File - C:\Users\taborovy\Downloads\3-Plotting-graph.py

```
'''

ALGORITH TO PLOT A GRAPH FROM A CSV FILE


PLEASE NOTE:

THE CSV FILE NAME MUST BE "FinData.csv"

THE CSV FILE MUST CONTAIN THE COLUMNS "Date" (yyyy-mm-dd), "Close" (a number in either int, float, or string format)

'''


import turtle
import csv


class CSVDataHandler:
  def __init__(self, csv_file_path):
      self.csv_file_path = csv_file_path

  def csv_to_dict(self):
      #method to convert the CSV file to dictionary
      data_dictionary = {}

      with open(self.csv_file_path, mode='r') as file:  # Open CSV file for reading
          csv_reader = csv.DictReader(file)

          for row in csv_reader:  # iterate over each row in the CSV
              date = row['Date']    # access using column names
              closing_price = row['Close']

              data_dictionary[date] = closing_price  # add them to the dictionary

      return data_dictionary


class TurtleGraph:
    # a class to represent and plot a graph using turtle graphics
    def __init__(self, screen_size_x=1000, screen_size_y=1000, title="Price - Time Graph"):
        # initialise the graph with a screen size and a title
        self.screen_size_x = screen_size_x
        self.screen_size_y = screen_size_y
        self.screen = self.initialise_screen()
        self.turtle = self.initialise_turtle()
        self.screen.title(title)

    def initialise_screen(self):
        # initialise the screen
        screen = turtle.Screen()
        screen.screensize(self.screen_size_x, self.screen_size_y)
        return screen

    def initialise_turtle(self):
```

```
        t = turtle.Turtle()
        t.speed(0) # fastest drawing speed
        t.width(2) # line width
        return t

    def find_axis_lenghts(self):
      # the subrouine name tells itself what it does
      return [self.screen_size_x * 5 / 11, self.screen_size_y * 5 / 11]

    def draw_axis(self):
        # Calculate axis lengths and positions
        axis_length_x, axis_length_y = self.find_axis_lenghts()
        arrow_size = 0.005  # Relative size of the arrow head

        # Draw X and Y axes with arrows
        self.draw_line_with_arrows(0, 0, axis_length_x, 0, arrow_size)
        self.draw_line_with_arrows(0, 0, 0, axis_length_y, arrow_size,
vertical=True)

    def draw_line_with_arrows(self, start_x, start_y, end_x, end_y, arrow_size
, vertical=False):
        self.turtle.penup()
        self.turtle.goto(start_x, start_y)
        self.turtle.pendown()
        self.turtle.goto(end_x, end_y)

        # Draw arrow head
        if vertical:
          self.draw_arrow_head(end_x, end_y, arrow_size, vertical=True)
        else:
          self.draw_arrow_head(end_x, end_y, arrow_size)

        self.turtle.penup()
        self.turtle.setpos(0,0)

    def draw_arrow_head(self, x, y, size, vertical=False):
        # Draw an arrow head at specified position
        adjust_x = self.screen_size_x * size
        adjust_y = self.screen_size_y * size
        if vertical:
            self.turtle.goto(x - adjust_x, y - adjust_y)
            self.turtle.goto(x, y)
            self.turtle.goto(x + adjust_x, y - adjust_y)
        else:
            self.turtle.goto(x - adjust_x, y + adjust_y)
            self.turtle.setposition(x, y)
            self.turtle.goto(x - adjust_x, y - adjust_y)

    def find_number_of_x_axis_points(self, dict):
      return len(dict)

    def plot_graph(self, data_points):
      '''
      Please note: the instructions to calculate the x-coordinate of the
points are
      commented out in the code, as dates instead of numerical values will be
used on
      the x-axis.  However, the code can still be used if there is a need to
plot a
```

```
110       graph with numerical values for the x-axis.
111       '''
112
113       # comment out the x axis (keys)
114
115       # find the lowest key in the dictionary
116       # lowest_key = self.find_min_or_max_key_in_dictionary(data_points) #
117       # find the highest key in the dictionary
118       # highest_key = self.find_min_or_max_key_in_dictionary(data_points,
    False) #
119       # find the lowest value in the dictionary
120       lowest_value = self.find_min_or_max_value_in_dictionary(data_points)
121       # find the highest value in the dictionary
122       highest_value = self.find_min_or_max_value_in_dictionary(data_points,
    False)
123
124       dist_between_the_points_on_x = (self.find_axis_lenghts()[0]) / self.
    find_number_of_x_axis_points(data_points)
125
126       k = 0 # counter for the x-axis
127
128       for x_value, y_value in data_points.items():
129           x_real_to_label = str(x_value)
130           y_real_to_label = float(y_value)
131
132           x_relative_to_plot = k * dist_between_the_points_on_x
133           y_relative_to_plot = (y_real_to_label - lowest_value) / (
    highest_value)*(self.find_axis_lenghts()[1])
134
135           # Plot the actual point
136           self.turtle.pendown()
137           self.turtle.goto(x_relative_to_plot, y_relative_to_plot)
138
139           # Label the point on the x-axis
140           self.label_point_x(x_relative_to_plot, x_real_to_label)
141
142           # Label the point on the y-axis
143           self.label_point_y(y_relative_to_plot, y_real_to_label)
144
145           # Prepare for the next point
146           self.turtle.penup()
147           self.turtle.goto(x_relative_to_plot, y_relative_to_plot)
148
149           k += 1  # increment the counter for the x-axis
150
151
152     def label_point_x(self, x, x_value):
153       # Label the point on the x axis
154       self.turtle.penup()
155       self.turtle.goto(x, 0)
156       self.turtle.dot(5)  # Mark the point on the x axis
157       self.turtle.goto(x, -20)
158       self.turtle.pendown()
159       self.turtle.write(x_value, align="center")  # Center align the x axis
    label
160
161     def label_point_y(self, y, y_value):
162       # Label the point on the y-axis
163       self.turtle.penup()
```

```
164         self.turtle.goto(0, y)
165         self.turtle.dot(5)  # Mark the point on the y axis
166         self.turtle.goto(-(self.screen_size_x /20), (y-y*0.05))
167         self.turtle.pendown()
168         self.turtle.write(y_value, align="center")  # Center align the y axis
    label
169 
170     def find_min_or_max_key_in_dictionary(self, data_points, minimum=True):
171         return float(min(data_points.keys()) if minimum else max(data_points.
    keys()))
172 
173     def find_min_or_max_value_in_dictionary(self, data_points, minimum=True):
174         return float(min(data_points.values()) if minimum else max(data_points.
    values()))
175 
176     def finish(self):
177         #clean up by hiding the turtle and displaying the plot until closed
178         turtle.done()
179         self.turtle.hideturtle()
180 
181 if __name__ == "__main__":
182     graph = TurtleGraph()
183     graph.draw_axis()
184     csv_file = 'FinData.csv'
185     data_handler = CSVDataHandler(csv_file)
186     data_in_dict = data_handler.csv_to_dict()
187     graph.plot_graph(data_in_dict)
188     graph.finish()
189 
190 
191 
192 
193 
194 
195 
196 
197 
198 # import turtle
199 
200 # class TurtleGraph:
201 #     def __init__(self, screen_size_x=2000, screen_size_y=2000, title="Data
    Points Plotter"):
202 #         self.screen_size_x = screen_size_x
203 #         self.screen_size_y = screen_size_y
204 #         self.screen = self.initialise_screen()
205 #         self.turtle = self.initialise_turtle()
206 #         self.screen.title(title)
207 
208 #     def initialise_screen(self):
209 #         screen = turtle.Screen()
210 #         screen.screensize(self.screen_size_x, self.screen_size_y)
211 #         return screen
212 
213 #     def initialise_turtle(self):
214 #         t = turtle.Turtle()
215 #         t.speed(0)
216 #         t.width(2)
217 #         return t
218 
```

```
219 #     def draw_axis(self):
220 #         # Calculate axis lengths and positions
221 #         axis_length_x = self.screen_size_x * 5 / 11
222 #         axis_length_y = self.screen_size_y * 5 / 11
223 #         arrow_size = 0.01  # Relative size of the arrow head
224 
225 #         # Draw X axis
226 #         self.draw_line_with_arrows(-axis_length_x, -axis_length_y,
    axis_length_x, -axis_length_y, arrow_size)
227 
228 #         # Draw Y axis
229 #         self.draw_line_with_arrows(-axis_length_x, -axis_length_y, -
    axis_length_x, axis_length_y, arrow_size, vertical=True)
230 
231 #     def draw_line_with_arrows(self, start_x, start_y, end_x, end_y,
    arrow_size, vertical=False):
232 #         self.turtle.penup()
233 #         self.turtle.goto(start_x, start_y)
234 #         self.turtle.pendown()
235 #         self.turtle.goto(end_x, end_y)
236 
237 #         # Draw arrow head
238 #         if vertical:
239 #             self.turtle.goto(end_x - self.screen_size_x * arrow_size, end_y
     - self.screen_size_y * arrow_size)
240 #             self.turtle.goto(end_x, end_y)
241 #             self.turtle.goto(end_x + self.screen_size_x * arrow_size, end_y
     - self.screen_size_y * arrow_size)
242 #         else:
243 #             self.turtle.goto(end_x - self.screen_size_x * arrow_size, end_y
     + self.screen_size_y * arrow_size)
244 #             self.turtle.setposition(end_x, end_y)
245 #             self.turtle.goto(end_x - self.screen_size_x * arrow_size, end_y
     - self.screen_size_y * arrow_size)
246 
247 #         self.turtle.penup()
248 #         self.turtle.setpos(0,0)
249 
250 
251 #     def plot_graph(self, data_points):
252 #       for x_value, y_value in data_points.items():
253 #           x = float(x_value)
254 #           y = float(y_value)
255 
256 #           # Plot the actual point
257 #           self.turtle.pendown()
258 #           self.turtle.goto(x, y)
259 
260 #           # Label the point on the x-axis
261 #           self.label_point_x(x, -10, x_value)
262 
263 #           # Label the point on the y-axis
264 #           self.label_point_y(-20, y, y_value)
265 
266 #           # Prepare for the next point
267 #           self.turtle.penup()
268 #           self.turtle.goto(x, y)
269 
270 
```

```
271 #     def label_point_x(self, x, y, x_value):
272 #       # Label the point on the x axis
273 #       self.turtle.penup()
274 #       self.turtle.goto(x, 0)
275 #       self.turtle.dot(5)  # Mark the point on the x axis
276 #       self.turtle.goto(x, -20)
277 #       self.turtle.pendown()
278 #       self.turtle.write(x_value, align="center")  # Center align the x axis
    label
279 
280 #     def label_point_y(self, x, y, y_value):
281 #       # Label the point on the y-axis
282 #       self.turtle.penup()
283 #       self.turtle.goto(0, y)
284 #       self.turtle.dot(5)  # Mark the point on the y axis
285 #       self.turtle.goto(-20, y)
286 #       self.turtle.pendown()
287 #       self.turtle.write(y_value, align="center")  # Center align the y axis
    label
288 
289 #     def finish(self):
290 #       #clean up by hiding the turtle and displaying the plot until closed
291 #       turtle.done()
292 #       self.turtle.hideturtle()
293 
294 # if __name__ == "__main__":
295 #     graph = TurtleGraph()
296 #     graph.draw_axis()
297 #     data_points = {"100": "100", "140": "180", "150": "250" , "200": "350
    ", "400": "450"}
298 #     graph.plot_graph(data_points)
299 #     graph.finish()
300 
301 
302 
303 # def plot_graph(data_points):
304 #     """
305 #     Plot and label a graph based on a dictionary of data points.
306 
307 #     Args:
308 #         data_points (dict): A dictionary where keys are x-axis values and
    values are y-axis values.
309 #     """
310 
311 #     # Ensure turtle graphics window is open and turtle is initialised
312 #     turtle_screen = turtle.Screen()
313 #     turtle_screen.title("Data Points Plotter")
314 #     plotter = turtle.Turtle()
315 #     plotter.speed(0)  # drawing speed
316 
317 #     for x_value, y_value in data_points.items():
318 #         # Convert string to float for plotting
319 #         x = float(x_value)
320 #         y = float(y_value)
321 
322 #         # Plot the actual point on the graph
323 #         plotter.pendown()
324 #         plotter.goto(x, y)
325 
```

```
326
327 #         # Label the point on the x-axis
328 #         plotter.penup()
329 #         plotter.goto(x, 0)
330 #         plotter.dot(5)  # Mark the point on the x axis
331 #         plotter.goto(x, -20)
332 #         plotter.pendown()
333 #         plotter.write(x_value, align="center")  # Center align the x-axis
    label
334
335 #         # Label the point on the y-axis
336 #         plotter.penup()
337 #         plotter.goto(0, y)
338 #         plotter.dot(5)  # Mark the point on the y axis
339 #         plotter.goto(-20, y)
340 #         plotter.pendown()
341 #         plotter.write(y_value, align="center")  # Right align the y-axis
    label
342
343 #         plotter.penup()  # Prepare for the next point
344 #         plotter.goto(x, y)
345
346 #     plotter.hideturtle()
347 #     turtle.done()
348
349 # Example usage
350 # data_points = {"100": "100", "140": "180", "150": "250" , "200": "350", "400
    ": "450"}
351 # plot_graph(data_points)
352
```