> 教你学会 Pandas 不是我的目的，**教你轻松玩转 Pandas 才是我的目的。**我会通过一系列实例来带入 Pandas 的知识点，让你在学习 Pandas 的路上不再枯燥。
>
> 声明：我所写的**轻松玩转 Pandas 教程都是免费的**，如果对你有帮助，你可以持续关注我。

在数据处理过程中，经常会遇到要筛选不同要求的数据，通过 Pandas 可以轻松时间，这一篇我们来看下如何使用 Pandas 来完成数据筛选吧。

In [1]:
```
# 导入相关库
import numpy as np
import pandas as pd
```

executed in 5ms, finished 10:23:56 2018-07-22

Pandas 中除了支持 Python 和 Numpy 的索引运算符[]和属性运算符.来访问数据之外，还有很多其他的方式来访问数据，我们一起来看看吧。

In [2]:
```
index = pd.Index(data=["Tom", "Bob", "Mary", "James", "Andy", "Alice"], name="name")

data = {
    "age": [18, 30, np.nan, 40, np.nan, 30],
    "city": ["Bei Jing ", "Shang Hai ", "Guang Zhou", "Shen Zhen", np.nan, " "],
    "sex": [None, "male", "female", "male", np.nan, "unknown"],
    "birth": ["2000-02-10", "1988-10-17", None, "1978-08-08", np.nan, "1988-10-17"]
}

user_info = pd.DataFrame(data=data, index=index)

# 将出生日期转为时间戳
user_info["birth"] = pd.to_datetime(user_info.birth)
user_info
```

executed in 57ms, finished 10:23:56 2018-07-22

Out[2]:

name	age	birth	city	sex
Tom	18.0	2000-02-10	Bei Jing	None
Bob	30.0	1988-10-17	Shang Hai	male
Mary	NaN	NaT	Guang Zhou	female
James	40.0	1978-08-08	Shen Zhen	male
Andy	NaN	NaT	NaN	NaN
Alice	30.0	1988-10-17		unknown

字典式 get 访问

我们都知道，Python 中的字典要获取 value 时可以通过 `get` 方法来获取，对于 Series 和 DataFrame 也一样，他们一样可以通过 `get` 方法来获取。

In [3]:
```
# 获取得到所有年龄相关的这一列的信息，结果为一个 Series
user_info.get("age")
```

executed in 13ms, finished 10:23:56 2018-07-22

Out[3]:
```
name
Tom      18.0
Bob      30.0
Mary      NaN
James    40.0
Andy      NaN
Alice    30.0
Name: age, dtype: float64
```

In [4]:
```
# 从包含所有的年龄信息的 Series 中得到 Tom 的年龄
user_info.get("age").get("Tom")
```

executed in 22ms, finished 10:23:56 2018-07-22

Out[4]:
```
18.0
```

属性访问

除了可以通过 `get` 方法来获取数据之外，还可以通过属性的方式来访问，同样完成上面的功能，来看下如何通过属性访问的方式来实现。

In [5]:
```
# 获取得到所有年龄相关的这一列的信息，结果为一个 Series
user_info.age
```

executed in 34ms, finished 10:23:56 2018-07-22

Out[5]:
```
name
Tom      18.0
Bob      30.0
Mary      NaN
James    40.0
Andy      NaN
Alice    30.0
Name: age, dtype: float64
```

In [6]:
```
# 从包含所有的年龄信息的 Series 中得到 Tom 的年龄
user_info.age.Tom
```

executed in 22ms, finished 10:23:56 2018-07-22

Out[6]: 18.0

切片操作

在学习 Python 时，会发现列表的切片操作非常地方便，Series 和 DataFrame 同样也有切片操作。

对于 Series 来说，通过切片可以完成选择指定的行，对于 Series 来说，通过切片可以完成选择指定的行或者列，来看看怎么玩吧。

In [7]:
```
# 获取年龄的前两行
user_info.age[:2]
```

executed in 21ms, finished 10:23:56 2018-07-22

Out[7]:
```
name
Tom    18.0
Bob    30.0
Name: age, dtype: float64
```

In [8]:
```
# 获取所有信息的前两行
user_info[:2]
```

executed in 29ms, finished 10:23:56 2018-07-22

Out[8]:

	age	birth	city	sex
name				
Tom	18.0	2000-02-10	Bei Jing	None
Bob	30.0	1988-10-17	Shang Hai	male

In [9]:
```
# 所有信息每两行选择一次数据
user_info[::2]
```

executed in 29ms, finished 10:23:56 2018-07-22

Out[9]:

name	age	birth	city	sex
Tom	18.0	2000-02-10	Bei Jing	None
Mary	NaN	NaT	Guang Zhou	female
Andy	NaN	NaT	NaN	NaN

In [10]:
```
# 对所有信息进行反转
user_info[::-1]
```

executed in 33ms, finished 10:23:56 2018-07-22

Out[10]:

name	age	birth	city	sex
Alice	30.0	1988-10-17		unknown
Andy	NaN	NaT	NaN	NaN
James	40.0	1978-08-08	Shen Zhen	male
Mary	NaN	NaT	Guang Zhou	female
Bob	30.0	1988-10-17	Shang Hai	male
Tom	18.0	2000-02-10	Bei Jing	None

上面都是筛选行，如何筛选 DataFrame 中的列呢?

只需要将列名传入切片即可完成筛选。

In [11]:
```
user_info["age"]
```

executed in 25ms, finished 10:23:56 2018-07-22

Out[11]:
```
name
Tom      18.0
Bob      30.0
Mary      NaN
James    40.0
Andy      NaN
Alice    30.0
Name: age, dtype: float64
```

如何筛选出多列的数据呢？只需要将对应的列名传入组成一个列表，传入切片中即可。

In [12]:
```
user_info[["city", "age"]]
```

executed in 28ms, finished 10:23:56 2018-07-22

Out[12]:

	city	age
name		
Tom	Bei Jing	18.0
Bob	Shang Hai	30.0
Mary	Guang Zhou	NaN
James	Shen Zhen	40.0
Andy	NaN	NaN
Alice		30.0

可以看到，列表中的列名的顺序会影响最后的结果。

通过数字筛选行和列

通过切片操作可以完成筛选行或者列，如何同时筛选出行和列呢？

通过 iloc 即可实现， iloc 支持传入行和列的筛选器，并用，隔开。无论是行或者里筛选器，都可以为以下几种情况：

- 一个整数，如 2
- 一个整数列表，如 [2, 1, 4]
- 一个整数切片对象，如 2:4
- 一个布尔数组
- 一个callable

先来看下前3种的用法。

In [13]:
```
# 筛选出第一行数据
user_info.iloc[0]
```
executed in 20ms, finished 10:23:56 2018-07-22

Out[13]:
```
age                       18
birth    2000-02-10 00:00:00
city                Bei Jing
sex                     None
Name: Tom, dtype: object
```

In [14]:
```
# 筛选出第二行第一列的数据
user_info.iloc[1, 0]
```
executed in 17ms, finished 10:23:56 2018-07-22

Out[14]:
```
30.0
```

In [15]:
```
# 筛选出第二行、第一行、第三行对应的第一列的数据
user_info.iloc[[1, 0, 2], 0]
```
executed in 23ms, finished 10:23:56 2018-07-22

Out[15]:
```
name
Bob     30.0
Tom     18.0
Mary     NaN
Name: age, dtype: float64
```

In [16]:
```
# 筛选出第一行至第三行以及第一列至第二列的数据
user_info.iloc[0:3, 0:2]
```

executed in 22ms, finished 10:23:56 2018-07-22

Out[16]:

name	age	birth
Tom	18.0	2000-02-10
Bob	30.0	1988-10-17
Mary	NaN	NaT

In [17]:
```
# 筛选出第一列至第二列的数据
user_info.iloc[:, 0:2]
```

executed in 25ms, finished 10:23:56 2018-07-22

Out[17]:

name	age	birth
Tom	18.0	2000-02-10
Bob	30.0	1988-10-17
Mary	NaN	NaT
James	40.0	1978-08-08
Andy	NaN	NaT
Alice	30.0	1988-10-17

通过名称筛选行和列

虽然通过 iloc 可以实现同时筛选出行和列，但是它接收的是输入，非常不直观， 通过 loc 可实现传入名称来筛选数据，loc 支持传入行和列的筛选器，并用，隔开。无论是行或者里筛选器，都可以为以下几种情况：

- 一个索引的名称，如："Tom"
- 一个索引的列表，如：["Bob", "Tom"]
- 一个标签范围，如："Tom": "Mary"
- 一个布尔数组
- 一个callable

先来看下前3种的用法。

In [18]:
```
# 筛选出名称为 Tom 的数据一行数据
user_info.loc["Tom"]
```

executed in 24ms, finished 10:23:56 2018-07-22

Out[18]:
```
age                       18
birth    2000-02-10 00:00:00
city                Bei Jing
sex                      None
Name: Tom, dtype: object
```

In [19]:
```
# 筛选出名称为 Tom 的年龄
user_info.loc["Tom", "age"]
```

executed in 16ms, finished 10:23:56 2018-07-22

Out[19]:
```
18.0
```

In [20]:
```
# 筛选出名称在 ["Bob", "Tom"] 中的两行数据
user_info.loc[["Bob", "Tom"]]
```

executed in 29ms, finished 10:23:56 2018-07-22

Out[20]:

name	age	birth	city	sex
Bob	30.0	1988-10-17	Shang Hai	male
Tom	18.0	2000-02-10	Bei Jing	None

In [21]:
```
# 筛选出索引名称在 Tom 到 Mary 之间的数据
user_info.loc["Tom": "Mary"]
```

executed in 27ms, finished 10:23:56 2018-07-22

Out[21]:

name	age	birth	city	sex
Tom	18.0	2000-02-10	Bei Jing	None
Bob	30.0	1988-10-17	Shang Hai	male
Mary	NaN	NaT	Guang Zhou	female

In [22]:
```
# 筛选出年龄这一列数据
user_info.loc[:, ["age"]]
```

executed in 25ms, finished 10:23:56 2018-07-22

Out[22]:

name	age
Tom	18.0
Bob	30.0
Mary	NaN
James	40.0
Andy	NaN
Alice	30.0

In [23]:
```
# 筛选出所有 age 到 birth 之间的这几列数据
user_info.loc[:, "age": "birth"]
```

executed in 27ms, finished 10:23:56 2018-07-22

Out[23]:

name	age	birth
Tom	18.0	2000-02-10
Bob	30.0	1988-10-17
Mary	NaN	NaT
James	40.0	1978-08-08
Andy	NaN	NaT
Alice	30.0	1988-10-17

你可能已经发现了，通过名称来筛选时，传入的切片是左右都包含的。

布尔索引

通过布尔操作我们一样可以进行筛选操作，布尔操作时，& 对应 and，| 对应 or，~ 对应 not。

当有多个布尔表达式时，需要通过小括号来进行分组。

In [24]:
```
user_info[user_info.age > 20]
```

executed in 31ms, finished 10:23:56 2018-07-22

Out[24]:

name	age	birth	city	sex
Bob	30.0	1988-10-17	Shang Hai	male
James	40.0	1978-08-08	Shen Zhen	male
Alice	30.0	1988-10-17		unknown

In [25]:
```
# 筛选出年龄在20岁以上，并且性别为男性的数据
user_info[(user_info.age > 20) & (user_info.sex == "male")]
```

executed in 30ms, finished 10:23:56 2018-07-22

Out[25]:

name	age	birth	city	sex
Bob	30.0	1988-10-17	Shang Hai	male
James	40.0	1978-08-08	Shen Zhen	male

In [26]:
```
# 筛选出性别不为 unknown 的数据
user_info[~(user_info.sex == "unknown")]
```

executed in 29ms, finished 10:23:56 2018-07-22

Out[26]:

name	age	birth	city	sex
Tom	18.0	2000-02-10	Bei Jing	None
Bob	30.0	1988-10-17	Shang Hai	male
Mary	NaN	NaT	Guang Zhou	female
James	40.0	1978-08-08	Shen Zhen	male
Andy	NaN	NaT	NaN	NaN

除了切片操作可以实现之外， loc 一样可以实现。

In [27]:
```
user_info.loc[user_info.age > 20, ["age"]]
```

executed in 22ms, finished 10:23:56 2018-07-22

Out[27]:

name	age
Bob	30.0
James	40.0
Alice	30.0

isin 筛选

Series 包含了 isin 方法，它能够返回一个布尔向量，用于筛选数据。

In [28]:
```
# 筛选出性别属于 male 和 female的数据
user_info[user_info.sex.isin(["male", "female"])]
```

executed in 28ms, finished 10:23:56 2018-07-22

Out[28]:

name	age	birth	city	sex
Bob	30.0	1988-10-17	Shang Hai	male
Mary	NaN	NaT	Guang Zhou	female
James	40.0	1978-08-08	Shen Zhen	male

对于索引来说，一样可以使用 isin 方法来筛选。

In [29]:
```
user_info[user_info.index.isin(["Bob"])]
```

executed in 26ms, finished 10:23:57 2018-07-22

Out[29]:

name	age	birth	city	sex
Bob	30.0	1988-10-17	Shang Hai	male

通过Callable筛选

loc、iloc、切片操作都支持接收一个 callable 函数，callable必须是带有一个参数（调用Series，DataFrame）的函数，并且返回用于索引的有效输出。

In [30]:
```
user_info[lambda df: df["age"] > 20]
```

executed in 25ms, finished 10:23:57 2018-07-22

Out[30]:

name	age	birth	city	sex
Bob	30.0	1988-10-17	Shang Hai	male
James	40.0	1978-08-08	Shen Zhen	male
Alice	30.0	1988-10-17		unknown

In [31]:
```
user_info.loc[lambda df: df.age > 20, lambda df: ["age"]]
```

executed in 22ms, finished 10:23:57 2018-07-22

Out[31]:

name	age
Bob	30.0
James	40.0
Alice	30.0

In [32]:
```
user_info.iloc[lambda df: [0, 5], lambda df: [0]]
```

executed in 22ms, finished 10:23:57 2018-07-22

Out[32]:

	age
name	
Tom	18.0
Alice	30.0

想要学习更多关于人工智能的知识，请关注公众号：**AI派**

这里我将整篇文章的内容整理成了pdf，想要pdf文件的可以在公众号后台回复关键字：**pandas**。