3-D Plots

This lesson discusses line plots and surface plots as part of the 3-D plots in Python.

WE'LL COVER THE FOLLOWING

Until now, we have been learning to make 2-D plots in Python using the `matplotlib` package. 3-D plotting is similar except for a few additional commands that we will learn in this lesson.

3-D line plots

Consider the x, y, z coordinates of a three-dimensional spiral

$$x(t) = a\ cos(t)$$

$$y(t) = a\ sin(t)$$

$$z(t) = bt$$

where a and b are constants and t is a parameter that varies. Write a Python function that takes a, b, and an array t as input arguments and returns arrays x, y, and z.

For 3-D plots, we import the 3-D plotting capabilities of `matplotlib` with the command

```
from mpl_toolkits.mplot3d import Axes3D
```

To plot a three-dimensional curve, specify the keyword `projection='3d'` when creating the axis. We can then plot on that axis using `ax.plot` by specifying

x, y, and z.

We will be plotting two spirals on the same graph. Using $a = 4$ and $b = 1$ for the first spiral, and $a = 2, b = 2$ for the second spiral; vary t from 0 to 20 with 100 points.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def spiral(a, b, t):
    x = a * np.cos(t)
    y = a * np.sin(t)
    z = b * t
    return x, y, z

t = np.linspace(0, 20, 100)
x1, y1, z1 = spiral(4, 1, t)
x2, y2, z2 = spiral(2, 2, t)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x1, y1, z1)
ax.plot(x2, y2, z2);
```

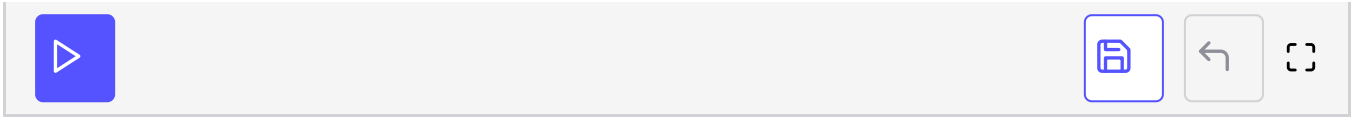
3-D surface plots

Surface plots are great for visualizing relationships across the entire 3D landscape since they provide the full structure. We will be using the same technique that we used above, except we won't be converting our data into 2-D using the `meshgrid()` function.

$$f(x, y) = sin(\sqrt{x^2 + y^2})$$

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from mpl_toolkits.mplot3d import Axes3D

def func(x, y):
  z = np.sin(np.sqrt(x ** 2 + y ** 2))
  return z

x = np.linspace(-6, 6, 30)
y = np.linspace(-6, 6, 30)
X, Y = np.meshgrid(x, y)
Z = func(X, Y)
```

```
fig = plt.figure()


ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap=cm.cool)
```



> 💡 Use the `help('mpl_toolkits.mplot3d.axes3d.Axes3D.plot_surface')` command to explore more properties.

You might have noticed lines 3 and 17. These are colormaps in `matplotlib` that help visualize the 3-D plot. In this example, we have chosen the property `cool` from the `cm` module. You can learn about more colormaps over here.

Changing the view of 3-D plots

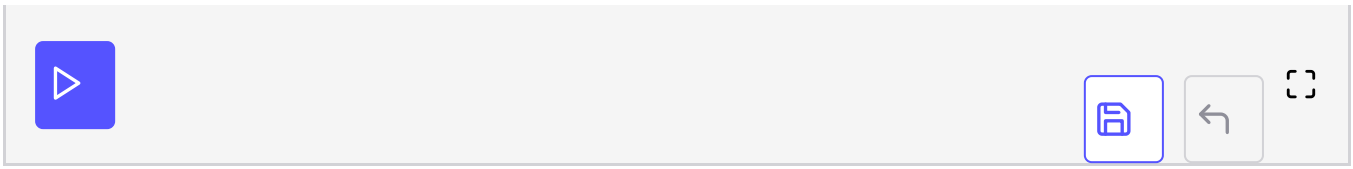
We can change the viewing perspective of a 3D plot using the `view_init()` method. This method takes two arguments:

1. elevation angle
2. azimuth angle

The unit for both these angles is degrees.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def spiral(a, b, t):
    x = a * np.cos(t)
    y = a * np.sin(t)
    z = b * t
    return x, y, z

t = np.linspace(0, 20, 100)
x, y, z = spiral(4, 2, t)

fig = plt.figure()

ax1 = fig.add_subplot(121, projection='3d')
ax1.plot(x, y, z)
ax1.view_init(45, 45)

ax2 = fig.add_subplot(122, projection='3d')
ax2.plot(x, y, z, color='r')
ax1.view_init(30, 90)
```

```
ax1.view_init(30, 90)
```



Let’s test your understanding of this chapter with a quiz in the next lesson.