Key Term

Comparison operators - Operators like ==, !=, >, < etc. used to compare values and return Boolean true/false results. Important for filtering data.

Boolean operators - Operators like &, |, ~ used to combine comparison expressions and return true/false results. Important for complex filter logic.

Filters - Boolean indexed arrays that allow selecting subsets of data meeting comparison criteria. Created using comparison/Boolean operators.

loc accessor - Accessor used to select/filter data from a DataFrame by label or Boolean array.

isin() method - Method to filter data using a list of values to check for set membership. Useful alternative to repeated equality checks.

```python
import pandas as pd

df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})

# Comparison operators
print(df[df['A'] == 2]) # Simple filter

# Boolean operators
filter = (df['A'] > 1) & (df['B'] == 6)
print(df[filter]) # Complex filter

# Filters with loc accessor
filter = df['A'] > 1
print(df.loc[filter])

# isin() method
values = [2, 5]
filter = df['A'].isin(values)
print(df[filter])
```

```
   A  B
1  2  5
   A  B
2  3  6
   A  B
1  2  5
2  3  6
   A  B
1  2  5
```

Manipulating Pandas DataFrame

Manipulating DataFrames

```
[40]: import pandas as pd
      data = {'first': ['Carl', 'Francis', 'Sam'],
              'last':  ['Po',   'Nyguen',  'Smith'],
              'age':   ['32',   '45',      '22']}
      clients = pd.DataFrame(data)
      clients
```

[40]:

	first	last	age
0	Carl	Po	32
1	Francis	Nyguen	45
2	Sam	Smith	22

Rename

Renaming

```
[41]: clients.rename(columns={'first':'First Name'})
```

current column name, new column name

↳ uses dictionary to map

[41]:

	First Name	last	age
0	Carl	Po	32
1	Francis	Nyguen	45
2	Sam	Smith	22

new column name

also use with rows

```
[42]: clients.rename(index={0:'a', 1:'b', 2:'c'})
```

[42]:

	first	last	age
a	Carl	Po	32
b	Francis	Nyguen	45
c	Sam	Smith	22

```
[43]: clients
```

it doesn't change because we didn't capture the result of the rename function

[43]:

	first	last	age
0	Carl	Po	32
1	Francis	Nyguen	45
2	Sam	Smith	22

put the optional argument 'inplace'

```
[44]: clients.rename(index={0:'a', 1:'b', 2:'c'}, inplace=True)
[45]: clients
```

[45]:

	first	last	age
a	Carl	Po	32
b	Francis	Nyguen	45
c	Sam	Smith	22

it has renamed indexes

```
[46]: clients.reset_index(inplace=True)
[47]: clients
```

set to the original data frame

[47]:

	index	first	last	age
0	a	Carl	Po	32
1	b	Francis	Nyguen	45
2	c	Sam	Smith	22

Drop

Dropping

drop column

```
[48]: clients.drop(columns='first')
```

[48]:

	index	last	age
0	a	Po	32
1	b	Nyguen	45
2	c	Smith	22

drop row

```
[49]: clients.drop(index=0)
```

[49]:

	index	first	last	age
1	b	Francis	Nyguen	45
2	c	Sam	Smith	22

set

Set Type

```
[50]: clients.age
```

↳ original types are strings

```
[50]: 0    32
      1    45
      2    22
      Name: age, dtype: object
```

```
[51]: clients.age.astype(int)
```

```
[51]: 0    32
      1    45
      2    22
      Name: age, dtype: int64
```

set them to int

Updating Pandas DataFrame Data

Updating DataFrame Data

```
[43]: import pandas as pd
      data = {'first': ['Carl', 'Francis', 'Sam'],
              'last':  ['Po',   'Nyguen',  'Smith'],
              'age':   ['32',   '45',      '22'],
              'CH_count':[12,  14, 39]}
      clients = pd.DataFrame(data)
      clients
```

[43]:

	first	last	age	CH_count
0	Carl	Po	32	12
1	Francis	Nyguen	45	14
2	Sam	Smith	22	39

Adding Rows

Notice! keys in Dic. are the same as the column → Dictionary with new data

```
[44]: new_data = {'first':['Sue', 'Boya'],
                  'last':['Rankler', 'Maple'],
                  'age':[93, 12],
                  'CH_count':[22, 1]}
      new_clients = pd.DataFrame(new_data)
      new_clients
```

create to new variable

[44]:

	first	last	age	CH_count
0	Sue	Rankler	93	22
1	Boya	Maple	12	1

then append to existing data frame

```
[45]: clients.append(new_client)
```

[45]:

	first	last	age	CH_count
0	Carl	Po	32	12.0
1	Francis	Nyguen	45	14.0
2	Sam	Smith	22	39.0
0	Sue	Rankler	93	NaN

We can use reset method from previous lesson

Setting specific value

specific index con → `clients.loc[1, 'first'] = 'Frankie'` ← specific column name, new value

```
[46]: clients.loc[1, 'first'] = 'Frankie'
      clients
```

[46]: updated

	first	last	age	CH_count
0	Carl	Po	32	12
1	Frankie	Nyguen	45	14
2	Sam	Smith	22	39

we can also specify a range for multiple updating cells

```
[47]: clients.loc[0:1, 'CH_count'] = -1
      clients
```

0:1 → row; 'CH_count' → column

[47]:

	first	last	age	CH_count
0	Carl	Po	32	-1
1	Frankie	Nyguen	45	-1
2	Sam	Smith	22	39

Math Operations

add in every rows in the specific column

```
[48]: clients.CH_count + 1
```

```
[48]: 0     0
      1     0
      2    40
      Name: CH_count, dtype: int64
```

```
[49]: clients
```

still has the original value

[49]:

	first	last	age	CH_count
0	Carl	Po	32	-1
1	Frankie	Nyguen	45	-1
2	Sam	Smith	22	39

use equality operation

```
[50]: clients.CH_count -= 3
      clients
```

original has changed

[50]:

	first	last	age	CH_count
0	Carl	Po	32	-4
1	Frankie	Nyguen	45	-4
2	Sam	Smith	22	36

Replace

to replace specific values

Value we wish to replace → `clients.replace(-4, 0)` ← to replace it with it

```
[51]: clients.replace(-4, 0)
```

[51]:

	first	last	age	CH_count
0	Carl	Po	32	0
1	Frankie	Nyguen	45	0
2	Sam	Smith	22	36

It will searches the whole data frame and replace them with the second argument

Applying Functions in a Pandas DataFrame

DataFrame Apply

```
[7]: import pandas as pd
     data = { 'even': range(20,0,-2),
              'odd':  range(1,21,2)
            }
     df = pd.DataFrame(data)
     df
```

[7]:

	even	odd
0	20	1
1	18	3
2	16	5
3	14	7
4	12	9
5	10	11
6	8	13
7	6	15
8	4	17
9	2	19

built in function

0+1+2+...+9

```
[8]: sum(range(10))
```

[8]: 45

Apply

apply to df

```
[9]: df.apply(sum)
```

applies the function in each column

[9]:
```
even    110
odd     100
dtype: int64
```

```
[10]: df
```

[10]:

	even	odd
0	20	1
1	18	3
2	16	5
3	14	7
4	12	9
5	10	11
6	8	13
7	6	15
8	4	17
9	2	19

we can specify axis

```
[11]: df.apply(sum, axis=1)
```

[11]:
```
0    21
1    21
2    21
3    21
4    21
5    21
6    21
7    21
8    21
9    21
dtype: int64
```

← it added up the value from all the columns in each row

we can use 'apply' with built-in or third-party or define our own functions

Define Column Function

```
[12]: def hundred_plus(col):
          if sum(col) > 100:
              return "Greater than 100"
          return "Not greater than 100"

[13]: df.apply(hundred_plus)
```

return to each column seperately

[13]:
```
even        Greater than 100
odd     Not greater than 100
dtype: object
```

Define Row Function

the row has function for each column

```
[14]: def label(row):
          if row['even'] % 3 == 0:
              return True
          elif row['odd'] % 3 == 0:
              return True
          return False

[15]: df.apply(label, axis=1)
```

[15]:
```
0    False
1     True
2    False
3    False
4     True
5    False
6    False
7     True
8    False
9    False
dtype: bool
```

Apply to Column

```
[19]: def div_three(row):
          if row %3 == 0:
              return 'Divisible by 3'
          return 'Not divisible by 3'

[20]: df.even.apply(div_three)
```

column

[20]:
```
0    Not divisible by 3
1        Divisible by 3
2    Not divisible by 3
3    Not divisible by 3
4        Divisible by 3
5    Not divisible by 3
6    Not divisible by 3
7        Divisible by 3
8    Not divisible by 3
9    Not divisible by 3
Name: even, dtype: object
```

Expanding Results

```
[16]: def ret_list(row):
          ret_val = [False, False]
          if row['even'] > 6:
              ret_val[0] = True

          if row['odd'] > 6:
              ret_val[1] = True

          return ret_val

[17]: df.apply(ret_list, axis=1)
```

[17]:
```
0     [True, False]
1     [True, False]
2     [True, False]
3      [True, True]
4      [True, True]
5      [True, True]
6     [False, True]
7     [False, True]
8     [False, True]
9     [False, True]
dtype: object
```

```
[18]: df.apply(ret_list, axis=1, result_type='expand')
```

[18]:

	0	1
0	True	False
1	True	False
2	True	False
3	True	True
4	True	True
5	True	True
6	True	True
7	False	True
8	False	True
9	False	True