df.infer_objects()

基于数据特征进行自动转换
0.21版以后新增

In [ ]:

```
df = pd.DataFrame({"A": ["a", 1, 2, 3]})
df = df.iloc[1:]
df.dtypes
```

In [ ]:

```
df.infer_objects().dtypes
```

3.6 实战：对PM2.5数据做简单清理

要求:

在数据中删除对后续分析无用的Parameter、Duration、QC Name等变量列
尝试对Date (LST)、Value等变量进行重命名
尝试对数据做各种类型的转换

4 胖哒黑魔法：索引

索引的用途:

用于在分析、可视化、数据展示、数据操作中标记数据行
提供数据汇总、合并、筛选时的关键依据
提供数据重构时的关键依据

注意事项:

索引是不可直接修改的，只能增、删、替换
逻辑上索引不应当出现重复值，Pandas对这种情况不会报错，但显然有潜在风险

4.1 建立索引

4.1.1 新建数据框时建立索引

所有的数据框默认都已经拥有流水号格式的索引，因此这里的“建立”索引指的是自定义索引

In [ ]:

```
df1 = pd.DataFrame(
{'var1' : 1.0, 'var2' : [1,2,3,4],
 'var3' :["test","train","test","train"], 'var4' : 'cons'},
 index = [0, 1, 2, 5]
)

df1
```

4.1.2 读入数据时建立索引

数据列直接提供索引值，因此指明相应的数据列即可

Q：如果没有现成的变量列，需要直接提供索引值，该如何操作？

In [ ]:

```
df2 = pd.read_csv("univ.csv", encoding ="gbk", index_col="学校名称" )
df2
```

In [ ]:

```
# 生成复合索引
df2 = pd.read_csv("univ.csv", encoding ="gbk",
                  index_col=["类型", "学校名称"] )
df2
```

4.1.3 指定某列为索引列

df.set_index(

keys : 被指定为索引的列名，复合索引用list格式提供
drop = True : 建立索引后是否删除该列
append = False : 是否在原索引基础上添加索引，默认是直接替换原索引
inplace = False : 是否直接修改原数据框

)

In [ ]:

```
df_new = df2.set_index(['名次','学校名称'], drop = False)
df_new
```

In [ ]:

```
# 生成复合索引
df_new = df2.set_index('名次', append=True, drop=False)# 不删除变量列
df_new
```

4.2 将索引还原回变量列

df.reset_index(

```
    drop = False : 是否将索引直接删除，而不是还原为变量列
    inplace = False : 是否直接修改原数据框
    level = None : 对于多重索引，确定转换哪个级别为变量
)
```

In [ ]:

```
df_new2 = df2.copy() # 真正生成副本，而不是指定另一个别名
df_new2.set_index(['名次','类型','所在省份'], inplace = True)
df_new2
```

In [ ]:

```
df_new2.reset_index(inplace = True) # 将索引全部还原为变量
df2
```

In [ ]:

```
df_new2.reset_index(level = ['类型']) # 筛选其中一个进行还原
df_new2
```

In [ ]:

```
df_new2.reset_index(level = ['名次','所在省份']) # 保留一个作为索引
```

4.3 引用和修改索引

4.3.1 引用索引

注意：索引仍然是有存储格式的，注意区分数值型和字符型的引用方式

In [ ]:

```
df1.index
```

In [ ]:

```
df2.index
```

In [ ]:

```
df_new2.index
```

4.3.2 修改索引

4.3.2.1 修改索引名

本质上和变量列名的修改方式相同

In [ ]:

```
df2.index.names # 为什么是复数形式的names？
```

In [ ]:

```
df2.index.names = ['idx']
df2.index.names
```

In [ ]:

```
df_new2.index.names = [None, None, None] # None代表无名称
df_new2
```

4.3.2.2 修改索引值

这里的修改本质上是全部替换

In [ ]:

```
df1.index[3] = 6 # 此处无法直接赋值
```

In [ ]:

```
df1.index = ['a', 'b', 'c', 6]
df1.index
```

4.3.3 强行更新索引

reindex则可以使用数据框中不存在的数值建立索引，并据此扩充新索引值对应的索引行/列，同时进行缺失值填充操作

df.reindex(

```
    labels : 类数组结构的数值，将按此数值重建索引，非必需
    axis : 针对哪个轴进行重建
        ('index', 'columns') or number (0, 1).
    copy = True : 建立新对象而不是直接更改原df/series
    level : 考虑被重建的索引级别

    缺失数据的处理方式
        method : 针对已经排序过的索引，确定数据单元格无数据时的填充方法，非必需
            pad / ffill: 用前面的有效数值填充
            backfill / bfill: 用后面的有效数值填充
            nearest: 使用最接近的数值进行填充
        fill_value = np.NaN : 将缺失值用什么数值替代
        limit = None : 向前/向后填充时的最大步长
    )
```

In [ ]:

```
df2.set_index('名次')
```

In [ ]:

```
df2.reindex()
```

In [ ]:

```
df2.reindex([1,2,99,101])
```

In [ ]:

```
df2.reindex([1,2,99,101], method = 'ffill' )
```

In [ ]:

```
df2.reindex([1,2,99,101], fill_value= "不知道" )
```

In [ ]:

```
df2.reindex([1,2,99,101], fill_value= "不知道" ).dtypes
```

4.4 实战：为PM2.5数据建立索引

要求:

尝试在读入文件时直接建立索引
尝试使用Date (LST)建立单一索引
    提示：为便于操作，最好先重命名
尝试使用Year,Month,Day,Hour建立复合索引
尝试修改索引名

5 案例行的基本操作

5.1 案例排序

5.1.1 用索引排序

df.sort_index(

level ： （多重索引时）指定用于排序的级别顺序号/名称
ascending = True ： 是否为升序排列，多列时以表形式提供
inplace = False ：
na_position = 'last': 缺失值的排列顺序，first/last

)

In [ ]:

```
df2 = pd.read_excel("高校信息.xlsx", sheet_name = 0)
df2.set_index(['类型','学校名称'], inplace = True)
df2
```

Q：课程学习遇到不懂怎么办？

- ✧ 本课程提供额外福利：QQ 群供学员交流心得，群号：630030855，可直接扫下方的二维码进入。
- ✧ 老师有空时也会参与讨论，但请不要把你的工作问题直接让老师解决。
- ✧ 老师鼓励学员多思考、多动手，通过自己努力解决问题，这样能发现乐趣、有成就感、成长快。