Lists, Control Structures & Tables

Prof. Abby Stylianou

Logistics

- Homework 1 out today
  - Due at beginning of class next Thursday
- Journal prompt available on course website (https://cs.slu.edu/~stylianou/1070/index.html)
- 20 minutes to complete

TECHNOLOGY

How a Feel-Good AI Story Went Wrong in Flint

A machine-learning model showed promising results, but city officials and their engineering contractor abandoned it.

ALEXIS C. MADRIGAL JAN 3, 2019

Workers in Flint, Michigan, replace a lead water-service pipe. (BILL PUGLIANO / GETTY)

MORE THAN A thousand days after the water problems in Flint, Michigan, became national news, thousands of homes in the city still have lead pipes, from which the toxic metal can leach into the water supply. To remedy the problem, the lead pipes need to be replaced with safer, copper ones. That sounds straightforward, but it is a challenge to figure out which homes have lead pipes in the first place. The City's records are incomplete and inaccurate. And digging up all the pipes would be costly and time-consuming.

MORE IN THIS SERI

Technology is tra
for better and wo

Tech Is Killing St
CHRISTINE RO

An Augmented Re
SARAH GAILEY

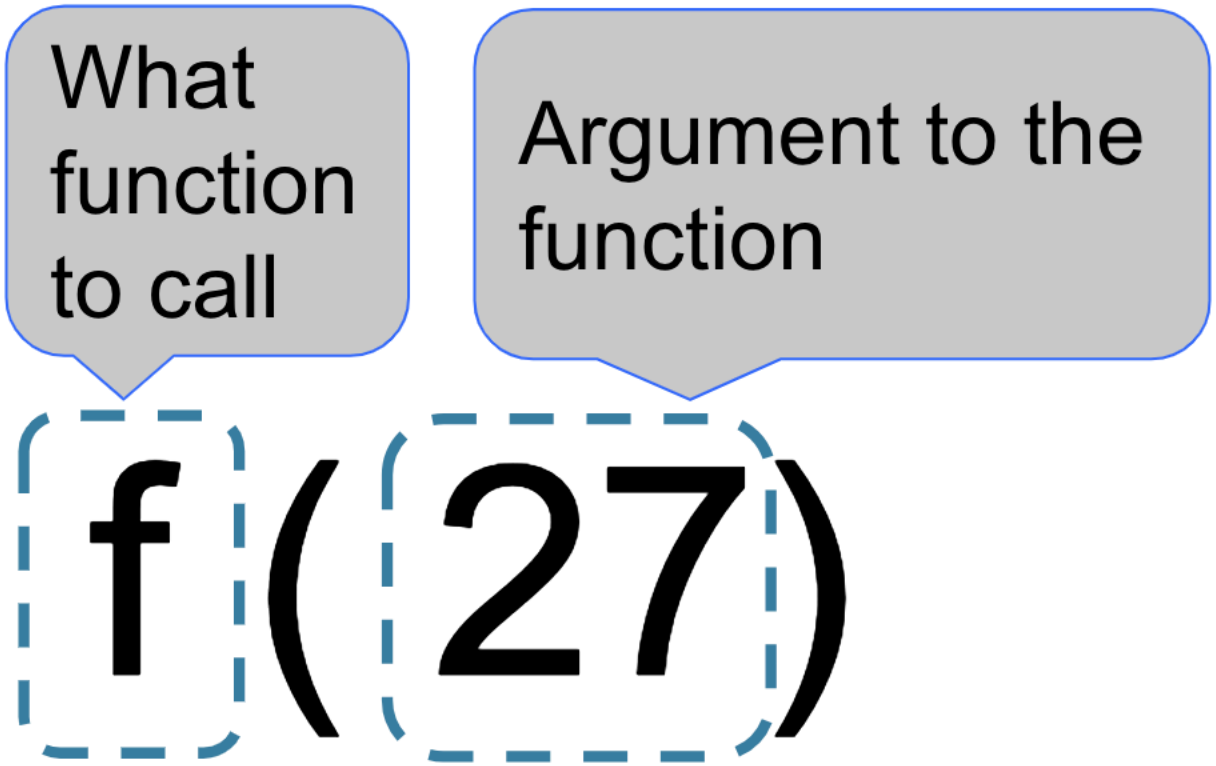
Meet the Safecrac
GEOFF MANAUGH

Review from Last Class

Operations

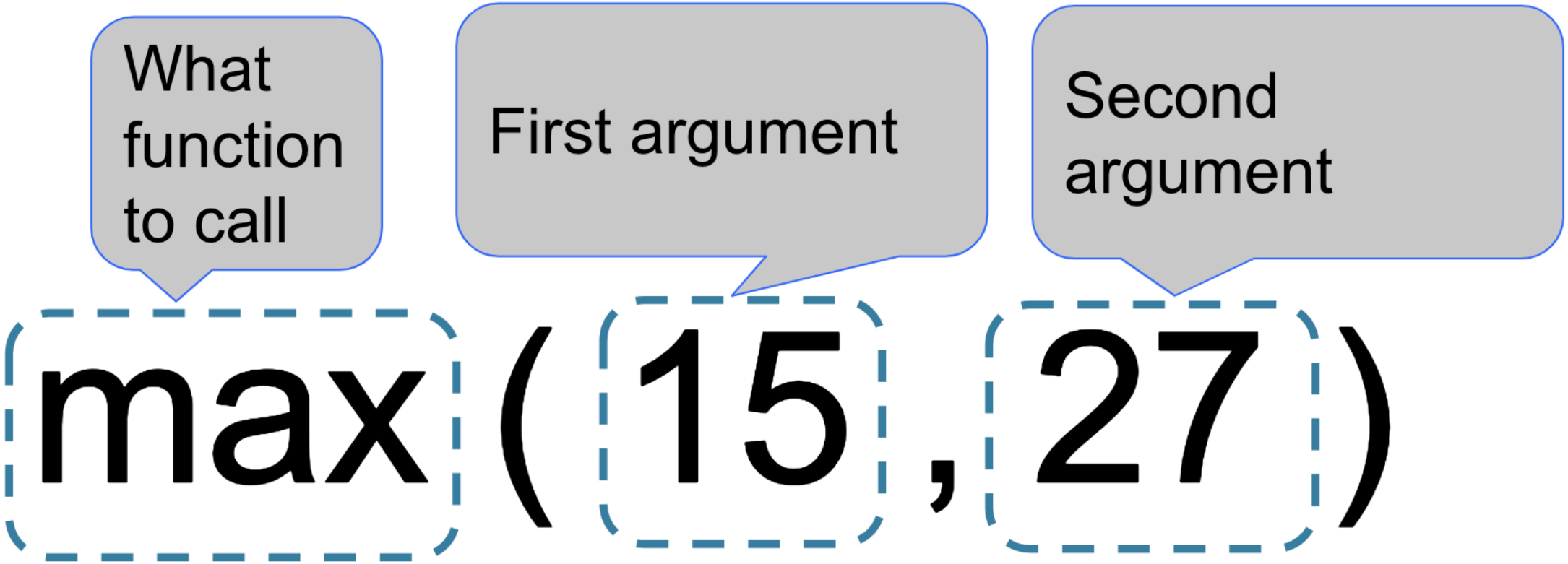
Operation	Operator	Example	Value
Addition	+	2 + 3	5
Subtraction	-	2 - 3	-1
Multiplication	*	2 * 3	6
Division	/	7 / 3	2.66667
Remainder	%	7 % 3	1
Exponentiation	**	2 ** 0.5	1.41421

Functions

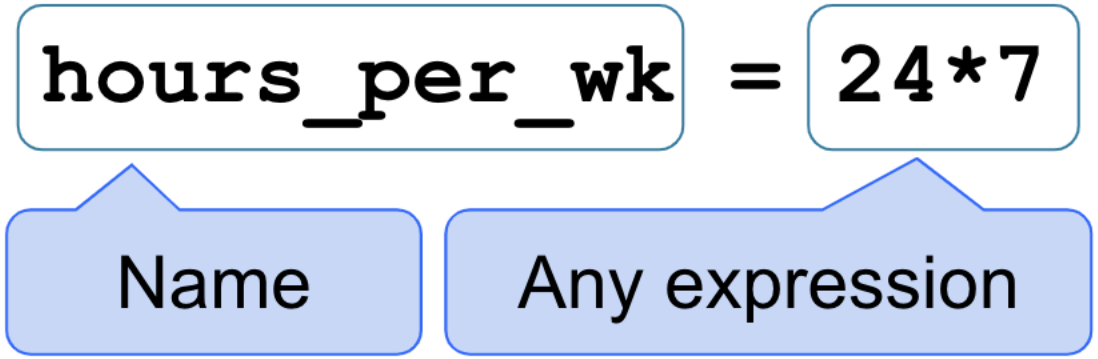


“Call f on 27”

Functions


What function to call
First argument
Second argument
max(15,27)

Assignment Statements



- An assignment statement changes the meaning of the name to the left of the = symbol

- The name is bound to a value (not an equation)

Data Types

We've seen 5 types so far:

- `int: 2`
- `float: 2.2`
- `bool: True`
- `str: 'Red fish, blue fish'`
- `Builtin_function_or_method: abs`

The `type` function can tell you the type of a value

- `type(2)`
- `type(2.2)`
- `type(True)`
- `type('Red fish, blue fish')`
- `type(abs)`

Conversions

- Strings that contain numbers can be converted to numbers
  - `int('12')`
  - `float('1.2')`
- Any value can be converted to a string
  - `str(5)`
  - `str(True)`
  - `str(abs)` ← anyone know what this would return?
- Numbers can be converted to other numeric types
  - `float(1)`
  - `int(1.2)`
  - `round(1.2)`

Lists

- Container that holds a number of objects in an order

```
L = [‘yellow’, ‘red’, ‘blue’, ‘green’, ‘black’]
```

- Accessing / Indexing

`L[0]`	’yellow’
`L[1:4]`	[‘red’, blue’, ‘green’]
`L[3:]`	[‘green’, ‘black’]
`L[-1]`	[‘black’]

- Length

`len(L)`	5

Lists

- Built-in methods for adding objects

```
L.append('pink')
print(L)
```

['yellow', 'red', 'blue', 'green', 'black', 'pink']

```
L.insert(0,'white')
print(L)
```

['white', 'yellow', 'red', 'blue', 'green', 'black', 'pink']

```
L2 = ['orange', 'cyan', 'magenta']
L.extend(L2)
print(L)
```

['white', 'yellow', 'red', 'blue', 'green', 'black', 'pink', 'orange', 'cyan', 'magenta']

Lists

```
L = ['white', 'yellow', 'red', 'blue', 'green', 'black',
     'pink', 'orange', 'cyan', 'magenta']
```

- Built-in methods for removing objects

```
L.remove('white')
print(L)
```

['yellow', 'red', 'blue', 'green', 'black', 'pink', 'orange', 'cyan', 'magenta']

```
del L[0]
print(L)
```

['red', 'blue', 'green', 'black', 'pink', 'orange', 'cyan', 'magenta']

```
L.pop()
```

'magenta'

```
print(L)
```

['yellow', 'red', 'blue', 'green', 'black', 'pink', 'orange', 'cyan']

Lists

```
L = [‘yellow’, ‘red’, ‘blue’, ‘green’, ‘black’, ‘pink’,
                    ‘orange’, ‘cyan’]
```

- Other built in methods

```
L.sort()
print(L)
```

['black', 'blue', 'cyan', 'green', 'orange', 'pink', 'red', 'yellow’]

```
L.count(‘red’)
```

1

```
L.reverse()
```

['yellow', 'red', 'pink', 'orange', 'green', 'cyan', 'blue', 'black']

Control Structures

- Control structure: direct the order of execution of statements in a program

- `if / else`

  - “If the weather is nice, I will mow the lawn, otherwise I’ll watch tv”

```
nice_weather = True
if nice_weather:
    mow_lawn()
else:
    watch_tv()
```

Control Structures

- Control structure: direct the order of execution of statements in a program

- `if / else`

  - “If the weather is nice, I will mow the lawn, otherwise I’ll watch tv”

```
nice_weather = True
if nice_weather:
    mow_lawn()
else:
    watch_tv()
```

```
L = [‘yellow’, ‘red’, ‘blue’, ‘green’, ‘black’, ‘pink’, ‘orange’, ‘cyan’]
```

```
if ‘blue’ in L:
    blue_exists_in_L = True
else:
    blue_exists_in_L = False
```

```
blue_exists_in_L = ‘blue’ in L
```

Control Structures

- `if / elif / else`
  - “If the temperature is above 80, I will swim; if it’s between 60 and 80, I will hike; otherwise, I will watch tv.”

```
if temperature > 80:
    swim()
elif temperature >= 60:
    hike()
else:
    watch_tv()
```

Control Structures

- `for` loops

```
L = [‘yellow’, ‘red’, ‘blue’, ‘green’, ‘black’, ‘pink’,
‘orange’, ‘cyan’]

for color in L:
    print color
```

yellow
red
pink
orange
green
cyan
blue
black

Control Structures

```
L = [‘yellow’, ‘red’, ‘blue’, ‘green’, ‘black’, ‘pink’, ‘orange’, ‘cyan’]

for color in L:
    if ‘e’ in color:
        print(color)
```

Control Structures

```
L = [‘yellow’, ‘red’, ‘blue’, ‘green’, ‘black’, ‘pink’,
‘orange’, ‘cyan’]

for color in L:
    if ‘e’ in color:
        print(color)
```

yellow
red
orange
green
blue

Control Structures

- `while` loops

```
L = [‘yellow’, ‘red’, ‘blue’, ‘green’, ‘black’, ‘pink’,
‘orange’, ‘cyan’]


idx = 0
while idx < 3:
    print(L[idx])
    idx += 1
```

Control Structures

- `while` loops

```
L = [‘yellow’, ‘red’, ‘blue’, ‘green’, ‘black’, ‘pink’, ‘orange’, ‘cyan’]


idx = 0
while idx < 3:
    print(L[idx])
    idx += 1
```

yellow
red
pink

List Comprehension

- A concise way to create lists

```
list = [ expression for item in list]
```

List Comprehension

- A concise way to create lists

```
list = [ expression for item in list]
```

```
squares = [x**2 for x in range(10)]
```

List Comprehension

- A concise way to create lists

```
list = [ expression for item in list if conditional ]
```

```
squares = [x**2 for x in range(10)]
```

```
even_squares = [x**2 for x in range(10) if x**2 % 2 == 0]
```

List Comprehension

- A concise way to create lists

```
list = [ expression for item in list if conditional ]
```

```
squares = [x**2 for x in range(10)]
```

```
even_squares = [x**2 for x in range(10) if x**2 % 2 == 0]
```

```
odd_squares = [x**2 for x in range(10) if x**2 % 2 != 0]
```

or:

```
odd_squares = [x for x in squares if x not in even_squares]
```

Control Structures

- Loop control statements: change execution from its normal sequence

- `Break`: exit out of a loop when a condition is triggered

```
for number in range(10):
    if number == 5:
        break
    print('Number is ' + str(number))
print('Out of loop')
```

Control Structures

- Loop control statements: change execution from its normal sequence

- `Break`: exit out of a loop when a condition is triggered

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

```
for number in range(10):
    if number == 5:
        break
    print('Number is ' + str(number))
print('Out of loop')
```

Control Structures

- Loop control statements: change execution from its normal sequence

- `break`: exit out of a loop when a condition is triggered

```
for number in range(10):
    if number == 5:
        break
    print('Number is ' + str(number))
print('Out of loop')
```

```
Number is 0
Number is 1
Number is 2
Number is 3
Number is 4
Out of loop
```

Control Structures

- Loop control statements: change execution from its normal sequence

- `continue`: skip over the part of the loop where a condition is triggered, but complete the rest of the loop

```
for number in range(10):
    if number == 5:
        continue
    print('Number is ' + str(number))
print('Out of loop')
```

Control Structures

- Loop control statements: change execution from its normal sequence

- `continue`: skip over the part of the loop where a condition is triggered, but complete the rest of the loop

```
for number in range(10):
    if number == 5:
        continue
    print('Number is ' + str(number))
print('Out of loop')
```

```
Number is 0
Number is 1
Number is 2
Number is 3
Number is 4
Number is 6
Number is 7
Number is 8
Number is 9
Out of loop
```

Control Structures

- Loop control statements: change execution from its normal sequence

- `pass`: handle the condition without the loop being impacted at all

```
for number in range(10):
    if number == 5:
        pass
    print('Number is ' + str(number))
print('Out of loop')
```

Control Structures

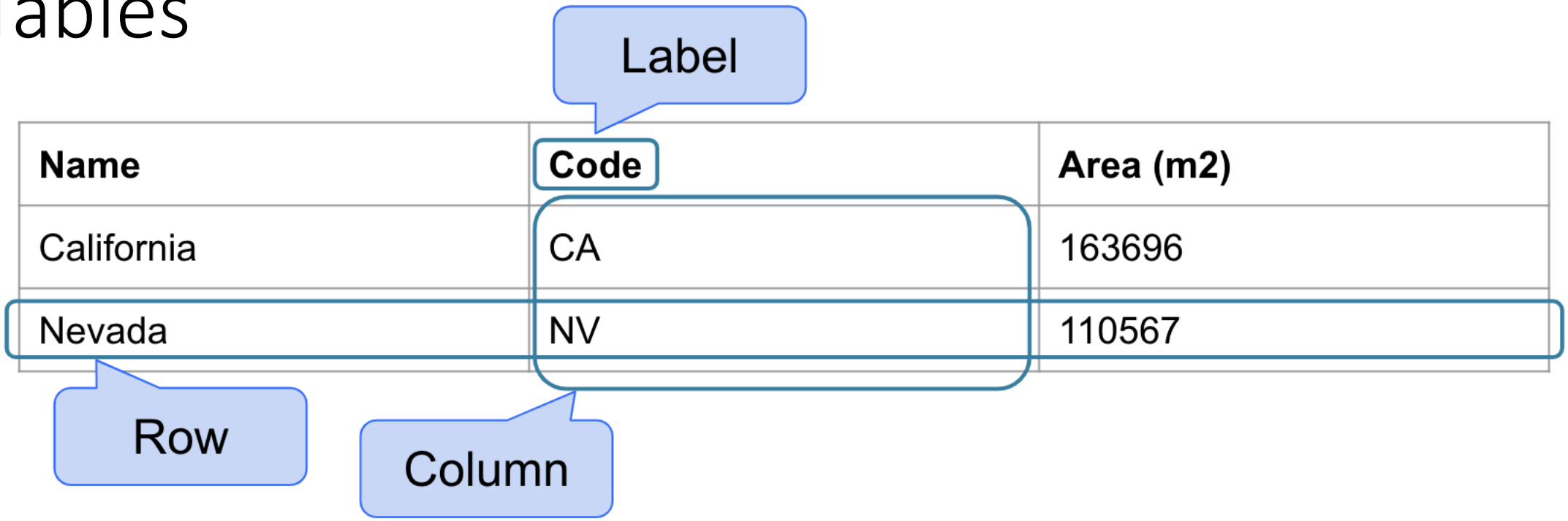
- Loop control statements: change execution from its normal sequence

- `pass`: handle the condition without the loop being impacted at all

```
for number in range(10):
    if number == 5:
        pass
    print('Number is ' + str(number))
print('Out of loop')
```

```
Number is 0
Number is 1
Number is 2
Number is 3
Number is 4
Number is 5
Number is 6
Number is 7
Number is 8
Number is 9
Out of loop
```

Tables



- A Table is a sequence of labeled columns
- Each row represents one individual
- Data within a column represents one attribute of the individuals

Table Operations

- **`t.select(label)`** - constructs a new table with just the specified columns
- **`t.drop(label)`** - constructs a new table in which the specified columns are omitted
- **`t.sort(label)`** - constructs a new table with rows sorted by the specified column
- **`t.where(label, condition)`** - constructs a new table with just the rows that match the condition

Table Operations

- **`t.select(label)`** - constructs a new table with just the specified columns
- **`t.drop(label)`** - constructs a new table in which the specified columns are omitted
- **`t.sort(label)`** - constructs a new table with rows sorted by the specified column
- **`t.where(label, condition)`** - constructs a new table with just the rows that match the condition

Demo: link for your JupyterHub on course website