CE 88

9.12.16

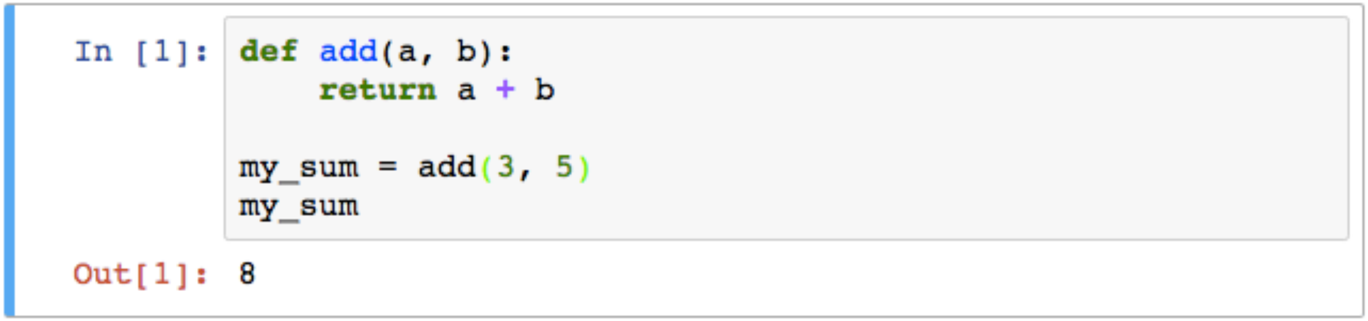
Mini-lab 2: Concepts covered

- Coding concepts
  - Function
  - .apply()
  - .where()
  - .select()
- Other concepts
  - Census data and census tracts
  - Distance on a sphere
  - Interpreting bar graphs

Functions

Functions

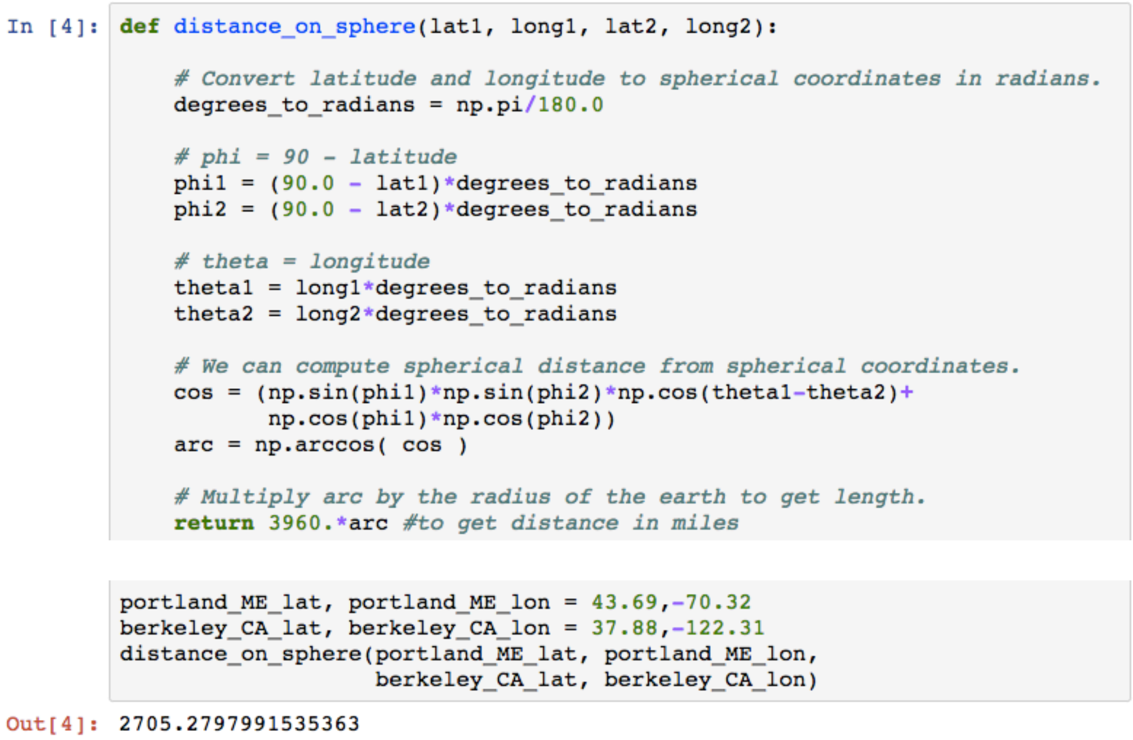
A function called add():



```
In [1]: def add(a, b):
            return a + b

        my_sum = add(3, 5)
        my_sum

Out[1]: 8
```

Function

A function called distance_on_sphere():

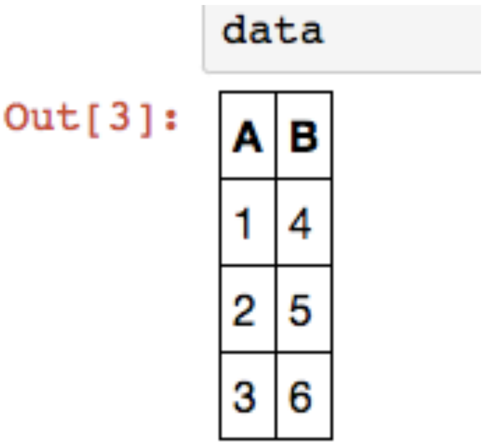


```
In [4]: def distance_on_sphere(lat1, long1, lat2, long2):

            # Convert latitude and longitude to spherical coordinates in radians.
            degrees_to_radians = np.pi/180.0

            # phi = 90 - latitude
            phi1 = (90.0 - lat1)*degrees_to_radians
            phi2 = (90.0 - lat2)*degrees_to_radians

            # theta = longitude
            theta1 = long1*degrees_to_radians
            theta2 = long2*degrees_to_radians

            # We can compute spherical distance from spherical coordinates.
            cos = (np.sin(phi1)*np.sin(phi2)*np.cos(theta1-theta2)+
                   np.cos(phi1)*np.cos(phi2))
            arc = np.arccos( cos )

            # Multiply arc by the radius of the earth to get length.
            return 3960.*arc #to get distance in miles

        portland_ME_lat, portland_ME_lon = 43.69,-70.32
        berkeley_CA_lat, berkeley_CA_lon = 37.88,-122.31
        distance_on_sphere(portland_ME_lat, portland_ME_lon,
                           berkeley_CA_lat, berkeley_CA_lon)

Out[4]: 2705.2797991535363
```

Table.apply()

Table.Apply()

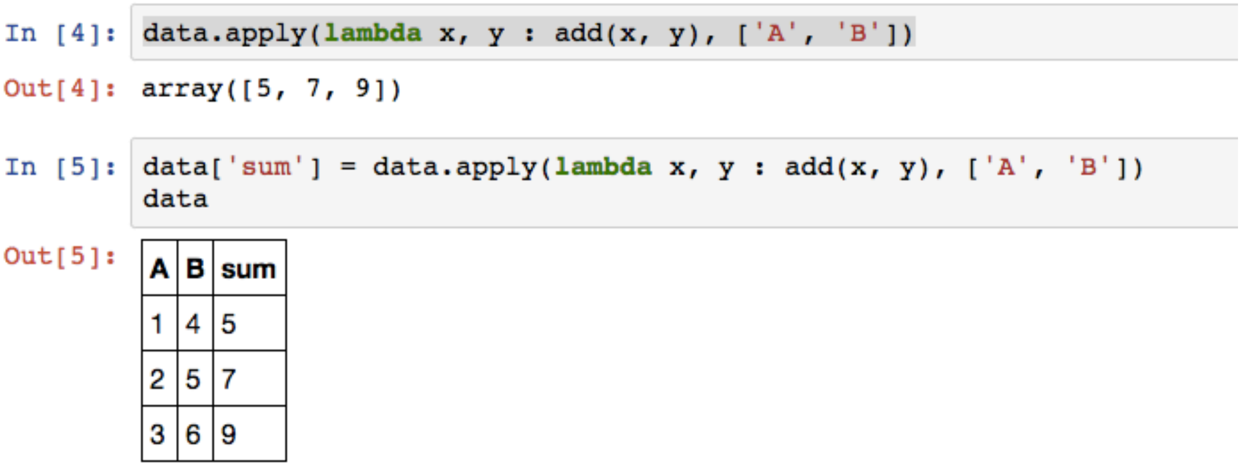
- Table.apply(lambda x, y : my_method(x,y), [‘col_x_name’, ‘col_y_name’])



```
data
```

Out[3]:

A	B
1	4
2	5
3	6



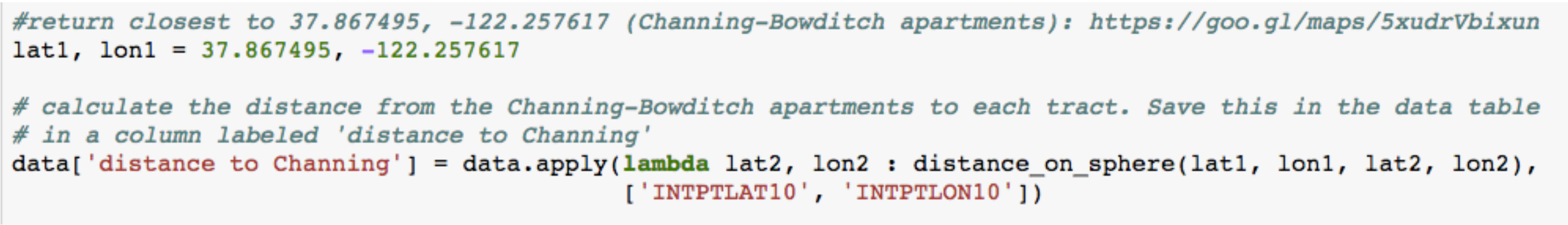
```
In [4]: data.apply(lambda x, y : add(x, y), ['A', 'B'])
Out[4]: array([5, 7, 9])

In [5]: data['sum'] = data.apply(lambda x, y : add(x, y), ['A', 'B'])
        data
```

Out[5]:

A	B	sum
1	4	5
2	5	7
3	6	9

Table.Apply()

- Table.apply(lambda x, y : my_method(x,y), [‘col_x_name’, ‘col_y_name’])



```
#return closest to 37.867495, -122.257617 (Channing-Bowditch apartments): https://goo.gl/maps/5xudrVbixun
lat1, lon1 = 37.867495, -122.257617

# calculate the distance from the Channing-Bowditch apartments to each tract. Save this in the data table
# in a column labeled 'distance to Channing'
data['distance to Channing'] = data.apply(lambda lat2, lon2 : distance_on_sphere(lat1, lon1, lat2, lon2),
                                          ['INTPTLAT10', 'INTPTLON10'])
```

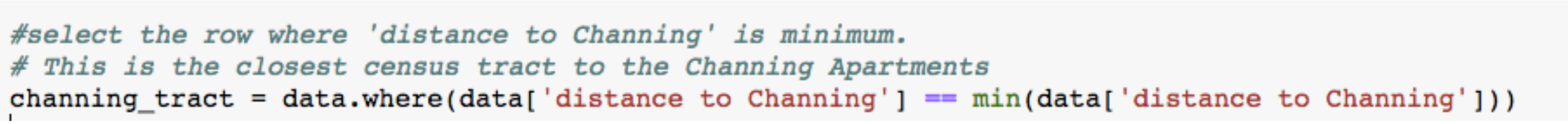
Table.where()

Table.where()

To select only certain rows of the table:

The two lines below are equivalent statements, both return a table with only rows where the statement (data[col_name] == value) is true

- Table.where(col_name, value)
- Table.where(data[col_name] == value)

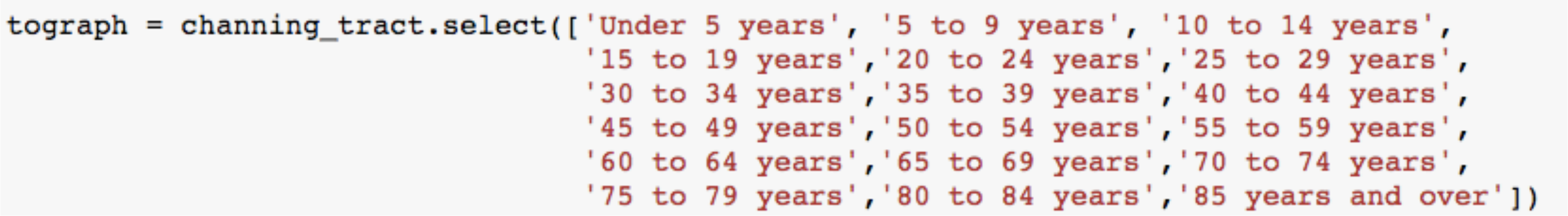


```
#select the row where 'distance to Channing' is minimum.
# This is the closest census tract to the Channing Apartments
channing_tract = data.where(data['distance to Channing'] == min(data['distance to Channing']))
```

Table.select()

Table.select()

- To select only certain columns of the table



```
tograph = channing_tract.select(['Under 5 years', '5 to 9 years', '10 to 14 years',
                                 '15 to 19 years','20 to 24 years','25 to 29 years',
                                 '30 to 34 years','35 to 39 years','40 to 44 years',
                                 '45 to 49 years','50 to 54 years','55 to 59 years',
                                 '60 to 64 years','65 to 69 years','70 to 74 years',
                                 '75 to 79 years','80 to 84 years','85 years and over'])
```