并且b满足条件2
    if a == 条件1 and b == 条件2
        # 筛选后执行的代码
```

要想进行数据筛选首先得获取到数据，所以在4-8行，我们通过load_workbook()方法打开一个承载着目标数据的工作簿。

打开工作表之后，在11行用iter_rows()方法逐行获取工作表当中的数据，然后在13–15行通过索引把获取到的数据赋值给变量a和b。

然后便来到了最关键的一步，使用数据。这一步实际上也就是数据筛选的核心动作，设置合适的筛选条件对获取到的数据进行筛选。

```
    # 构造筛选条件，如a满足条件1并且b满足条件2
    if a == 条件1 and b == 条件2
        # 筛选后执行的代码
```

既然要设置条件，那自然会想到用条件判断语句，具体的条件可以运用之前所学的python基础知识，借助比较运算符、成员运算符和逻辑运算符等设置出符合要求的条件来对数据进行筛选～

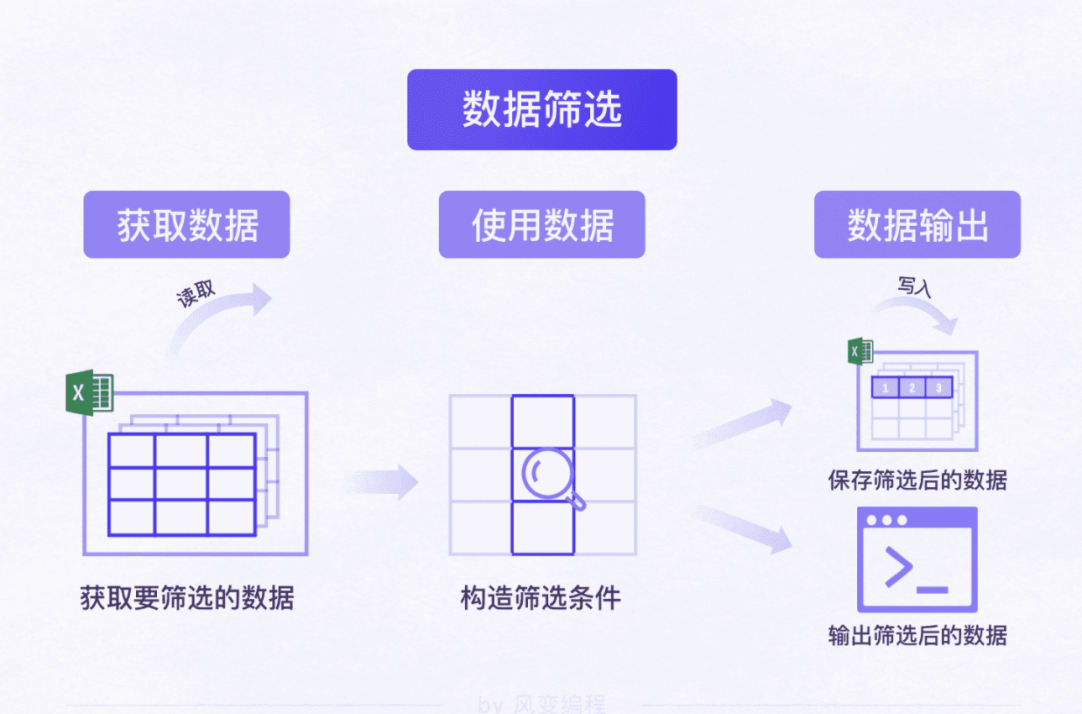
构造筛选条件的常见Python基础语法

Python基础语法	筛选条件	示例
条件判断	条件	If 条件
比较运算	条件：A等于B	If A == B
	条件：A不等于B	If A != B
	条件：A大于B	If A > B
	条件：A小于B	If A < B
	条件：A大于等于B	If A >= B
	条件：A小于等于B	If A <= B
成员运算	条件：A在指定的序列B中	If A in B
	条件：A不在指定的序列B中	If A not in B
逻辑运算	条件1与条件2同时成立	If 条件1 and 条件2
	条件1成立或条件2成立	If 条件1 or 条件2
	不满足条件1	If not 条件1

by 风变编程

也就是说，筛选的条件是开放性的，只要你能用所学的知识构造出符合你要求的条件，那自然就可以从大量数据中筛选出符合你所设置的条件的数据

整个数据筛选的逻辑过程就可以看做是这样:



然后再来看看数据匹配的逻辑代码:

```
from openpyxl import load_workbook

# a为工作簿1
a_wb = load_workbook('工作簿路径1')
a_ws = a_wb.active

# b为工作簿2
b_wb =load_workbook('工作簿路径2')
b_ws = b_wb.active
info_dict = {}

# 取出工作簿1中的匹配源数据
for a_row in a_ws.iter_rows(min_row=2, values_only=True):
    # 将横坐标1的值作为字典的键
    a_key = a_row[索引1]
    # 将横坐标2的值作为字典的值
    a_value = a_row[索引2]
    # 键值对写入字典info_dict
    info_dict[a_key] = a_value

# 取出工作簿2中的待匹配数据，并进行匹配
for b_row in b_ws.iter_rows(min_row=1, values_only=True):
    # 将横坐标3的值作为字典的键
    b_key = b_row[索引3]
    # 数据匹配，看待匹配数据b_key，是否在工作簿1的源数据中
    info_dict[b_key] 或 info_dict.get(b_key)
```

数据匹配比数据筛选要稍微复杂一点，因为需要针对多个不同的数据源进行处理

以上面的代码为例，在3-10行我们分别打开两个不同的但是又具有一定关联的工作簿，并提前创建好一个空字典备用

```
# a为工作簿1
a_wb = load_workbook('工作簿路径1')
a_ws = a_wb.active

# b为工作簿2
b_wb =load_workbook('工作簿路径2')
b_ws = b_wb.active
info_dict = {}
```

这里为什么要选择字典呢？ 这是因为我们做数据匹配的时候，用到的都是具有一定关联性的数据源，所谓的一定关联性就是说它们有一些相同的地方，但是也有很多不同的地方。

它们之间通过某项数据作为桥梁，构成一定的关联性，而这个桥梁数据一定是相同的，这样我们自然就想到了字典的键，它在字典中也是唯一不可重复的，但是一个键却可以对应多个不同的值。

在上面的代码中，12-19行先获取工作簿1当中的源数据,并把桥梁数据a_key作为字典的键，把需要用来做匹配核对的数据a_value作为字典的值

```
# 取出工作簿1中的匹配源数据
for a_row in a_ws.iter_rows(min_row=2, values_only=True):
    # 将横坐标1的值作为字典的键
    a_key = a_row[索引1]
    # 将横坐标2的值作为字典的值
    a_value = a_row[索引2]
    # 键值对写入字典info_dict
    info_dict[a_key] = a_value
```

之后再打开另一个工作簿2，取出桥梁数据b_key，并判断b_key是否在工作簿1当中存在，如果存在的话info_dict[b_key]或info_dict.get(b_key)不会报错，反之则说明不存在

```
# 取出工作簿2中的待匹配数据，并进行匹配
for b_row in b_ws.iter_rows(min_row=1, values_only=True):
    # 将横坐标3的值作为字典的键
    b_key = b_row[索引3]
    # 数据匹配，看待匹配数据b_key，是否在工作簿1的源数据中
    info_dict[b_key] 或 info_dict.get(b_key)
```

需要注意的是，匹配的最终目的是将不同表格中需要的数据关联起来。上面的代码中，只是匹配一个数据，但是很多时候我们需要将一组的多个数据进行匹配。
比如课程中具体的代码，其实是先匹配键再比较值：

```
# 源数据字典，键staff_id和值staff_late
info_dict[staff_id] = staff_late

# 待匹配数据，键member_id和值member_late
for monthly_row in monthly_ws.iter_rows(min_row=3, max_col=13,
values_only=True):
        member_id = monthly_row[0]
        member_late = monthly_row[-1]
        # 数据匹配
        if member_late != info_dict[member_id]:
            print('工号{}迟到情况不匹配，请核查后更新'.format(member_id))
```

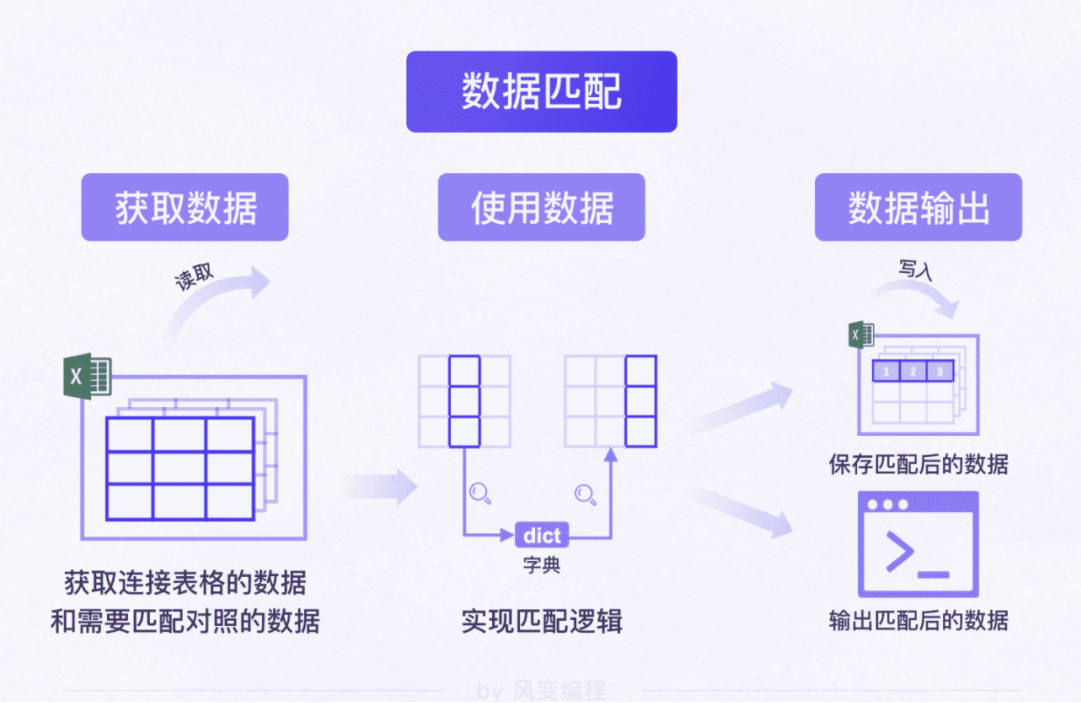
2.值与值相等，说明值能够相互匹配

1.能取到值，说明键能够相互匹配

这时候，桥梁数据的作用就突显出来了，对吧?

当桥梁数据匹配上了之后，我们就可以继续比较，两个桥梁数据（键）后面对应的值是否也能匹配上。也就是说：如果将匹配数据也视为一对键值对，就可以关联多个需要的数据进行匹配。

那么，数据匹配的逻辑过程是这样的:



有得同学可能会说那用excle的if函数和vlookup函数也能实现这个，干嘛要特地学python写代码去做呢?

首先要明确一点，用python肯定比用excel更好，因为你用excel能实现的操作用python都可以做到，excel做不到的python也可以做到，甚至于后面要学习的数据分析，也都可以用python完成，python比excel更加简单高效

最后再说一点，学习代码切忌纸上谈兵，大家一定要多思考，然后多实操，实实在在的去运行代码并观察，这样才能更好的理解和使用它~