Knowledge Discovery and Data Mining

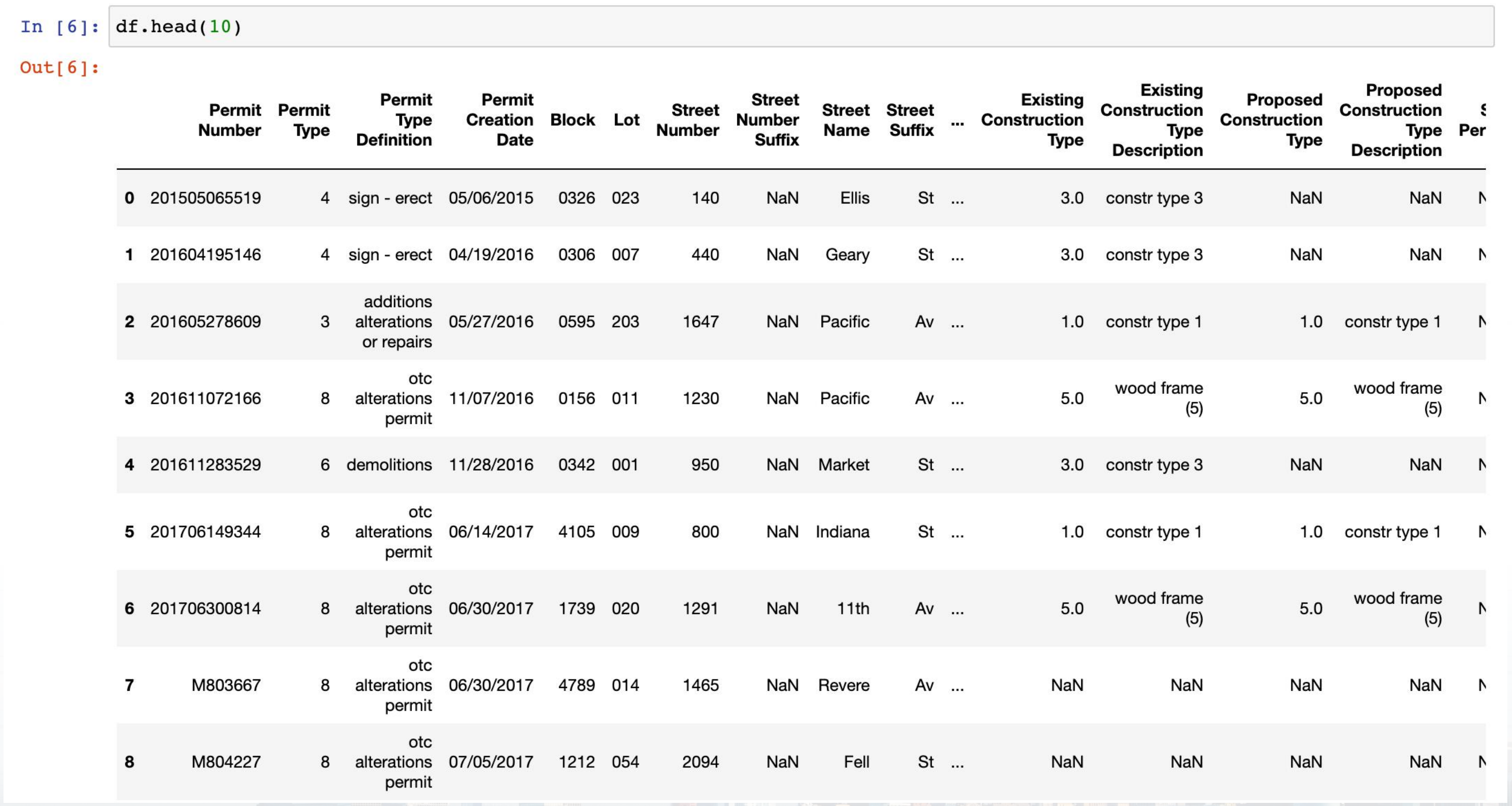
Lab 4 Data Cleaning II Dates, Encoding Types and Remove Duplications

Tianyue Zheng
zhengty@sustech.edu.cn

Topics

1.Tips for using pandas
2.Play with Datetime type in pandas Dataframe
3. Understand different kinds of character encodings
4. Remove duplicate records

Data



```
In [6]: df.head(10)
```

Out[6]:

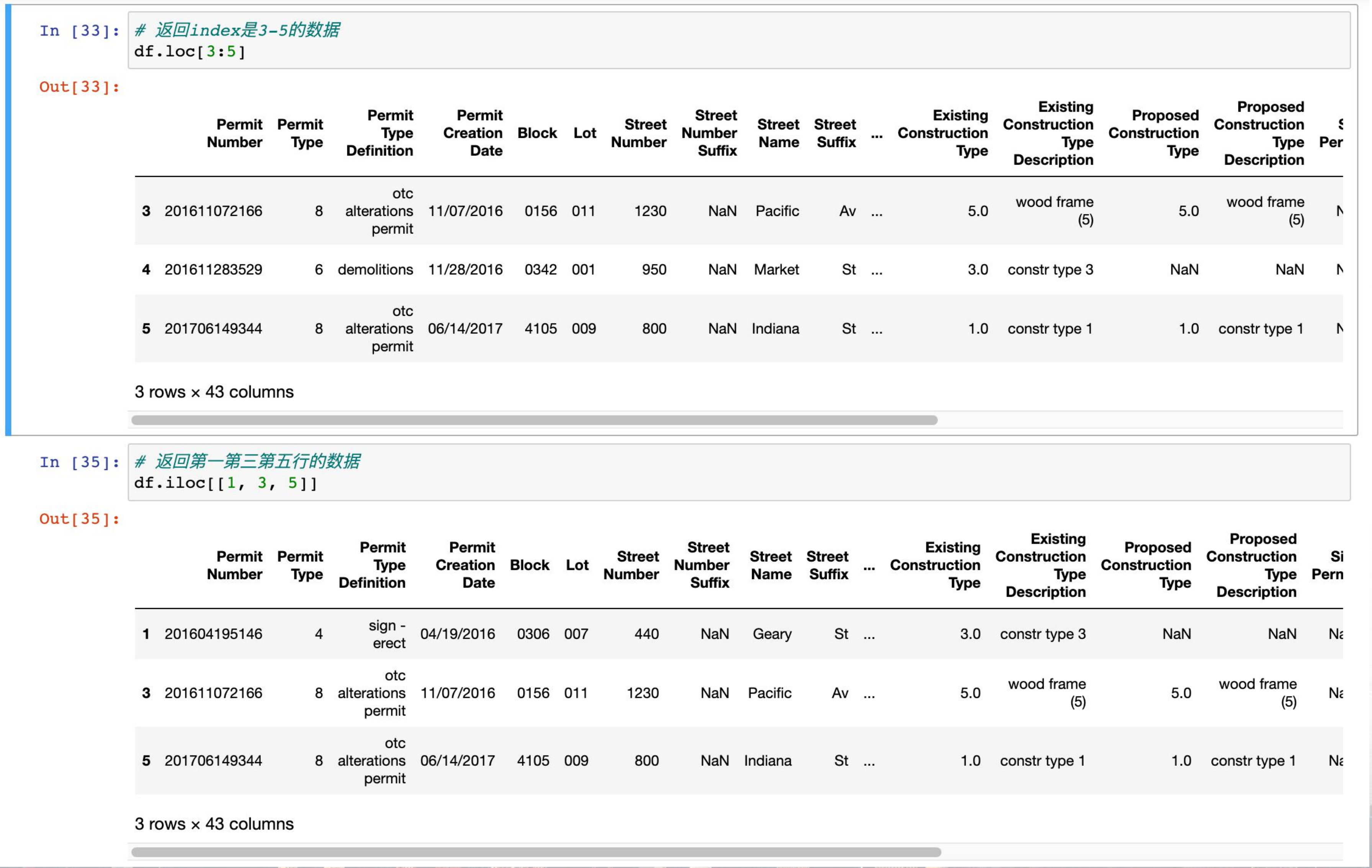
	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...	Existing Construction Type	Existing Construction Type Description	Proposed Construction Type	Proposed Construction Type Description	S Per
0	201505065519	4	sign - erect	05/06/2015	0326	023	140	NaN	Ellis	St	...	3.0	constr type 3	NaN	NaN	N
1	201604195146	4	sign - erect	04/19/2016	0306	007	440	NaN	Geary	St	...	3.0	constr type 3	NaN	NaN	N
2	201605278609	3	additions alterations or repairs	05/27/2016	0595	203	1647	NaN	Pacific	Av	...	1.0	constr type 1	1.0	constr type 1	N
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	Av	...	5.0	wood frame (5)	5.0	wood frame (5)	N
4	201611283529	6	demolitions	11/28/2016	0342	001	950	NaN	Market	St	...	3.0	constr type 3	NaN	NaN	N
5	201706149344	8	otc alterations permit	06/14/2017	4105	009	800	NaN	Indiana	St	...	1.0	constr type 1	1.0	constr type 1	N
6	201706300814	8	otc alterations permit	06/30/2017	1739	020	1291	NaN	11th	Av	...	5.0	wood frame (5)	5.0	wood frame (5)	N
7	M803667	8	otc alterations permit	06/30/2017	4789	014	1465	NaN	Revere	Av	...	NaN	NaN	NaN	NaN	N
8	M804227	8	otc alterations permit	07/05/2017	1212	054	2094	NaN	Fell	St	...	NaN	NaN	NaN	NaN	N

Access data in pandas

Row_Data = DataFrame.loc[index_list]

Row_Data = DataFrame.iloc[location_list]

visit row



In [33]:
```
# 返回index是3-5的数据
df.loc[3:5]
```

Out[33]:

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...	Existing Construction Type	Existing Construction Type Description	Proposed Construction Type	Proposed Construction Type Description	S Per
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	Av	...	5.0	wood frame (5)	5.0	wood frame (5)	N
4	201611283529	6	demolitions	11/28/2016	0342	001	950	NaN	Market	St	...	3.0	constr type 3	NaN	NaN	N
5	201706149344	8	otc alterations permit	06/14/2017	4105	009	800	NaN	Indiana	St	...	1.0	constr type 1	1.0	constr type 1	N

3 rows × 43 columns

In [35]:
```
# 返回第一第三第五行的数据
df.iloc[[1, 3, 5]]
```

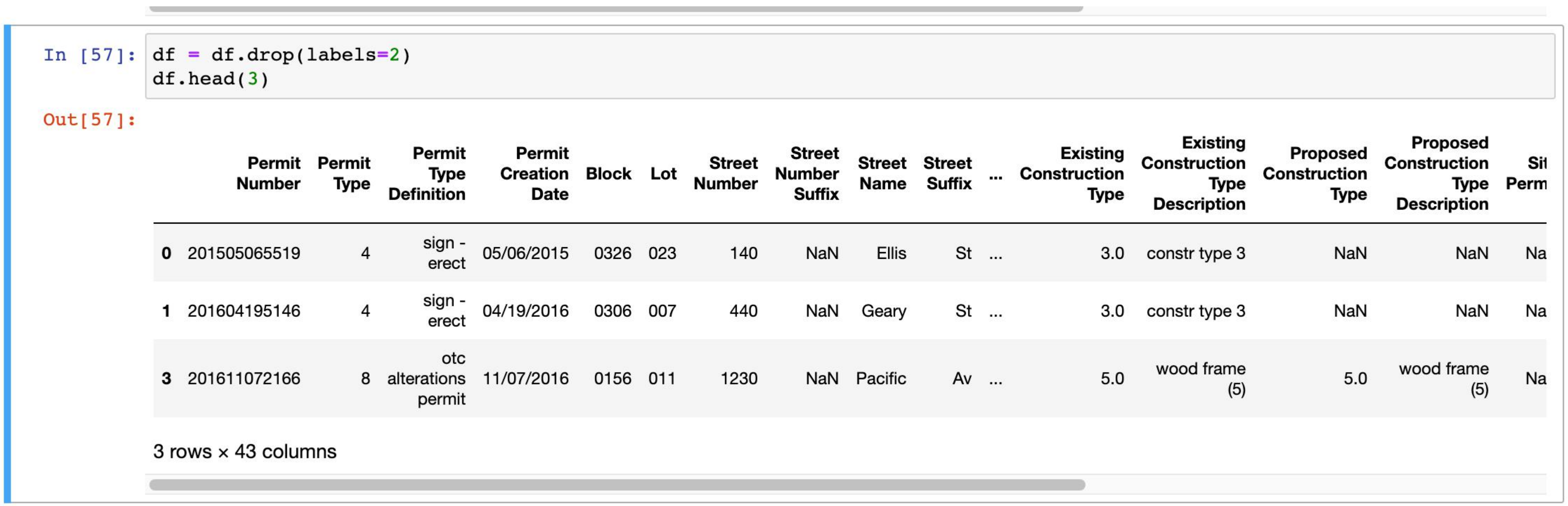
Out[35]:

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...	Existing Construction Type	Existing Construction Type Description	Proposed Construction Type	Proposed Construction Type Description	Si Pern
1	201604195146	4	sign - erect	04/19/2016	0306	007	440	NaN	Geary	St	...	3.0	constr type 3	NaN	NaN	Na
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	Av	...	5.0	wood frame (5)	5.0	wood frame (5)	Na
5	201706149344	8	otc alterations permit	06/14/2017	4105	009	800	NaN	Indiana	St	...	1.0	constr type 1	1.0	constr type 1	Na

3 rows × 43 columns

Access data in pandas

visit row



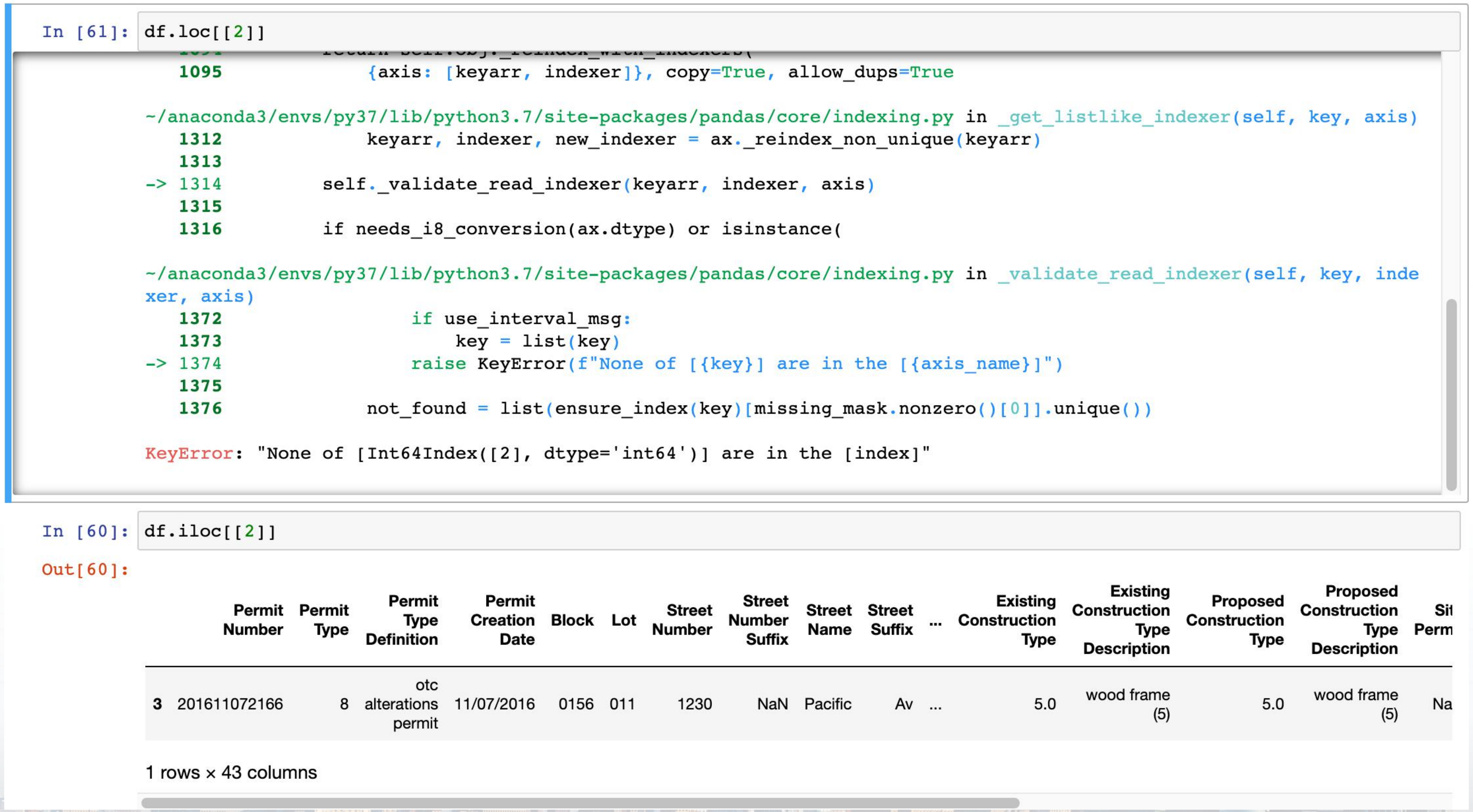
```
In [57]: df = df.drop(labels=2)
         df.head(3)
Out[57]:
```

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...	Existing Construction Type	Existing Construction Type Description	Proposed Construction Type	Proposed Construction Type Description	Sit Perm
0	201505065519	4	sign - erect	05/06/2015	0326	023	140	NaN	Ellis	St	...	3.0	constr type 3	NaN	NaN	Na
1	201604195146	4	sign - erect	04/19/2016	0306	007	440	NaN	Geary	St	...	3.0	constr type 3	NaN	NaN	Na
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	Av	...	5.0	wood frame (5)	5.0	wood frame (5)	Na

3 rows × 43 columns

Access data in pandas

visit row



```
In [61]: df.loc[[2]]
   1095                 {axis: [keyarr, indexer]}, copy=True, allow_dups=True

~/anaconda3/envs/py37/lib/python3.7/site-packages/pandas/core/indexing.py in _get_listlike_indexer(self, key, axis)
   1312             keyarr, indexer, new_indexer = ax._reindex_non_unique(keyarr)
   1313 
-> 1314         self._validate_read_indexer(keyarr, indexer, axis)
   1315 
   1316         if needs_i8_conversion(ax.dtype) or isinstance(

~/anaconda3/envs/py37/lib/python3.7/site-packages/pandas/core/indexing.py in _validate_read_indexer(self, key, inde
xer, axis)
   1372                 if use_interval_msg:
   1373                     key = list(key)
-> 1374                 raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   1375 
   1376             not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())

KeyError: "None of [Int64Index([2], dtype='int64')] are in the [index]"
```

```
In [60]: df.iloc[[2]]
```

Out[60]:

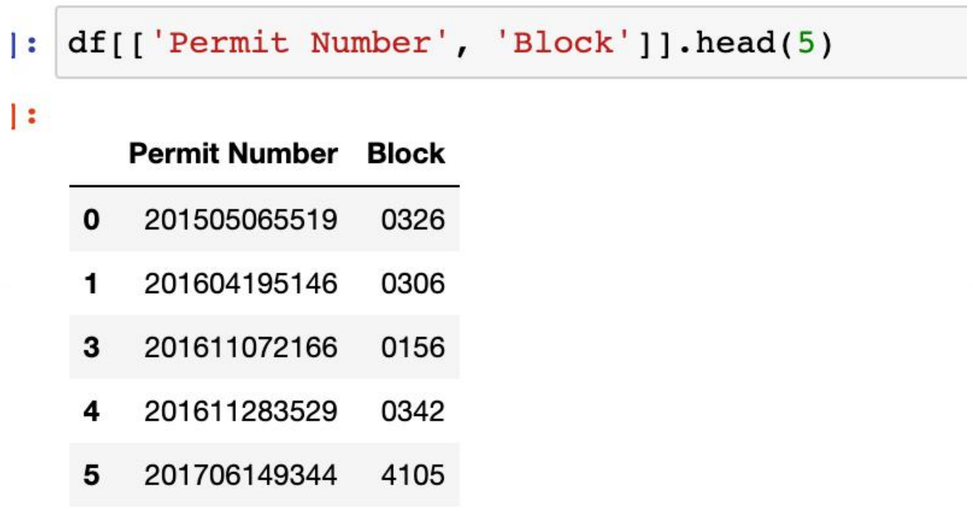
	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...	Existing Construction Type	Existing Construction Type Description	Proposed Construction Type	Proposed Construction Type Description	Sit Perm
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	Av	...	5.0	wood frame (5)	5.0	wood frame (5)	Na

1 rows × 43 columns

Access data in pandas

visit column

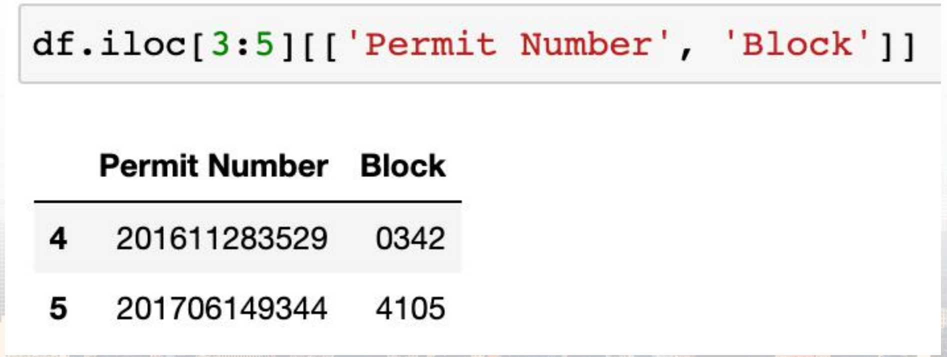
Column_Data = DataFrame[Column_Name_lsit]



```
]: df[['Permit Number', 'Block']].head(5)
]:
```

	Permit Number	Block
0	201505065519	0326
1	201604195146	0306
3	201611072166	0156
4	201611283529	0342
5	201706149344	4105

visit a special location data=visit row+visit column



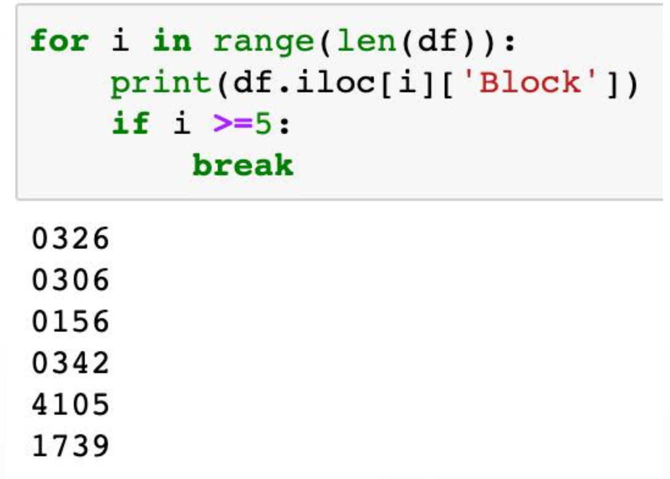
```
df.iloc[3:5][['Permit Number', 'Block']]
```

	Permit Number	Block
4	201611283529	0342
5	201706149344	4105

Access data in pandas

loop visit pandas data*3

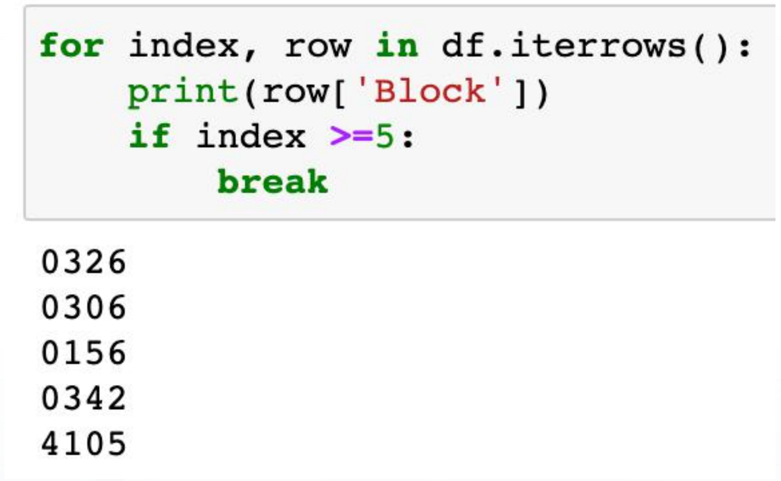
for loop on dataframe



```
for i in range(len(df)):
    print(df.iloc[i]['Block'])
    if i >=5:
        break
```

```
0326
0306
0156
0342
4105
1739
```

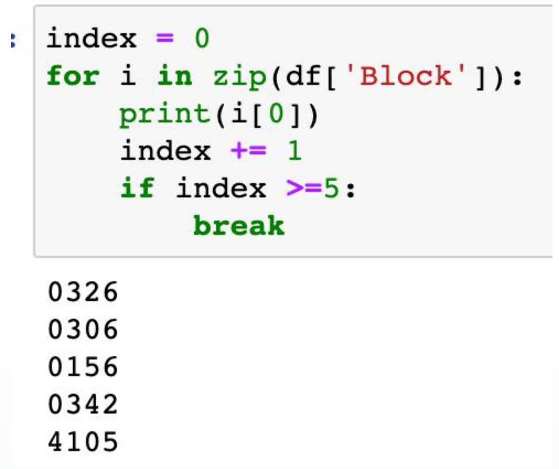
iteration on dataframe



```
for index, row in df.iterrows():
    print(row['Block'])
    if index >=5:
        break
```

```
0326
0306
0156
0342
4105
```

for loop on zip data

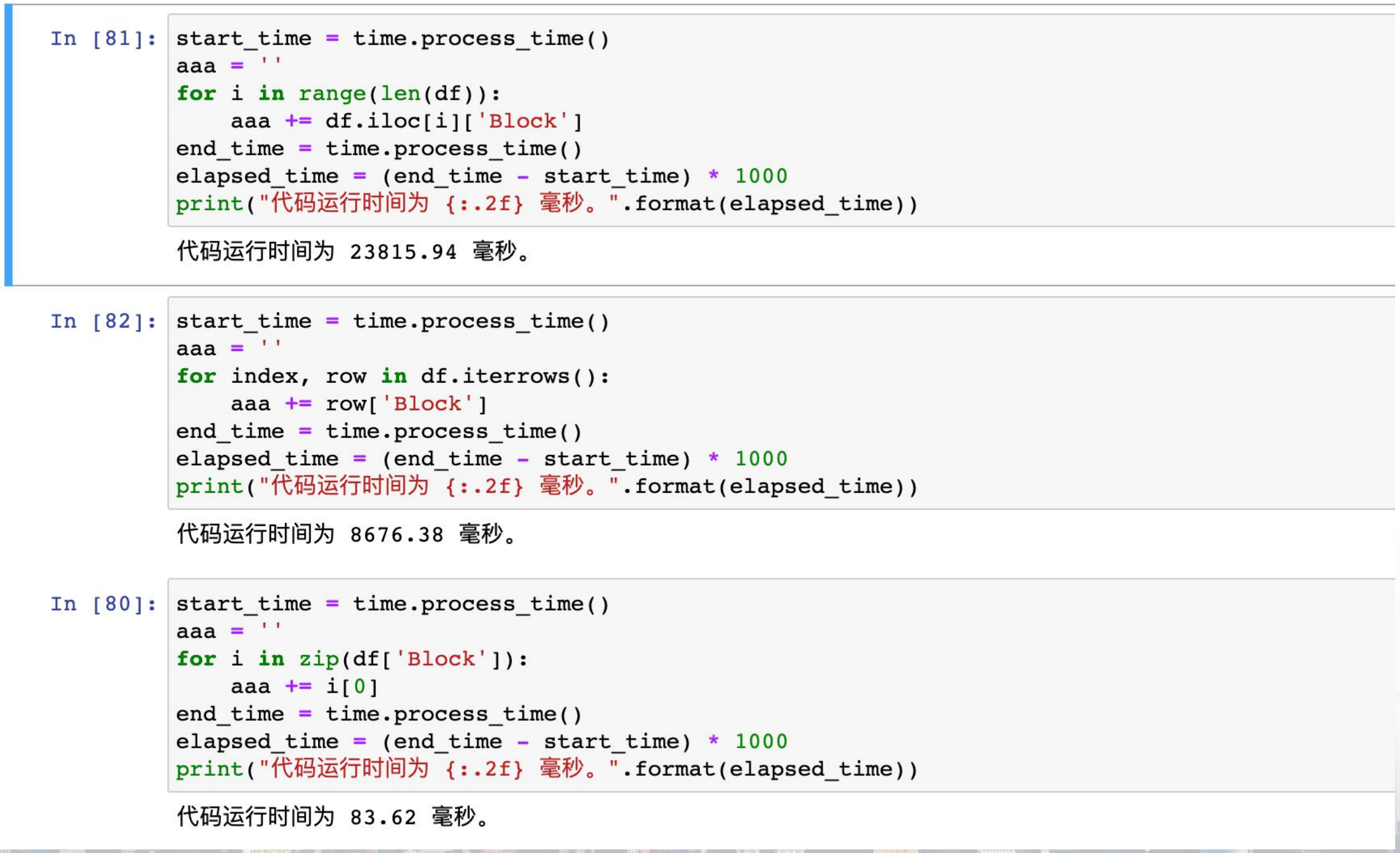


```
index = 0
for i in zip(df['Block']):
    print(i[0])
    index += 1
    if index >=5:
        break
```

```
0326
0306
0156
0342
4105
```

Access data in pandas

loop visit pandas data*3



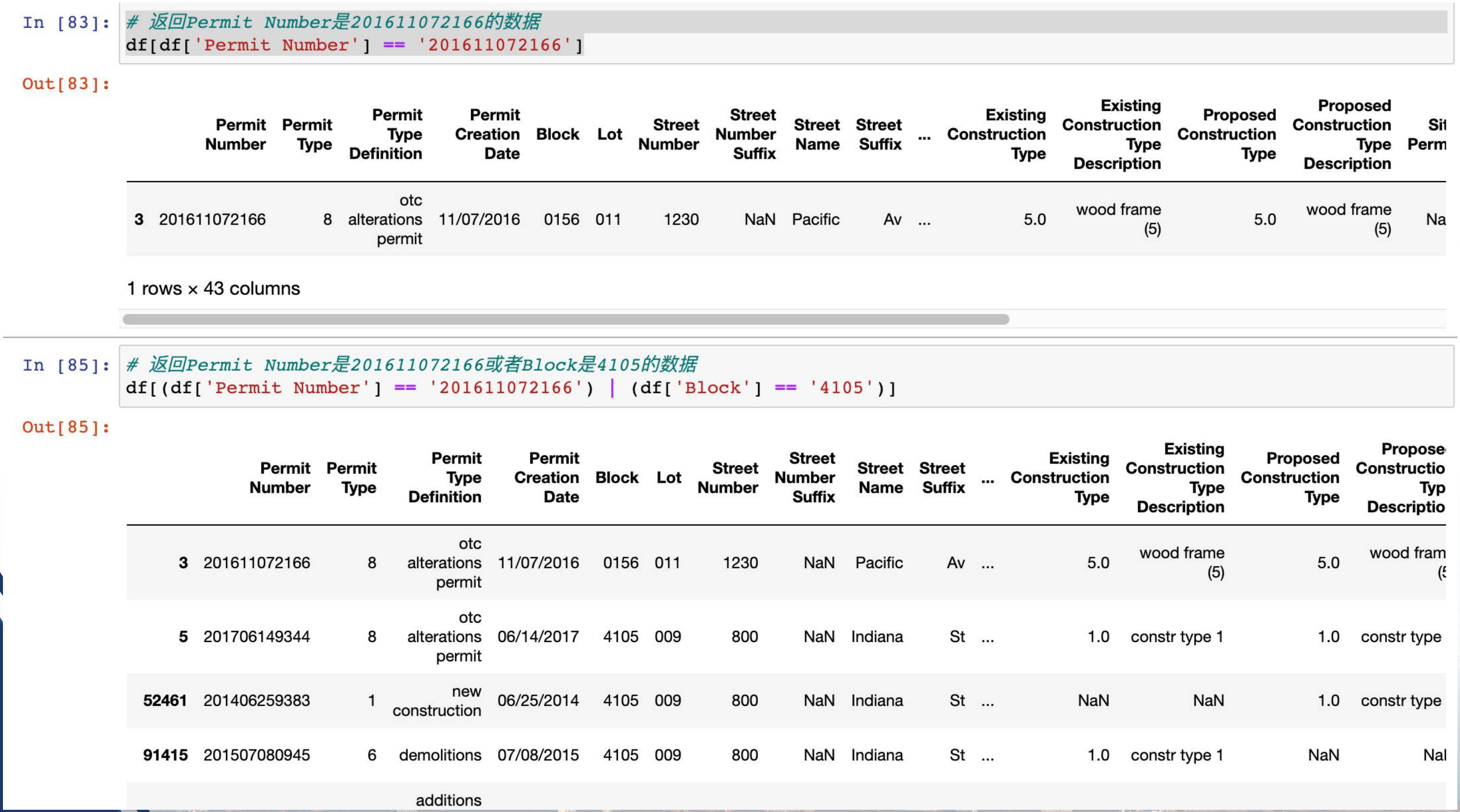
```
In [81]: start_time = time.process_time()
         aaa = ''
         for i in range(len(df)):
             aaa += df.iloc[i]['Block']
         end_time = time.process_time()
         elapsed_time = (end_time - start_time) * 1000
         print("代码运行时间为 {:.2f} 毫秒。".format(elapsed_time))

         代码运行时间为 23815.94 毫秒。

In [82]: start_time = time.process_time()
         aaa = ''
         for index, row in df.iterrows():
             aaa += row['Block']
         end_time = time.process_time()
         elapsed_time = (end_time - start_time) * 1000
         print("代码运行时间为 {:.2f} 毫秒。".format(elapsed_time))

         代码运行时间为 8676.38 毫秒。

In [80]: start_time = time.process_time()
         aaa = ''
         for i in zip(df['Block']):
             aaa += i[0]
         end_time = time.process_time()
         elapsed_time = (end_time - start_time) * 1000
         print("代码运行时间为 {:.2f} 毫秒。".format(elapsed_time))

         代码运行时间为 83.62 毫秒。
```

Access data in pandas

visit data with condition

Row_Data = DataFrame[Boolean_lsit]

and->&
or->|
not->-



```
In [83]: # 返回Permit Number是201611072166的数据
         df[df['Permit Number'] == '201611072166']
```

Out[83]:

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...	Existing Construction Type	Existing Construction Type Description	Proposed Construction Type	Proposed Construction Type Description	Si Perm
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	Av	...	5.0	wood frame (5)	5.0	wood frame (5)	Na

1 rows × 43 columns

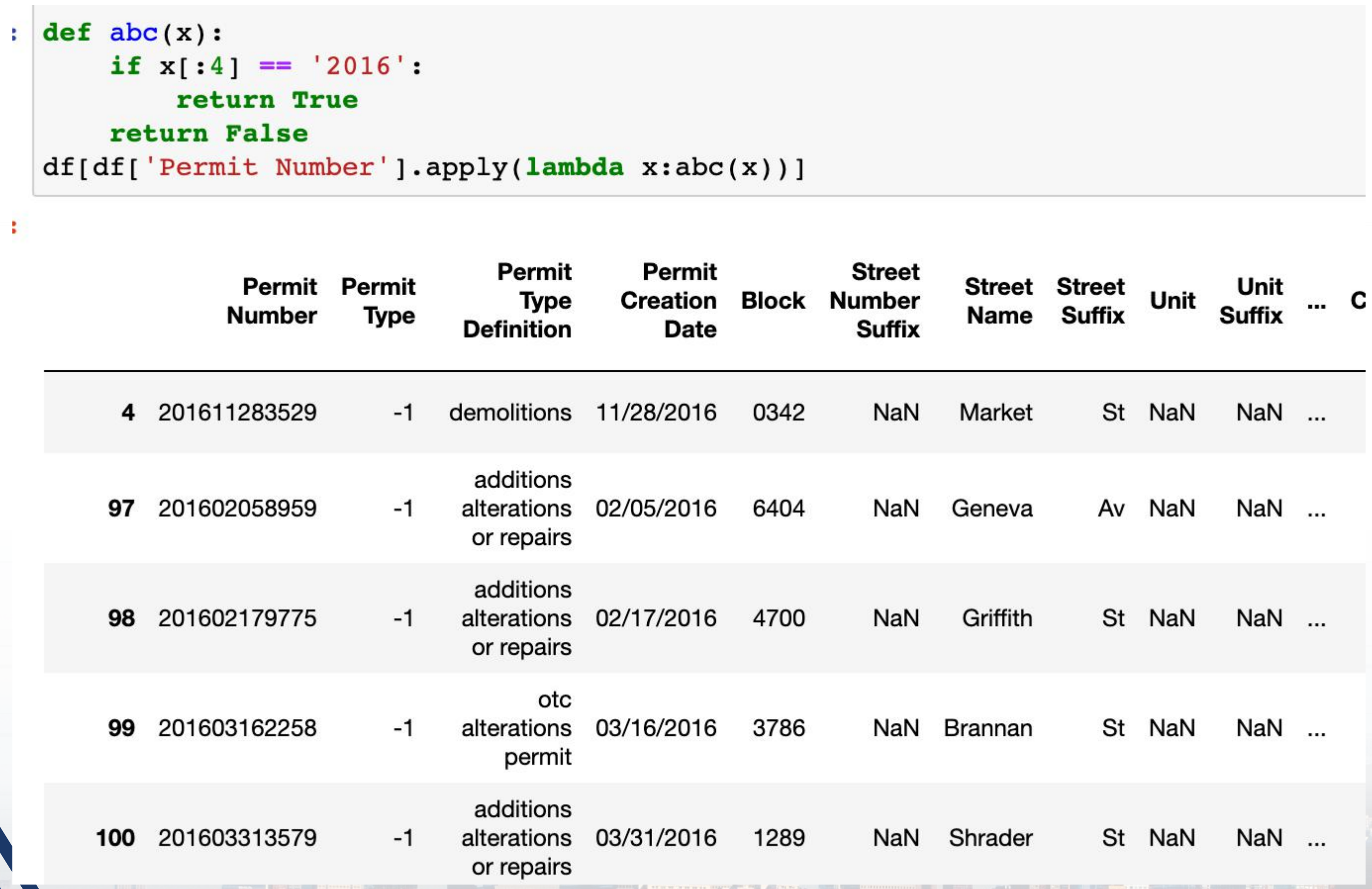
```
In [85]: # 返回Permit Number是201611072166或者Block是4105的数据
         df[(df['Permit Number'] == '201611072166') | (df['Block'] == '4105')]
```

Out[85]:

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...	Existing Construction Type	Existing Construction Type Description	Proposed Construction Type	Propose Constructio Typ Descriptio
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	Av	...	5.0	wood frame (5)	5.0	wood fram (
5	201706149344	8	otc alterations permit	06/14/2017	4105	009	800	NaN	Indiana	St	...	1.0	constr type 1	1.0	constr type
52461	201406259383	1	new construction	06/25/2014	4105	009	800	NaN	Indiana	St	...	NaN	NaN	1.0	constr type
91415	201507080945	6	demolitions	07/08/2015	4105	009	800	NaN	Indiana	St	...	1.0	constr type 1	NaN	Na
			additions												

Access data in pandas

visit data with condition



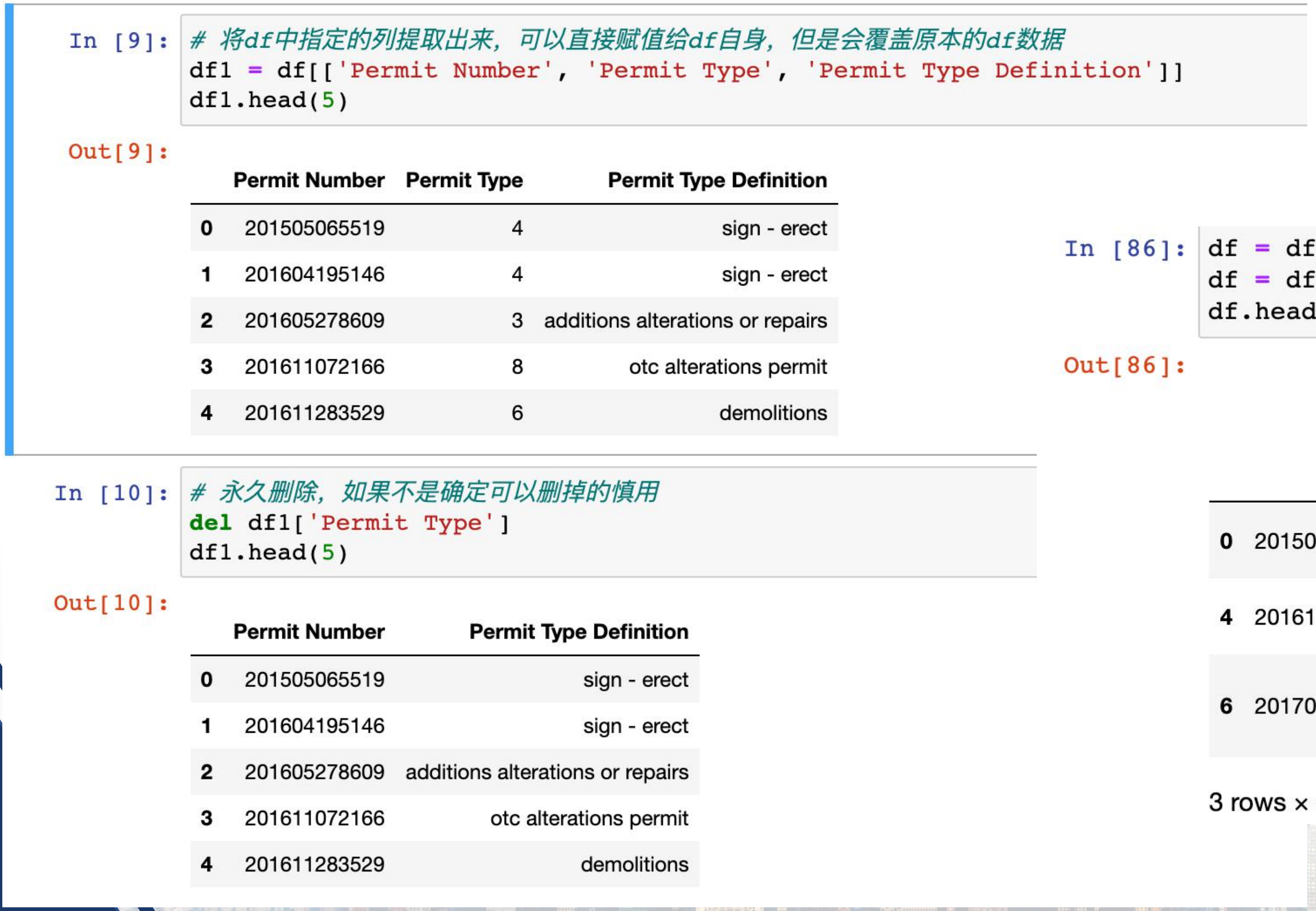
```python
def abc(x):
    if x[:4] == '2016':
        return True
    return False
df[df['Permit Number'].apply(lambda x:abc(x))]
```

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Street Number Suffix	Street Name	Street Suffix	Unit	Unit Suffix	...	C
4	201611283529	-1	demolitions	11/28/2016	0342	NaN	Market	St	NaN	NaN	...	
97	201602058959	-1	additions alterations or repairs	02/05/2016	6404	NaN	Geneva	Av	NaN	NaN	...	
98	201602179775	-1	additions alterations or repairs	02/17/2016	4700	NaN	Griffith	St	NaN	NaN	...	
99	201603162258	-1	otc alterations permit	03/16/2016	3786	NaN	Brannan	St	NaN	NaN	...	
100	201603313579	-1	additions alterations or repairs	03/31/2016	1289	NaN	Shrader	St	NaN	NaN	...	

df.apply(lambda x:funcname(x))
means every row in this df will execute this function and get a result

Data Selection

data selection=data access and save or delete some unuseful data



In [9]:
```python
# 将df中指定的列提取出来，可以直接赋值给df自身，但是会覆盖原本的df数据
df1 = df[['Permit Number', 'Permit Type', 'Permit Type Definition']]
df1.head(5)
```

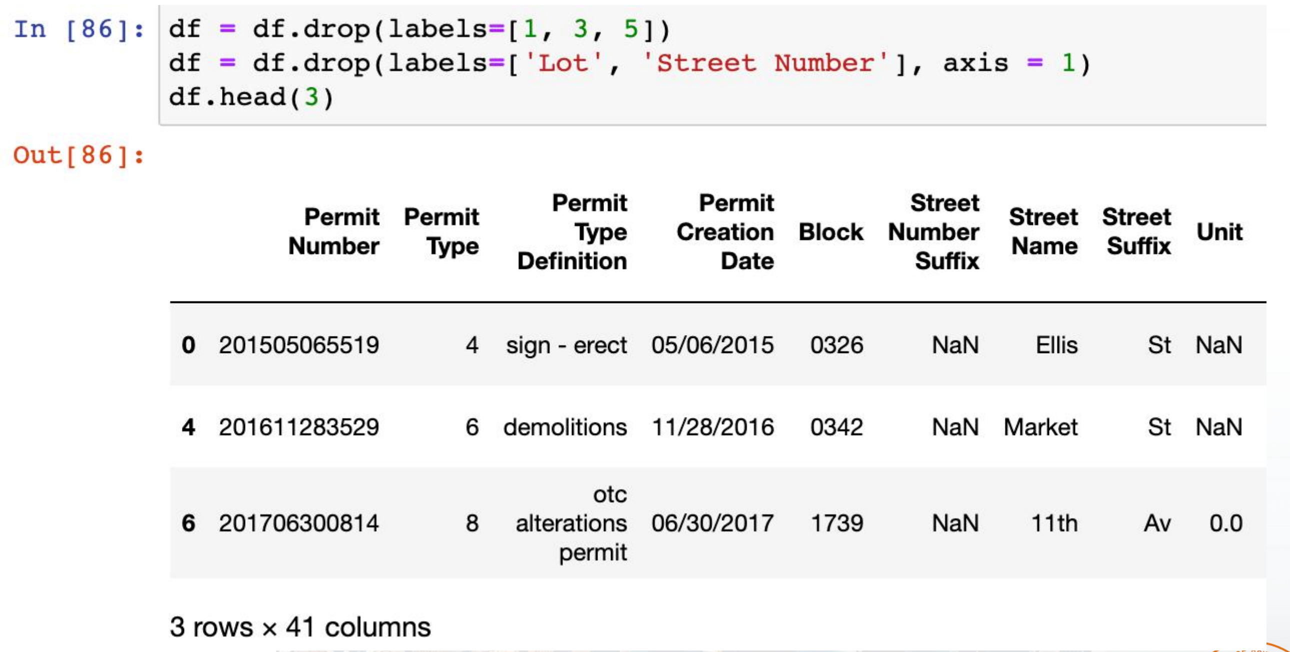
Out[9]:

	Permit Number	Permit Type	Permit Type Definition
0	201505065519	4	sign - erect
1	201604195146	4	sign - erect
2	201605278609	3	additions alterations or repairs
3	201611072166	8	otc alterations permit
4	201611283529	6	demolitions

In [10]:
```python
# 永久删除，如果不是确定可以删掉的慎用
del df1['Permit Type']
df1.head(5)
```

Out[10]:

	Permit Number	Permit Type Definition
0	201505065519	sign - erect
1	201604195146	sign - erect
2	201605278609	additions alterations or repairs
3	201611072166	otc alterations permit
4	201611283529	demolitions



In [86]:
```python
df = df.drop(labels=[1, 3, 5])
df = df.drop(labels=['Lot', 'Street Number'], axis = 1)
df.head(3)
```

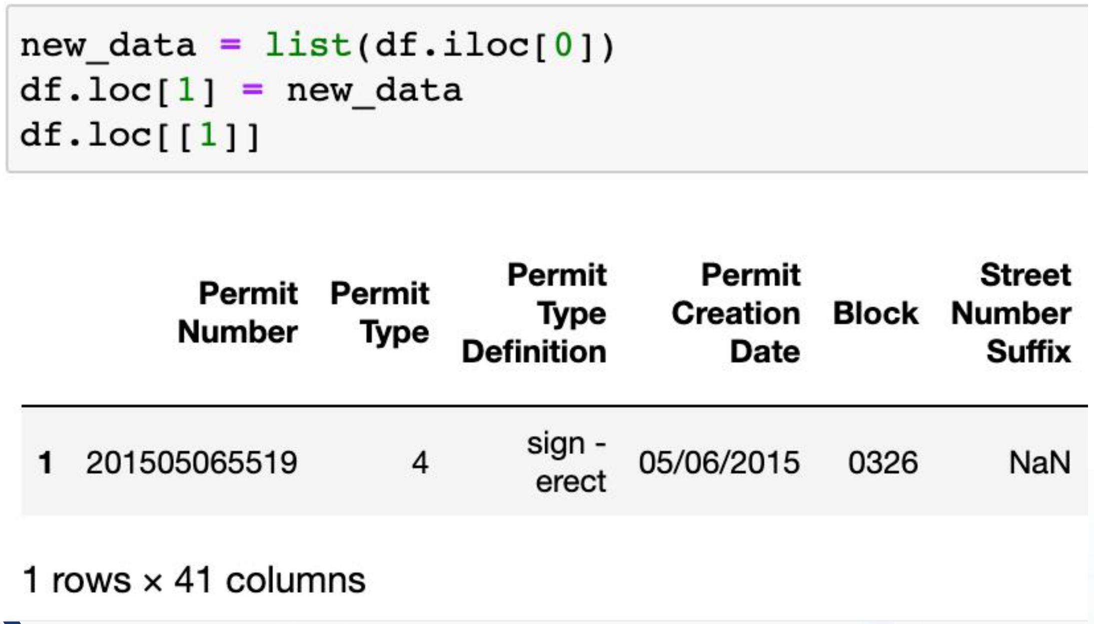
Out[86]:

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Street Number Suffix	Street Name	Street Suffix	Unit
0	201505065519	4	sign - erect	05/06/2015	0326	NaN	Ellis	St	NaN
4	201611283529	6	demolitions	11/28/2016	0342	NaN	Market	St	NaN
6	201706300814	8	otc alterations permit	06/30/2017	1739	NaN	11th	Av	0.0

3 rows × 41 columns

Data Addition or Modification

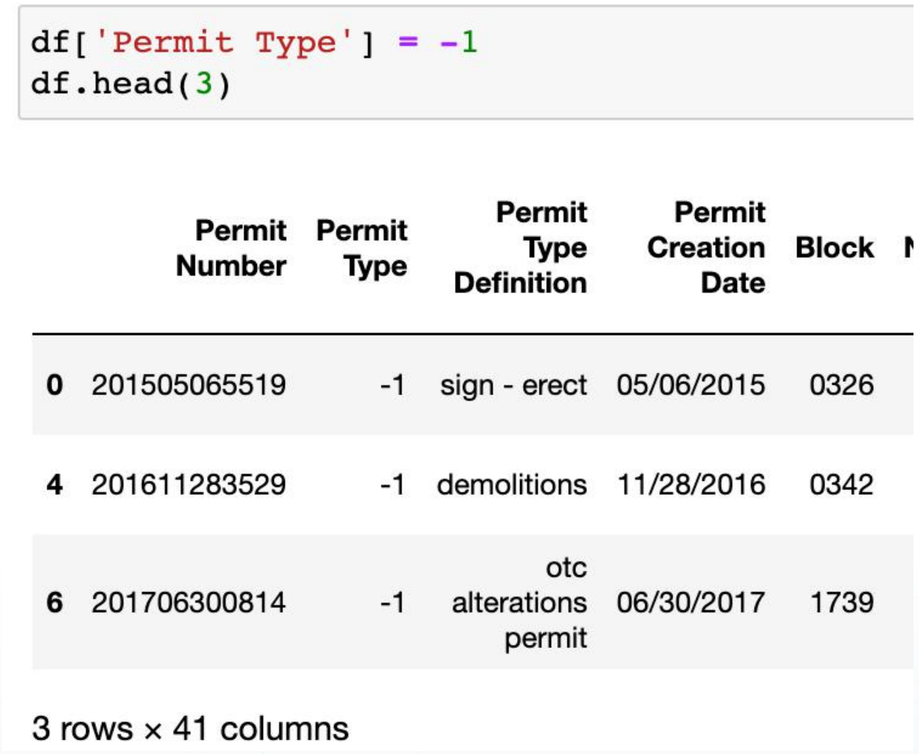
Select a row or a column and save data in it



```
new_data = list(df.iloc[0])
df.loc[1] = new_data
df.loc[[1]]
```

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Street Number Suffix
1	201505065519	4	sign - erect	05/06/2015	0326	NaN

1 rows × 41 columns

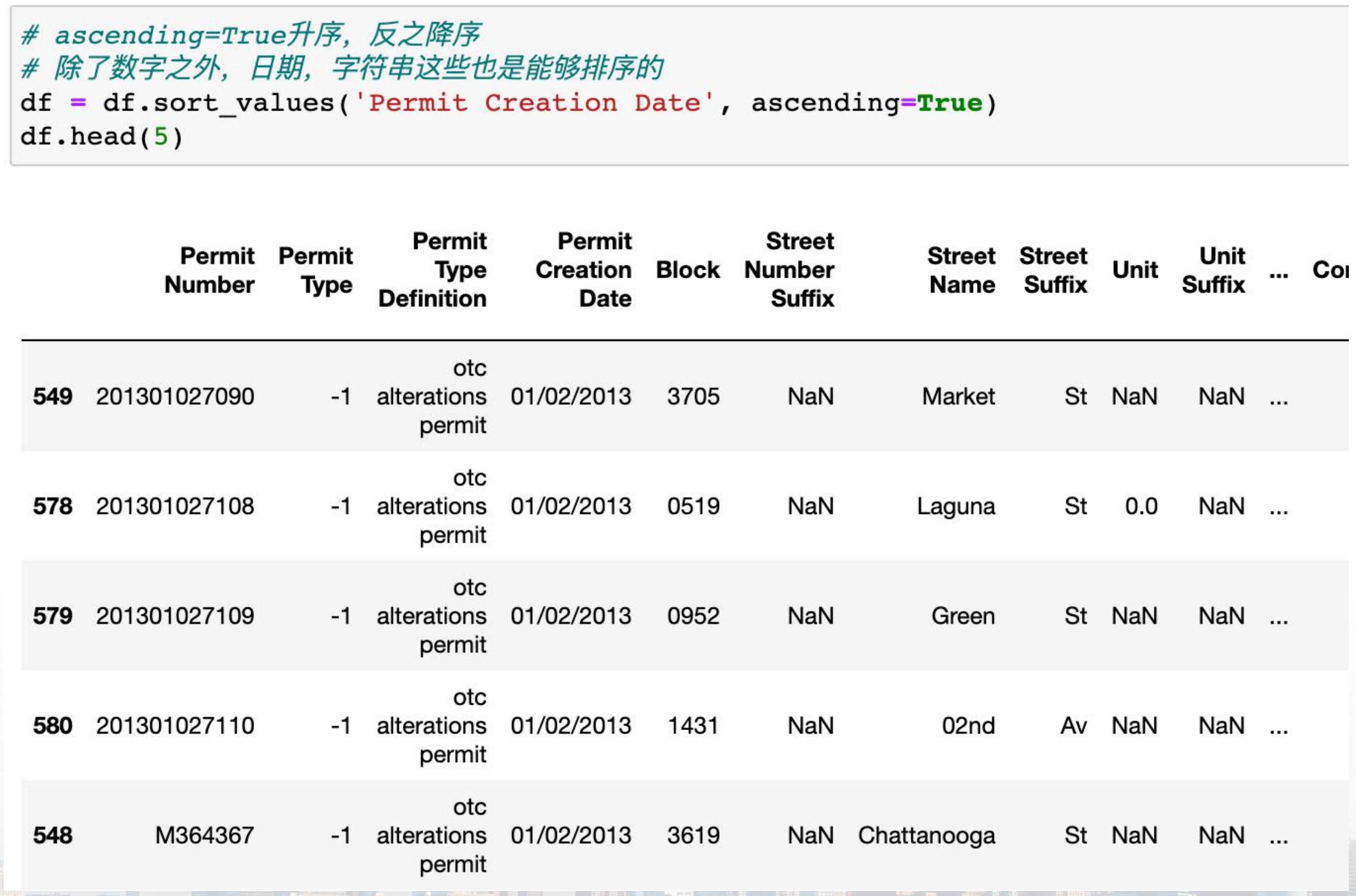


```
df['Permit Type'] = -1
df.head(3)
```

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block
0	201505065519	-1	sign - erect	05/06/2015	0326
4	201611283529	-1	demolitions	11/28/2016	0342
6	201706300814	-1	otc alterations permit	06/30/2017	1739

3 rows × 41 columns

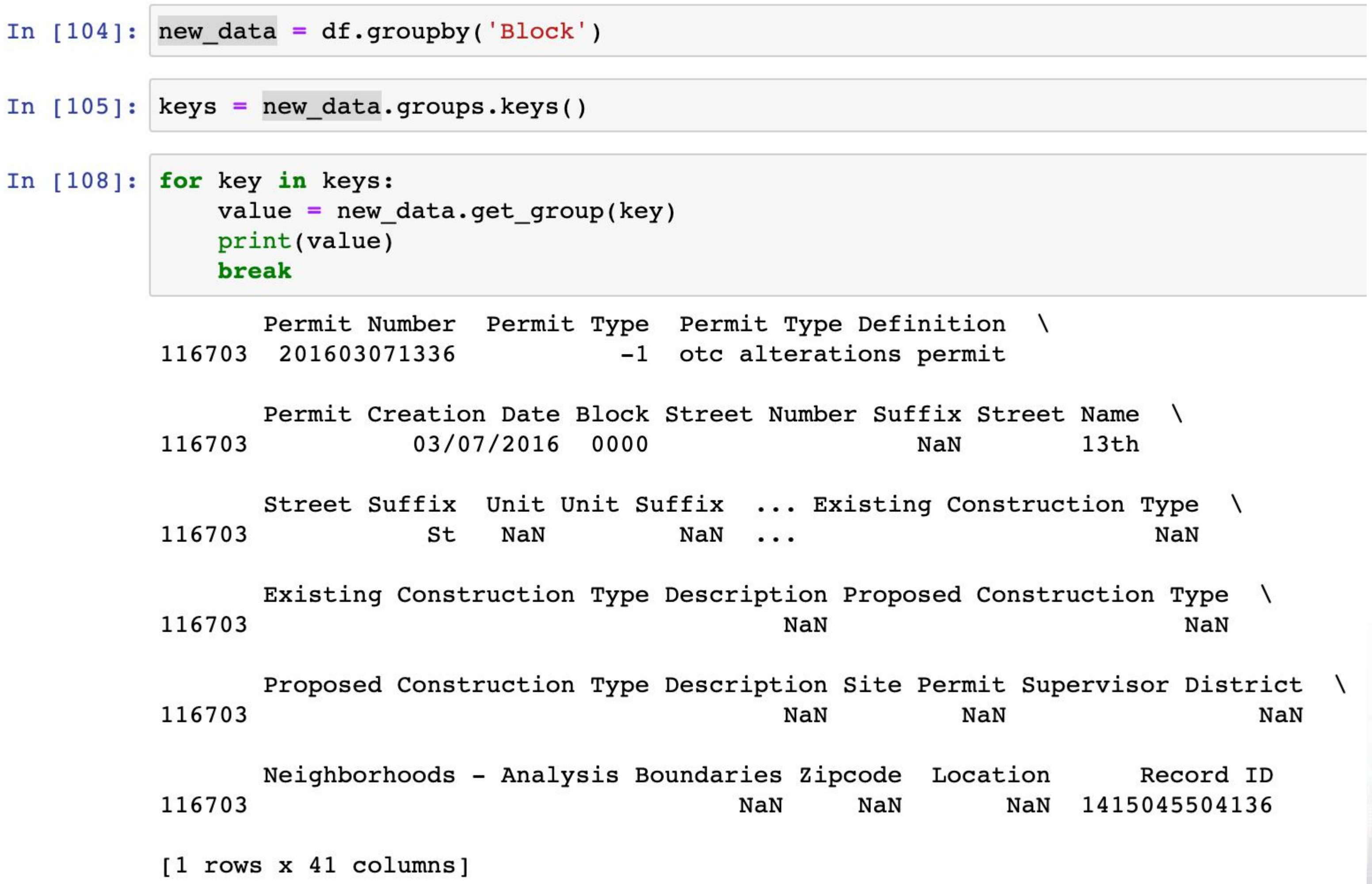
Sort



```python
# ascending=True升序，反之降序
# 除了数字之外，日期，字符串这些也是能够排序的
df = df.sort_values('Permit Creation Date', ascending=True)
df.head(5)
```

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Street Number Suffix	Street Name	Street Suffix	Unit	Unit Suffix	...	Co
549	201301027090	-1	otc alterations permit	01/02/2013	3705	NaN	Market	St	NaN	NaN	...	
578	201301027108	-1	otc alterations permit	01/02/2013	0519	NaN	Laguna	St	0.0	NaN	...	
579	201301027109	-1	otc alterations permit	01/02/2013	0952	NaN	Green	St	NaN	NaN	...	
580	201301027110	-1	otc alterations permit	01/02/2013	1431	NaN	02nd	Av	NaN	NaN	...	
548	M364367	-1	otc alterations permit	01/02/2013	3619	NaN	Chattanooga	St	NaN	NaN	...	

Groupby



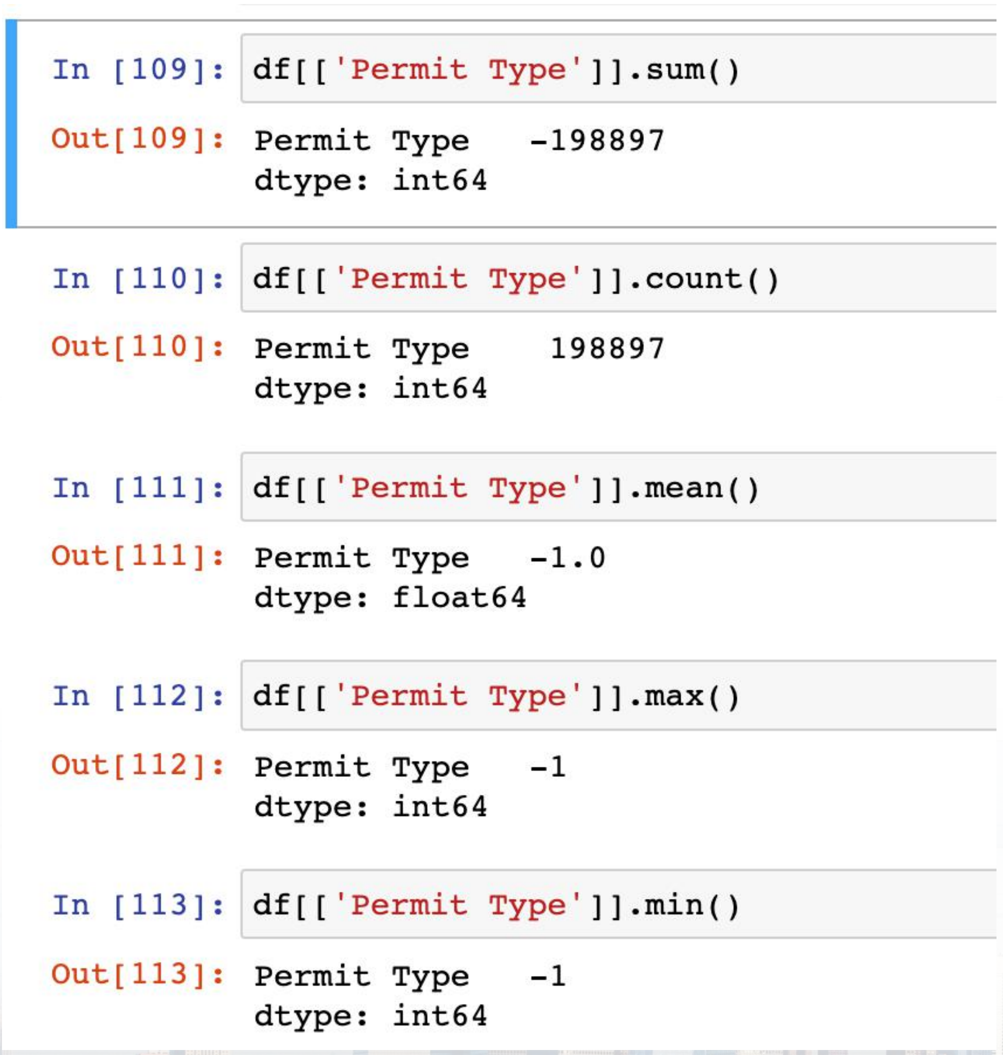
```
In [104]: new_data = df.groupby('Block')

In [105]: keys = new_data.groups.keys()

In [108]: for key in keys:
              value = new_data.get_group(key)
              print(value)
              break
```

```
       Permit Number  Permit Type  Permit Type Definition  \
116703  201603071336           -1  otc alterations permit

       Permit Creation Date Block Street Number Suffix Street Name  \
116703           03/07/2016  0000                  NaN        13th

       Street Suffix  Unit Unit Suffix  ... Existing Construction Type  \
116703            St   NaN         NaN  ...                        NaN

       Existing Construction Type Description Proposed Construction Type  \
116703                                    NaN                        NaN

       Proposed Construction Type Description Site Permit Supervisor District  \
116703                                    NaN         NaN                 NaN

       Neighborhoods - Analysis Boundaries Zipcode  Location      Record ID
116703                                 NaN     NaN       NaN  1415045504136

[1 rows x 41 columns]
```

Combine rows with the same element into a new table using one or more columns.

You can loop through the values of the column, and then loop to extract to achieve the same effect, but it will be very slow.

Statistical Methods


```
In [109]: df[['Permit Type']].sum()

Out[109]: Permit Type   -198897
          dtype: int64

In [110]: df[['Permit Type']].count()

Out[110]: Permit Type    198897
          dtype: int64

In [111]: df[['Permit Type']].mean()

Out[111]: Permit Type   -1.0
          dtype: float64

In [112]: df[['Permit Type']].max()

Out[112]: Permit Type   -1
          dtype: int64

In [113]: df[['Permit Type']].min()

Out[113]: Permit Type   -1
          dtype: int64
```

You can use this behind any DataFrame
But if it has some data which not a number, it will go wrong

Dates

Let's start by printing out the date column, shall we?

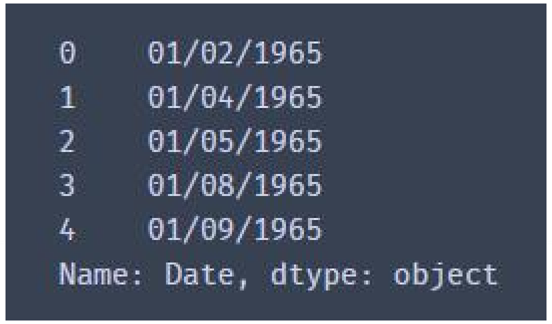
```
0    01/02/1965
1    01/04/1965
2    01/05/1965
3    01/08/1965
4    01/09/1965
Name: Date, dtype: object
```

We can clearly see that a string like "01/02/1965" to be a date. In python, this is called a "datetime" type. However, when we read the csv file, this structure is not automatically maintained, and instead, we just get the default "object" type.

```
dtype('O')
```

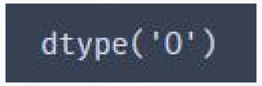
Dates

Let’s start by printing out the date column, shall we?



```
0    01/02/1965
1    01/04/1965
2    01/05/1965
3    01/08/1965
4    01/09/1965
Name: Date, dtype: object
```

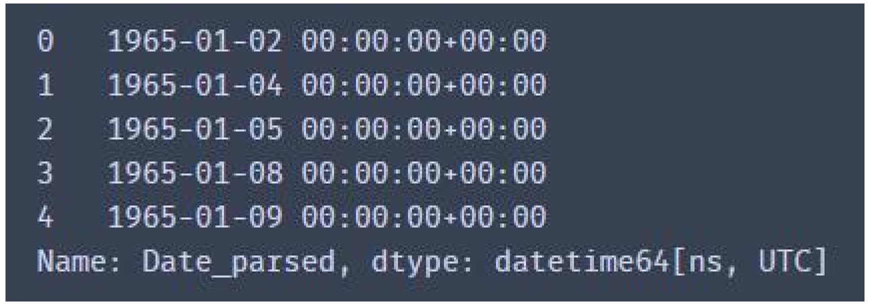
We can clearly see that a string like “01/02/1965” to be a date. In python, this is called a “datetime” type. However, when we read the csv file, this structure is not automatically maintained, and instead, we just get the default “object” type.



```
dtype('O')
```

Date

We will use the **pandas.to_datetime()** function to convert the object type column into datetime type column.



```
0   1965-01-02 00:00:00+00:00
1   1965-01-04 00:00:00+00:00
2   1965-01-05 00:00:00+00:00
3   1965-01-08 00:00:00+00:00
4   1965-01-09 00:00:00+00:00
Name: Date_parsed, dtype: datetime64[ns, UTC]
```

If you encounter problems when converting datetime, refer to these 2 following links:

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html

https://docs.python.org/zh-cn/3/library/datetime.html#strftime-and-strptime-format-codes

Date

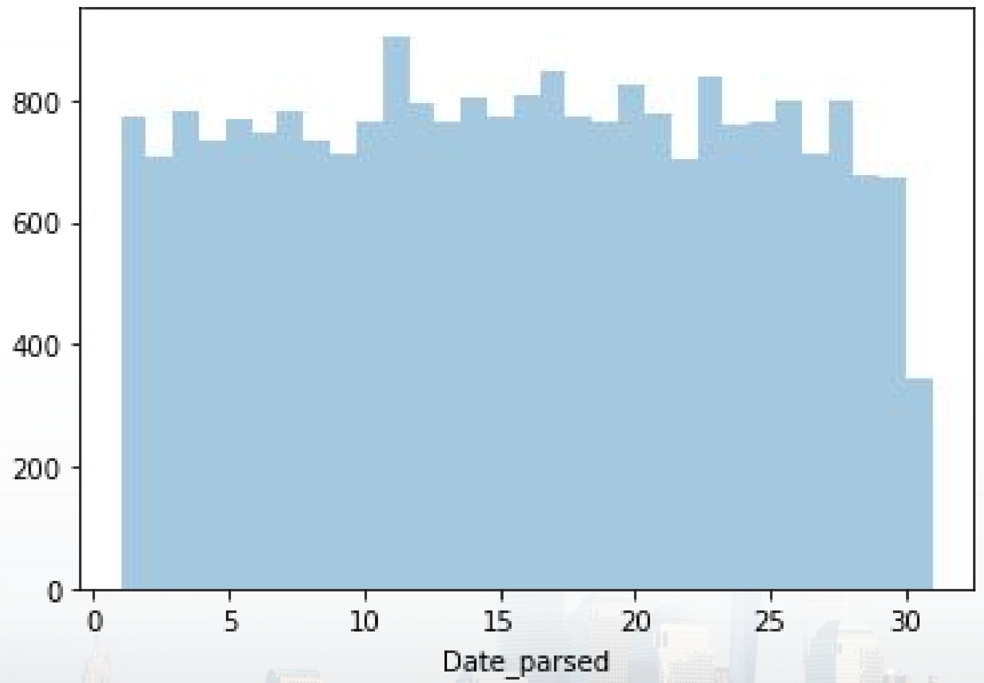
pandas.to_datetime(arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, format=None, exact=True, unit=None, infer_datetime_format=False, origin='unix', cache=True)

errors: 版本足够新的pandas，errors会默认设置为'raise'（遇到无法识别的字符会弹出报错）;在旧版本的pandas中，这个errors会默认设置为'ignore'（遇到错误直接跳过，也不报错，但也没成功转换为datetime格式）;最后一个errors选项为'coerce'，会将无法转换的时间转换为一个专有的NaT格式
dayfirst, yearfirst: 一些相对不是很重要的选项
utc: 是否强制使用UTC标准时
format: 定制arg的时间格式strftime(format)，举例格式可以是'%Y-%m-%d %H:%M:%S'或者'%Y-%m-%d'（根据你的arg的格式来定）
exact, unit: exact控制上述format是完全对应还是部分对应，unit控制转换出来的时间精度，一般都用默认的True和ns就好
infer_datetime_format: 懒人方法，根据第一个处理的时间格式来自动决定用什么format，用这个处理时间会更加耗时
origin: 处理时间时使用的默认参考0时，一般都是用默认的unix
cache: 是否使用缓存来加快处理时间，默认选是，一般不用改

Date

Cool, we get the date column in format “datetime”, now what?

We can start extracting the day information from the column and plot out the day distribution.



Hint:
Use **pandas.Series.dt.day()** to extract each datetime from the column.
Use **seaborn.distplot()** to make the plot.

```
day_of_month_earthquakes = earthquakes['Date_parsed'].dt.day
```

```
sns.distplot(day_of_month_earthquakes, kde=False, bins=31)
```

Date – Lab Exercise

Make a day plot AND a week-of-day plot of both data:

Data 1: landslide_catalog.csv

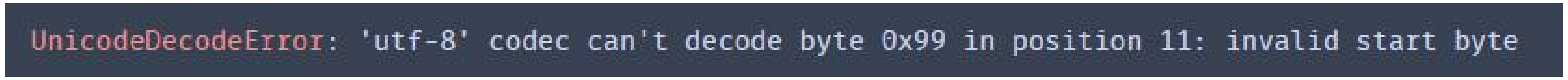
landslide_
catalog.cs
v

Data 2: volcano_database.csv

volcano_d
atabase.cs
v

Character Encoding

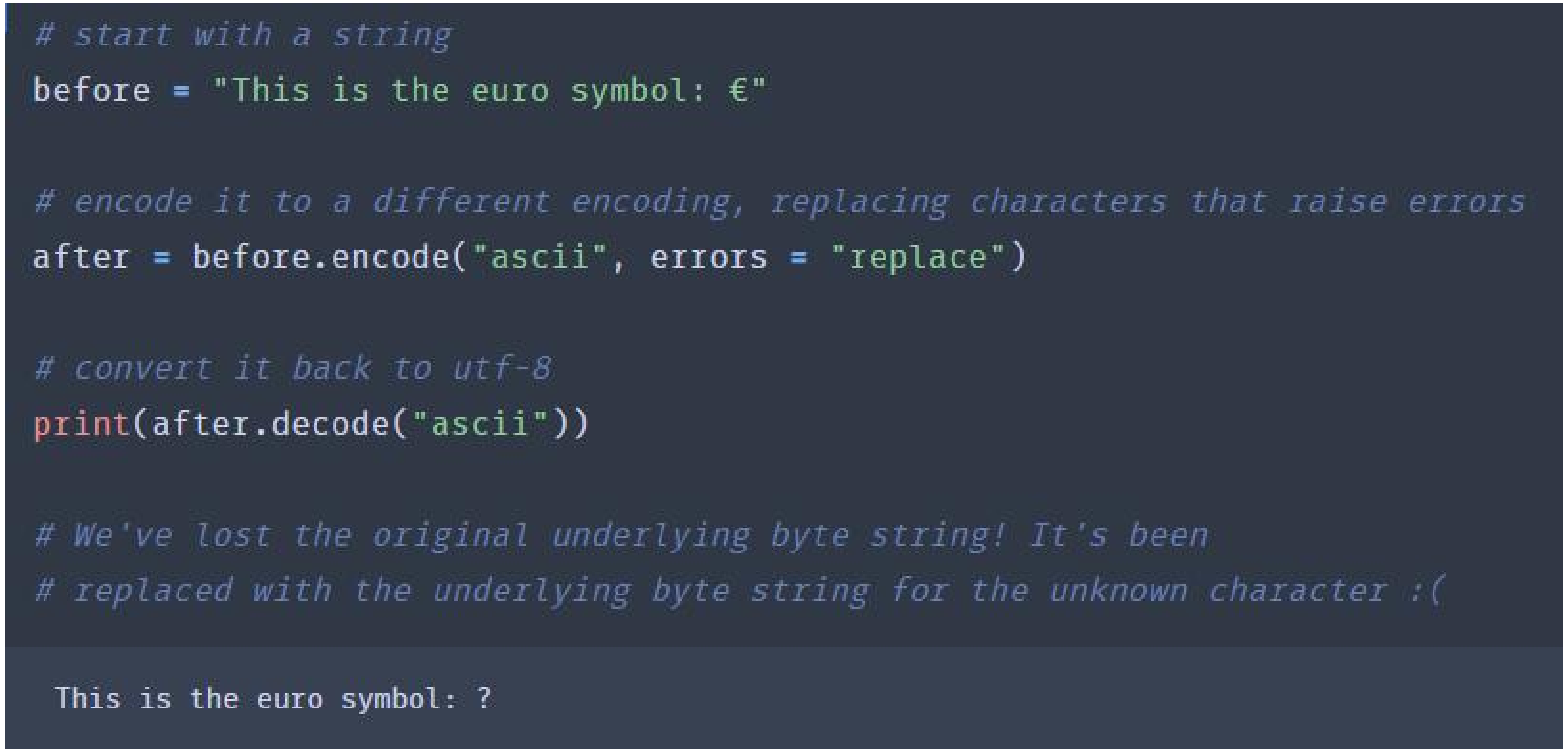
Sometimes, the file you try to read in might not be the convenient encoding type (the default standard encoding is type ‘utf-8’).



```
UnicodeDecodeError: 'utf-8' codec can't decode byte 0x99 in position 11: invalid start byte
```

But let’s first play with the character codings first: Try encoding and decoding different symbols to ASCII and see what happens. I'd recommend $, #, 你好 and न म स् ते but feel free to try other characters as well.

Character Encoding



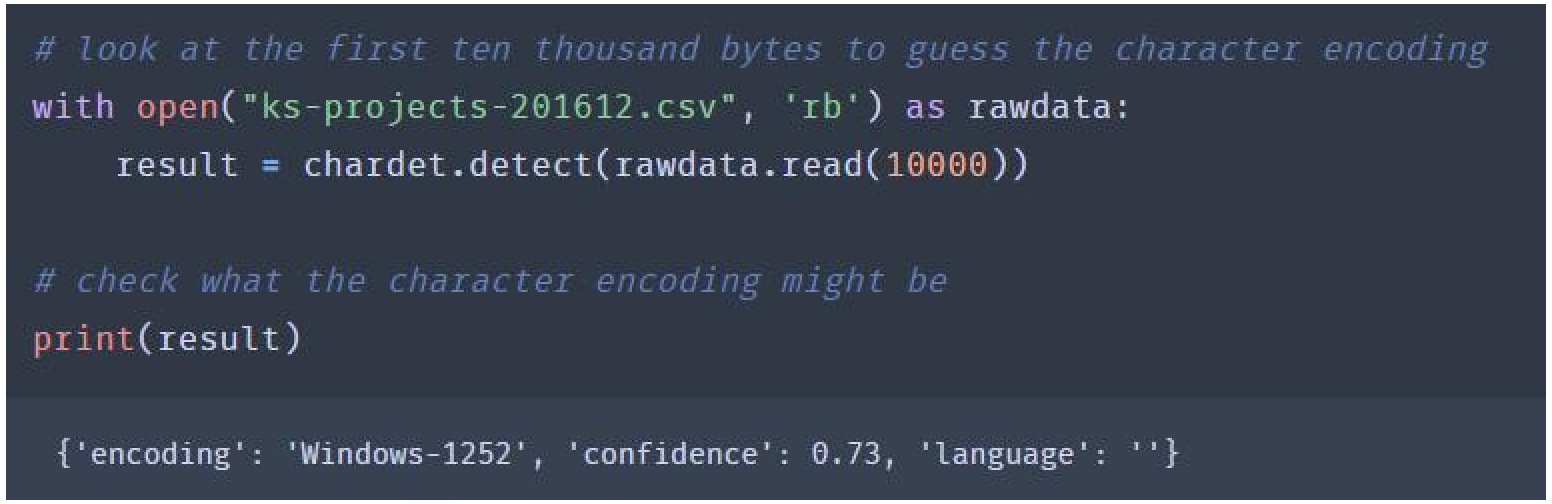
```
# start with a string
before = "This is the euro symbol: €"

# encode it to a different encoding, replacing characters that raise errors
after = before.encode("ascii", errors = "replace")

# convert it back to utf-8
print(after.decode("ascii"))

# We've lost the original underlying byte string! It's been
# replaced with the underlying byte string for the unknown character :(
```

```
This is the euro symbol: ?
```

https://docs.python.org/zh-cn/3/library/codecs.html#standard-encodings

Character Encoding

One way to find out which character encoding your file contains is by utilizing the python chardet function.



```python
# look at the first ten thousand bytes to guess the character encoding
with open("ks-projects-201612.csv", 'rb') as rawdata:
    result = chardet.detect(rawdata.read(10000))

# check what the character encoding might be
print(result)
```

```
{'encoding': 'Windows-1252', 'confidence': 0.73, 'language': ''}
```

Now we have our initial guess to how to correctly decode the file!

Character Encoding – Lab Exercise

Successfully read in these two data:

Data 1: ks-projects-201801.csv

ks-project
s-201801.
csv

Data 2: PoliceKillingsUS.csv

PoliceKillin
gsUS.csv

Duplication – Lab Exercise

This one is relatively easy, just use the pandas default **drop_duplicates()** function.

Now, calculate the percentage of data retained after deduplication:

Data to use: Reviews.csv

Reviews.cs
v

Hint: use len(your_dataframe) to get the length.

Class Work

As explained above in the 3 sessions.

No extra challenge this week, but you are more than welcome to play around with the given datasets.

Starting next week we will begin model training ☺

Homework 1

Homework 1 is also up!

Make sure you check out Blackboard and start working on it!

End of Lab 4