Module 4: Working with Text Data

Preprocessing our data is very important.

We may encounter data that is not clean. As we can see in DataFrame below:

- First name and last name are separated by a comma.
- Employee Annual Salary has a dollar ($) sign, which means we cannot do mathematical operation on it.

In [1]:

```
import pandas as pd
df = pd.read_csv("data/chicago.csv")
df.head()
```

Out[1]:

	Name	Position Title	Department	Employee Annual Salary
0	AARON, ELVIA J	WATER RATE TAKER	WATER MGMNT	$90744.00
1	AARON, JEFFERY M	POLICE OFFICER	POLICE	$84450.00
2	AARON, KARINA	POLICE OFFICER	POLICE	$84450.00
3	AARON, KIMBERLEI R	CHIEF CONTRACT EXPEDITER	GENERAL SERVICES	$89880.00
4	ABAD JR, VICENTE M	CIVIL ENGINEER IV	WATER MGMNT	$106836.00

In [2]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32063 entries, 0 to 32062
Data columns (total 4 columns):
Name                     32062 non-null object
Position Title           32062 non-null object
Department               32062 non-null object
Employee Annual Salary   32062 non-null object
dtypes: object(4)
memory usage: 1002.0+ KB
```

In Department column, we have 35 categories. It is much better to change the column type to **category** as we have 32,063 rows of data. Rather than having 32,063 strings, we can classify them into 35 categories.

You can see that the memory usage decreases from 1002KB to 784.4KB. This may speed up our operation so much more.

In [3]:

```
df["Department"].nunique()
```

Out[3]:

```
35
```

In [4]:

```
df["Department"] = df["Department"].astype("category")
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32063 entries, 0 to 32062
Data columns (total 4 columns):
Name                      32062 non-null object
Position Title            32062 non-null object
Department                32062 non-null category
Employee Annual Salary    32062 non-null object
dtypes: category(1), object(3)
memory usage: 784.4+ KB
```

Remove null values

In [6]:

```
df.tail()
```

Out[6]:

	Name	Position Title	Department	Employee Annual Salary
32058	ZYGOWICZ, PETER J	POLICE OFFICER	POLICE	$87384.00
32059	ZYMANTAS, MARK E	POLICE OFFICER	POLICE	$84450.00
32060	ZYRKOWSKI, CARLO E	POLICE OFFICER	POLICE	$87384.00
32061	ZYSKOWSKI, DARIUSZ	CHIEF DATA BASE ANALYST	DoIT	$113664.00
32062	NaN	NaN	NaN	NaN

In [7]:

```
df.dropna(how="all", inplace=True)
df.tail()
```

Out[7]:

	Name	Position Title	Department	Employee Annual Salary
32057	ZYGADLO, MICHAEL J	FRM OF MACHINISTS - AUTOMOTIVE	GENERAL SERVICES	$99528.00
32058	ZYGOWICZ, PETER J	POLICE OFFICER	POLICE	$87384.00
32059	ZYMANTAS, MARK E	POLICE OFFICER	POLICE	$84450.00
32060	ZYRKOWSKI, CARLO E	POLICE OFFICER	POLICE	$87384.00
32061	ZYSKOWSKI, DARIUSZ	CHIEF DATA BASE ANALYST	DoIT	$113664.00

Python built-in function : lower(), upper(), title() , len()

- lower() : returns the string in lowercase.
- upper() : returns a copy of the string in uppercase.
- title() : convert the first character in each word to uppercase and remaining characters to lowercase in string
- len() : calculate the length of string/ list/ etc. (In string, space also counts as a character)

In [8]:

```
"HELLO WORLD".lower()
```

Out[8]:

```
'hello world'
```

In [9]:

```
"hello world".upper()
```

Out[9]:

```
'HELLO WORLD'
```

In [10]:

```
"hello world".title()
```

Out[10]:

```
'Hello World'
```

In [11]:

```
len("HELLO WORLD")
```

Out[11]:

```
11
```

These functions allow us to change the format in Name column into something more readable.

In [12]:

```
df["Name"] = df["Name"].str.title()
df.head()
```

Out[12]:

	Name	Position Title	Department	Employee Annual Salary
0	Aaron, Elvia J	WATER RATE TAKER	WATER MGMNT	$90744.00
1	Aaron, Jeffery M	POLICE OFFICER	POLICE	$84450.00
2	Aaron, Karina	POLICE OFFICER	POLICE	$84450.00
3	Aaron, Kimberlei R	CHIEF CONTRACT EXPEDITER	GENERAL SERVICES	$89880.00
4	Abad Jr, Vicente M	CIVIL ENGINEER IV	WATER MGMNT	$106836.00

The same goes for Position Title column and Department column.

In [13]:

```
df["Position Title"] = df["Position Title"].str.title()
df.head()
```

Out[13]:

	Name	Position Title	Department	Employee Annual Salary
0	Aaron, Elvia J	Water Rate Taker	WATER MGMNT	$90744.00
1	Aaron, Jeffery M	Police Officer	POLICE	$84450.00
2	Aaron, Karina	Police Officer	POLICE	$84450.00
3	Aaron, Kimberlei R	Chief Contract Expediter	GENERAL SERVICES	$89880.00
4	Abad Jr, Vicente M	Civil Engineer Iv	WATER MGMNT	$106836.00

In [14]:

```
df["Department"] = df["Department"].str.title()
df.head()
```

Out[14]:

	Name	Position Title	Department	Employee Annual Salary
0	Aaron, Elvia J	Water Rate Taker	Water Mgmnt	$90744.00
1	Aaron, Jeffery M	Police Officer	Police	$84450.00
2	Aaron, Karina	Police Officer	Police	$84450.00
3	Aaron, Kimberlei R	Chief Contract Expediter	General Services	$89880.00
4	Abad Jr, Vicente M	Civil Engineer Iv	Water Mgmnt	$106836.00

replace() method

The function is used to replace a string, regex, list, dictionary, series, or number.

In the Department column, "Management“ is shortened to "Mgmnt“ while in the Employee Annual Salary column, there is dollar sign ($) in each values. These data can be cleaned using replace() method.

Firstly, we have to understand how the method works. The first argument is the pattern that we want to replace. The second argument is what we want it to be replaced with.

In [15]:

```
"Hello World".replace("l", "!")
```

Out[15]:

```
'He!!o Wor!d'
```

In [16]:

```
df["Department"] = df["Department"].str.replace("Mgmnt", "Management")
df.head(3)
```

Out[16]:

	Name	Position Title	Department	Employee Annual Salary
0	Aaron, Elvia J	Water Rate Taker	Water Management	$90744.00
1	Aaron, Jeffery M	Police Officer	Police	$84450.00
2	Aaron, Karina	Police Officer	Police	$84450.00

Let's remove the dollar sign in Employee Annual Salary column as well and change the data type to float.

In [17]:

```
df["Employee Annual Salary"] = df["Employee Annual Salary"].str.replace("$", "").astype(flo
df.head(3)
```

Out[17]:

	Name	Position Title	Department	Employee Annual Salary
0	Aaron, Elvia J	Water Rate Taker	Water Management	90744.0
1	Aaron, Jeffery M	Police Officer	Police	84450.0
2	Aaron, Karina	Police Officer	Police	84450.0

Now, we can do mathematical operations on that column. We can get the average annual salary which is 80,204.

In [18]:

```
df["Employee Annual Salary"].mean()
```

Out[18]:

```
80204.178633899
```

.contains() , endswith() startswith() methods

These methods return Boolean values, either True or False.

Let see how many jobs that are related to water. Out of 32,061, only 111 jobs are related to water.

In [19]:

```
mask = df["Position Title"].str.contains("Water")
df[mask]
```

Out[19]:

	Name	Position Title	Department	Employee Annual Salary
0	Aaron, Elvia J	Water Rate Taker	Water Management	90744.0
554	Aluise, Vincent G	Foreman Of Water Pipe Construction	Water Management	102440.0
671	Ander, Perry A	Water Chemist Ii	Water Management	82044.0
685	Anderson, Andrew J	District Superintendent Of Water Distribution	Water Management	109272.0
702	Anderson, Donald	Foreman Of Water Pipe Construction	Water Management	102440.0
1054	Ashley, Karma T	Water Chemist Ii	Water Management	82044.0
1079	Atkins, Joanna M	Water Chemist Ii	Water Management	82044.0
1181	Azeem, Mohammed A	Water Chemist Ii	Water Management	53172.0
1285	Bajic, John A	Water Meter Machinist	Water Management	82576.0
2400	Bolton, Brian E	Water Rate Taker	Water Management	78948.0
2586	Boyce, Adner L	Water Chemist Ii	Water Management	82044.0
2745	Brandys, Daniel	Water Chemist Ii	Water Management	53172.0
3143	Brown, Sharon L	Water Rate Taker	Water Management	82728.0
3246	Buchanan, Anthony L	Water Chemist Ii	Water Management	66780.0
3456	Burt, Carl S	Water Rate Taker	Water Management	90744.0
3626	Cage, Ophelia	Water Rate Taker	Water Management	90744.0
3663	Calderone, Michael P	Foreman Of Water Pipe Construction	Water Management	102440.0
4889	Clark, Mark A	Water Meter Machinist	Water Management	82576.0
5506	Cooper Jr, Eddie	Water Chemist Ii	Water Management	82044.0
6337	Davenport, Clarence	Water Meter Machinist	Water Management	82576.0
6421	Davis Iii, Wallace	Gen Supt Of Water Management	Water Management	119208.0

	Name	Position Title	Department	Employee Annual Salary
6564	Decaluwe, Stanley J	Foreman Of Water Pipe Construction	Water Management	102440.0
6954	Diaz, Oscar A	Water Rate Taker	Water Management	90744.0
7356	Dowdy, Timothy J	Foreman Of Water Pipe Construction	Water Management	102440.0
7463	Driver, Claude A	Foreman Of Water Pipe Construction	Water Management	102440.0
7641	Durant, Patricia B	Water Rate Taker	Water Management	90744.0
7650	Durham, Patrick L	Water Meter Machinist	Water Management	82576.0
8092	Espinosa, Rodolfo	Water Rate Taker	Water Management	78948.0
8660	Fitzpatrick, Charles W	Foreman Of Water Pipe Construction	Water Management	102440.0
9325	Gaither, Rasheda K	Chief Water Chemist	Water Management	96840.0
...	...	...	...	...
23187	Quintanilla, Edwin	Water Chemist Ii	Water Management	70152.0
24003	Rios, Francisco	Water Rate Taker	Water Management	78948.0
24178	Roberts, Akilah	Water Chemist Ii	Water Management	66780.0
24266	Robinson, Jerry	Water Meter Assessor	Water Management	90744.0
24422	Rodriguez, Edward L	Water Meter Assessor	Water Management	78948.0
24501	Rodriguez, Marco A	Water Rate Taker	Water Management	90744.0
24945	Rudmin, Kenneth D	Water Quality Inspector I/C	Water Management	65172.0
25055	Russnak, Thomas W	Water Meter Assessor	Water Management	90744.0
25233	Saleh, Hassen	Dir Of Water Purification Laboratories	Water Management	109008.0
25260	Salinas, Eduardo S	Engineer Of Water Purification	Water Management	118656.0
26372	Siddiqui, Abdurrazzaq	Water Chemist Iii	Water Management	89676.0
26519	Sims, Demetrius	Water Rate Taker	Water Management	90744.0
26625	Skowron, John	Water Rate Taker	Water Management	90744.0
26871	Smith, Nancy A	Water Rate Taker	Water Management	90744.0
27015	Sojka, Jeffrey A	Water Meter Assessor	Water Management	90744.0

	Name	Position Title	Department	Employee Annual Salary
27368	Stang, Christopher	Foreman Of Water Pipe Construction	Water Management	102440.0
27537	Stevenson Jr, Walter	Foreman Of Water Pipe Construction	Water Management	102440.0
27635	Stokes, Carolyn M	Water Chemist Ii	Water Management	82044.0
27755	Stroud Jr, James E	Water Meter Machinist	Water Management	82576.0
28185	Tate, Gary	Water Rate Taker	Water Management	82728.0
28443	Thomas, Howard	Water Rate Taker	Water Management	78948.0
28574	Threatt, Denise R	Water Quality Inspector	Water Management	62004.0
28602	Tignor, Darryl B	Water Rate Taker	Water Management	78948.0
28955	Travis Cook, Leslie R	Water Rate Taker	Water Management	78948.0
29584	Velazquez, John	Water Rate Taker	Water Management	78948.0
29669	Verma, Anupam	Managing Engineer - Water Management	Water Management	111192.0
30239	Washington, Joseph	Water Chemist Iii	Water Management	89676.0
30544	West, Thomas R	Gen Supt Of Water Management	Water Management	115704.0
30991	Williams, Matthew	Foreman Of Water Pipe Construction	Water Management	102440.0
31405	Woodridge, Robert L	Foreman Of Water Pipe Construction	Water Management	102440.0

111 rows × 4 columns

We can identify specialists by using **endswith()** method. Here, we have 100 specialists from different departments

In [20]:

```
mask = df["Position Title"].str.endswith("Specialist")
df[mask]
```

Out[20]:

	Name	Position Title	Department	Employee Annual Salary
308	Alarcon, Luis J	Loan Processing Specialist	Community Development	81948.00
422	Allain, Carolyn	Senior Telecommunications Specialist	Doit	89880.00
705	Anderson, Edward M	Sr Procurement Specialist	Procurement	91476.00
1163	Ayala Jr, Juan	Field Sanitation Specialist	Streets & San	78948.00
1558	Barrett, Barbara J	Technical Training Specialist	Police	94200.00
1869	Beltran, Mauricio	Procurement Specialist	Procurement	79596.00
2095	Biggane, Thomas M	Security Specialist	City Council	52000.08
2319	Bobba, Mahita	Grants Research Specialist	Health	84924.00
2519	Boston, Nathaniel K	Field Sanitation Specialist	Streets & San	75384.00
2678	Bradley, Jeena	Grants Research Specialist	Health	97812.00
4109	Carter, Alice	Safety Specialist	Water Management	86580.00
4479	Chan, Joseph	Sr Procurement Specialist	Procurement	83340.00
5043	Coco, Michelle T	Investigator Specialist	Fire	91476.00
5307	Condon, Kristin	Hr Records Specialist	Human Resources	48852.00
5633	Cortes, Jezieel T	Sr Procurement Specialist	Procurement	68556.00
5921	Crump, Carolyn R	Field Sanitation Specialist	Streets & San	71976.00
6136	Czahor, Patrick J	Field Sanitation Specialist	Streets & San	75384.00
6571	Decker, Grace A	Hr Records Specialist	Human Resources	54108.00
6599	Degard, Christopher L	Sr Procurement Specialist	Procurement	68556.00
7317	Dotson, William L	Procurement Specialist	Procurement	75960.00
7387	Doyle, George D	Safety Specialist	Aviation	68028.00
7423	Drain, Josalyn T	Field Sanitation Specialist	Streets & San	75384.00
8753	Flores, Archibaldo	Safety Specialist	Streets & San	56508.00
8760	Flores, Eduardo S	Safety Specialist	Aviation	68688.00
9122	Freelon, Lisa L	Sr Procurement Specialist	Procurement	95820.00
9366	Gallagher, John E	Safety Specialist	Water Management	78948.00
9484	Gapinski, Frank J	Senior Telecommunications Specialist	Doit	94200.00

	Name	Position Title	Department	Employee Annual Salary
9562	Garcia, Joseph E	Rehabilitation Construction Specialist	Community Development	81948.00
10374	Gonzalez, Lylianis	Procurement Specialist	Procurement	75960.00
10581	Graham, Kendra D	Safety Specialist	Aviation	68688.00
...	...	...	...	...
23533	Reckinger, Jeffry D	Grants Research Specialist	Community Development	97812.00
23952	Riley, Altha S	Sr Procurement Specialist	Procurement	83340.00
24156	Roa, Edward J	Senior Telecommunications Specialist	Doit	81948.00
24525	Rodriguez, Ralph	Supervising Rehabilitation Construction Specia...	Community Development	79596.00
24645	Rollins, Delia	Grants Research Specialist	Police	97812.00
25054	Russ, Johanna M	Senior Archival Specialist	Public Library	63528.00
25445	Sanders, Reginald	Audio-Visual Specialist	Public Library	74676.00
25720	Schauf, Ralph G	Security Specialist	City Council	52000.08
26863	Smith, Michael L	Procurement Specialist	Procurement	75960.00
26982	Snow, Michael J	Field Sanitation Specialist	Streets & San	78948.00
27306	Sriver, Nikki K	Pr Telecommunications Specialist	Doit	98664.00
27576	Stewart Jr, John	Sr Procurement Specialist	Procurement	83340.00
27593	Stewart, Stacy P	Sr Procurement Specialist	Procurement	83340.00
28077	Szymusiak, Diane	Procurement Specialist	Procurement	61584.00
28353	Terrell, Lynnette	Sr Procurement Specialist	Procurement	87324.00
28477	Thomas, Michael E	Rehabilitation Construction Specialist	Community Development	89880.00
28578	Thrower, Derwin K	Safety Specialist	Water Management	90744.00
29502	Vaughn Jr, Joseph T	Rehabilitation Construction Specialist	Community Development	81948.00
30060	Walsh, Morag	Senior Archival Specialist	Public Library	82044.00
30156	Ward, Sonji S	Sr Procurement Specialist	Procurement	95820.00
30287	Watkins, Gary G	Sr Labor Relations Specialist	Law	82668.00
30914	Williams, Jana Y	Procurement Specialist	Procurement	58800.00
31118	Willis, Kirk J	Crimes Surveillance Specialist	Oemc	19676.80
31296	Wisniewski, Jerry V	Field Sanitation Specialist	Streets & San	75384.00
31320	Wittman, Elisabeth C	Archival Specialist	City Clerk	70152.00
31653	Yeksigian, John P	Rehabilitation Construction Specialist	Community Development	85764.00
31667	Yoder, Teresa G	Archival Specialist	Public Library	74304.00
31688	Youngbloom, Laurence G	Crimes Surveillance Specialist	Oemc	19676.80

	Name	Position Title	Department	Employee Annual Salary
31717	Young, Kimberly M	Sr Procurement Specialist	Procurement	68556.00
31837	Zapata, Hugo	Sr Procurement Specialist	Procurement	87324.00

100 rows × 4 columns

We can also do identification by using **startswith()** method.

In [21]:

```
mask = df["Position Title"].str.startswith("Water")
df[mask]
```

Out[21]:

	Name	Position Title	Department	Employee Annual Salary
0	Aaron, Elvia J	Water Rate Taker	Water Management	90744.0
671	Ander, Perry A	Water Chemist Ii	Water Management	82044.0
1054	Ashley, Karma T	Water Chemist Ii	Water Management	82044.0
1079	Atkins, Joanna M	Water Chemist Ii	Water Management	82044.0
1181	Azeem, Mohammed A	Water Chemist Ii	Water Management	53172.0
1285	Bajic, John A	Water Meter Machinist	Water Management	82576.0
2400	Bolton, Brian E	Water Rate Taker	Water Management	78948.0
2586	Boyce, Adner L	Water Chemist Ii	Water Management	82044.0
2745	Brandys, Daniel	Water Chemist Ii	Water Management	53172.0
3143	Brown, Sharon L	Water Rate Taker	Water Management	82728.0
3246	Buchanan, Anthony L	Water Chemist Ii	Water Management	66780.0
3456	Burt, Carl S	Water Rate Taker	Water Management	90744.0
3626	Cage, Ophelia	Water Rate Taker	Water Management	90744.0
4889	Clark, Mark A	Water Meter Machinist	Water Management	82576.0
5506	Cooper Jr, Eddie	Water Chemist Ii	Water Management	82044.0
6337	Davenport, Clarence	Water Meter Machinist	Water Management	82576.0
6954	Diaz, Oscar A	Water Rate Taker	Water Management	90744.0
7641	Durant, Patricia B	Water Rate Taker	Water Management	90744.0
7650	Durham, Patrick L	Water Meter Machinist	Water Management	82576.0
8092	Espinosa, Rodolfo	Water Rate Taker	Water Management	78948.0
10170	Godinez, Julio A	Water Meter Machinist	Water Management	82576.0

	Name	Position Title	Department	Employee Annual Salary
10457	Gordon, Donald R	Water Meter Machinist	Water Management	82576.0
10707	Green, John F	Water Meter Machinist	Water Management	82576.0
11857	Hedrick, Lee A	Water Rate Taker	Water Management	78948.0
12408	Hjelmgren, James A	Water Meter Machinist	Water Management	82576.0
12487	Ho, Kwok L	Water Conservation Engineer	Water Management	104748.0
12594	Ho, Nam C	Water Chemist Ii	Water Management	53172.0
12781	Howell Jr, George B	Water Chemist Ii	Water Management	66780.0
13361	Jacob, Lovely	Water Chemist Iii	Water Management	89676.0
13476	Jansky, Thomas A	Water Meter Machinist	Water Management	82576.0
...	...	...	...	...
20936	O Connor Jr, Michael J	Water Meter Machinist	Water Management	82576.0
20962	Odibo, Grace E	Water Chemist Ii	Water Management	82044.0
21078	Olbera Sr, Martin R	Water Meter Machinist	Water Management	82576.0
21212	O Neal, Rozella	Water Meter Assessor	Water Management	86580.0
21788	Parker, Terry	Water Chemist Ii	Water Management	70152.0
22384	Pfannkuche, Joseph W	Water Meter Machinist	Water Management	82576.0
22916	Price, Johnny L	Water Meter Machinist	Water Management	82576.0
23102	Putz, Andrea R	Water Quality Manager	Water Management	114552.0
23187	Quintanilla, Edwin	Water Chemist Ii	Water Management	70152.0
24003	Rios, Francisco	Water Rate Taker	Water Management	78948.0
24178	Roberts, Akilah	Water Chemist Ii	Water Management	66780.0
24266	Robinson, Jerry	Water Meter Assessor	Water Management	90744.0
24422	Rodriguez, Edward L	Water Meter Assessor	Water Management	78948.0
24501	Rodriguez, Marco A	Water Rate Taker	Water Management	90744.0
24945	Rudmin, Kenneth D	Water Quality Inspector I/C	Water Management	65172.0

	Name	Position Title	Department	Employee Annual Salary
25055	Russnak, Thomas W	Water Meter Assessor	Water Management	90744.0
26372	Siddiqui, Abdurrazzaq	Water Chemist Iii	Water Management	89676.0
26519	Sims, Demetrius	Water Rate Taker	Water Management	90744.0
26625	Skowron, John	Water Rate Taker	Water Management	90744.0
26871	Smith, Nancy A	Water Rate Taker	Water Management	90744.0
27015	Sojka, Jeffrey A	Water Meter Assessor	Water Management	90744.0
27635	Stokes, Carolyn M	Water Chemist Ii	Water Management	82044.0
27755	Stroud Jr, James E	Water Meter Machinist	Water Management	82576.0
28185	Tate, Gary	Water Rate Taker	Water Management	82728.0
28443	Thomas, Howard	Water Rate Taker	Water Management	78948.0
28574	Threatt, Denise R	Water Quality Inspector	Water Management	62004.0
28602	Tignor, Darryl B	Water Rate Taker	Water Management	78948.0
28955	Travis Cook, Leslie R	Water Rate Taker	Water Management	78948.0
29584	Velazquez, John	Water Rate Taker	Water Management	78948.0
30239	Washington, Joseph	Water Chemist Iii	Water Management	89676.0

75 rows × 4 columns

strip() , lstrip() , rstrip()

Strip means removing white space at the start or end of a string.

lstrip() stands for left strip. It will only remove white space on the left. On the other hand, **rstrip()** will only remove white space on the right.

These methods are very handy to clean unneeded spaces in the data. Sometimes users tend to make mistakes on input.

In [22]:

```
"         Hello Word   ".strip()
```

Out[22]:

```
'Hello Word'
```

In [23]:

```
"         Hello Word   ".lstrip()
```

Out[23]:

```
'Hello Word   '
```

In [24]:

```
"         Hello Word   ".rstrip()
```

Out[24]:

```
'         Hello Word'
```

In [25]:

```
df["Name"] = df["Name"].str.strip()
df["Name"].head()
```

Out[25]:

```
0        Aaron,  Elvia J
1      Aaron,  Jeffery M
2         Aaron,  Karina
3    Aaron,  Kimberlei R
4    Abad Jr,  Vicente M
Name: Name, dtype: object
```

Spilit String by Character using .str.split() method

Split string by a character using **.str.split()** method. It will split strings around given separator/delimiter.

The split() method will divide the string based on the separator that we specify. In the example, the separator is a space (" ")

In [26]:

```
"My name is Ali bin Abu".split(" ")
```

Out[26]:

```
['My', 'name', 'is', 'Ali', 'bin', 'Abu']
```

We can separate the Last name and first name in Name column.

In [27]:

```
lastName =  df["Name"].str.split(",").str.get(0).str.strip()
lastName
```

Out[27]:

```
0                Aaron
1                Aaron
2                Aaron
3                Aaron
4             Abad Jr
5               Abarca
6               Abarca
7              Abascal
8               Abbasi
9           Abbatacola
10        Abbatemarco
11             Abbate
12             Abbott
13             Abbott
14         Abbruzzese
15           Abdallah
16          Abdelhadi
17         Abdellatif
18        Abdelmajeid
19       Abdollahzadeh
20        Abdul-Karim
21           Abdullah
22           Abdullah
23           Abdullah
24           Abdullah
25        Abdulsattar
26       Abdul-Shakur
27         Abdulwahab
28            Abejero
29      Abercrombie Iv
            ...
32032       Zuniga Jr
32033          Zuniga
32034          Zuniga
32035          Zuniga
32036          Zuniga
32037          Zuniga
32038            Zuno
32039           Zupan
32040        Zupancic
32041        Zurawski
32042        Zurawski
32043        Zurawski
32044        Zurawski
32045           Zurek
32046           Zurek
32047          Zurita
32048          Zvanja
32049     Zwarycz Mann
32050          Zwarycz
32051        Zwiesler
32052            Zwit
32053         Zwolfer
32054            Zych
```

```
32055            Zydek
32056          Zygadlo
32057          Zygadlo
32058         Zygowicz
32059         Zymantas
32060        Zyrkowski
32061        Zyskowski
Name: Name, Length: 32062, dtype: object
```

In [28]:

```
firstName =  df["Name"].str.split(",").str.get(1).str.strip().str.split(" ").str.get(0)
firstName.head(20)
```

Out[28]:

```
0          Elvia
1         Jeffery
2          Karina
3       Kimberlei
4         Vicente
5          Anabel
6        Emmanuel
7           Reece
8     Christopher
9          Robert
10          James
11          Terry
12          Betty
13         Lynise
14        William
15           Zaid
16      Abdalmahd
17           Aref
18           Aziz
19            Ali
Name: Name, dtype: object
```

After we have extracted the First Name and Last Name, we can add a new column into the DataFrame using the insert() method.

In [29]:

```
df.insert(0,column = "First Name" , value = firstName)
df.insert(1,column = "Last Name" ,value = lastName)
df.head()
```

Out[29]:

	First Name	Last Name	Name	Position Title	Department	Employee Annual Salary
0	Elvia	Aaron	Aaron, Elvia J	Water Rate Taker	Water Management	90744.0
1	Jeffery	Aaron	Aaron, Jeffery M	Police Officer	Police	84450.0
2	Karina	Aaron	Aaron, Karina	Police Officer	Police	84450.0
3	Kimberlei	Aaron	Aaron, Kimberlei R	Chief Contract Expediter	General Services	89880.0
4	Vicente	Abad Jr	Abad Jr, Vicente M	Civil Engineer Iv	Water Management	106836.0

Once the new columns are added, we may drop the Name column to avoid redundancy.

In [30]:

```
# df.drop(labels= "Name" , axis = 1, inplace=True)
# df.head()
```

In [31]:

```
df["First Name"].value_counts().head()
```

Out[31]:

```
Michael    1153
John        899
James       676
Robert      622
Joseph      537
Name: First Name, dtype: int64
```

In **split()**, the **expand parameter** will expand the split strings into separate columns.

In [32]:

```
df["Name"].str.split(",", expand=True).head()
```

Out[32]:

	0	1
0	Aaron	Elvia J
1	Aaron	Jeffery M
2	Aaron	Karina
3	Aaron	Kimberlei R
4	Abad Jr	Vicente M

split() has **n parameter** which we can use to decide how many occurrences to split. The example shows that we are splitting strings around a space. Since, we give argument of 1 to the n parameter, it will only split the strings around the first space it finds.

In [33]:

```
df["Position Title"].str.split(" ", expand=True, n = 1).head()
```

Out[33]:

	0	1
0	Water	Rate Taker
1	Police	Officer
2	Police	Officer
3	Chief	Contract Expediter
4	Civil	Engineer Iv

Regular Expression (ReGex) with Pandas

Regex is a powerful library to search for word(s) in the string. It can also find sequence of characters/words.

Example: Validating a password format. When we want to create a new password, there are some requirements to be fulfilled like it should include uppercase, lowercase, numbers, and unique character.

Python has a library specifically for regex.

Python ReGex Cheat Sheet : https://www.cheatography.com/mutanclan/cheat-sheets/python-regular-expression-regex/ (https://www.cheatography.com/mutanclan/cheat-sheets/python-regular-expression-regex/)

In [34]:

```
import re
```

In **re module**, there is a few common libraries used.

match() function returns a matched object if the text matches the pattern
search() function checks for a match anywhere in the string.
compile() Compiles a regular expression pattern into a regular expression object.
findall() Return all non-overlapping matches of pattern in string, as a list of strings

Tips: When you need to use an expression several times in a single program, using the compile() function to save the resulting regular expression object for reuse is more efficient.

dot (.) : Match any character except newline

In [35]:

```
text = "Amin Hakim"
pattern = r"Ami."

pattern = re.compile(pattern)

result = pattern.search(text)
print(result.group())
```

```
Amin
```

In [36]:

```
text2 = "Amir Haiqal"

pattern2 = pattern.search(text2)
print(pattern2.group())
```

```
Amir
```

bracket ([]) : Match anything in the bracket.

In [37]:

```
text3 = "I'm 21 years old"

pattern = re.compile("[1-9]")
result = pattern.search(text3)
print(bool(result))
```

```
True
```

In Pandas, we can use regex to search the pattern that we want in the text. As I mentioned before, we can include regex in **str.contains() function**.

For example, we want to search for Position Title which has a digit in it.

In [38]:

```
df.loc[df["Position Title"].str.contains("[1-9]")].head(10)
```

Out[38]:

	First Name	Last Name	Name	Position Title	Department	Employee Annual Salary
275	Mark	Akana	Akana, Mark S	Police Officer/Explsv Detect K9 Hndlr	Police	95178.0
892	Gilbert	Arce	Arce, Gilbert	Police Officer/Explsv Detect K9 Hndlr	Police	98016.0
1151	Rhonda	Avina	Avina, Rhonda S	Communications Operator I - 311	Oemc	59184.0
1233	Gayla	Baggett	Baggett, Gayla D	Communications Operator I - 311	Oemc	62004.0
1934	Timothy	Beran	Beran, Timothy	Police Officer/Explsv Detect K9 Hndlr	Police	91752.0
2170	Eunice	Bishop Abren	Bishop Abren, Eunice	Communications Operator I - 311	Oemc	68028.0
2751	Michael	Bransfield	Bransfield, Michael P	Police Officer/Explsv Detect K9 Hndlr	Police	98016.0
3521	Sheila	Butler	Butler, Sheila	Police Officer/Explsv Detect K9 Hndlr	Police	95178.0
3561	James	Byrne	Byrne, James T	Police Officer/Explsv Detect K9 Hndlr	Police	98016.0
3764	Brian	Campbell	Campbell, Brian P	Police Officer/Explsv Detect K9 Hndlr	Police	95178.0

We can go very specific with additional regex. Say, we want to search for Position Title, which contains the word, "Police" as well as number(s).

- **dot (.)** : Match any character except newline
- **star (*)** : Zero or more occurrences
- **bracket ([ ])** : Match with anything in the bracket.

In [39]:

```
df.loc[df["Position Title"].str.contains("Police.*[1-9]")].head(10)
```

Out[39]:

	First Name	Last Name	Name	Position Title	Department	Employee Annual Salary
275	Mark	Akana	Akana, Mark S	Police Officer/Explsv Detect K9 Hndlr	Police	95178.0
892	Gilbert	Arce	Arce, Gilbert	Police Officer/Explsv Detect K9 Hndlr	Police	98016.0
1934	Timothy	Beran	Beran, Timothy	Police Officer/Explsv Detect K9 Hndlr	Police	91752.0
2751	Michael	Bransfield	Bransfield, Michael P	Police Officer/Explsv Detect K9 Hndlr	Police	98016.0
3521	Sheila	Butler	Butler, Sheila	Police Officer/Explsv Detect K9 Hndlr	Police	95178.0
3561	James	Byrne	Byrne, James T	Police Officer/Explsv Detect K9 Hndlr	Police	98016.0
3764	Brian	Campbell	Campbell, Brian P	Police Officer/Explsv Detect K9 Hndlr	Police	95178.0
5593	Rodolfo	Coronado	Coronado, Rodolfo E	Police Officer/Explsv Detect K9 Hndlr	Police	95178.0
6344	Paul	Davenport	Davenport, Paul B	Police Officer/Explsv Detect K9 Hndlr	Police	95178.0
6937	Joshua	Diaz	Diaz, Joshua	Police Officer/Explsv Detect K9 Hndlr	Police	88656.0

Let's search for Position Title that starts with **Water** and ends with **Taker**

- **^** : Start of string
- **Bracket []** : Match anything in the bracket. Eg: [Ww] means match any which have w or W
- **Dot (.)** : Match any single character except the newline character.
- **star (*)** : Zero or more occurrences.
- **\b** : Word boundary. Spaces between word.

In [40]:

```
df.loc[df["Position Title"].str.contains("^[wW]ater.*Taker")].head(15)
```

Out[40]:

	First Name	Last Name	Name	Position Title	Department	Employee Annual Salary
0	Elvia	Aaron	Aaron, Elvia J	Water Rate Taker	Water Management	90744.0
2400	Brian	Bolton	Bolton, Brian E	Water Rate Taker	Water Management	78948.0
3143	Sharon	Brown	Brown, Sharon L	Water Rate Taker	Water Management	82728.0
3456	Carl	Burt	Burt, Carl S	Water Rate Taker	Water Management	90744.0
3626	Ophelia	Cage	Cage, Ophelia	Water Rate Taker	Water Management	90744.0
6954	Oscar	Diaz	Diaz, Oscar A	Water Rate Taker	Water Management	90744.0