

# apply, map, and applymap explained

Created by Alyssa Harris  
for Python Data Science Training at Urban

# Apply, map, and applymap

- Apply: Iterate over every row (or column) in the data frame and apply an operation to that row (or column)
- Map: Iterate over one row (or column) in the data frame and operate on each element in that row (or column)
- ApplyMap: Iterate over every element in the data frame and apply an operation to each element

# Apply, map, and applymap examples

The following examples will use lines of code largely drawn from the class examples along with this data frame:

	state	city	value1	value2	date	citystate
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA

# Apply, map, and applymap examples

- **Apply: Iterate over every row (or column) in the data frame and apply an operation to that row (or column)**
- Map: Iterate over one row (or column) in the data frame and operate on each element in that row (or column)
- ApplyMap: Iterate over every element in the data frame and apply an operation to each element

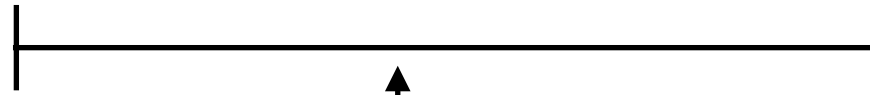
# Apply - syntax

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

**Let's break down what this line is saying**

# Apply - syntax

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```



Apply this function

# Apply - syntax

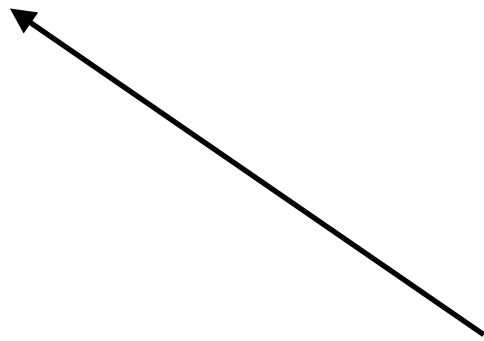
```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

To *this axis*



# Apply - syntax

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```



And save the result to this new column  
named "v1 + v2"



# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

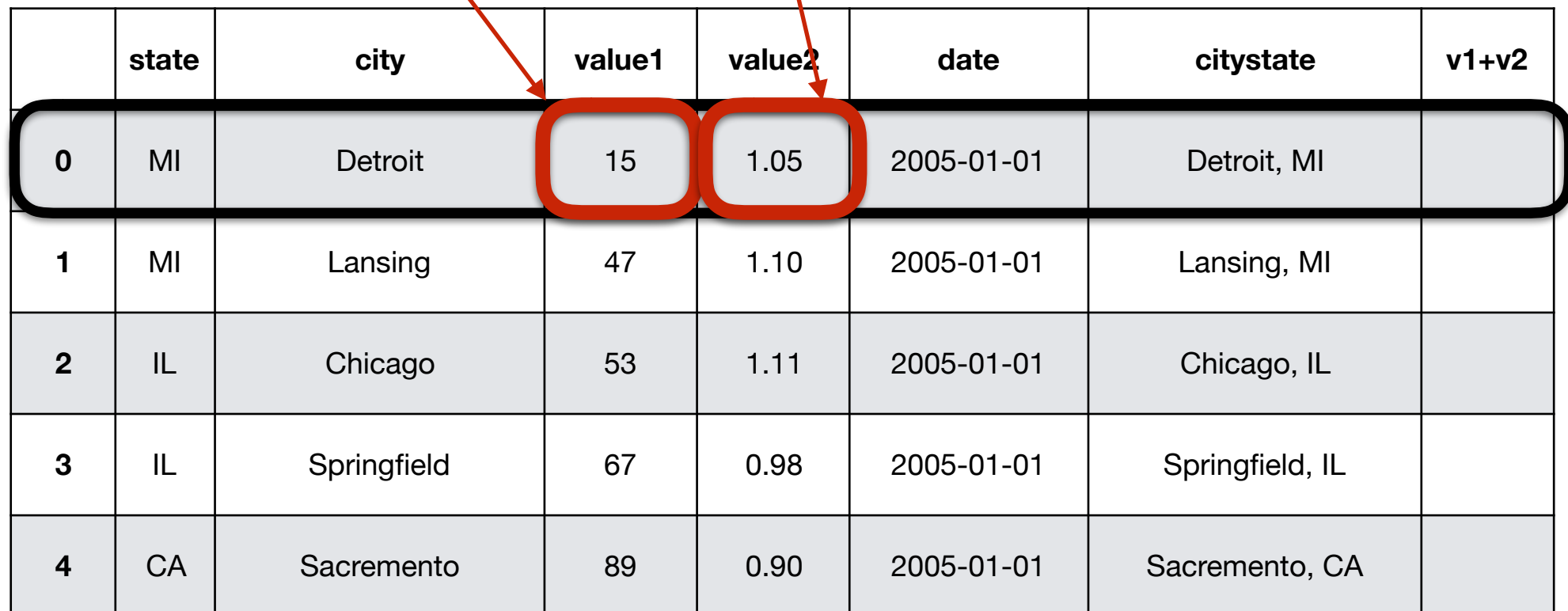
	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

← r

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

$r['value1'] + r['value2']$

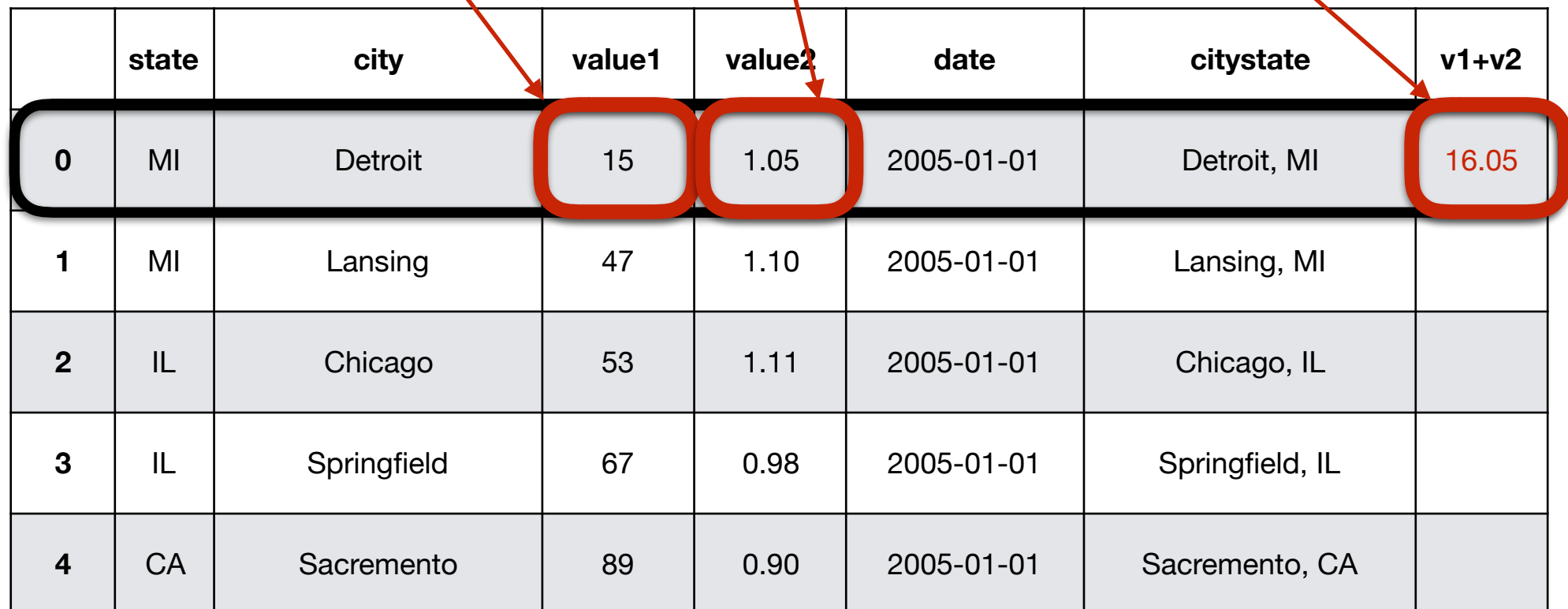


	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

$$r['value1'] + r['value2'] = 16.05$$



	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

← r

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

$r['value1'] + r['value2']$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

←  $r$

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

$$r['value1'] + r['value2'] = 48.10$$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

← r

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

← r



# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

$r['value1'] + r['value2']$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

←  $r$

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

$$r['value1'] + r['value2'] = 54.11$$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

← r

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

← r

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

$r['value1'] + r['value2']$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

←  $r$

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

$$r['value1'] + r['value2'] = 67.98$$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

← r

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

← r

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

$r['value1'] + r['value2']$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	

← r

# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

$$r['value1'] + r['value2'] = 89.90$$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90

← r



# Apply - in action

```
df['v1+v2'] = df.apply(lambda r: r['value1'] + r['value2'], axis=1)
```

**And we're done!**

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90

# Apply, map, and applymap examples

- Apply: Iterate over every row (or column) in the data frame and apply an operation to that row (or column)
- **Map: Iterate over one row (or column) in the data frame and operate on each element in that row (or column)**
- ApplyMap: Iterate over every element in the data frame and apply an operation to each element

# Map - syntax

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

**Again, let's break down this line into  
pieces**

# Map - syntax

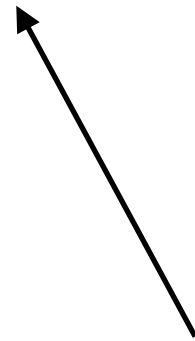
```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

A horizontal line with vertical end caps at both ends, spanning the width of the lambda function 'lambda i: round(i, 1)'.A diagonal arrow pointing from the text 'Apply this function' up towards the lambda function part of the code.

Apply this function

# Map - syntax

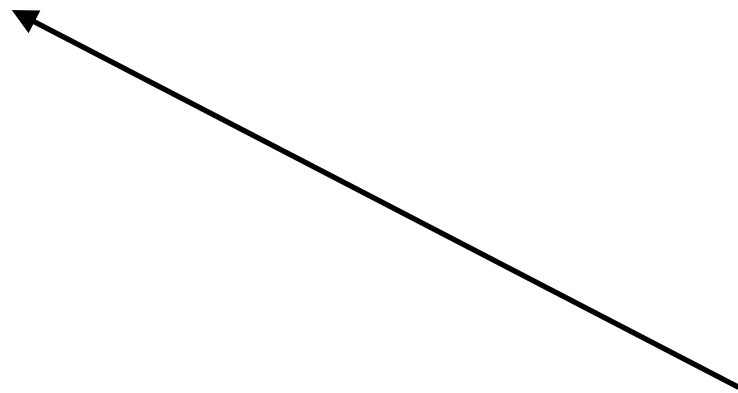
```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```



To every element in this column

# Map - syntax

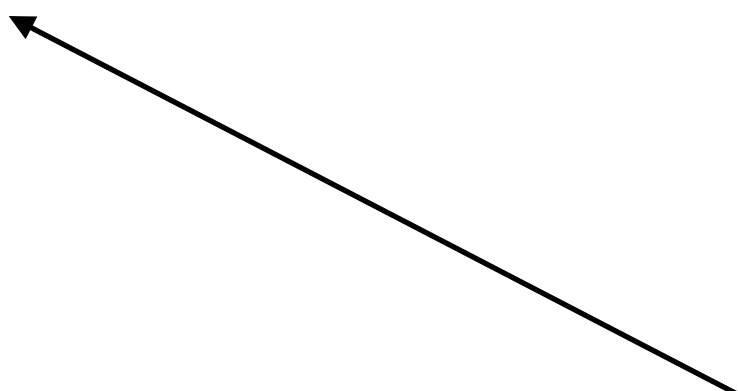
```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```



And save the result to this column

# Map - syntax

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

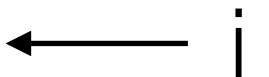


*Note:* this will overwrite the column 'v1+v2' with the results of the mapping function. If we wanted to keep both, we would need to create a column with a different name

# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90





# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

round(16.05, 1) = 16

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16.05
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90

# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

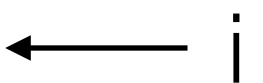
$\text{round}(16.05, 1) = 16$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90

# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90



# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

$\text{round}(48.10, 1) = 48$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48.10
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90

# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

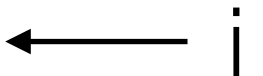
$\text{round}(48.10, 1) = 48$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90

# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90



# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

$\text{round}(54.11, 1) = 54$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54.11
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90

# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

$\text{round}(54.11, 1) = 54$

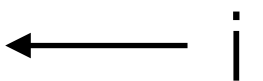
	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90



# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90



# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

$\text{round}(67.98, 1) = 68$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	67.98
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90

# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

$\text{round}(67.98, 1) = 68$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	68
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90

# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	68
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90

← i

# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

$\text{round}(89.90, 1) = 90$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	68
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	89.90

# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

$\text{round}(89.90, 1) = 90$

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	68
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	90

# Map - in action

```
df['v1+v2'] = df['v1+v2'].map(lambda i: round(i, 1))
```

**And we're done!**

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	68
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	90

# Apply, map, and applymap examples

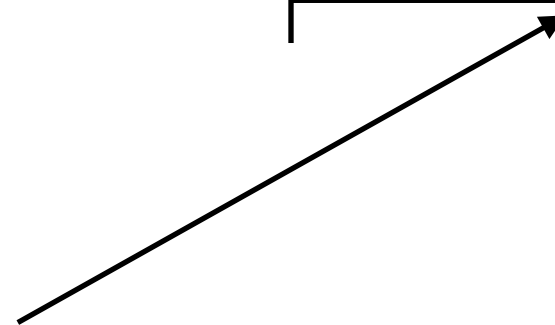
- Apply: Iterate over every row (or column) in the data frame and apply an operation to that row (or column)
- Map: Iterate over one row (or column) in the data frame and operate on each element in that row (or column)
- **ApplyMap: Iterate over every element in the data frame and apply an operation to each element**



# Applymap - syntax

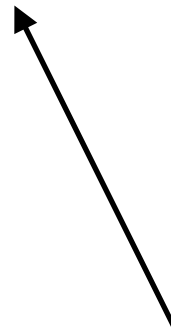
```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

Apply this function



# Applymap - syntax

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```



to every element of this data frame

# Applymap - syntax

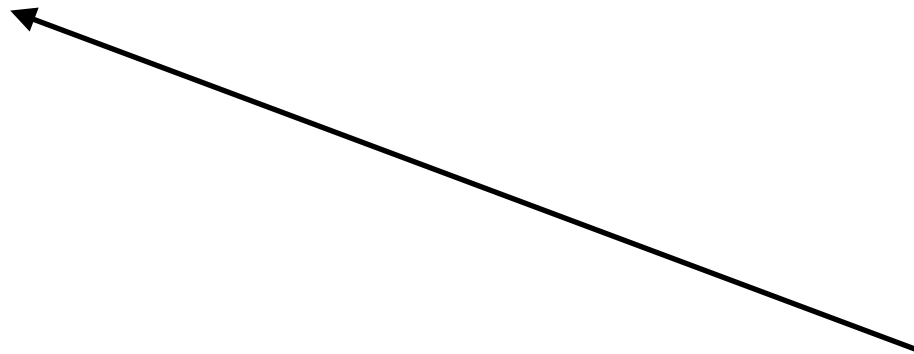
```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

*\*Note\** a 2-D subset of a  
data frame is another  
data frame

to every element of this data frame

# Applymap - syntax

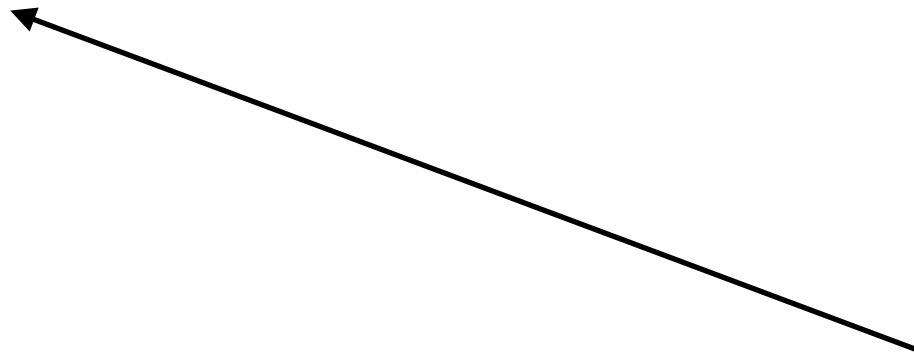
```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```



And save the results this data frame

# Applymap - syntax

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

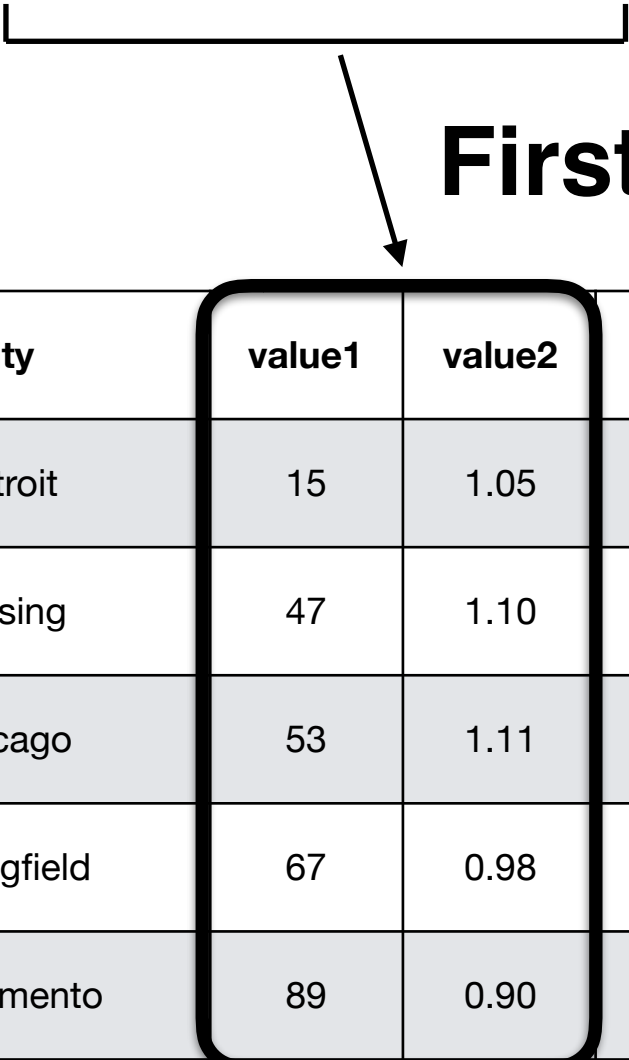


*Note:* This will overwrite the values that were in this data frame.

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

**First step:** subset the data frame



	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	15	1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	47	1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	53	1.11	2005-01-01	Chicago, IL	54
3	IL	Springfield	67	0.98	2005-01-01	Springfield, IL	68
4	CA	Sacramento	89	0.90	2005-01-01	Sacramento, CA	90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

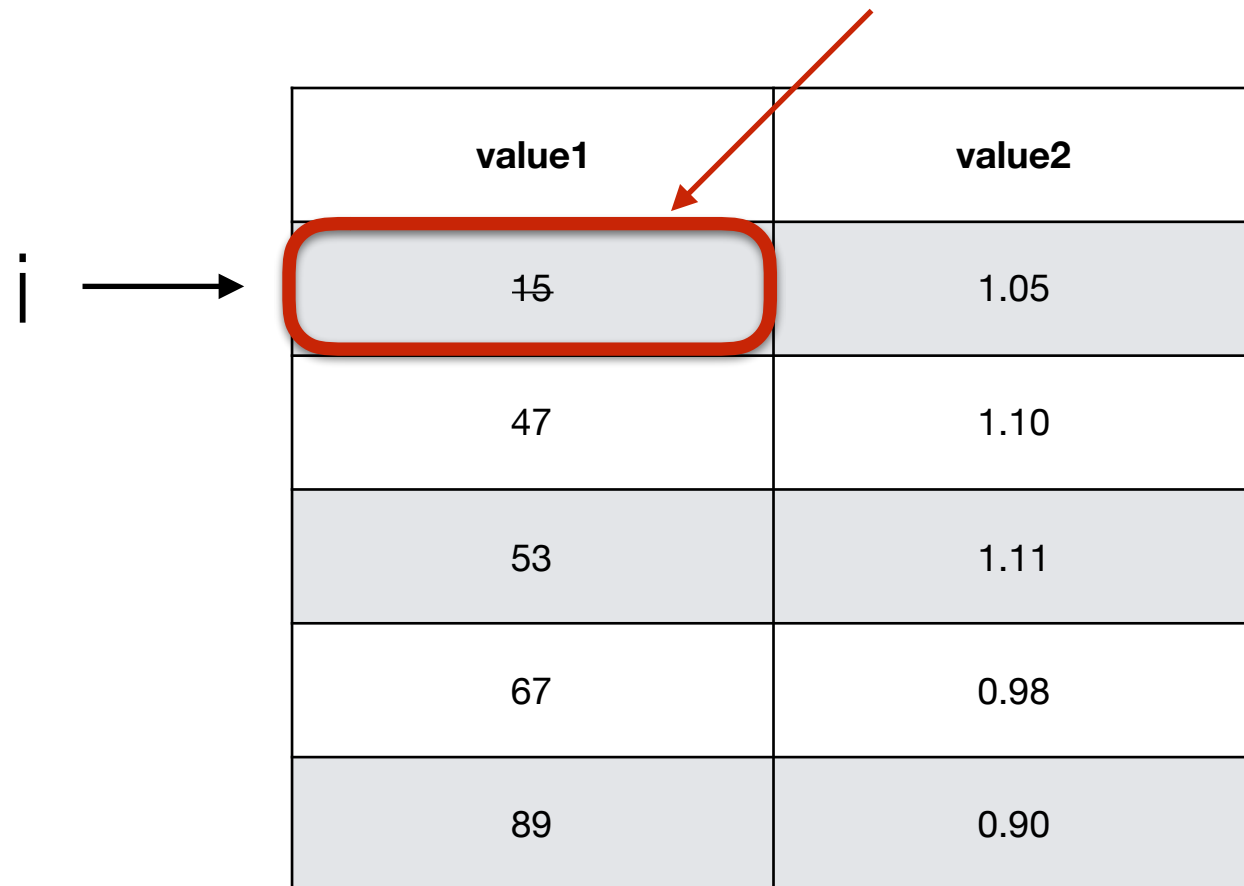
**Now:** iterate through  
all the elements

	value1	value2
i →	15	1.05
	47	1.10
	53	1.11
	67	0.98
	89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

lambda i: '\${:.2f}'.format(i) = \$15.00



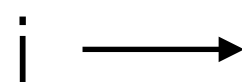
value1	value2
15	1.05
47	1.10
53	1.11
67	0.98
89	0.90



# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(15) = $15.00`




value1	value2
\$15.00	1.05
47	1.10
53	1.11
67	0.98
89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

value1	value2
\$15.00	1.05
47	1.10
53	1.11
67	0.98
89	0.90

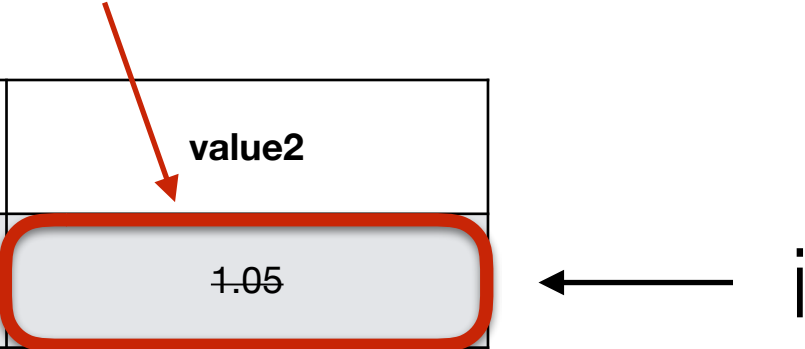


The diagram illustrates the 'i' parameter in the lambda function of the applymap operation. It shows a table with two columns, 'value1' and 'value2'. The first row of data is highlighted with a thick black border around the 'value2' cell, which contains the value '1.05'. An arrow points from the letter 'i' to this specific cell, indicating that 'i' represents the individual element being processed by the lambda function.

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(1.05) = $1.05`

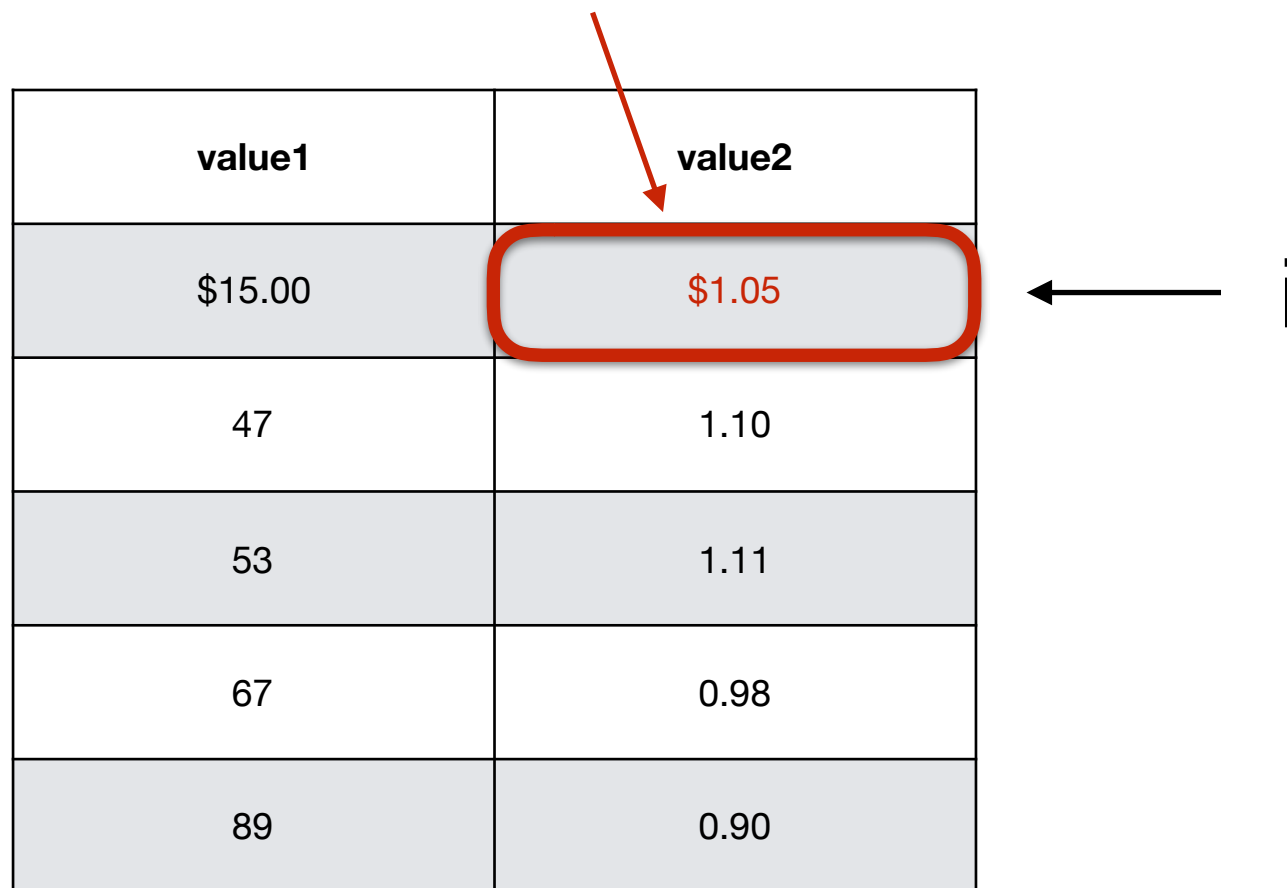


value1	value2
\$15.00	1.05
47	1.10
53	1.11
67	0.98
89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(1.05) = $1.05`

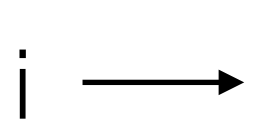


The diagram illustrates the application of a lambda function to a DataFrame column. A red arrow points from the lambda function `'${:.2f}'.format(i)` to the `value2` column header. A red rounded rectangle highlights the cell containing `$1.05` in the first row of the `value2` column. A black arrow points from the variable `i` to this highlighted cell, indicating that `i` represents the value being formatted.

value1	value2
\$15.00	\$1.05
47	1.10
53	1.11
67	0.98
89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

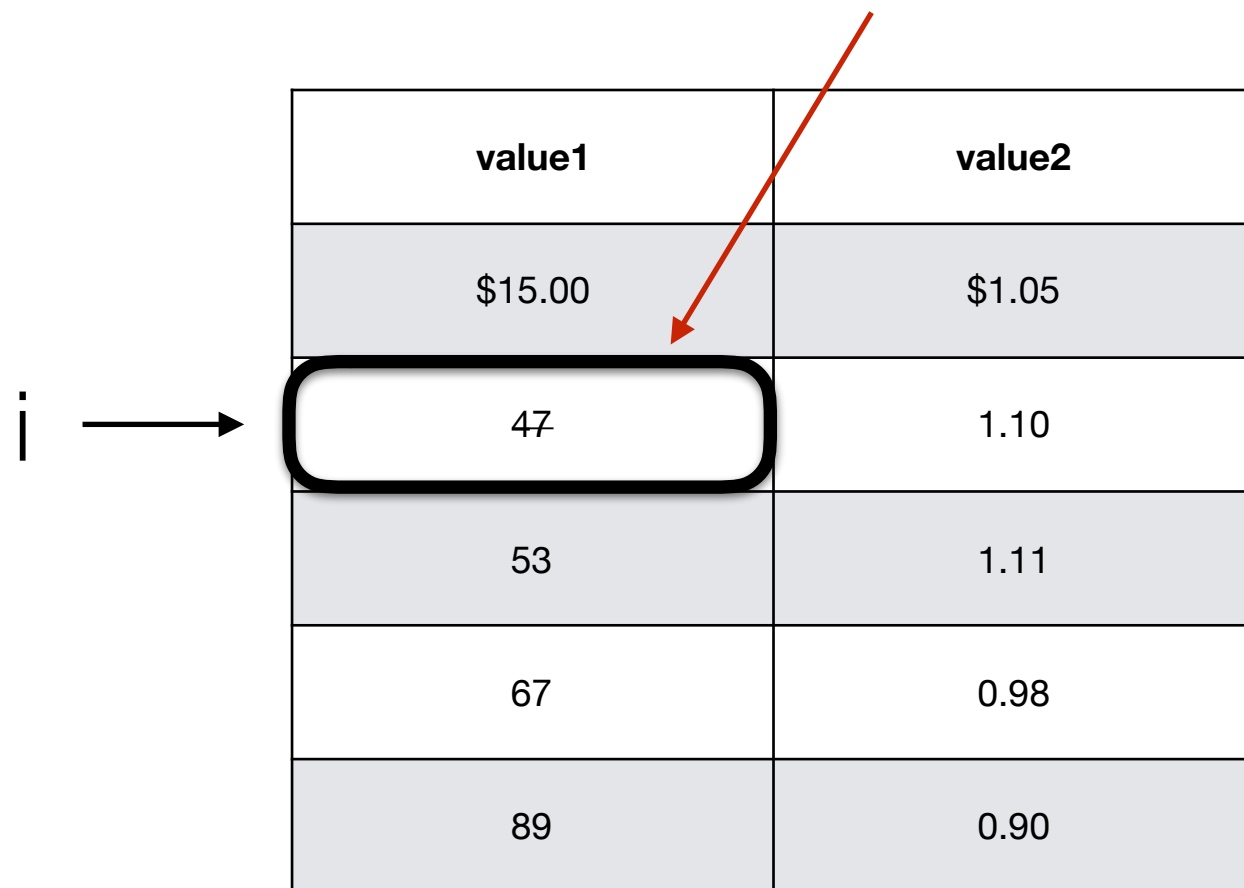


value1	value2
\$15.00	\$1.05
47	1.10
53	1.11
67	0.98
89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(47) = $47.00`

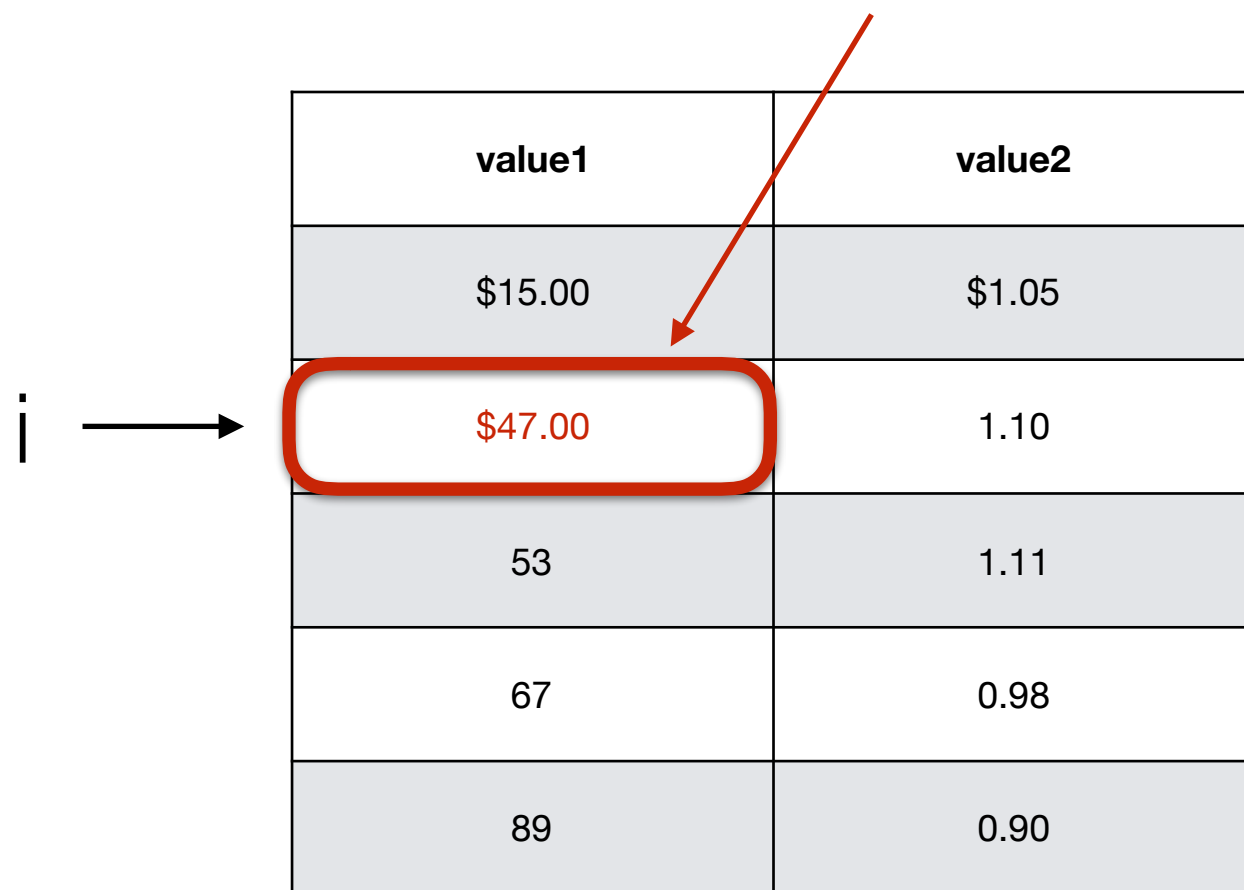


	value1	value2
	\$15.00	\$1.05
i →	47	1.10
	53	1.11
	67	0.98
	89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(47) = $47.00`

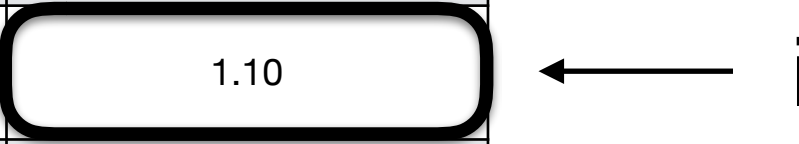


value1	value2
\$15.00	\$1.05
\$47.00	1.10
53	1.11
67	0.98
89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

value1	value2
\$15.00	\$1.05
\$47.00	1.10
53	1.11
67	0.98
89	0.90



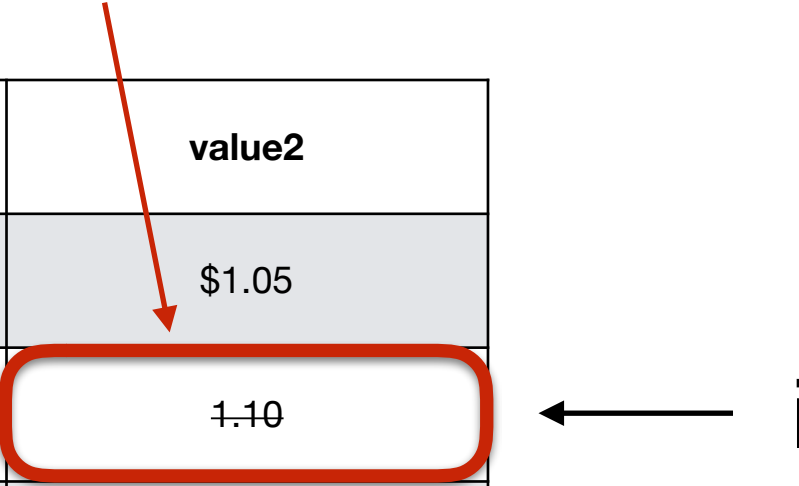
The diagram illustrates the 'i' parameter in the lambda function of the applymap operation. An arrow points from the variable 'i' to the cell containing the value '1.10' in the 'value2' column of the DataFrame. This cell is also highlighted with a thick black border, indicating it is the current element being processed by the function.



# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(1.10) = $1.10`

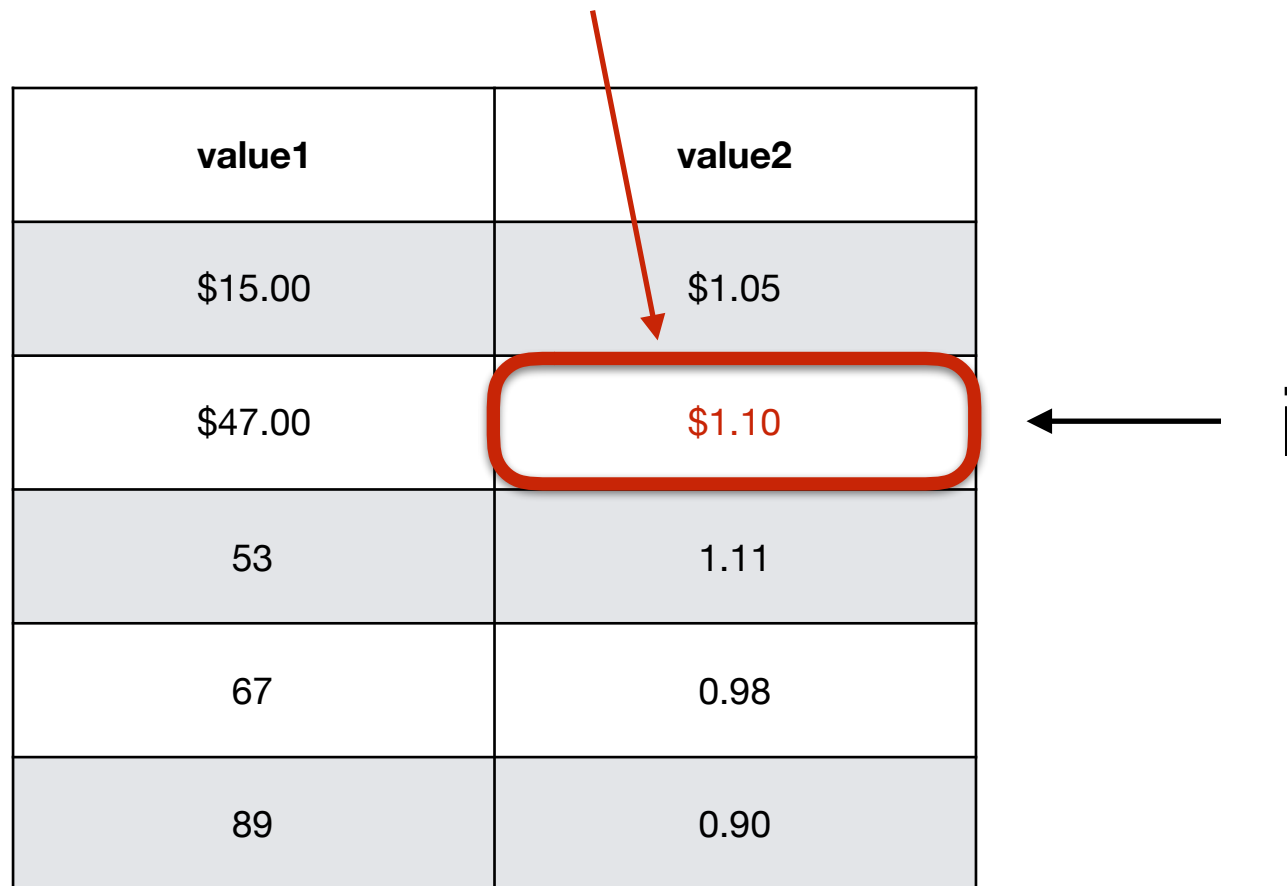


value1	value2
\$15.00	\$1.05
\$47.00	1.10
53	1.11
67	0.98
89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(1.10) = $1.10`



The diagram illustrates the application of a lambda function to a DataFrame column. A red arrow points from the lambda function in the code above to the 'value2' column of the DataFrame. The cell containing '1.10' is highlighted with a red rounded rectangle, and a black arrow points from the variable 'i' in the lambda function to this cell, indicating that 'i' represents the current value being processed.

value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
53	1.11
67	0.98
89	0.90

# Applymap - in action

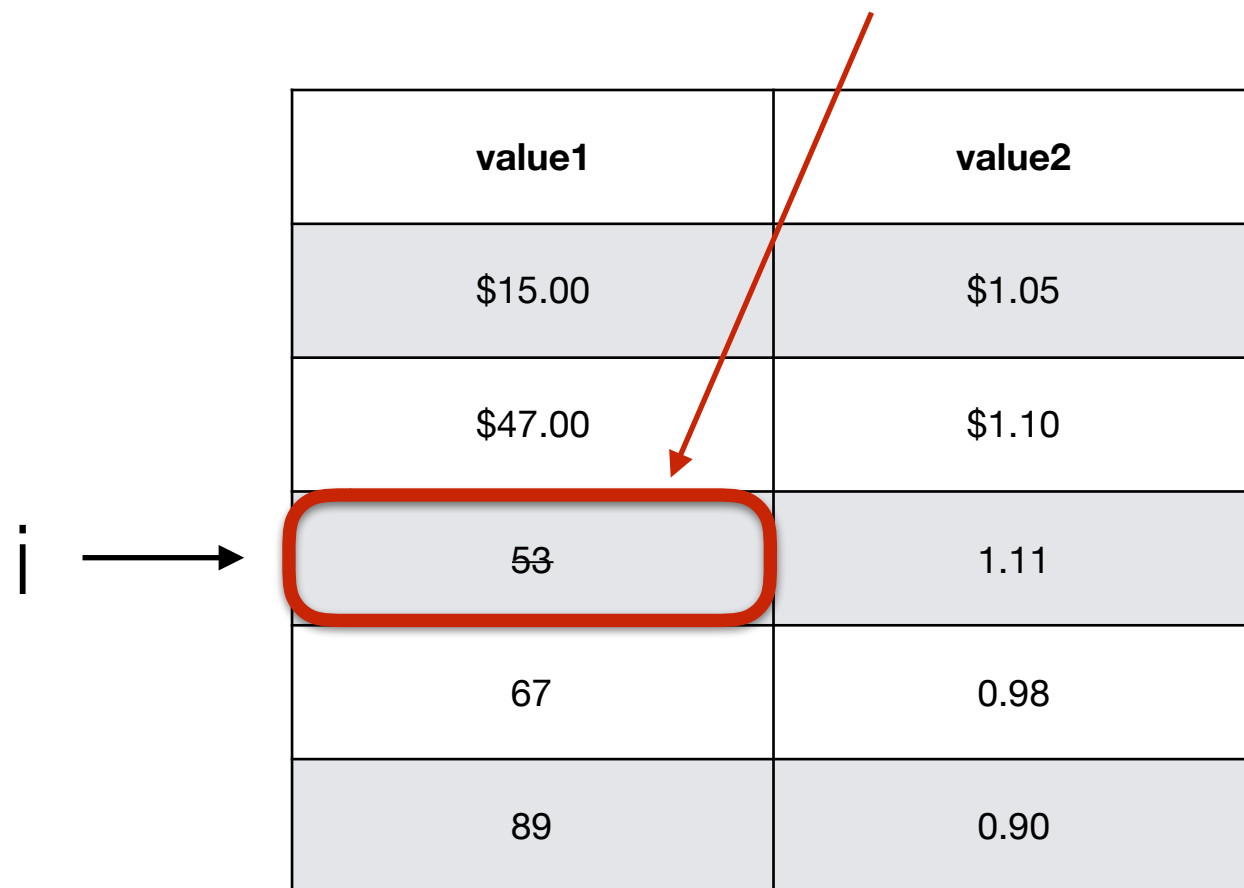
```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

	value1	value2
	\$15.00	\$1.05
	\$47.00	\$1.10
i →	53	1.11
	67	0.98
	89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(53) = $53.00`

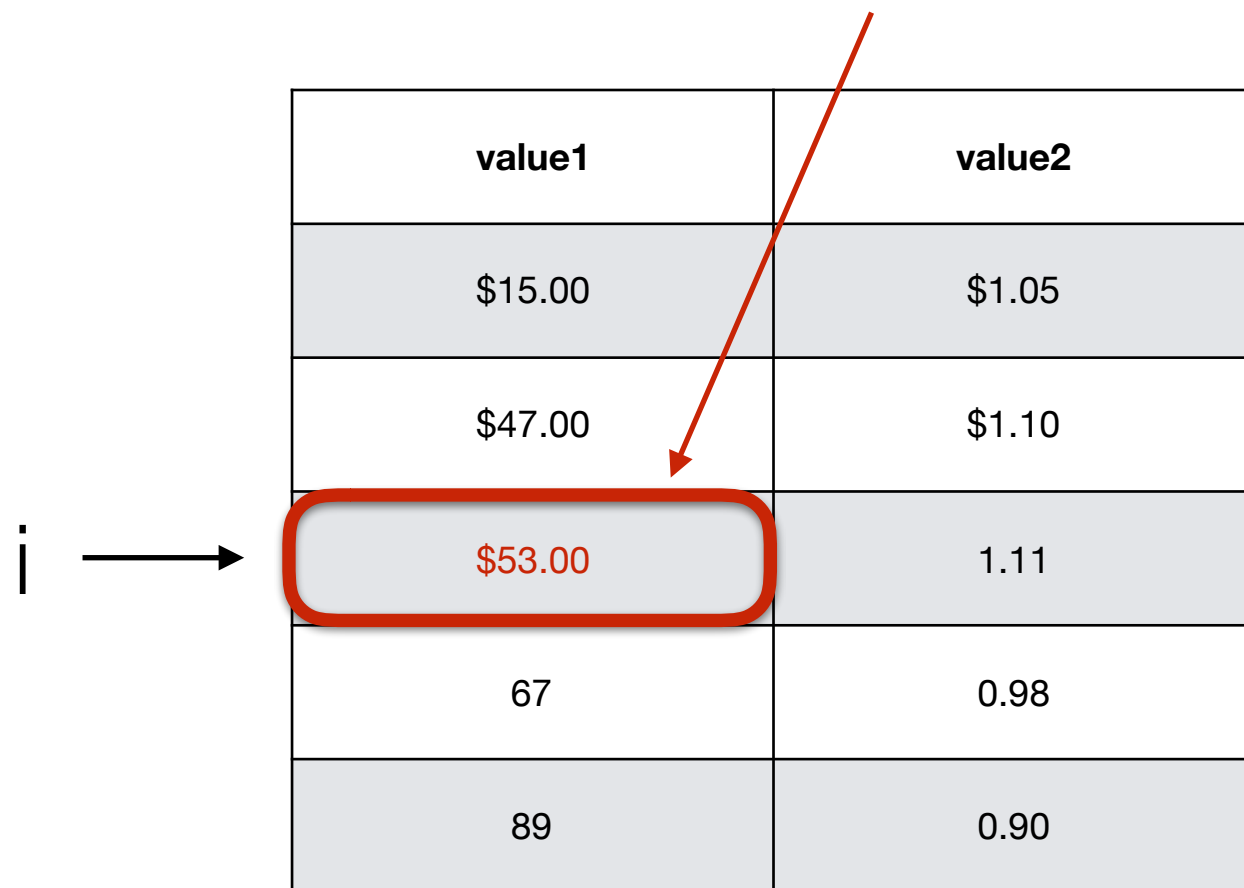


	value1	value2
	\$15.00	\$1.05
	\$47.00	\$1.10
i →	53	1.11
	67	0.98
	89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(53) = $53.00`




value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
<b>\$53.00</b>	1.11
67	0.98
89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
\$53.00	1.11
67	0.98
89	0.90

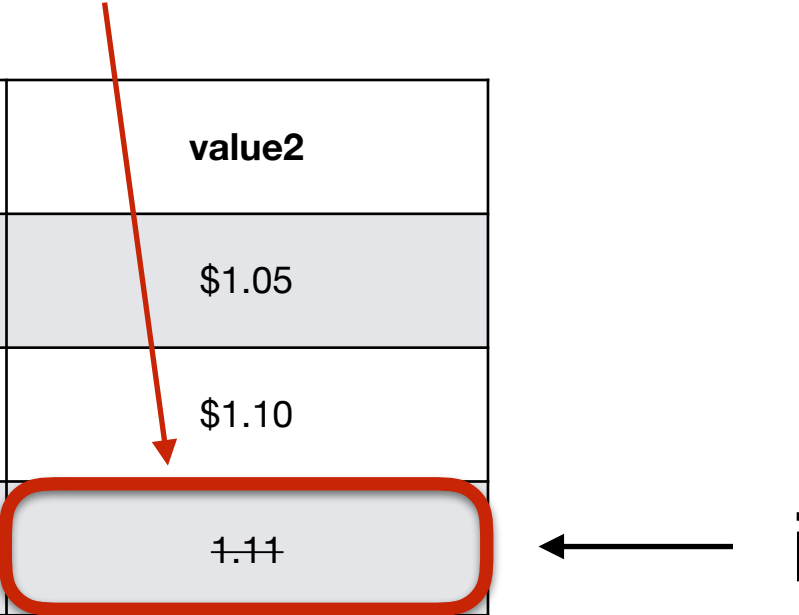


# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(1.11) = $1.11`

value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
\$53.00	1.11
67	0.98
89	0.90

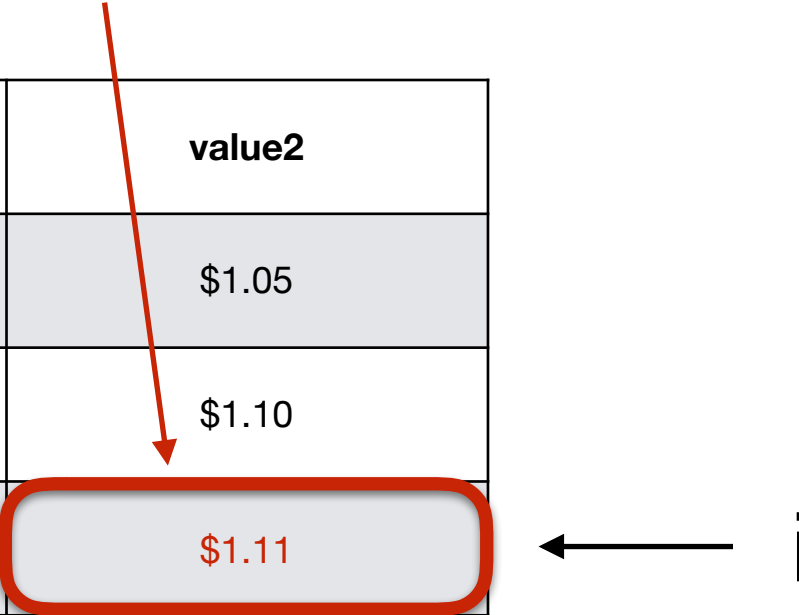


# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(1.11) = $1.11`

value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
\$53.00	\$1.11
67	0.98
89	0.90





# Applymap - in action

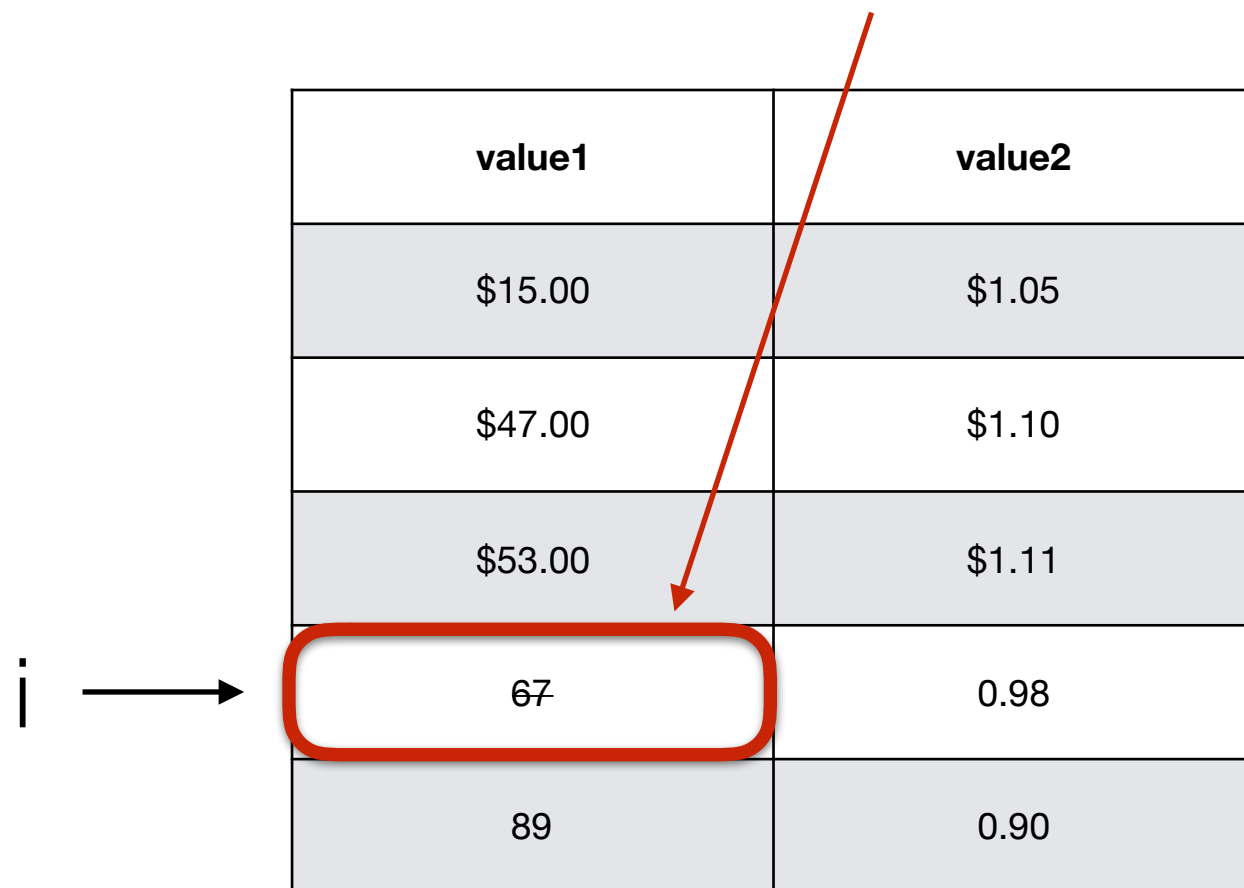
```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

	value1	value2
	\$15.00	\$1.05
	\$47.00	\$1.10
	\$53.00	\$1.11
i →	67	0.98
	89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(67) = $67.00`

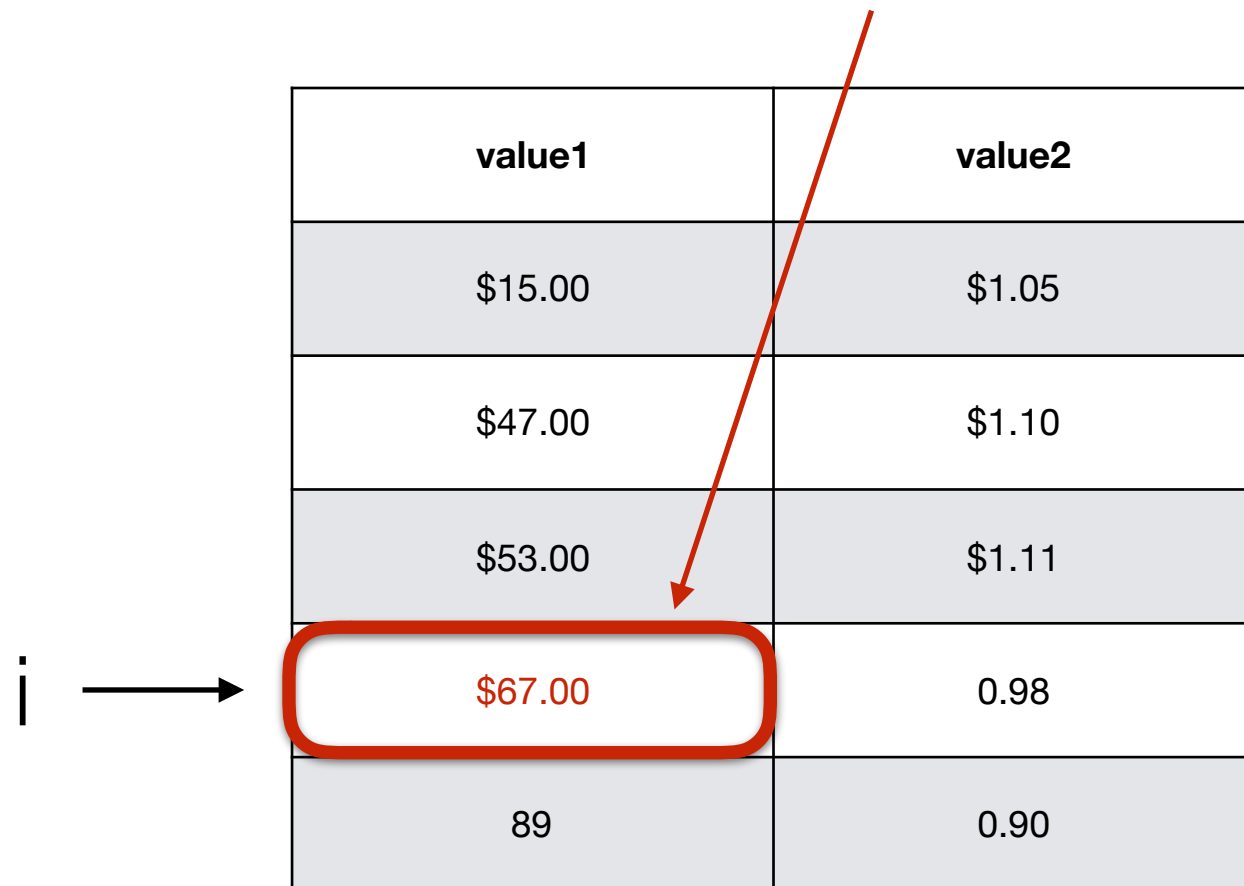


	value1	value2
	\$15.00	\$1.05
	\$47.00	\$1.10
	\$53.00	\$1.11
i →	67	0.98
	89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(67) = $67.00`



value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
\$53.00	\$1.11
<b>\$67.00</b>	0.98
89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
\$53.00	\$1.11
\$67.00	0.98
89	0.90

← i

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(0.98) = $0.98`

value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
\$53.00	\$1.11
\$67.00	0.98
89	0.90

← i

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(0.98) = $0.98`

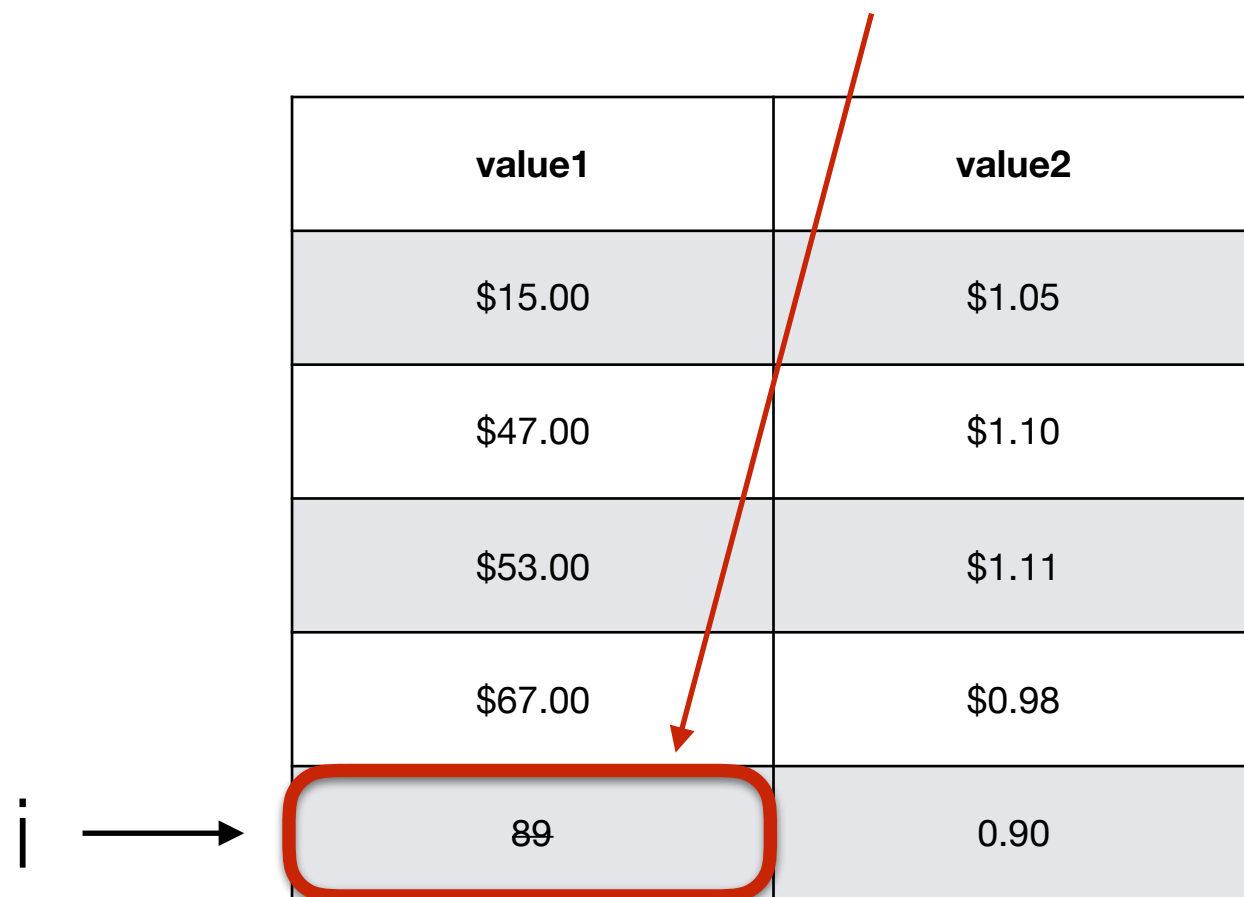
value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
\$53.00	\$1.11
\$67.00	\$0.98
89	0.90

← i

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(89) = $89.00`

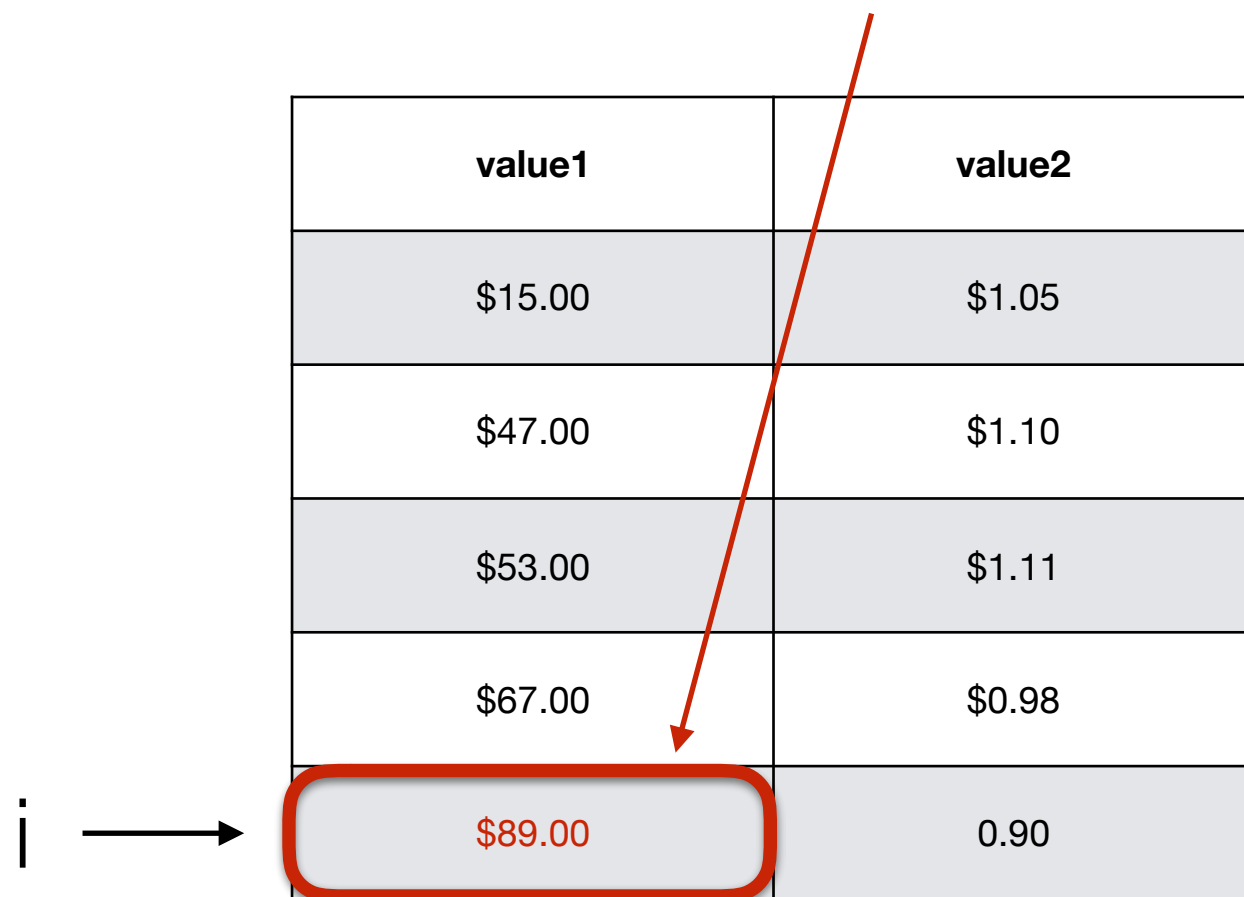


value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
\$53.00	\$1.11
\$67.00	\$0.98
89	0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(89) = $89.00`



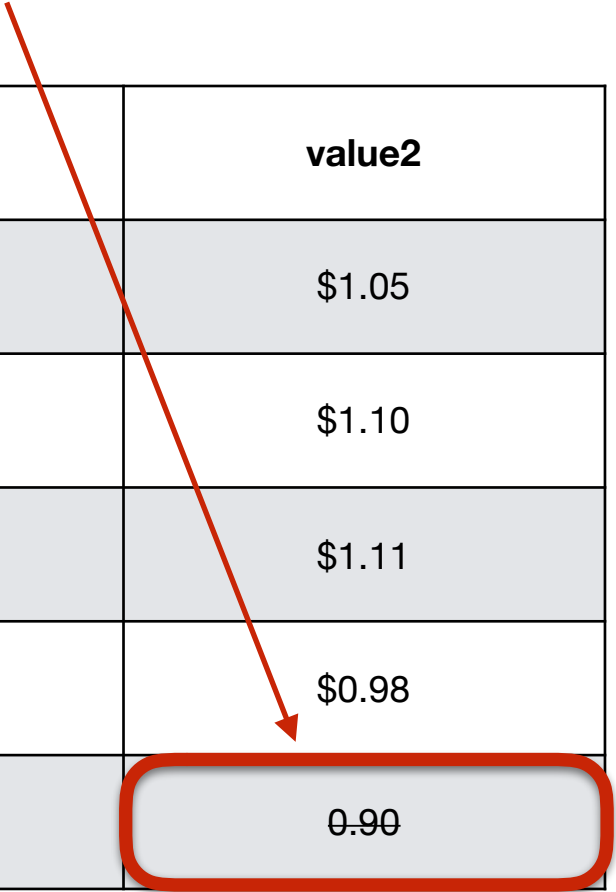
	value1	value2
	\$15.00	\$1.05
	\$47.00	\$1.10
	\$53.00	\$1.11
	\$67.00	\$0.98
i →	\$89.00	0.90



# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(89) = $89.00`



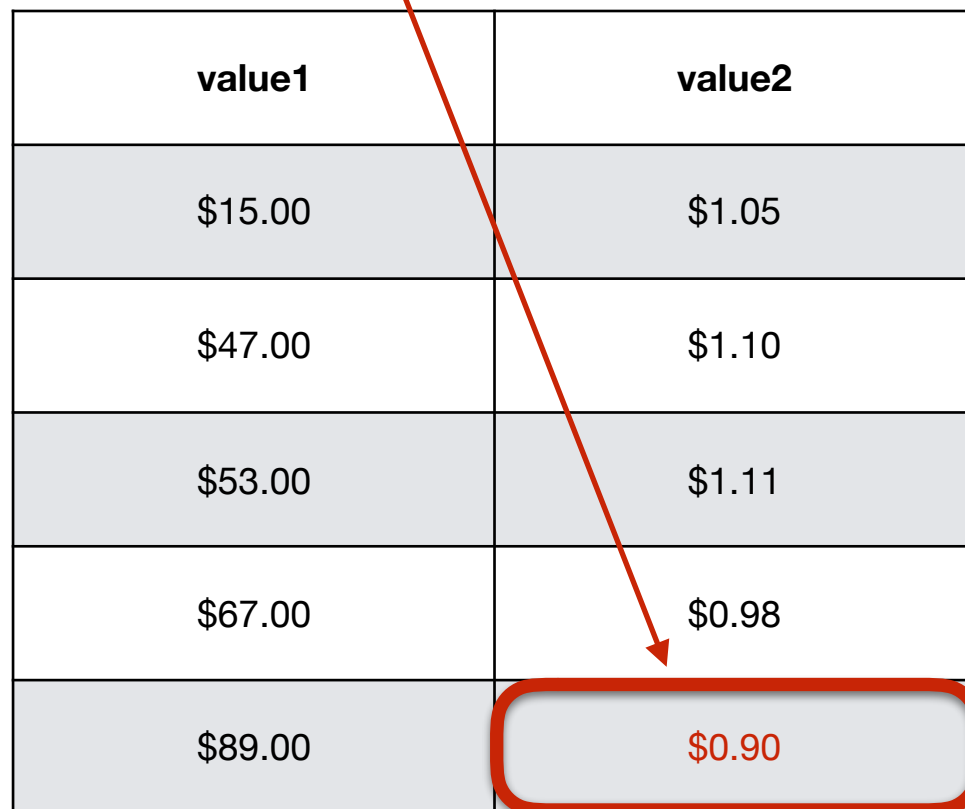
value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
\$53.00	\$1.11
\$67.00	\$0.98
\$89.00	0.90

← i

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

`'${:.2f}'.format(0.90) = $0.90`



value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
\$53.00	\$1.11
\$67.00	\$0.98
\$89.00	\$0.90

# Applymap - in action

```
df[['value1', 'value2']] = df[['value1', 'value2']].applymap(lambda i: '${:.2f}'.format(i))
```

**And we're done!**

value1	value2
\$15.00	\$1.05
\$47.00	\$1.10
\$53.00	\$1.11
\$67.00	\$0.98
\$89.00	\$0.90

# So our final data frame is:

	state	city	value1	value2	date	citystate	v1+v2
0	MI	Detroit	\$15.00	\$1.05	2005-01-01	Detroit, MI	16
1	MI	Lansing	\$47.00	\$1.10	2005-01-01	Lansing, MI	48
2	IL	Chicago	\$53.00	\$1.11	2005-01-01	Chicago, IL	54
3	IL	Springfield	\$67.00	\$0.98	2005-01-01	Springfield, IL	68
4	CA	Sacramento	\$89.00	\$0.90	2005-01-01	Sacramento, CA	90

# Documentation and references

10 Minutes to Pandas:

<http://pandas.pydata.org/pandas-docs/stable/10min.html>

A really relevant Stack Overflow forum:

<http://stackoverflow.com/questions/19798153/difference-between-map-applymap-and-apply-methods-in-pandas>

And the book that forum cites:

*Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython* by Wes McKinney