How to plot (x,y,z) coordinates in the shape of a hexagonal grid?

Asked 7 years, 2 months ago Modified 7 years, 2 months ago Viewed 15k times

1

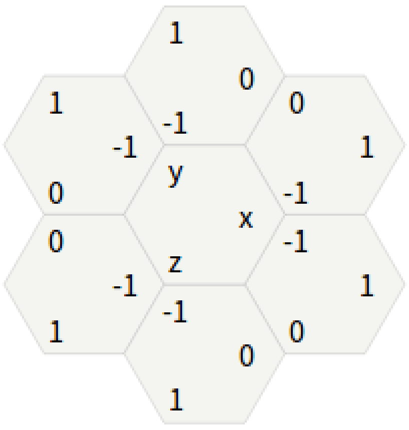
If for example, I have the following coordinates with corresponding colors which represent a hexagonal shaped grid of hexagons:

```
coord = [[0,0,0],[0,1,-1],[-1,1,0],[-1,0,1],[0,-1,1],[1,-1,0],[1,0,-1]]
colors = [["Green"],["Blue"],["Green"],["Green"],["Red"],["Green"],["Green"]]
```

How can one plot this in Python so that the points on the plot retain that hexagonal shape? Additionally how can one represent the 'colors' list on the hexagon.

Somewhat like this:

Simple Hexagonal grid



But the look doesn't matter, just a simple scatter plot type visualization would suffice, just so that one can see where in relation to other hexagons the colors lie.

python matplotlib coordinates hexagonal-tiles

Share Improve this question Follow

edited Oct 2, 2017 at 13:15

asked Oct 2, 2017 at 12:59
ishido
4,225 9 34 42

Are you using ASCII or do you wish to use the turtle graphics module? – EgMusic Oct 2, 2017 at 13:17

What do the coordinates represent? I cannot make the connection between the image and the `coord` . You would probably want to tell the convention used here, e.g. if I take a list `[[a,b,c],[d,e,f]]` , what are a,b,c,d,e,f? – ImportanceOfBeingErnest Oct 2, 2017 at 13:18

@ChaseBarnes i wish to use the simplest solution. ASCII would suffice. – ishido Oct 2, 2017 at 13:21

@ImportanceOfBeingErnest the coordinates are the different hexagons on the hexagonal shaped tiling. Each set of coordinates (x,y,z) has a property, represented here as a color, which I would like to display in form of a visualization – ishido Oct 2, 2017 at 13:23

I did ask the question: in a list [[a,b,c],[d,e,f]], what are a,b,c,d,e,f? – ImportanceOfBeingErnest Oct 2, 2017 at 13:26

3 Answers

Sorted by: Highest score (default)

11

You just need to turn the `(y, z)` coordinates from your hexagons into the `y` cartesian coordinate on the matplotlib axes.

I think the correct way to do that is using this formula:

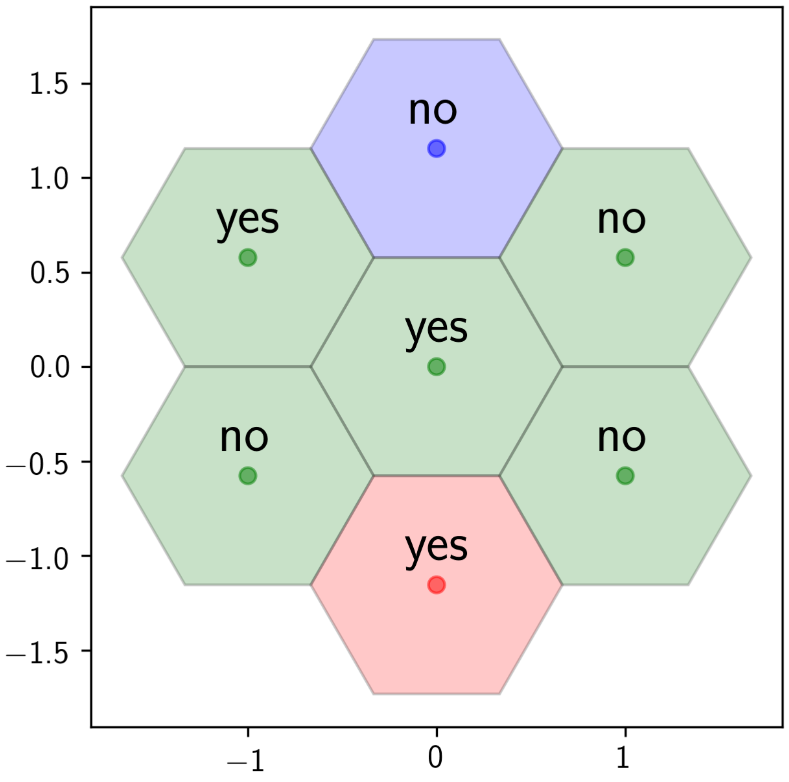
```
y_cartesian = (2 / 3) * sin(60) * (y_hex - z_hex)
```

You can then add the hexagons using a matplotlib `RegularPolygon` patch, or plot the centres using `scatter` .

Here's a script to make a plot from your lists:

```
import matplotlib.pyplot as plt
from matplotlib.patches import RegularPolygon
import numpy as np

coord = [[0,0,0],[0,1,-1],[-1,1,0],[-1,0,1],[0,-1,1],[1,-1,0],[1,0,-1]]
colors = [["Green"],["Blue"],["Green"],["Green"],["Red"],["Green"],["Green"]]
labels = [['yes'],['no'],['yes'],['no'],['yes'],['no'],['no']]

# Horizontal cartesian coords
hcoord = [c[0] for c in coord]

# Vertical cartersian coords
vcoord = [2. * np.sin(np.radians(60)) * (c[1] - c[2]) /3. for c in coord]

fig, ax = plt.subplots(1)
ax.set_aspect('equal')

# Add some coloured hexagons
for x, y, c, l in zip(hcoord, vcoord, colors, labels):
    color = c[0].lower()  # matplotlib understands lower case words for colours
    hex = RegularPolygon((x, y), numVertices=6, radius=2. / 3.,
                         orientation=np.radians(30),
                         facecolor=color, alpha=0.2, edgecolor='k')
    ax.add_patch(hex)
    # Also add a text label
    ax.text(x, y+0.2, l[0], ha='center', va='center', size=20)

# Also add scatter points in hexagon centres
ax.scatter(hcoord, vcoord, c=[c[0].lower() for c in colors], alpha=0.5)
```

```
plt.show()
```



Share Improve this answer Follow edited Oct 2, 2017 at 14:32 answered Oct 2, 2017 at 13:46

tmdavison
68.8k 12 200 180

Thank you very much for this. Is it also possible to additionally put a label in each hexagon? Say if I had a list of `labels = [['yes'],['no'],['yes'],['no'],['yes'],['no'],['no']]`. I saw that `RegularPolygon` does have a labels option – ishido Oct 2, 2017 at 14:27

1 `label` for `RegularPolygon` I think is to put a label in a legend. Instead, you could use `ax.text(x, y, label)`. See my edit – tmdavison Oct 2, 2017 at 14:31

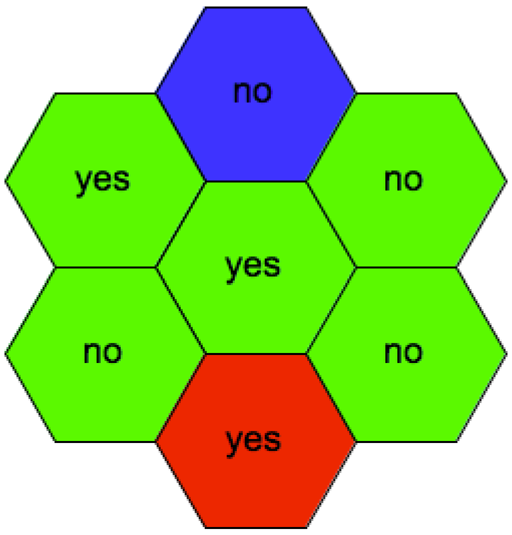
While this indeed works, removing the scatter plot completely messes up the displayed range. In general, `ax.autoscale_view()` should thus be called before `plt.show()`. – Eric O. Lebigot Aug 28, 2021 at 20:43

2

Below is my attempt to finish PM2Ring's turtle-based solution (+1) as well as fix what I see as a coordinate calculation error in his answer:

```
from math import sqrt
from turtle import Turtle, Screen

ROOT3_OVER_2 = sqrt(3) / 2

FONT_SIZE = 18
FONT = ('Arial', FONT_SIZE, 'normal')
```

```python
SIDE = 50  # the scale used for drawing

# Convert hex coordinates to rectangular
def hex_to_rect(coord):
    v, u, w = coord
    x = -u / 2 + v - w / 2
    y = (u - w) * ROOT3_OVER_2
    return x * SIDE, y * SIDE

def hexagon(turtle, radius, color, label):
    clone = turtle.clone()  # so we don't affect turtle's state
    xpos, ypos = clone.position()
    clone.setposition(xpos - radius / 2, ypos - ROOT3_OVER_2 * radius)
    clone.setheading(-30)
    clone.color('black', color)
    clone.pendown()
    clone.begin_fill()
    clone.circle(radius, steps=6)
    clone.end_fill()
    clone.penup()
    clone.setposition(xpos, ypos - FONT_SIZE / 2)
    clone.write(label, align="center", font=FONT)

# Initialize the turtle
tortoise = Turtle(visible=False)
tortoise.speed('fastest')
tortoise.penup()

coords = [[0, 0, 0], [0, 1, -1], [-1, 1, 0], [-1, 0, 1], [0, -1, 1], [1, -1, 0],
[1, 0, -1]]
colors = ["Green", "Blue", "Green", "Green", "Red", "Green", "Green"]
labels = ['yes', 'no', 'yes', 'no', 'yes', 'no', 'no']

# Plot the points
for hexcoord, color, label in zip(coords, colors, labels):
    tortoise.goto(hex_to_rect(hexcoord))
    hexagon(tortoise, SIDE, color, label)

# Wait for the user to close the window
screen = Screen()
screen.exitonclick()
```



Share Improve this answer Follow

answered Oct 2, 2017 at 18:33

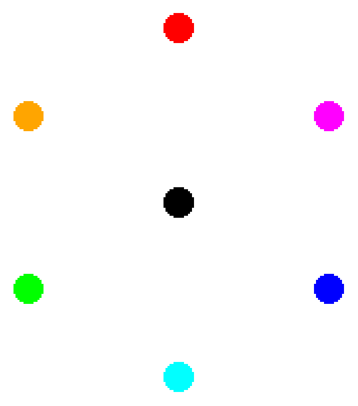
cdlane
41.7k 5 36 83

2

Here's a function that converts a (u, v, w) tuple of hex coordinates into rectangular coordinates. I'll illustrate it using the standard `turtle` module (I don't have the matplotlib module). I've changed the colours on the list so we can easily check that each point is plotted in the correct position.

```
import turtle
from math import sqrt

root3 = sqrt(3)

# the scale used for drawing
side = 50

# Convert hex coordinates to rectangular
def hex_to_rect(coord):
    u, v, w = coord
    x = u - v/2 - w/2
    y = (v - w) * root3 / 2
    return x * side, y * side

# Initialize the turtle
t = turtle.Turtle()
t.speed(0)
t.hideturtle()
t.up()

coords = [[0,0,0], [0,1,-1], [-1,1,0], [-1,0,1], [0,-1,1], [1,-1,0], [1,0,-1]]
colors = ['black', 'red', 'orange', 'green', 'cyan', 'blue', 'magenta']

#Plot the points
for hexcoord, color in zip(coords, colors):
    xy = hex_to_rect(hexcoord)
    t.goto(xy)
    t.dot(15, color)

# Wait for the user to close the window
turtle.done()
```

output



Share Improve this answer Follow edited Oct 2, 2017 at 19:08 answered Oct 2, 2017 at 13:54

PM 2Ring
55.3k 6 91 192

@Chase Barnes Thanks for fixing that typo. But the info about installing matplotlib is quite irrelevant to my answer. – PM 2Ring Oct 2, 2017 at 15:22

I understand, but just in case people want to know how to install modules, there it is. – EgMusic Oct 2, 2017 at 15:43

Based on my reading of the problem, you've a green circle where you should have a cyan one at [0, -1, 1] i.e. the bottom position. – cdlane Oct 2, 2017 at 18:06

I'm assuming the central hexagon in the OP's diagram shows how X, Y and Z are decoded which puts [0, -1, 1] (cyan) as the bottom hexagon. – cdlane Oct 2, 2017 at 18:36

@cdlane Fixed. I only just noticed that you've submitted an answer as well. – PM 2Ring Oct 2, 2017 at 19:09