

Bokeh 3.1.1 Dashboard

Part 2

```

1 from bokeh.io import curdoc
2 from bokeh.plotting import figure
3 from bokeh.models import ColumnDataSource, Select
4 from bokeh.layouts import row
5 import pandas as pd
6
7 # read the dataset
8 service_requests = pd.read_csv('../datasets/department-sr-a0.csv', index_col=0)
9
10 # create a blank ColumnDataSource object
11 source = ColumnDataSource(data=dict(x=[], y=[], dept=[]))
12
13 # create the blank figure
14 p = figure()
15
16 # create circle glyphs with wn_y and wn_x coordinates as x and y
17 p.circle('x', 'y', source=source, alpha=0.75)
18
19 # create a department drop down
20 dept = Select(title="Departments", value="INFO",
21              options=['INFO', 'ISD', 'PWDx', 'BTDT', 'PARK', 'PROP', 'ANML'])
22
23 # filter the DataFrame based on the Department Title

```

* Empty ColumnDataSource object

```

26 def select_requests():
27     dept_val = dept.value
28     filtered_df = service_requests
29     filtered_df = filtered_df[
30         filtered_df.Department.str.contains(dept_val) == True]
31     return filtered_df

```

* Filter DataFrame by the Department Value

```

32
33 # update the ColumnDataSource values based on the new filtered df
34
35
36 def update():
37     df = select_requests()
38     source.data = dict(
39         x=df['wn_x'],
40         y=df['wn_y'],
41         dept=df['Department']
42     )
43
44 dept.on_change('value', lambda attr, old, new: update())

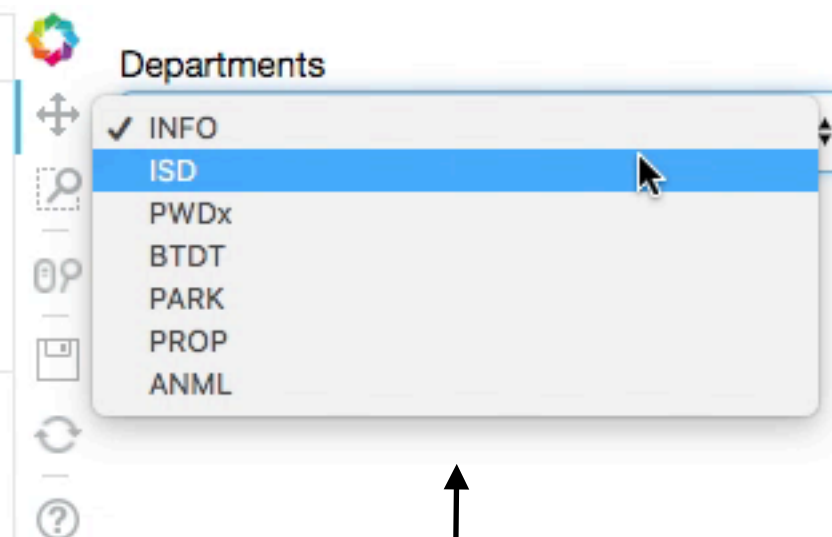
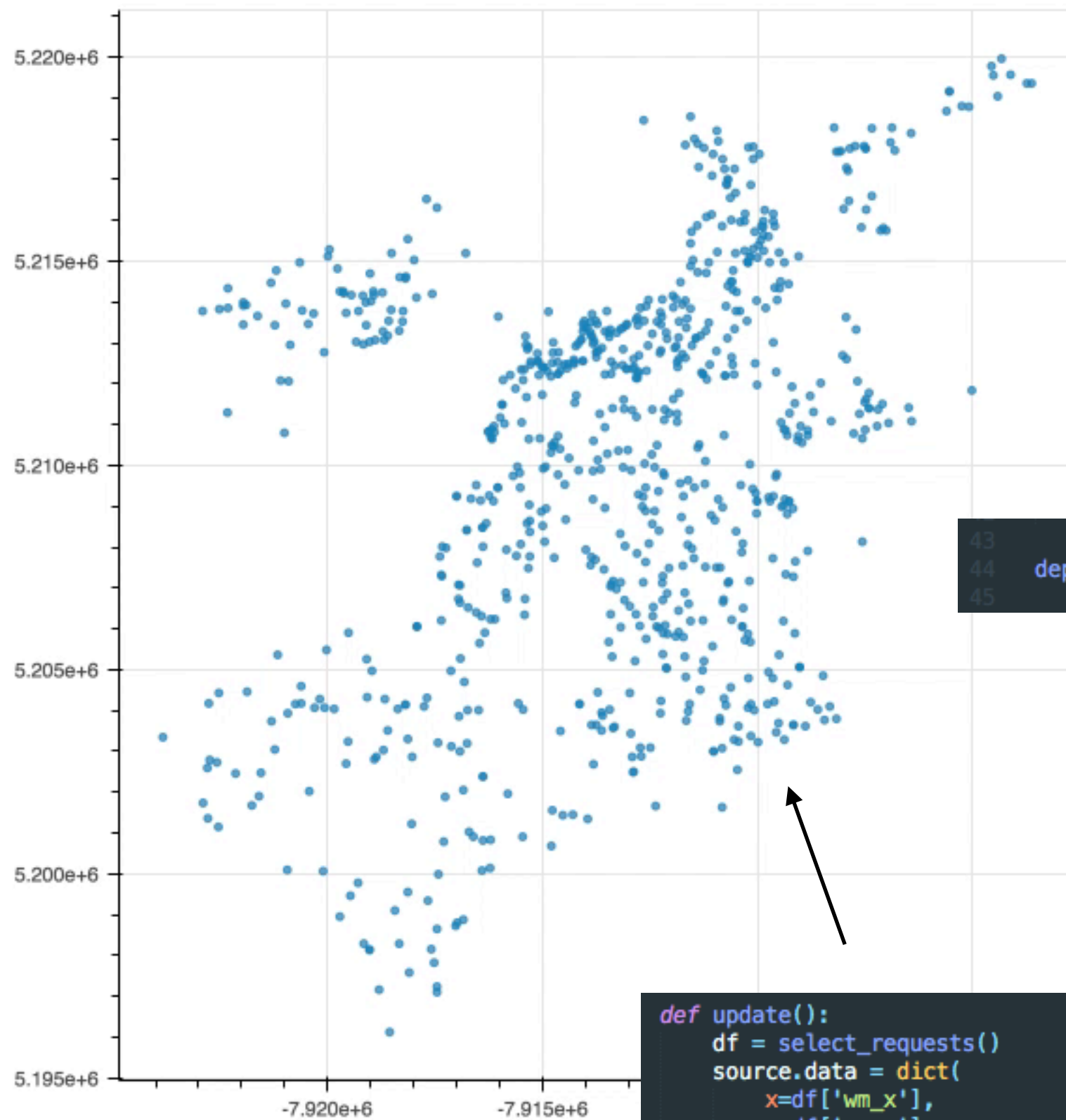
```

* Update the ColumnDataSource object with the values from the filtered DataFrame.

```

45
46 update() # initial load of the data
47
48 # create a layout with one row
49 layout = row(p, dept)
50
51 # add the layout to the current document
52 curdoc().add_root(layout)
53

```



```
43  
44 dept.on_change('value', lambda attr, old, new: update())  
45
```

```
def update():  
    df = select_requests()  
    source.data = dict(  
        x=df['wm_x'],  
        y=df['wm_y'],  
        dept=df['Department']  
    )
```

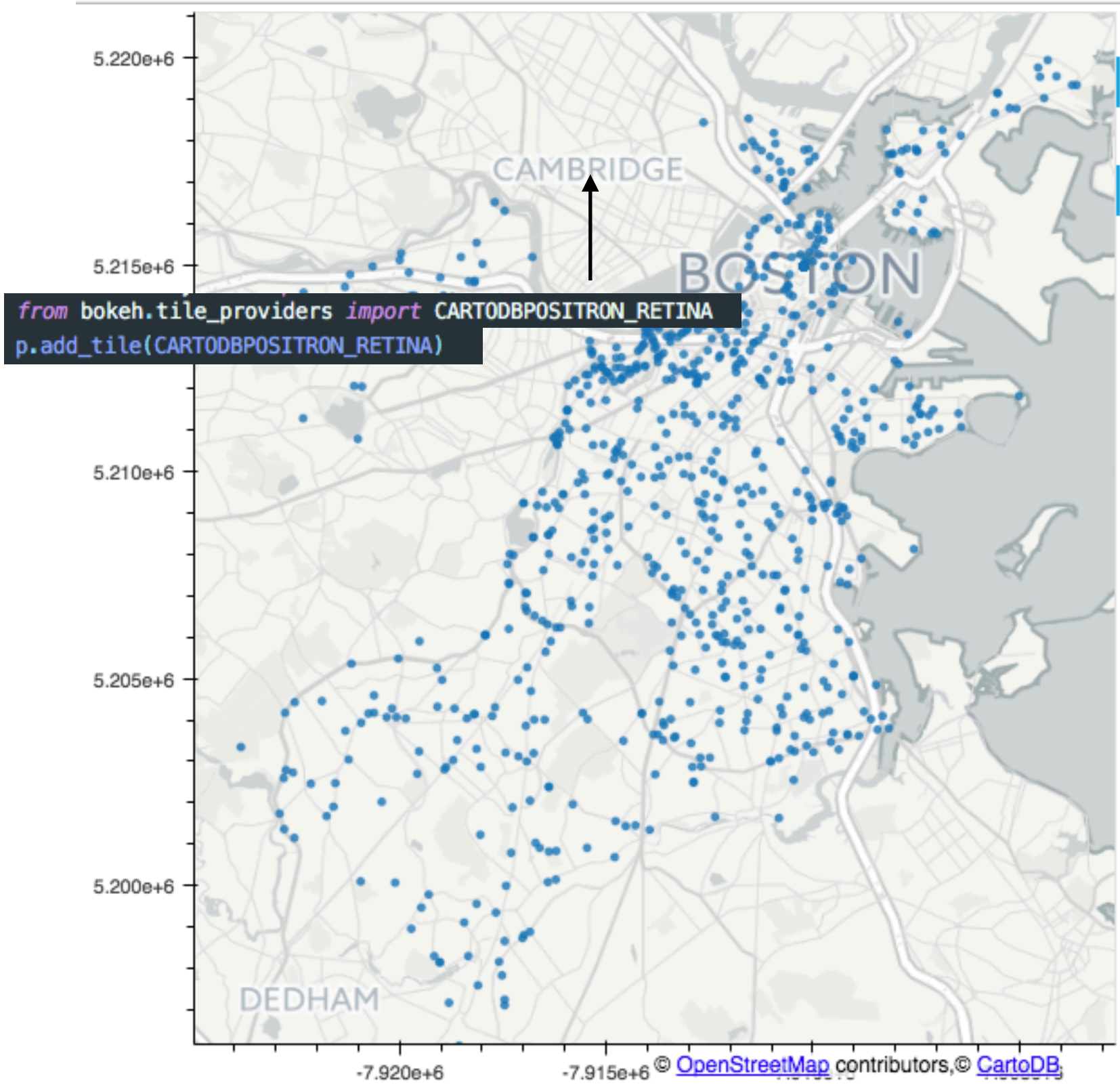
```

1 from bokeh.io import curdoc
2 from bokeh.plotting import figure
3 from bokeh.models import ColumnDataSource, Select
4 from bokeh.layouts import row
5 from bokeh.tile_providers import CARTODBPOSITRON_RETINA
6 import pandas as pd
7
8 # read the dataset
9 service_requests = pd.read_csv('../datasets/department-sr-ao.csv', index_col=0)
10
11 # create a blank ColumnDataSource object
12 source = ColumnDataSource(data=dict(x=[], y=[], dept=[]))
13
14 # create the blank figure & use webgl to use GPU rendering in browser
15 p = figure(webgl=True)
16
17 p.add_tile(CARTODBPOSITRON_RETINA)
18
19 # create circle glyphs with wm_y and wm_x coordinates as x and y
20 p.circle('x', 'y', source=source, alpha=0.8)
21
22 # create a department drop down
23 dept = Select(title="Departments", value="INFO",
24              options=['INFO', 'ISD', 'PWDx', 'BTDT', 'PARK', 'PROP', 'ANML'])
25
26 # filter the DataFrame based on the Department Title
27
28
29 def select_requests():
30     dept_val = dept.value
31     filtered_df = service_requests
32     filtered_df = filtered_df[
33         filtered_df.Department.str.contains(dept_val) == True]
34     return filtered_df
35
36 # update the ColumnDataSource values based on the new filtered df
37
38
39 def update():
40     df = select_requests()
41     source.data = dict(
42         x=df['wm_x'],
43         y=df['wm_y'],
44         dept=df['Department']
45     )
46
47 dept.on_change('value', lambda attr, old, new: update())
48
49 update() # initial load of the data
50
51 # create a layout with one row
52 layout = row(p, dept)
53
54 # add the layout to the current document

```

* Import the mapping tile

* Add the tile to the figure



Departments

INFO

Map navigation controls: pan, zoom, full screen, save, refresh, help.

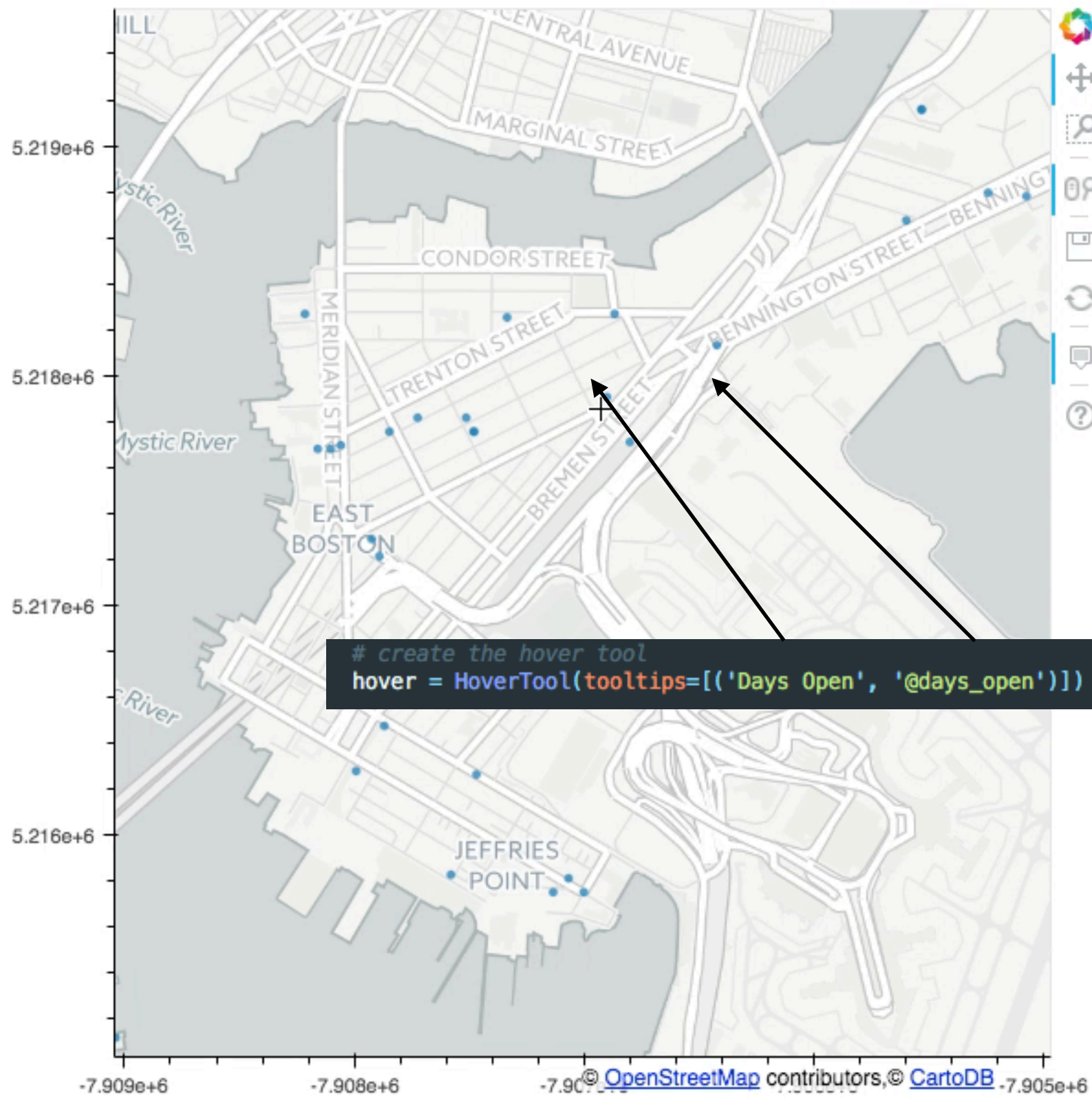

```

> departments-dashboard-103.py x departments-dashboard-104.py x departments-das
1 from bokeh.io import curdoc
2 from bokeh.plotting import figure
3 from bokeh.models import ColumnDataSource, Select, HoverTool
4 from bokeh.layouts import row
5 from bokeh.tile_providers import CARTODBPOSITRON_RETINA
6 import pandas as pd
7
8 # read the dataset
9 service_requests = pd.read_csv('../datasets/department-sr-a0.csv', index_col=0)
10
11 # create a blank ColumnDataSource object
12 source = ColumnDataSource(data=dict(x=[], y=[], dept=[], days_open=[]))
13
14 # create the blank figure & use webgl to use GPU rendering in browser
15 p = figure(webgl=True)
16
17 # create a department drop down
18 dept = Select(title="Departments", value="INFO",
19               options=['INFO', 'ISD', 'PWDx', 'BTDT', 'PARK', 'PROP', 'ANML'])
20
21 # create the hover tool
22 hover = HoverTool(tooltips=[('Days Open', '@days_open')])
23
24 # add layers to the figure
25 p.add_tile(CARTODBPOSITRON_RETINA)
26 p.add_tools(hover)
27
28 # create circle glyphs with wm_y and wm_x coordinates as x and y
29 p.circle('x', 'y', source=source, alpha=0.8)
30
31 # filter the DataFrame based on the Department Title
32
33
34 def select_requests():
35     dept_val = dept.value
36     filtered_df = service_requests
37     filtered_df = filtered_df[
38         filtered_df.Department.str.contains(dept_val) == True]
39     return filtered_df
40
41 # update the ColumnDataSource values based on the new filtered df
42
43
44 def update():
45     df = select_requests()
46     source.data = dict(
47         x=df['wm_x'],
48         y=df['wm_y'],
49         dept=df['Department'],
50         days_open=df['days_open']
51     )
52
53 dept.on_change('value', lambda attr, old, new: update())
54

```

Tooltip title

ColumnDataSource “key”



```
# create the hover tool  
hover = HoverTool(tooltips=[('Days Open', '@days_open')])
```



Departments

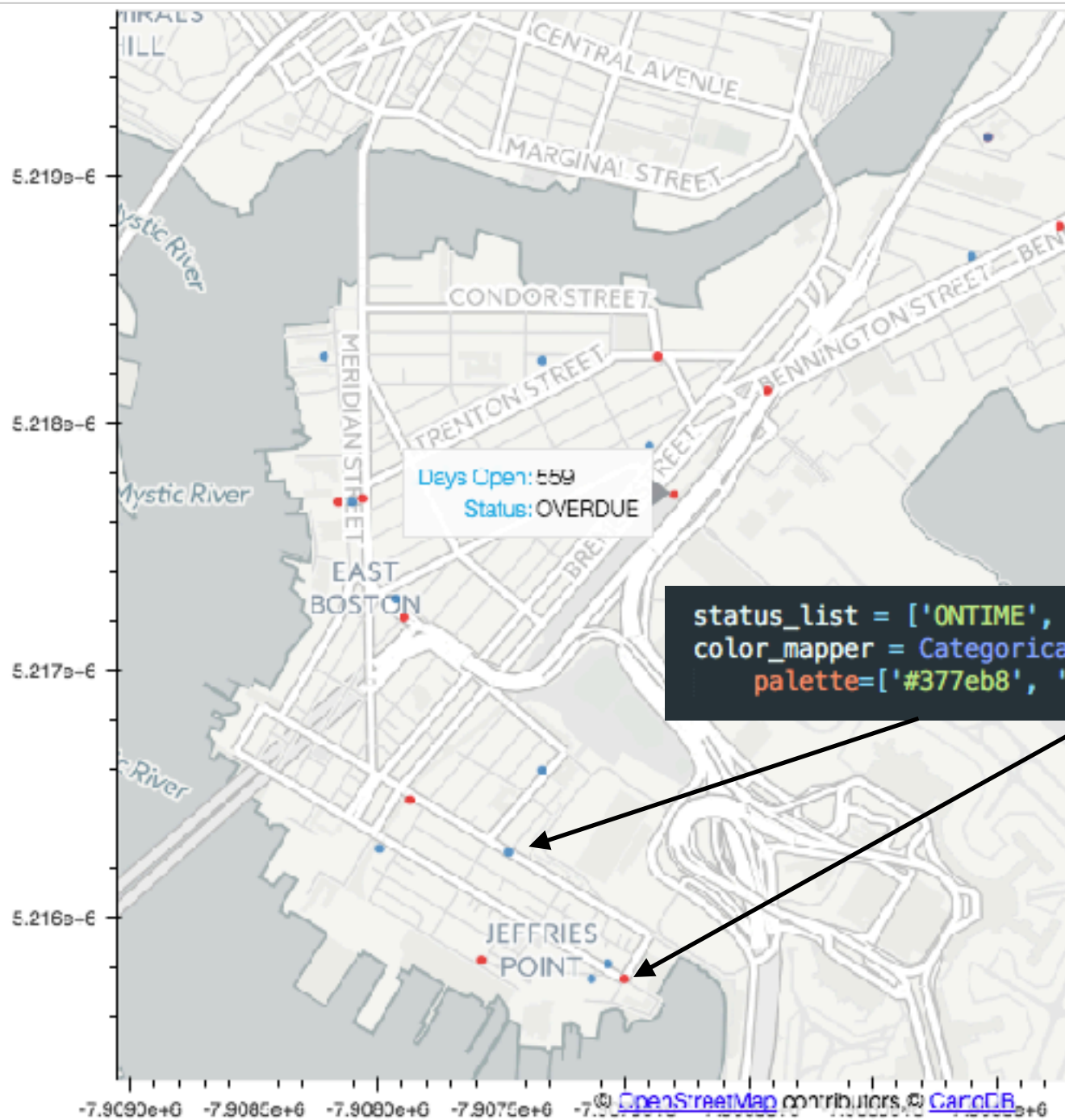
INFO

p.add_tools(hover)

```

1  from bokeh.io import curdoc
2  from bokeh.plotting import figure
3  from bokeh.models import ColumnDataSource, Select, HoverTool, CategoricalColorMapper
4  from bokeh.layouts import row
5  from bokeh.tile_providers import CARTODBPOSITRON_RETINA
6  from bokeh.palettes import Viridis3, Set1
7  import pandas as pd
8
9  # read the dataset
10 service_requests = pd.read_csv('../datasets/department-sr-ao.csv', index_col=0)
11
12 # create a blank ColumnDataSource object
13 source = ColumnDataSource(
14     data=dict(x=[], y=[], dept=[], days_open=[], status=[]))
15
16 # create the blank figure & use webgl to use GPU rendering in browser
17 p = figure(webgl=True)
18
19 # create a department drop down
20 dept = Select(title="Departments", value="INFO",
21              options=['INFO', 'ISD', 'PWX', 'BTDT', 'PARK', 'PROP', 'ANML'])
22
23 # create the hover tool
24 hover = HoverTool(tooltips=[
25     ('Days Open', '@days_open'),
26     ('Status', '@status')])
27
28 # add layers to the figure
29 p.add_tile(CARTODBPOSITRON_RETINA)
30 p.add_tools(hover)
31
32 status_list = ['ONTIME', 'OVERDUE']
33 color_mapper = CategoricalColorMapper(
34     palette=['#377eb8', '#e41alc'], factors=status_list)
35
36 # create circle glyphs with wm_y and wm_x coordinates as x and y
37 p.circle('x', 'y', source=source, alpha=0.8, color={
38     'field': 'status', 'transform': color_mapper})
39
40 # filter the DataFrame based on the Department title
41
42
43
44
45 def select_requests():
46     dept_val = dept.value
47     filtered_df = service_requests
48     filtered_df = filtered_df[
49         filtered_df.Department.str.contains(dept_val) == True]
50     return filtered_df
51
52 # update the ColumnDataSource values based on the new filtered df
53
54

```

Departments

INFO

```
status_list = ['ONTIME', 'OVERDUE']  
color_mapper = CategoricalColorMapper(  
    palette=['#377eb8', '#e41a1c'], factors=status_list)
```

```
1 from bokeh.io import curdoc
2 from bokeh.plotting import figure
3 from bokeh.models import ColumnDataSource, Select, HoverTool, CategoricalColorMapper
4 from bokeh.layouts import row, widgetbox, column
5 from bokeh.tile_providers import CARTODBPOSITRON_RETINA
6 from bokeh.palettes import Viridis3, Set1
7 from bokeh.models.widgets import DataTable, TableColumn
8 import pandas as pd
9
10 # read the dataset
11 service_requests = pd.read_csv('../datasets/department-sr-ao.csv', index_col=0)
12
13 # create a blank ColumnDataSource object
14 source = ColumnDataSource(data=dict(x=[], y=[], dept=[], days_open=[], status=[],
15                                     case_title=[], source=[], id=[], photo=[]))
16
17 # create the blank figure & use webgl to use GPU rendering in browser
18 p = figure(webgl=True)
19
20 # create a department drop down
21 dept = Select(title="Departments", value="INFO",
22               options=['INFO', 'ISD', 'PWX', 'BTDT', 'PARK', 'PROP', 'ANML'])
23
24 # create the hover tool
25 hover = HoverTool(tooltips=[
26     ('Days Open', '@days_open'),
27     ('Status', '@status'),
28     ('Case Title', '@title'),
29     ('Source', '@source')],)
30
31 # add layers to the figure
32 p.add_tile(CARTODBPOSITRON_RETINA)
33 p.add_tools(hover)
34
35 # add a data table
36 columns = [
37     TableColumn(field='id', title='Case ID'),
38     TableColumn(field="days_open", title="Days Open"),
39     TableColumn(field="title", title="Title "),
40     TableColumn(field='queue', title='Work Queue')
41 ]
42 data_table = DataTable(source=source, columns=columns, width=600, height=500)
43
44
45
46 status_list = ['ONTIME', 'OVERDUE']
47 color_mapper = CategoricalColorMapper(
48     palette=['#377eb8', '#e41a1c'], factors=status_list)
49
```

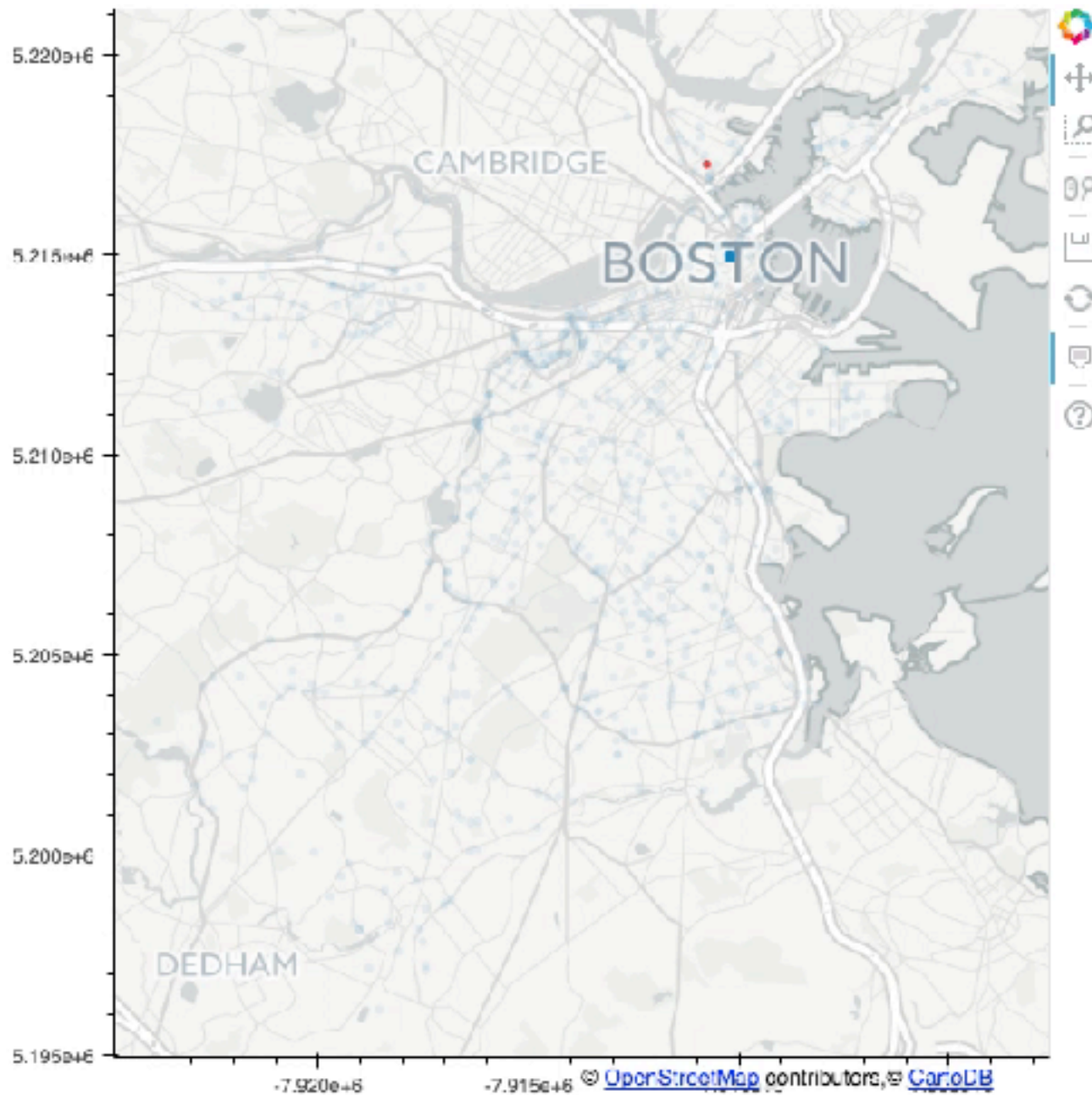
```

46 status_list = ['ONTIME', 'OVERDUE']
47 color_mapper = CategoricalColorMapper(
48     palette=['#377eb8', '#e41a1c'], factors=status_list)
49
50
51 # create circle glyphs with wm_y and wm_x coordinates as x and y
52 p.circle('x', 'y', source=source, alpha=0.8, color={
53     'field': 'status', 'transform': color_mapper})
54
55 # filter the DataFrame based on the Department Title
56
57
58 def select_requests():
59     dept_val = dept.value
60     filtered_df = service_requests
61     filtered_df = filtered_df[
62         filtered_df.Department.str.contains(dept_val) == True]
63     return filtered_df
64
65 # update the ColumnDataSource values based on the new filtered df
66
67
68 def update():
69     df = select_requests()
70     source.data = dict(
71         x=df['wm_x'],
72         y=df['wm_y'],
73         dept=df['Department'],
74         days_open=df['days_open'],
75         status=df['OnTime_Status'],
76         title=df['CASE_TITLE'],
77         source=df['Source'],
78         id=df['CASE_ENQUIRY_ID'],
79         queue=df['QUEUE'],
80     )
81
82 dept.on_change('value', lambda attr, old, new: update())
83
84 update() # initial load of the data
85
86 # create a layout with one row
87 layout = row(p, column(widgetbox(dept, data_table)))
88
89 # add the layout to the current document
90 curdoc().add_root(layout)
91

```



```
# create a layout with one row
layout = row(p, column(widgetbox(dept, data_table)))
```



Departments

INFO				
#	Case ID	Days Open	Title	Work Queue
871	101001000000	1525	Street Light Outages	INFO01_GenericForm...
576	101001000000	702	Street Light Outages	INFO01_GenericForm...
269	101002000000	429	Sign Repair	INFO01_GenericForm...
935	101002000000	507	Sign Repair	INFO01_GenericForm...
220	101001000000	1064	Sign Repair	INFO01_GenericForm...
1142	101002000000	360	Sign Repair	INFO_Mass DCR
1058	101002000000	232	Sign Repair	INFO_Mass DOT
883	101001000000	1203	Sign Repair	INFO01_GenericForm...
924	101001000000	1335	Sign Repair	INFO01_GenericForm...
778	101000000000	1822	Sign Repair	INFO01_GenericForm...
23	101001000000	725	Sign Repair	INFO01_GenericForm...
1282	101001000000	715	Sign Repair	INFO01_GenericForm...
60	101001000000	1388	Sign Repair	INFO01_GenericForm...
290	101002000000	415	Sign Repair	INFO01_GenericForm...
877	101002000000	419	Sign Repair	INFO01_GenericForm...
1224	101001000000	861	Sign Repair	INFO01_GenericForm...
554	101001000000	875	Sign Repair	INFO01_GenericForm...
349	101001000000	377	Sign Repair	INFO01_GenericForm...
447	101001000000	380	Sign Repair	INFO01_GenericForm...

```
# add a data table
columns = [
  TableColumn(field='id', title='Case ID'),
  TableColumn(field="days_open", title="Days Open"),
  TableColumn(field="title", title="Title"),
  TableColumn(field='queue', title='Work Queue')
]
data_table = DataTable(source=source, columns=columns, width=600, height=500)
```



```

< > README.md × departments-dashboard-106.py ×
1 import pandas as pd
2
3 from bokeh.io import curdoc
4 from bokeh.layouts import row, widgetbox, column
5 from bokeh.models import (
6     ColumnDataSource, Select, HoverTool, CategoricalColorMapper,
7     Slider
8 )
9 from bokeh.models.widgets import DataTable, TableColumn
10 from bokeh.plotting import figure
11 from bokeh.tile_providers import CARTODBPOSITRON_RETINA
12
13
14 # read the dataset
15 service_requests = pd.read_csv(
16     '../..../datasets/department-sr-ao.csv', index_col=0)
17
18 # create a blank ColumnDataSource object
19 source = ColumnDataSource(
20     data={'x': [], 'y': [], 'dept': [], 'days_open': [], 'status': [],
21          'case_title': [], 'source': [], 'id': [], 'photo': []})
22
23 # create the blank figure & use webgl to use GPU rendering in browser
24 p = figure(webgl=True)
25
26 # create a department drop down
27 dept = Select(title="Departments", value="INFO", options=[
28     'INFO', 'ISD', 'PWDx', 'BTDT', 'PARK', 'PROP', 'ANML'])
29
30 max_days_open = service_requests['days_open'].max()
31
32 # round down to the nearest 100 to ensure there is always at least one point
33 max_number_100 = max_days_open - (max_days_open % 100)
34
35 # create a slider
36 days_slider = Slider(title='# of days open', value=0,
37                      start=0, end=max_number_100, step=100)
38
39
40 # create the hover tool
41 hover = HoverTool(tooltips=[
42     ('Days Open', '@days_open'),
43     ('Status', '@status'),
44     ('Case Title', '@title'),
45     ('Source', '@source')],)

```

* Create a Slider Object

```

< > README.md x departments-dashboard-106.py x
62 status_list = ['ONTIME', 'OVERDUE']
63 color_mapper = CategoricalColorMapper(
64     palette=['#377eb8', '#e41a1c'], factors=status_list)
65
66
67 # create circle glyphs with wm_y and wm_x coordinates as x and y
68 p.circle('x', 'y', source=source, alpha=0.8, color={
69     'field': 'status', 'transform': color_mapper})
70
71 # filter the DataFrame based on the Department Title
72
73
74 def select_requests():
75     dept_val = dept.value
76     filtered_df = service_requests[
77         (service_requests['days_open'] >= days_slider.value)
78     ]
79     filtered_df = filtered_df[
80         filtered_df.Department.str.contains(dept_val) == True]
81     return filtered_df
82
83 # update the ColumnDataSource values based on the new filtered df
84
85
86 def update():
87     df = select_requests()
88     source.data = {
89         'x': df['wm_x'],
90         'y': df['wm_y'],
91         'dept': df['Department'],
92         'days_open': df['days_open'],
93         'status': df['OnTime_Status'],
94         'title': df['CASE_TITLE'],
95         'source': df['Source'],
96         'id': df['CASE_ENQUIRY_ID'],
97         'queue': df['QUEUE']}
98
99 dept.on_change('value', lambda attr, old, new: update())
100 days_slider.on_change('value', lambda attr, old, new: update())
101
102 update() # initial load of the data
103
104 # create a layout with one row
105 layout = row(p, column(widgetbox(dept, days_slider, data_table)))
106
107 # add the layout to the current document
108 curdoc().add_root(layout)
109

```

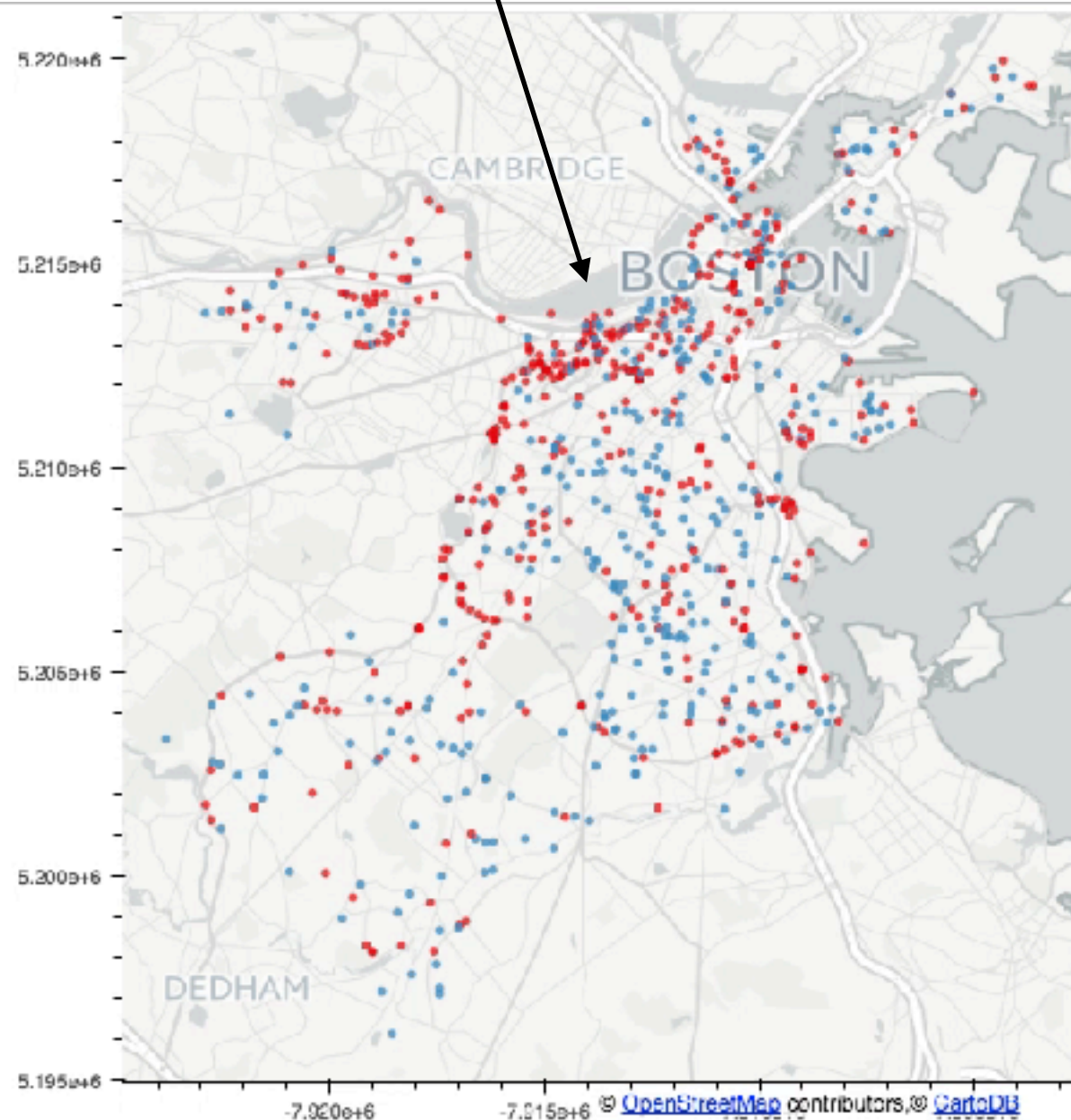
* Filters DataFrame based on days_open column

* Updates value on_change

```
def select_requests():
    dept_val = dept.value
    filtered_df = service_requests[
        (service_requests['days_open'] >= days_slider.value)
    ]
    filtered_df = filtered_df[
        filtered_df.Department.str.contains(dept_val) == True]
    return filtered_df

days_slider.on_change('value', lambda attr, old, new: update())
```

```
# create a slider
days_slider = Slider(title='# of days open', value=0,
                     start=0, end=max_number_100, step=100)
```



Departments

INFO

of days open: 0

#	Case ID	Days Open	Title	Work Queue
0	101002000000	125	Sidewalk Repair (Mak...	INFO_Mass DCR
1	101002000000	236	Other: Comment	INFO_General Comme...
2	101001000000	1053	General Comments Fo...	INFO_General Comme...
3	101001000000	1579	Generic Case	INFO01_Gener ceForm...
4	101001000000	1076	Pick up Dead Animal	INFO01_Gener ceForm...
5	101001000000	867	Graffiti Removal	INFO01_Gener ceForm...
6	101001000000	1212	Animal Generic Reque...	INFO01_Gener ceForm...
7	101003000000	1859	Other: Comment	INFO_General Comme...
8	101002000000	99	Request for Pothole R...	INFO_Mass DCR
9	101001000000	760	Animal Generic Reque...	INFO01_Gener ceForm...
10	101002000000	563	Animal Generic Reque...	INFO01_Gener ceForm...
11	101001000000	651	Animal Generic Reque...	INFO01_Gener ceForm...
12	101001000000	1138	New Sign Crosswalk a...	INFO01_Gener ceForm...
13	101001000000	713	Other: Comment	INFO_General Comme...
14	101001000000	903	Animal Generic Reque...	INFO01_Gener ceForm...
15	101002000000	345	Request for Pothole R...	INFO_Mass DCR
16	101001000000	761	Fire Department	INFO01_Gener ceForm...
17	101001000000	653	Sign Repair	INFO01_Gener ceForm...
18	101001000000	989	Animal Generic Reque...	INFO01_Gener ceForm...

Dashboard 2 - Workshop