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 &, |, \sim used to combine comparison expressions and return true/false results. Important for complex filter logic.

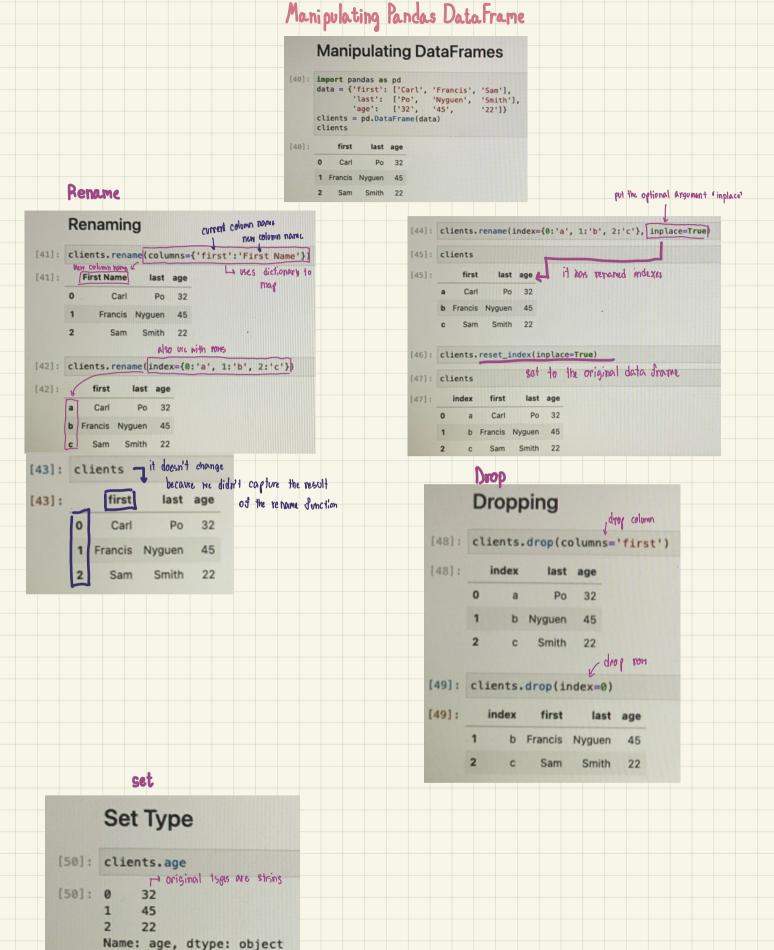
Filters - Boolean indexed arrays that allow selecting subsets of data meeting comparison criteria. Created using comparison/Boolean operators.

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.

```
1
     import pandas as pd
 2
 3
     df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
4
 5
     # Comparison operators
6
     print(df[df['A'] == 2]) # Simple filter
7
8
     # Boolean operators
     filter = (df['A'] > 1) & (df['B'] == 6)
9
10
     print(df[filter]) # Complex filter
11
12
     # Filters with loc accessor
     filter = df['A'] > 1
13
14
     print(df.loc[filter])
15
     # isin() method
16
17
     values = [2, 5]
     filter = df['A'].isin(values)
18
19
     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
1 2 5
```



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

Name: age, dtype: int64

set them to

32

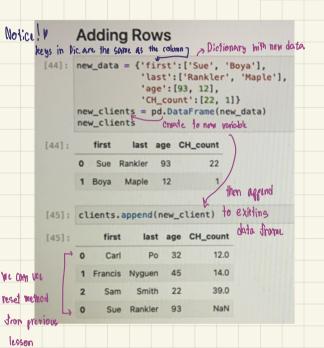
22

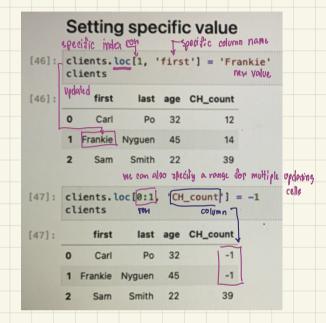
[51]: 0

Updating Pandas DataFrame Data

Updating DataFrame Data

[43]:		first	last	age	CH_count
	0	Carl	Po	32	12
	1	Francis	Nyguen	45	14
	-	Com	Cmith	22	30





Replace to replace speafic values

Smith

5 to replace it with it

0 4

0

36

last age CH_count

Value we mish to replace 7

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

first

Carl

Sam

Frankie Nyguen

[51]:

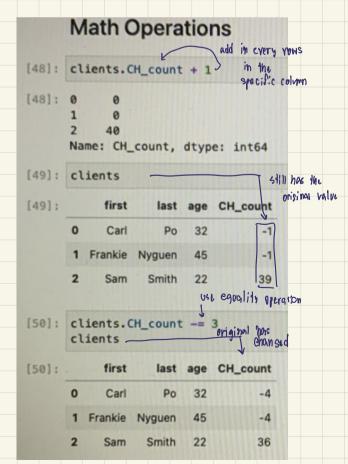
it will searches

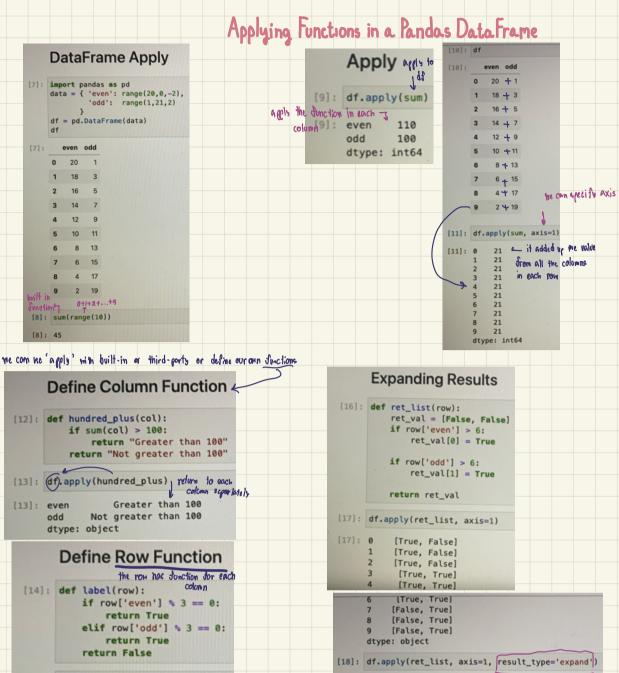
the whole data frame

Second orgament

and replace them

mits the





0 1 0 True False

1 True False

True True

True True

6 True True

8 False True

```
[14]: def label(row):
    if row['even'] % 3 == 0:
        return True
    elif row['odd'] % 3 == 0:
        return True
    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)
        Not divisible by 3
             Divisible by 3
          Not divisible by 3
         Not divisible by 3
            Divisible by 3
          Not divisible by 3
         Not divisible by 3
            Divisible by 3
         Not divisible by 3
         Not divisible by 3
     Name: even, dtype: object
```