Notebook inline interactive plot

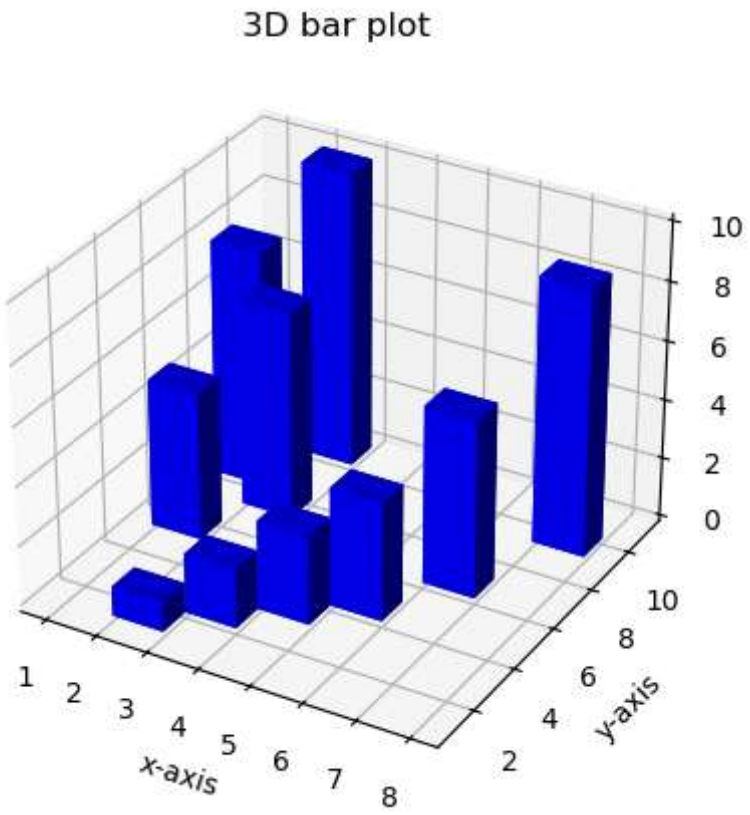
The following codes (cells) demonstrate the difference in
%matplotlib inline
and
%matplotlib widget
or
%matplotlib ipympl
For the first one (in general) nothing has to be modified as **matplotlib** module is installed with **Juypter Notebook** package.
While the latter two requires the installation or enabling the **widgets** or the **ipympl** package as Jupyter server service extension.

On possible issues with plotting (display) see bugfixes blog post.

In [ ]:
```
# creating predefined dataset
xs = [2, 3, 4, 5, 1, 6, 2, 1, 7, 2]
ys = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
zs = np.zeros(10)
dx = np.ones(10)
dy = np.ones(10)
dz = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Using
%matplotlib inline
creates a static (still-) image from a default viewpoint. Original code source

In [34]:
```
# Creating 3d bar plot using matplotlib in python. NOT INTERACTIVE!
%matplotlib inline

# importing required libraries
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np

# creating figure
figg = plt.figure()
ax = figg.add_subplot(111, projection='3d')

# creating the plot
plot_geeks = ax.bar3d(xs, ys, zs, dx,
                      dy, dz, color='blue')

# setting title and labels
ax.set_title("3D bar plot")
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')

# displaying the plot
plt.show()
```



With reference to the previously created variables and imported python modules it is enough to declare
%matplotlib ipympl
and to create the plot again. It is **interactive** in this case. Typically on the left side of the plot, different buttons show up when hovering over the plot.

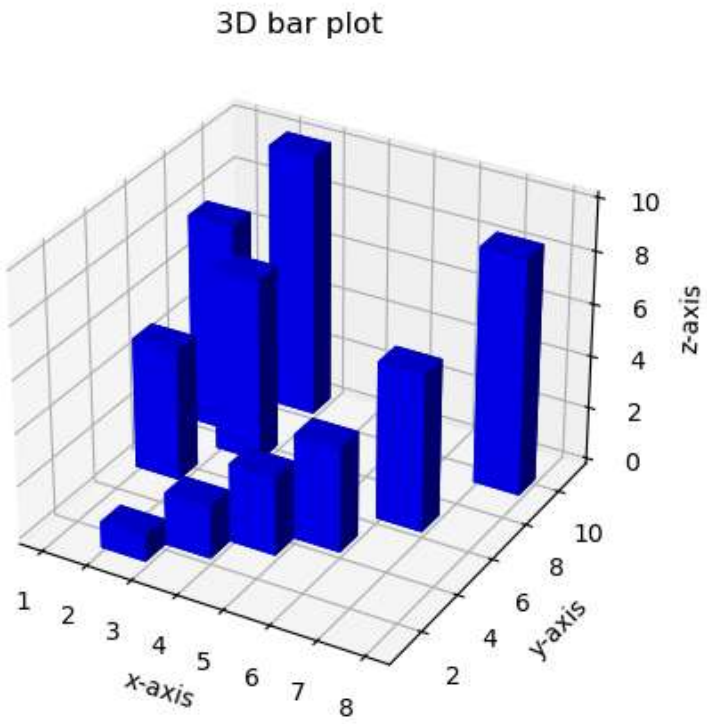
- Reset original view
- Back to / Forward to view
- Zoom to rectangle
- Download plot

Possible error: "*Warning: Cannot change to a different GUI toolkit: widget. Using widget instead.*" This indicates that your Jupyter has no **ipympl** but **widget** extension installed. Another error may arise if you have no appropriate module that could create interactive plots.

In [36]:
```
%matplotlib ipympl
figg = plt.figure()
ax = figg.add_subplot(111, projection='3d')

# creating the plot
plot_geeks = ax.bar3d(xs, ys, zs, dx,
                      dy, dz, color='blue')

# setting title and labels
ax.set_title("3D bar plot")
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
```

```
ax.set_zlabel('z-axis')
plt.show()
```

Figure

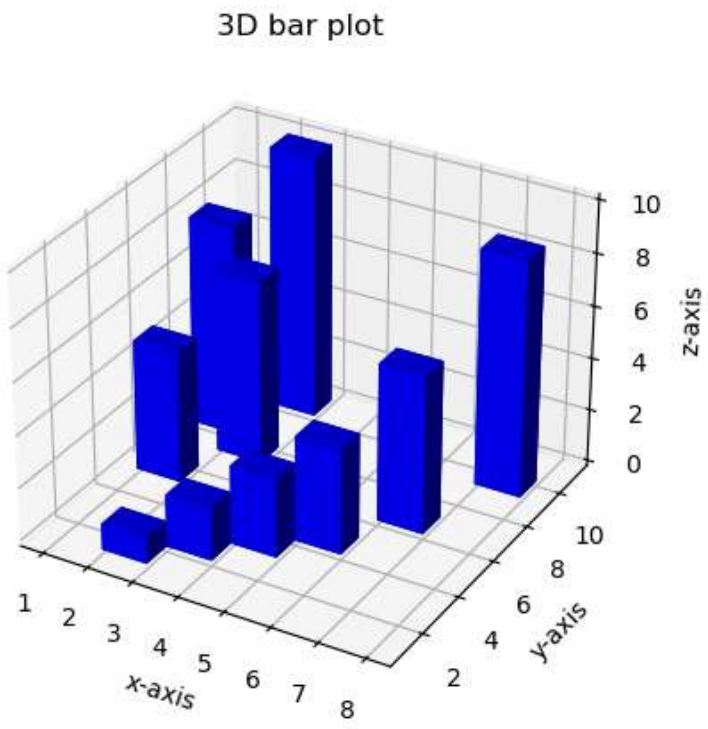


In [ ]:
```
Let's try with the widget option, using

Possible error: _"Warning: Cannot change to a different GUI toolkit: widget. Usi
Another error may arise if you have no appropriate module that could create inte
```

In [41]:
```
%matplotlib widget
figg = plt.figure()
ax = figg.add_subplot(111, projection='3d')

# creating the plot
plot_geeks = ax.bar3d(xs, ys, zs, dx,
                      dy, dz, color='blue')

# setting title and labels
ax.set_title("3D bar plot")
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
plt.show()
```

```
Warning: Cannot change to a different GUI toolkit: widget. Using ipympl instead.
```

Figure

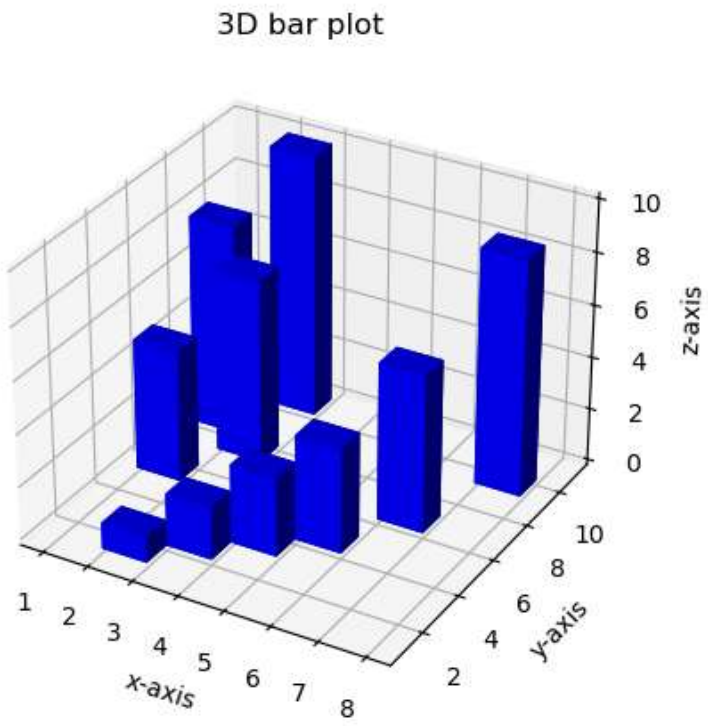


Full code with ipympl:

In [12]:
```
# creating 3d bar plot using matplotlib
# in python

# to interact with plot
%matplotlib ipympl

# importing required libraries
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np

# creating random dataset
xs = [2, 3, 4, 5, 1, 6, 2, 1, 7, 2]
ys = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
zs = np.zeros(10)
dx = np.ones(10)
dy = np.ones(10)
dz = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# creating figure
figg = plt.figure()
ax = figg.add_subplot(111, projection='3d')

# creating the plot
plot_geeks = ax.bar3d(xs, ys, zs, dx,
                      dy, dz, color='blue')

# setting title and labels
ax.set_title("3D bar plot")
```

```
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')

# displaying the plot
plt.show()
```

Figure



Further interactivity tests

Slider

A simple slider widget that provides easy interaction when values has to be modified to check the effect of the defined values. Typically useful in case of plotting functions.
min and max values define the accessible value range, while *readout_format='d'* declares integer value is returned. In case a float is required use *readout_format='.1f'*
Source

In [1]:
```
import ipywidgets as widgets
widgets.IntSlider(
    value=7,
    min=0,
    max=10,
    step=1,
    description='Test:',
    disabled=False,
    continuous_update=False,
```

```
    orientation='horizontal',
    readout=True,
    readout_format='d'
)
```

Out[1]:
```
IntSlider(value=7, continuous_update=False, description='Test:', max=10)
```

Plotting a line with variable parameters

Knowing the general equation of lines:
y = a*x + b
the steepness and the (Y axis) interception point can be modified. Grab one of the circles on the sliders and see how it effects the line plot. Check the other slider effect as well!
Source of code

In [61]:
```
%matplotlib ipympl
from ipywidgets import interactive, FloatSlider
import matplotlib.pyplot as plt
import numpy as np
m_widget = FloatSlider(min=-2, max=2, step=.5, readout_format='.3f', continuous_
b_widget = FloatSlider(min=-5, max=5, step=.5, readout_format='.1e', continuous_
def f(a, b):
    fig, ax = plt.subplots(figsize=(6,4))
    x = np.linspace(-10, 10, num=1000)
    ax.plot(x, a * x + b)
    ax.set_ylim(-5, 5)
    plt.show()

interactive_plot = interactive(f, a=m_widget, b=b_widget)
interactive_plot
```

Out[61]:
```
interactive(children=(FloatSlider(value=0.0, continuous_update=False, descripti
on='a', max=2.0, min=-2.0, read…
```

Further sources:
https://matplotlib.org/ipympl/
https://matplotlib.org/stable/users/explain/figure/interactive.html
https://www.geeksforgeeks.org/make-3d-interactive-matplotlib-plot-in-jupyter-notebook/

In [ ]: