```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 20 12:41:54 2020

@author: priyashaji
"""


## ------ Import the required packages and libraries


import pandas as pd
from pandas.api.types import CategoricalDtype
import numpy as np
import dash
import dash_auth
from dash.dependencies import Input, Output
import dash_core_components as dcc
import dash_html_components as html

import plotly.offline as py
import plotly.graph_objs as go

from pyproj import Proj, transform

import matplotlib.pyplot as plt

import datashader as ds
import datashader.transfer_functions as tf
import datashader.glyphs
from datashader import reductions
from datashader.core import bypixel
from datashader.utils import lnglat_to_meters as webm, export_image
from datashader.colors import colormap_select, Greys9, viridis, inferno
from functools import partial



## ------ Load the Trees dataset using Socrata API

url = 'https://data.cityofnewyork.us/resource/uvpi-gqnh.json'
trees = pd.read_json(url)
#trees.head(10)
#for col in trees.columns:
 #   print(col)


## ------ No. of rows and columns of trees dataset


trees.shape

## ------ Forming the categories of tree quality and replacing the `na` values



category_type = CategoricalDtype(categories=["Poor", "Fair", "Good"], ordered =
True)
```

```
trees['health'] = trees['health'].astype(category_type)
print(trees['health'].describe())
print(trees['health'].cat.codes[:10])
trees['health'].isna().sum()
trees['health'] = trees['health'].fillna('Fair')
trees['health'] = trees['health'].cat.codes
print(trees['health'][:10])
print(trees['health'].describe())



## ------ Replace `na` in `steward` column with `none`

trees['steward'] = trees['steward'].fillna('None')


## ------ Plot histogram for health category of trees


trees['health'].hist()
plt.show()


## ------ Initialize datashader plot by forming required parameters


background = "black"
export = partial(export_image, background = background, export_path="export")
cm = partial(colormap_select, reverse=(background!="black"))


## ------ Map to locate the Trees

NewYorkCity   = (( -74.29,  -73.69), (40.49, 40.92))
cvs = ds.Canvas(700, 700, *NewYorkCity)


agg = cvs.points(trees, 'longitude', 'latitude')
view = tf.shade(agg, cmap = cm(viridis), how='log')
export(tf.spread(view, px=2), 'trees')

## ------ Map for health of the trees with dark color showing trees in `good
health` and the light most color showing poor health trees


agg = cvs.points(trees, 'longitude', 'latitude')
view = tf.shade(agg, cmap = cm(viridis), how='eq_hist')
export(tf.spread(view, px=2), 'trees_health')


## ------ Dash app


external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)

server = app.server # the Flask app
```

```
#auth = dash_auth.BasicAuth(
 #   app,
  #  VALID_USERNAME_PASSWORD_PAIRS
#)

## ------ Dash app showing Tree Health by Borough and Stewardship

app.layout = html.Div(children=[
    html.H1(children='NYC Street Tree Health'),
    html.P('These graphics display the overall health of trees along city streets
in NYC'),
    html.P('First select a Borough: '),
    dcc.RadioItems(
        id='dropdown-a',
        options=[{'label': i, 'value': i} for i in ['Bronx', 'Brooklyn',
'Manhattan', 'Queens', 'Staten Island']],
        value='Queens'
    ),
    html.Div(id='output-a'),
    html.P("0 = Poor Health; 1 = Fair Health, 2 = Good Health"),
    dcc.RadioItems(
        id='dropdown-b',
        options=[{'label': i, 'value': i} for i in trees['steward'].unique()],
        value='None'
    ),
    html.Div(id='output-b'),
    html.P("0 = Poor Health; 1 = Fair Health, 2 = Good Health")
])





@app.callback(
    Output(component_id='output-a', component_property='children'),
    [Input(component_id='dropdown-a', component_property='value')]
)
def boro_graph(input_data):
    df = trees[trees.boroname == input_data]

    return dcc.Graph(
        id='Health by Borough',
        figure={
            'data': [
                {'x': df['health'], 'type': 'histogram', 'name': 'Health by
Borough'}
            ],
            'layout': {
                'title': "Health by Borough"
            }
        }
    )





@app.callback(
    Output(component_id='output-b', component_property='children'),
    [Input(component_id='dropdown-b', component_property='value')]
)
def steward_graph(input_data):
    df = trees[trees.steward == input_data]

    return dcc.Graph(
        id='Health by Steward',
        figure={
            'data': [
                {'x': df['health'], 'type': 'histogram', 'name': 'Health by
Stewardship'}
            ],
            'layout': {
                'title': "Health by Stewardship"
            }
        }
    )

if __name__ == '__main__':
    app.run_server(debug=True)
```