Bokeh 311 Dashboard Part 2

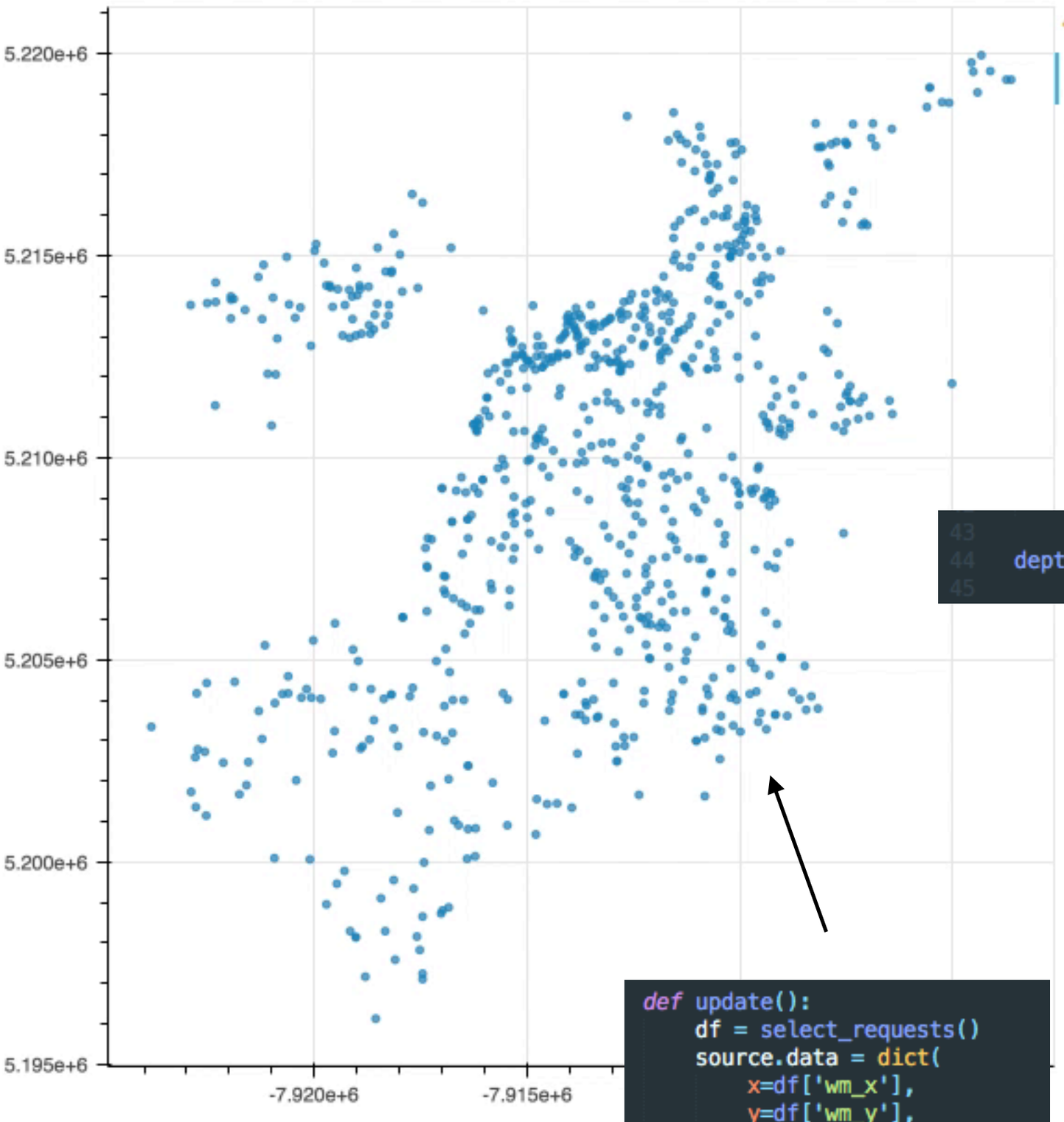
departments-dashboard-101.py × departments-dashboard-102.py × departments-dash

```python
from bokeh.io import curdoc
from bokeh.plotting import figure
from bokeh.models import ColumnDataSource, Select
from bokeh.layouts import row
import pandas as pd

# read the dataset
service_requests = pd.read_csv('../datasets/department-sr-ao.csv', index_col=0)

# create a blank ColumnDataSource object
source = ColumnDataSource(data=dict(x=[], y=[], dept=[]))

# create the blank figure
p = figure()

# create circle glyphs with wm_y and wm_x coordinates as x and y
p.circle('x', 'y', source=source, alpha=0.75)

# create a department drop down
dept = Select(title="Departments", value="INFO",
              options=['INFO', 'ISD', 'PWDx', 'BTDT', 'PARK', 'PROP', 'ANML'])

# filter the DataFrame based on the Department Title


def select_requests():
    dept_val = dept.value
    filtered_df = service_requests
    filtered_df = filtered_df[
        filtered_df.Department.str.contains(dept_val) == True]
    return filtered_df

# update the ColumnDataSource values based on the new filtered df


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

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

update()  # initial load of the data

# create a layout with one row
layout = row(p, dept)

# add the layout to the current document
curdoc().add_root(layout)
```

* Empty ColumnDataSource object

* Filter DataFrame by the Department Value

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



Departments

- ✓ INFO
- ISD
- PWDx
- BTDT
- PARK
- PROP
- ANML

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

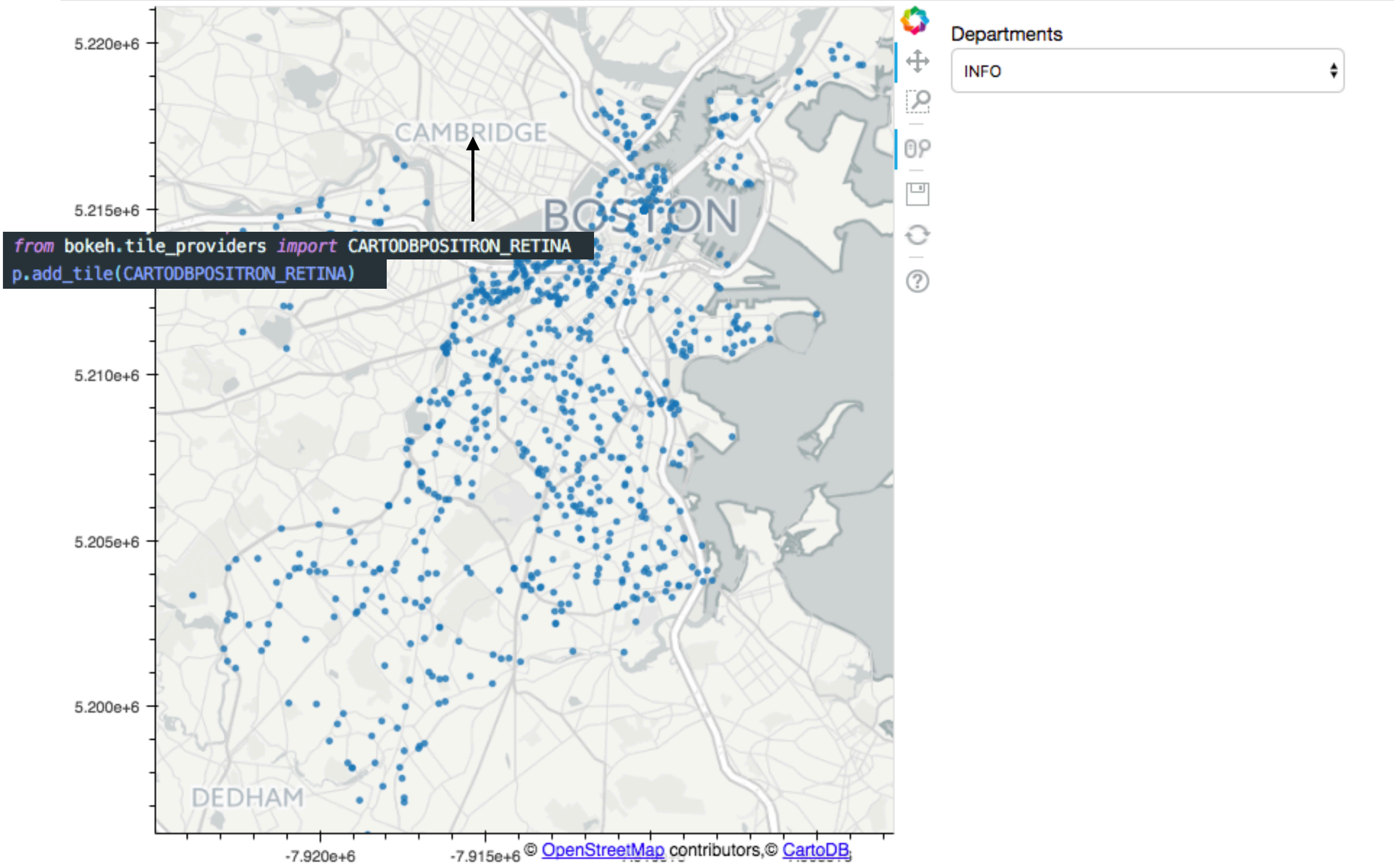
```
def update():
    df = select_requests()
    source.data = dict(
        x=df['wm_x'],
        y=df['wm_y'],
        dept=df['Department']
    )
```

departments-dashboard-101.py × departments-dashboard-102.py × departments-dashb

```python
from bokeh.io import curdoc
from bokeh.plotting import figure
from bokeh.models import ColumnDataSource, Select
from bokeh.layouts import row
from bokeh.tile_providers import CARTODBPOSITRON_RETINA
import pandas as pd

# read the dataset
service_requests = pd.read_csv('../datasets/department-sr-ab.csv', index_col=0)

# create a blank ColumnDataSource object
source = ColumnDataSource(data=dict(x=[], y=[], dept=[]))

# create the blank figure & use webgl to use GPU rendering in browser
p = figure(webgl=True)

p.add_tile(CARTODBPOSITRON_RETINA)

# create circle glyphs with wm_y and wm_x coordinates as x and y
p.circle('x', 'y', source=source, alpha=0.8)

# create a department drop down
dept = Select(title="Departments", value="INFO",
              options=['INFO', 'ISD', 'PWDx', 'BTDT', 'PARK', 'PROP', 'ANML'])

# filter the DataFrame based on the Department Title


def select_requests():
    dept_val = dept.value
    filtered_df = service_requests
    filtered_df = filtered_df[
        filtered_df.Department.str.contains(dept_val) == True]
    return filtered_df

# update the ColumnDataSource values based on the new filtered df


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

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

update()  # initial load of the data

# create a layout with one row
layout = row(p, dept)

# add the layout to the current document
```

* Import the mapping tile

* Add the tile to the figure



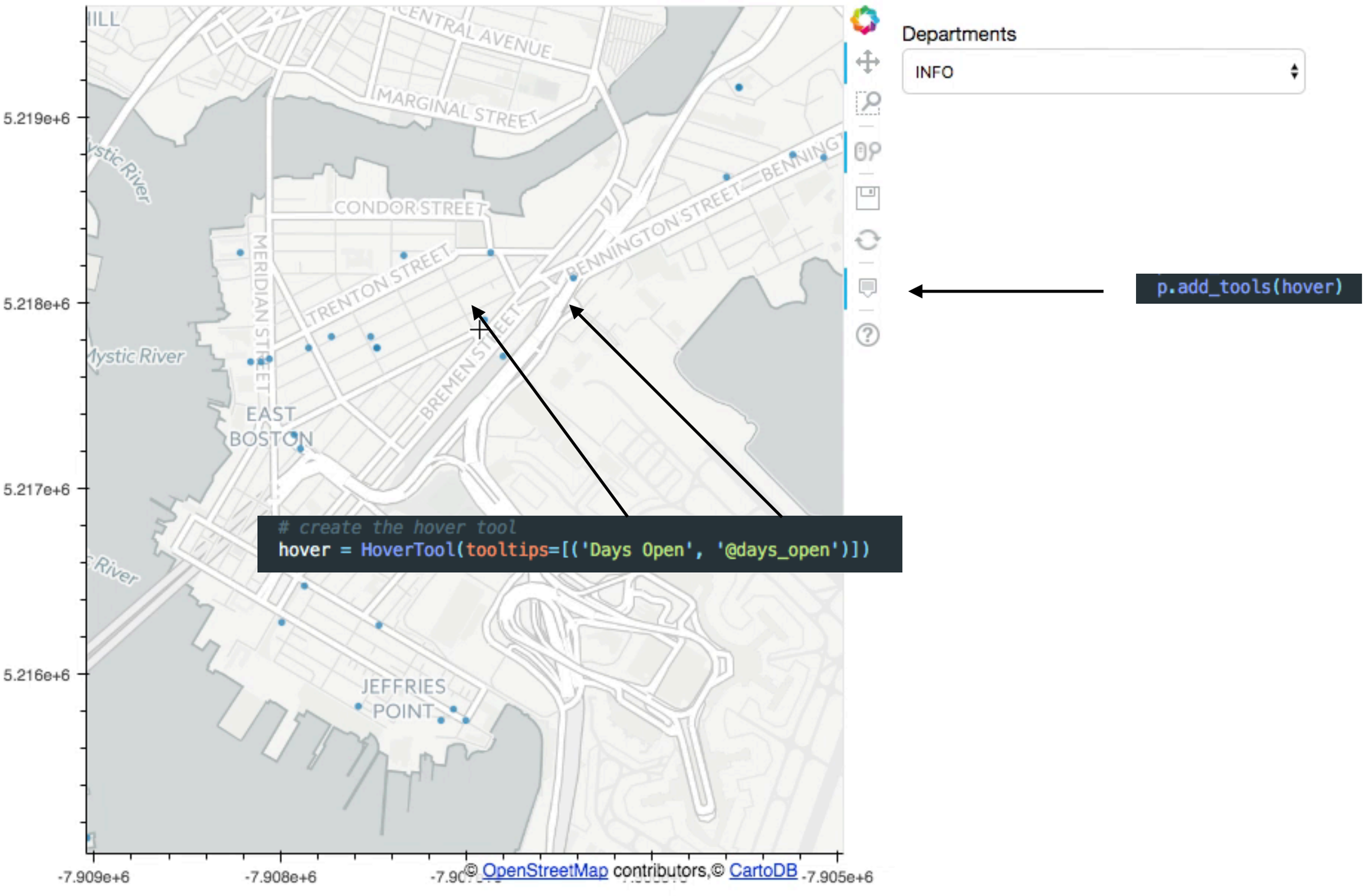

```
from bokeh.tile_providers import CARTODBPOSITRON_RETINA
p.add_tile(CARTODBPOSITRON_RETINA)
```

departments-dashboard-103.py × departments-dashboard-104.py × departments-das

```python
from bokeh.io import curdoc
from bokeh.plotting import figure
from bokeh.models import ColumnDataSource, Select, HoverTool
from bokeh.layouts import row
from bokeh.tile_providers import CARTODBPOSITRON_RETINA
import pandas as pd

# read the dataset
service_requests = pd.read_csv('../datasets/department-sr-ao.csv', index_col=0)

# create a blank ColumnDataSource object
source = ColumnDataSource(data=dict(x=[], y=[], dept=[], days_open=[]))

# create the blank figure & use webgl to use GPU rendering in browser
p = figure(webgl=True)


# create a department drop down
dept = Select(title="Departments", value="INFO",
              options=['INFO', 'ISD', 'PWD', 'BTDT', 'PARK', 'PROP', 'ANML'])
# create the hover tool
hover = HoverTool(tooltips=[('Days Open', '@days_open')])

# add layers to the figure
p.add_tile(CARTODBPOSITRON_RETINA)
p.add_tools(hover)

# create circle glyphs with wm_y and wm_x coordinates as x and y
p.circle('x', 'y', source=source, alpha=0.8)

# filter the DataFrame based on the Department Title


def select_requests():
    dept_val = dept.value
    filtered_df = service_requests
    filtered_df = filtered_df[
        filtered_df.Department.str.contains(dept_val) == True]
    return filtered_df

# update the ColumnDataSource values based on the new filtered df


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

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

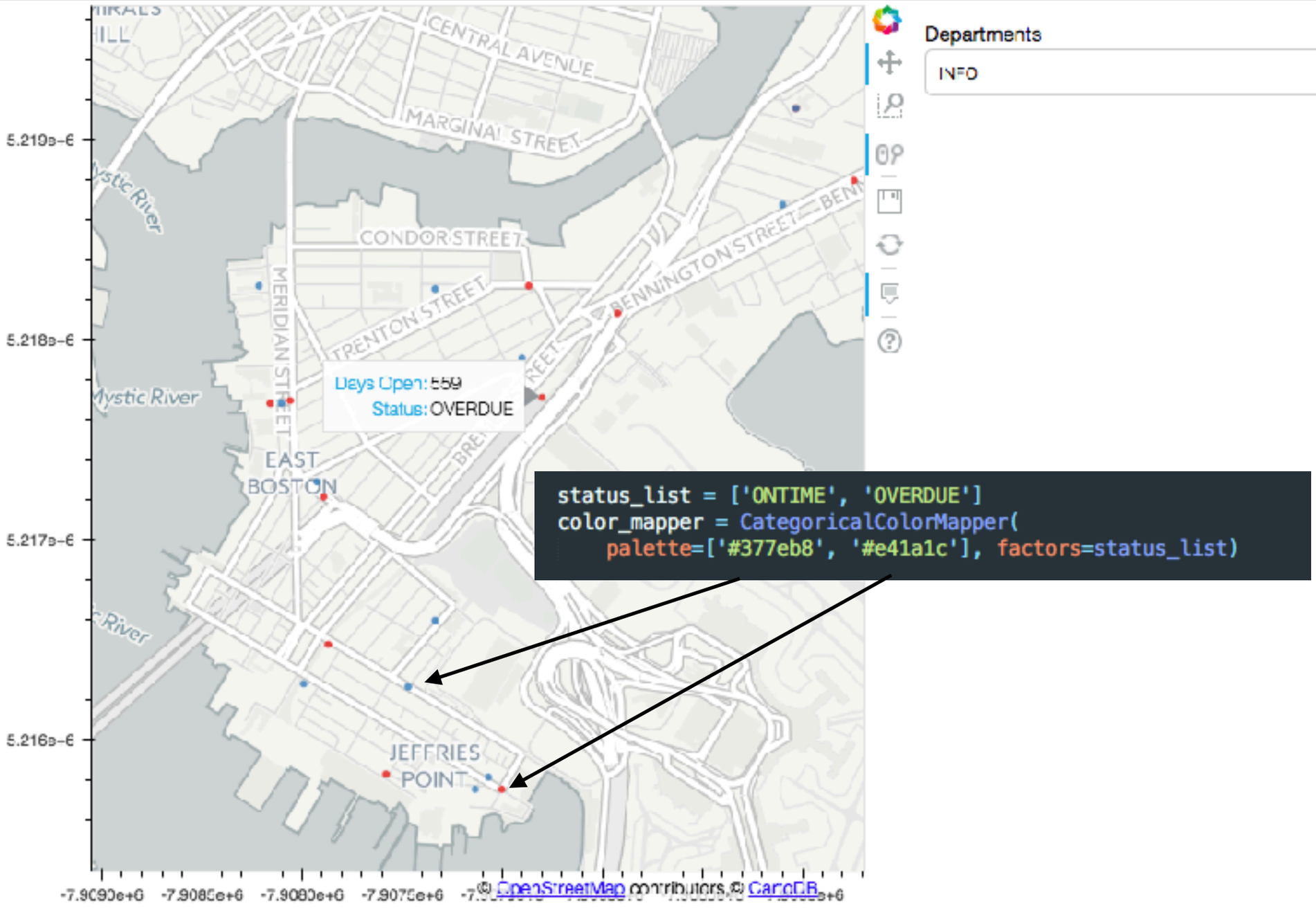
Tooltip title

ColumnDataSource “key”


Departments
INFO
p.add_tools(hover)
# create the hover tool
hover = HoverTool(tooltips=[('Days Open', '@days_open')])
5.219e+6
5.218e+6
5.217e+6
5.216e+6
-7.909e+6
-7.908e+6
-7.905e+6
CENTRAL AVENUE
MARGINAL STREET
CONDOR STREET
TRENTON STREET
BENNINGTON STREET
MERIDIAN STREET
BREMEN STREET
EAST BOSTON
JEFFRIES POINT
Mystic River
© OpenStreetMap contributors,© CartoDB

departments-dashboard-103.py × departments-dashboard-104.py × departments-dashboar

```python
from bokeh.io import curdoc
from bokeh.plotting import figure
from bokeh.models import ColumnDataSource, Select, HoverTool, CategoricalColorMapper
from bokeh.layouts import row
from bokeh.tile_providers import CARTODBPOSITRON_RETINA
from bokeh.palettes import Viridis3, Set1
import pandas as pd

# read the dataset
service_requests = pd.read_csv('../datasets/department-sr-ao.csv', index_col=0)

# create a blank ColumnDataSource object
source = ColumnDataSource(
    data=dict(x=[], y=[], dept=[], days_open=[], status=[]))

# create the blank figure & use webgl to use GPU rendering in browser
p = figure(webgl=True)

# create a department drop down
dept = Select(title="Departments", value="INFO",
              options=['INFO', 'ISD', 'PWDx', 'BTDT', 'PARK', 'PROP', 'ANML'])

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

# add layers to the figure
p.add_tile(CARTODBPOSITRON_RETINA)
p.add_tools(hover)

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

# create circle glyphs with wm_y and wm_x coordinates as x and y
p.circle('x', 'y', source=source, alpha=0.8, color={
         'field': 'status', 'transform': color_mapper})

# filter the DataFrame based on the Department Title


def select_requests():
    dept_val = dept.value
    filtered_df = service_requests
    filtered_df = filtered_df[
        filtered_df.Department.str.contains(dept_val) == True]
    return filtered_df

# update the ColumnDataSource values based on the new filtered df
```

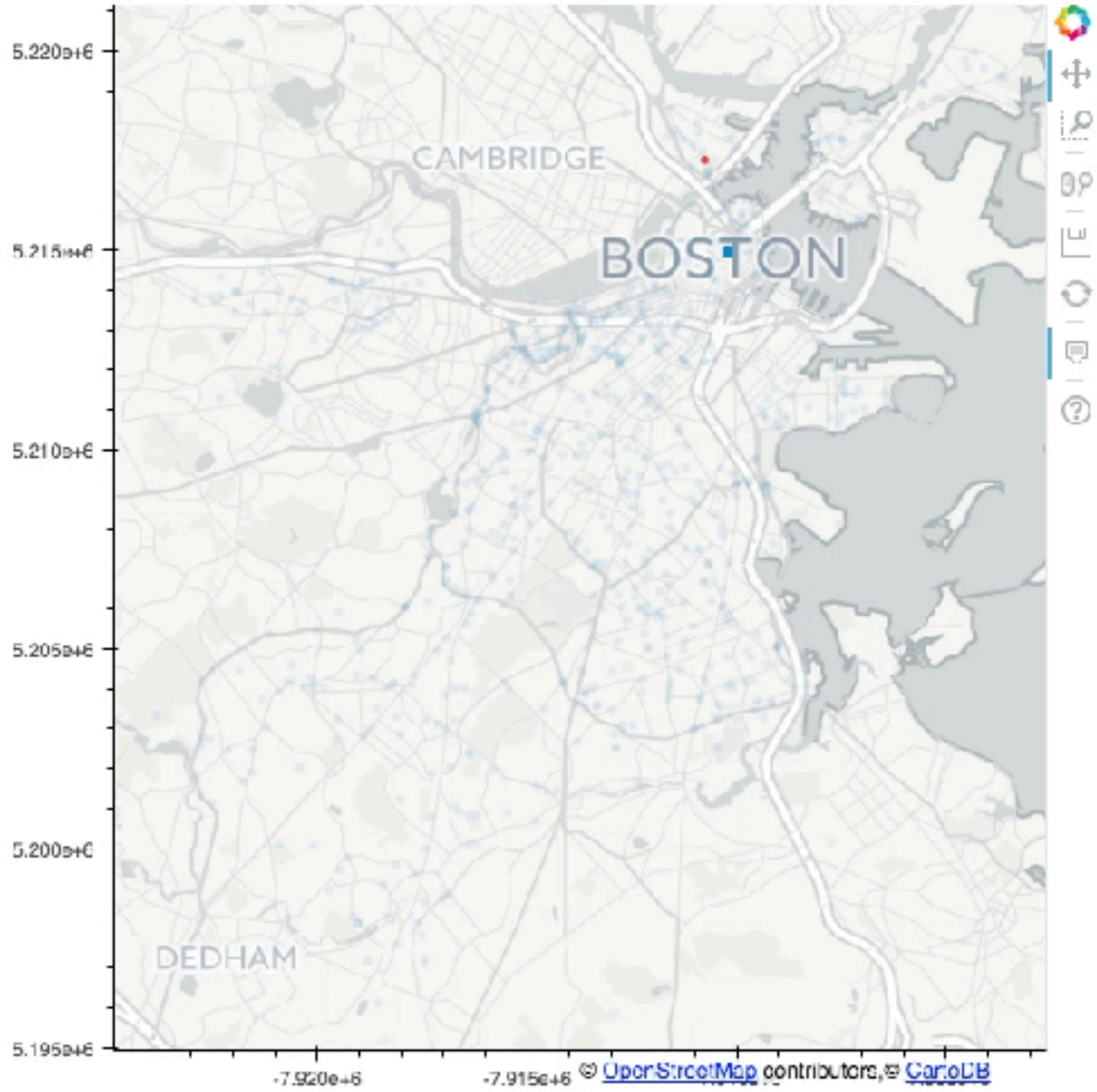

Departments
INFO
CENTRAL AVENUE
MARGINAL STREET
CONDOR STREET
TRENTON STREET
BENNINGTON STREET
MERIDIAN STREET
EAST BOSTON
JEFFRIES POINT
Mystic River
Days Open: 559
Status: OVERDUE
5.219e-6
5.218e-6
5.217e-6
5.216e-6
-7.909e+6
-7.908e+6
-7.908e+6
-7.907e+6
© OpenStreetMap contributors © CartoDB
status_list = ['ONTIME', 'OVERDUE']
color_mapper = CategoricalColorMapper(
palette=['#377eb8', '#e41a1c'], factors=status_list)

departments-dashboard-105.py

```python
from bokeh.io import curdoc
from bokeh.plotting import figure
from bokeh.models import ColumnDataSource, Select, HoverTool, CategoricalColorMapper
from bokeh.layouts import row, widgetbox, column
from bokeh.tile_providers import CARTODBPOSITRON_RETINA
from bokeh.palettes import Viridis3, Set1
from bokeh.models.widgets import DataTable, TableColumn
import pandas as pd

# read the dataset
service_requests = pd.read_csv('../datasets/department-sr-ao.csv', index_col=0)

# create a blank ColumnDataSource object
source = ColumnDataSource(data=dict(x=[], y=[], dept=[], days_open=[], status=[],
                                    case_title=[], source=[], id=[], photo=[]))

# create the blank figure & use webgl to use GPU rendering in browser
p = figure(webgl=True)

# create a department drop down
dept = Select(title="Departments", value="INFO",
              options=['INFO', 'ISD', 'PWDx', 'BTDT', 'PARK', 'PROP', 'ANML'])

# create the hover tool
hover = HoverTool(tooltips=[
    ('Days Open', '@days_open'),
    ('Status', '@status'),
    ('Case Title', '@title'),
    ('Source', '@source')],)

# add layers to the figure
p.add_tile(CARTODBPOSITRON_RETINA)
p.add_tools(hover)


# 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)


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

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


# create circle glyphs with wm_y and wm_x coordinates as x and y
p.circle('x', 'y', source=source, alpha=0.8, color={
          'field': 'status', 'transform': color_mapper})

# filter the DataFrame based on the Department Title


def select_requests():
    dept_val = dept.value
    filtered_df = service_requests
    filtered_df = filtered_df[
        filtered_df.Department.str.contains(dept_val) == True]
    return filtered_df

# update the ColumnDataSource values based on the new filtered df


def update():
    df = select_requests()
    source.data = dict(
        x=df['wm_x'],
        y=df['wm_y'],
        dept=df['Department'],
        days_open=df['days_open'],
        status=df['OnTime_Status'],
        title=df['CASE_TITLE'],
        source=df['Source'],
        id=df['CASE_ENQUIRY_ID'],
        queue=df['QUEUE'],
    )

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

update()  # initial load of the data

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

# add the layout to the current document
curdoc().add_root(layout)

```

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



Departments

INFO

#	Case ID	Days Open	Title	Work Queue
871	10100100C000	1525	Street Light Outages	INFOC1_GenericsForm..
576	10100100C000	702	Street Light Outages	INFOC1_GenericsForm..
269	10100200C000	429	Sign Repair	INFOC1_GenericsForm..
935	10100200C000	507	Sign Repair	INFOC1_GenericsForm..
220	10100100C000	1061	Sign Repair	INFOC1_GenericsForm..
1142	10100200C000	360	Sign Repair	INFO_Mass DCR
1058	10100200C000	232	Sign Repair	INFO_Mass DOT
863	10100100C000	1293	Sign Repair	INFOC1_GenericsForm..
924	10100100C000	1335	Sign Repair	INFOC1_GenericsForm..
778	10100000C000	1822	Sign Repair	INFOC1_GenericsForm..
23	10100100C000	725	Sign Repair	INFOC1_GenericsForm..
1262	10100100C000	715	Sign Repair	INFOC1_GenericsForm..
60	10100100C000	1388	Sign Repair	INFOC1_GenericsForm..
250	10100200C000	415	Sign Repair	INFOC1_GenericsForm..
877	10100200C000	419	Sign Repair	INFOC1_GenericsForm..
1224	10100100C000	861	Sign Repair	INFOC1_GenericsForm..
554	10100100C000	875	Sign Repair	INFOC1_GenericsForm..
349	10100100C000	377	Sign Repair	INFOC1_GenericsForm..
447	10100100C000	880	Sign Repair	INFOC1_GenericsForm..

```
# 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 ×

```python
import pandas as pd

from bokeh.io import curdoc
from bokeh.layouts import row, widgetbox, column
from bokeh.models import (
    ColumnDataSource, Select, HoverTool, CategoricalColorMapper,
    Slider
)
from bokeh.models.widgets import DataTable, TableColumn
from bokeh.plotting import figure
from bokeh.tile_providers import CARTODBPOSITRON_RETINA


# read the dataset
service_requests = pd.read_csv(
    '../../datasets/department-sr-ao.csv', index_col=0)

# create a blank ColumnDataSource object
source = ColumnDataSource(
    data={'x': [], 'y': [], 'dept': [], 'days_open': [], 'status': [],
          'case_title': [], 'source': [], 'id': [], 'photo': []})

# create the blank figure & use webgl to use GPU rendering in browser
p = figure(webgl=True)

# create a department drop down
dept = Select(title="Departments", value="INFO", options=[
                'INFO', 'ISD', 'PWDx', 'BTDT', 'PARK', 'PROP', 'ANML'])

max_days_open = service_requests['days_open'].max()

# round down to the nearest 100 to ensure there is always at least one point
max_number_100 = max_days_open - (max_days_open % 100)

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


# create the hover tool
hover = HoverTool(tooltips=[
    ('Days Open', '@days_open'),
    ('Status', '@status'),
    ('Case Title', '@title'),
    ('Source', '@source')],)
```

* Create a Slider Object

README.md | departments-dashboard-106.py

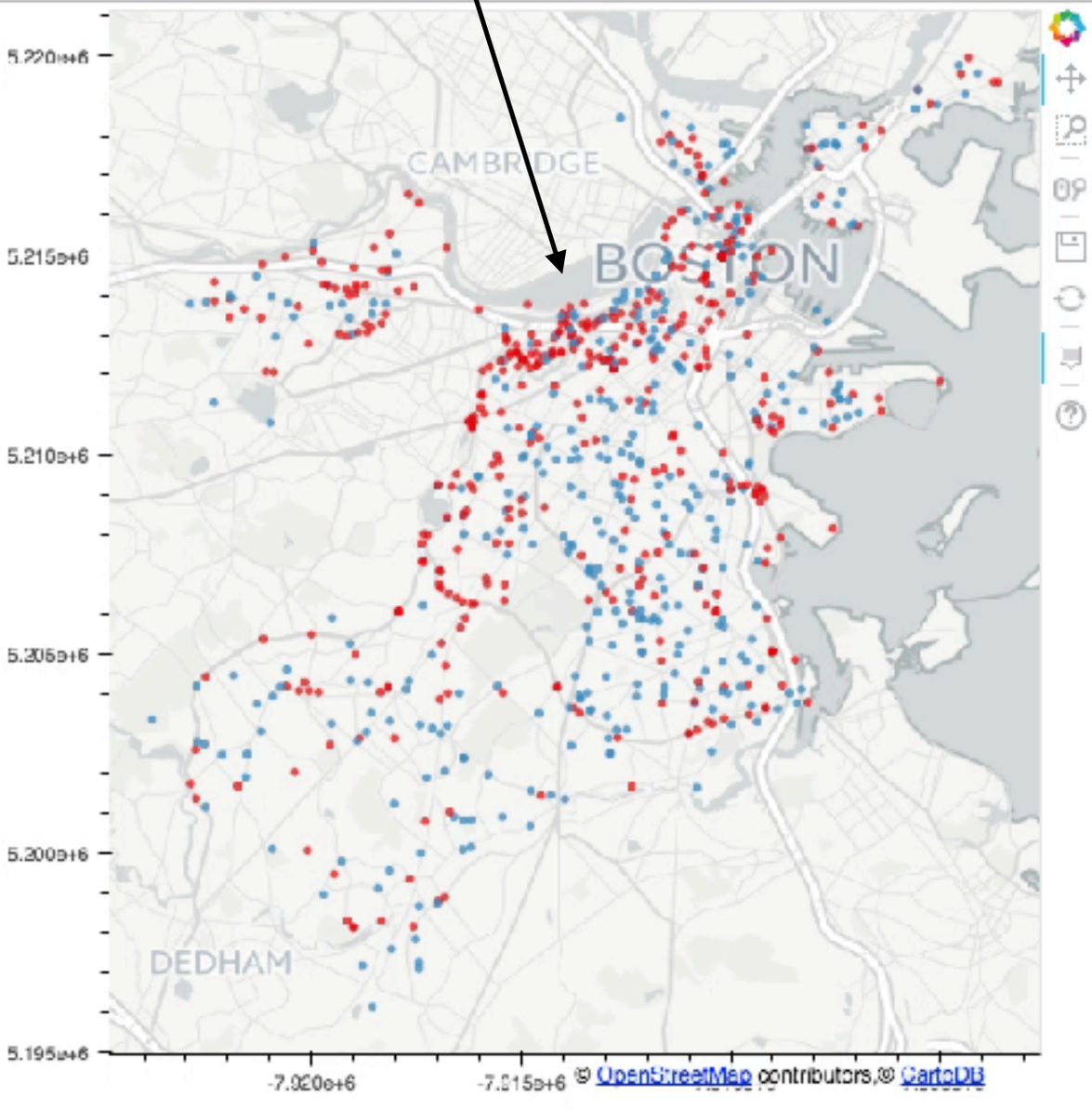
```
status_list = ['ONTIME', 'OVERDUE']
color_mapper = CategoricalColorMapper(
    palette=['#377eb8', '#e41a1c'], factors=status_list)


# create circle glyphs with wm_y and wm_x coordinates as x and y
p.circle('x', 'y', source=source, alpha=0.8, color={
         'field': 'status', 'transform': color_mapper})

# filter the DataFrame based on the Department Title


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

# update the ColumnDataSource values based on the new filtered df


def update():
    df = select_requests()
    source.data = {
        'x': df['wm_x'],
        'y': df['wm_y'],
        'dept': df['Department'],
        'days_open': df['days_open'],
        'status': df['OnTime_Status'],
        'title': df['CASE_TITLE'],
        'source': df['Source'],
        'id': df['CASE_ENQUIRY_ID'],
        'queue': df['QUEUE']}

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

update()  # initial load of the data

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

# add the layout to the current document
curdoc().add_root(layout)
```

* 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	101002030000	125	Sidewalk Repair (Mak...	INFO_Mass DCR
1	101002030000	206	Other: Comment	INFO_General Comme...
2	101001030000	1053	General Comments Fo...	INFO_General Comme...
3	101001030000	1579	Generic Case	INFO01_GenericForm...
4	101001030000	1076	Pick up Dead Animal	INFO01_GenericForm...
5	101001030000	867	Graffiti Removal	INFO01_GenericForm...
6	101001030000	1212	Animal Generic Reque...	INFO01_GenericForm...
7	101003030000	1859	Other: Comment	INFO_General Comme...
8	101002030000	99	Request for Pothole R...	INFO_Mass DCR
9	101001030000	760	Animal Generic Reque...	INFO01_GenericForm...
10	101002030000	563	Animal Generic Reque...	INFO01_GenericForm...
11	101001030000	651	Animal Generic Reque...	INFO01_GenericForm...
12	101001030000	1138	New Sign Crosswalk o...	INFO01_GenericForm...
13	101001030000	713	Other: Comment	INFO_General Comme...
14	101001030000	903	Animal Generic Reque...	INFO01_GenericForm...
15	101002030000	345	Request for Pothole R...	INFO_Mass DCR
16	101001030000	761	Fire Department	INFO01_GenericForm...
17	101001030000	658	Sign Repair	INFO01_GenericForm...
18	101001030000	989	Animal Generic Reque...	INFO01_GenericForm...

Dashboard 2 - Workshop