Matplotlib Practice - Hints - Unibs 2021

In [ ]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

Import matplotlib.pyplot package under name `plt` and print version

`hint: import ... as, plt.__version__`

In [ ]:
```
import matplotlib as mat
print(mat.__version__)
import matplotlib.pyplot as plt
```

```
3.7.1
```

Activate matplotlib inline

`hint: ... inline`

In [ ]:
```
%matplotlib inline
```

Base

Plot a line with formula $y = 2x + 1$

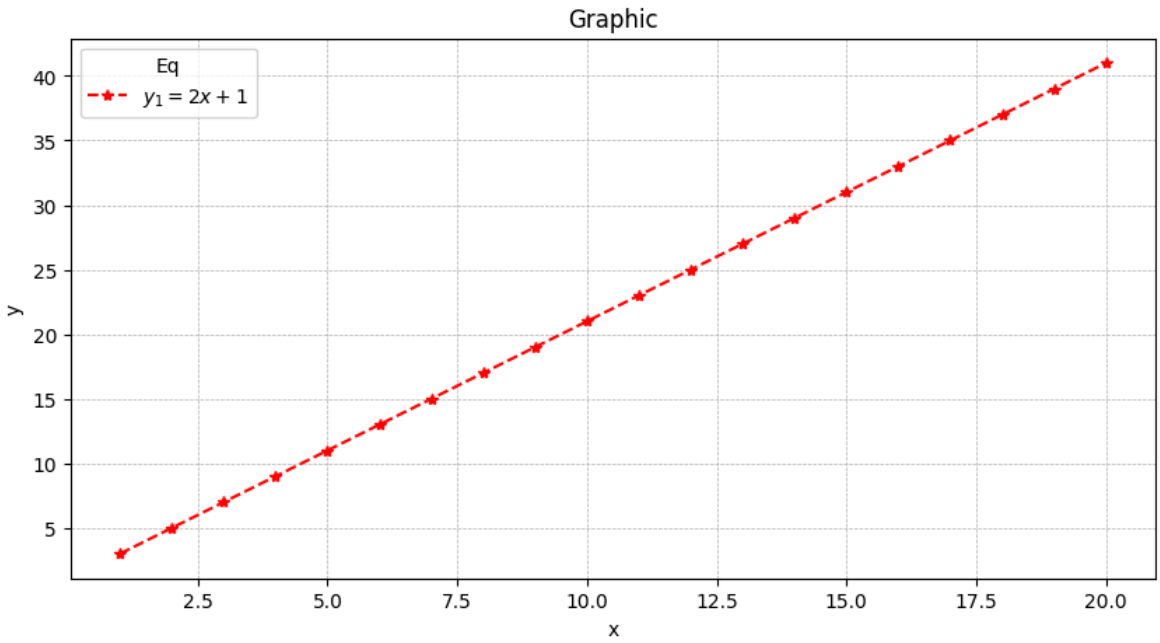
`hint: np.arange, plt.figure, plt.plot, plt.title, plt.xlabel, plt.ylabel, plt.grid, plt.legend, plt.show`

- Use points in range [1, 20]
- Figure size of (10,5)
- Set axis labels
- Set plot title
- Set line color as red
- Discontinued line (--) with star (*) on point
- Plot the legend with the formula in latex version (`r"$ ... $"`)
- Set a dashed grid

In [ ]:
```
x = np.arange(1, 21)
y1 = 2 * x + 1
```

In [ ]:
```
plt.figure(figsize=(10, 5))
plt.xlabel('x')
plt.ylabel('y')
plt.title('Graphic')

plt.plot(x, y1, color='red', linestyle='--', marker='*',label=r"$y_1 = 2x + 1$")
```

```
plt.legend(title="Eq")
plt.grid(True, linestyle='--', linewidth=0.5)

plt.show()
```

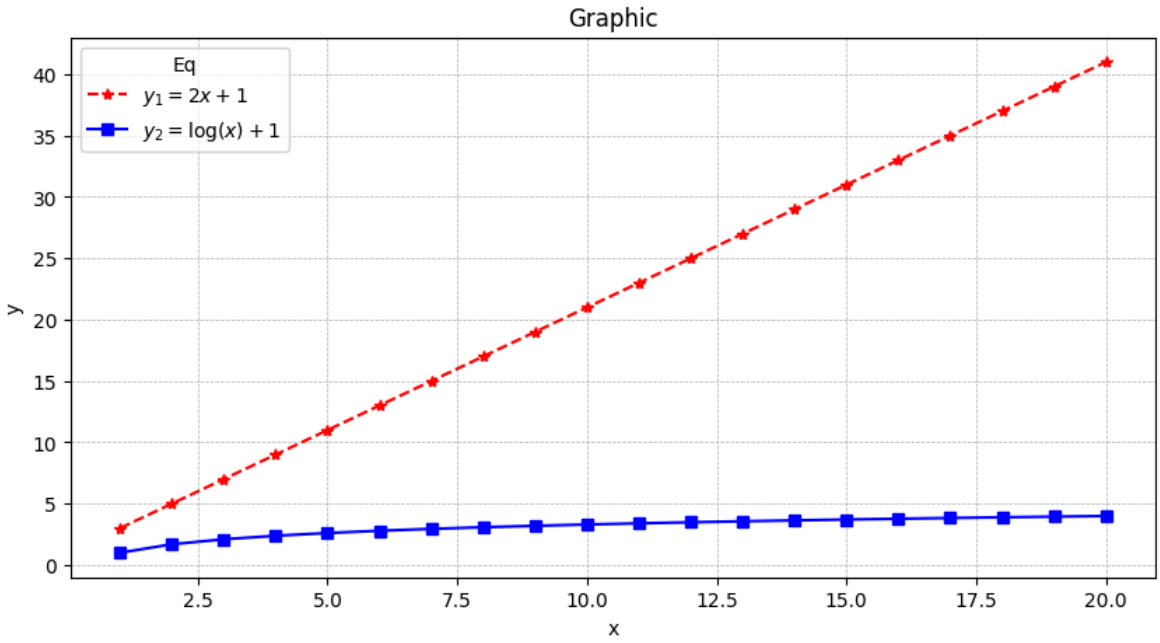


Add to the previous plot the line with formula $y = log(x) + 1$ in the same range

`hint: np.log, plt.plot`

- Set line color as blue
- Normal line (-) with square (s) char on point

In [ ]:
```
# grafico precedente
plt.figure(figsize=(10, 5))
plt.xlabel('x')
plt.ylabel('y')
plt.title('Graphic')

plt.plot(x, y1, color='red', linestyle='--', marker='*',label=r"$y_1 = 2x + 1$")

# grafico nuovo
y2 = np.log(x) + 1
plt.plot(x, y2, color='blue', linestyle='-', marker='s', label=r"$y_2 = \log(x)

plt.legend(title="Eq")
plt.grid(True, linestyle='--', linewidth=0.5)
plt.show()
```

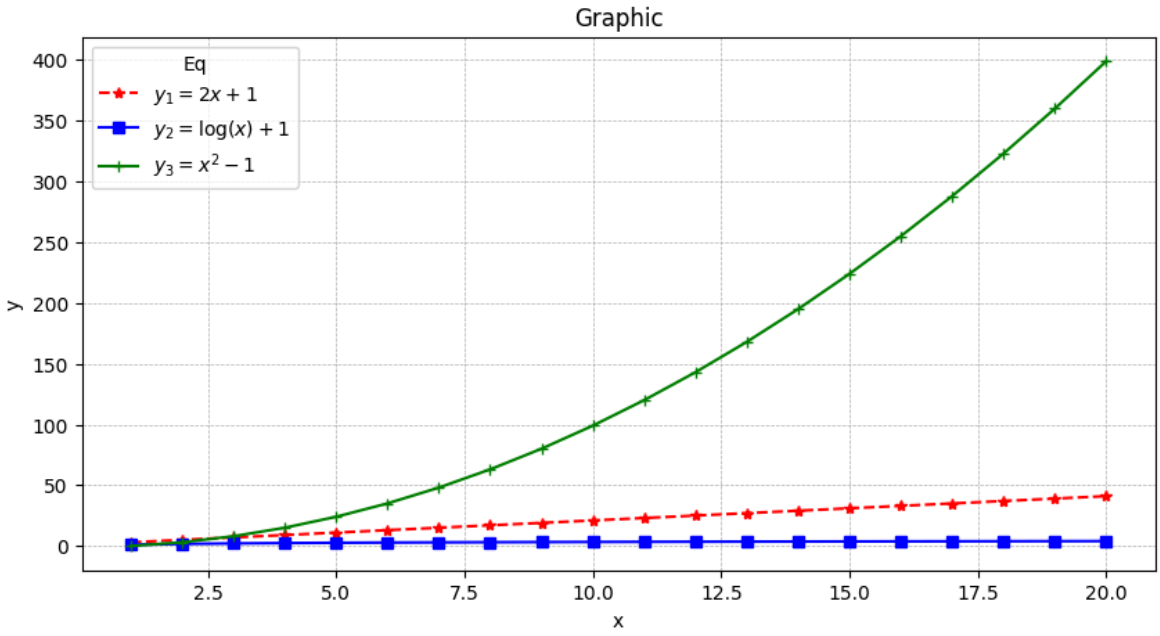


Add to the previous plot the line with formula $y = x^2 - 1$ in the same range

`hint: x**2, plt.plot`

- Set line color as green
- Normal line (-) with + (plus) char on point

In [ ]:
```
# grafici precedenti
plt.figure(figsize=(10, 5))
plt.xlabel('x')
plt.ylabel('y')
plt.title('Graphic')

plt.plot(x, y1,
         color='red',
         linestyle='--',
         marker='*',
         label=r"$y_1 = 2x + 1$")
plt.plot(x, y2,
         color='blue',
         linestyle='-',
         marker='s',
         label=r"$y_2 = \log(x) + 1$")

# grafico nuovo
y3 = x**2-1
plt.plot(x, y3,
         color='green',
         linestyle='-',
         marker='+',
         label=r"$y_3 = x^2 - 1$")

plt.legend(title="Eq")
plt.grid(True, linestyle='--', linewidth=0.5)
plt.show()
```



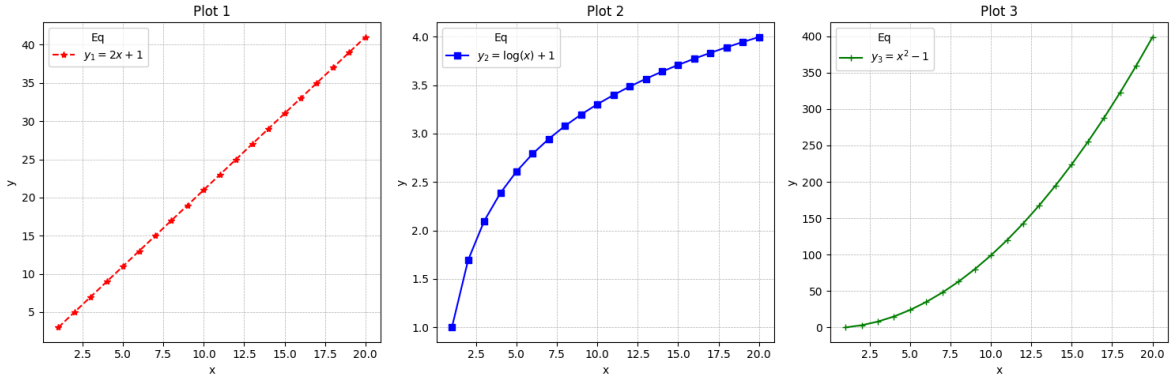
Replot the previous plot in different subplots in the same line without sharing axes

`hint: plt.subplots, axes[].plot, axes[].set_title, axes[].set_xlabel, axes[].set_ylabel, axes[].grid, axes[].legend, fig.tight_layout`

In [ ]:

```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Primo sottoplot
axes[0].plot(x, y1,
             color='red',
             linestyle='--',
             marker='*',
             label=r"$y_1 = 2x + 1$")
axes[0].set_title('Plot 1')
axes[0].set_xlabel('x')
axes[0].set_ylabel('y')
axes[0].legend(title="Eq")
axes[0].grid(True, linestyle='--', linewidth=0.5)

# Secondo sottoplot
axes[1].plot(x, y2,
             color='blue',
             linestyle='-',
             marker='s',
             label=r"$y_2 = \log(x) + 1$")
axes[1].set_title('Plot 2')
axes[1].set_xlabel('x')
axes[1].set_ylabel('y')
axes[1].legend(title="Eq")
axes[1].grid(True, linestyle='--', linewidth=0.5)

# Terzo sottoplot
axes[2].plot(x, y3,
             color='green',
             linestyle='-',
             marker='+',
             label=r"$y_3 = x^2 - 1$")
```

```
axes[2].set_title('Plot 3')
axes[2].set_xlabel('x')
axes[2].set_ylabel('y')
axes[2].legend(title="Eq")
axes[2].grid(True, linestyle='--', linewidth=0.5)

fig.tight_layout()
plt.show()
```

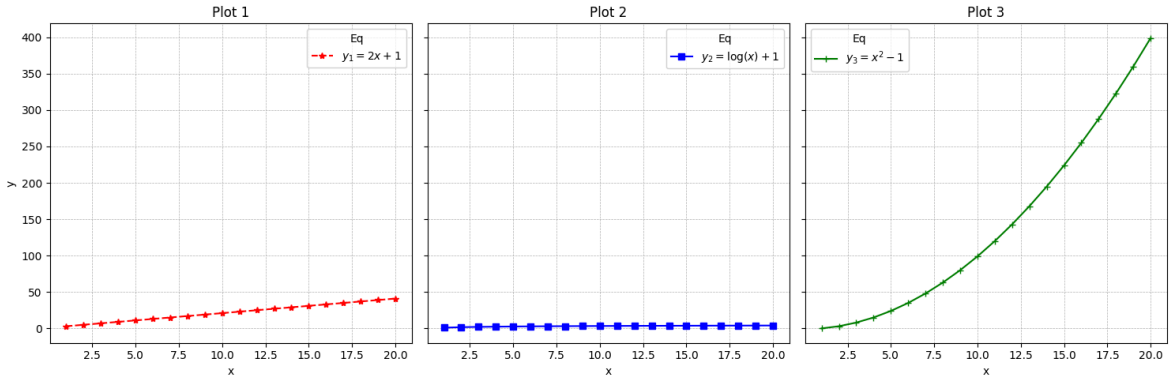


Replot the previous plot sharing y between suplots

`hint: sharey=True`

In [ ]:
```
fig, axes = plt.subplots(1, 3, figsize=(15, 5), sharey=True)

# Primo sottoplot
axes[0].plot(x, y1,
             color='red',
             linestyle='--',
             marker='*',
             label=r"$y_1 = 2x + 1$")
axes[0].set_title('Plot 1')
axes[0].set_xlabel('x')
axes[0].set_ylabel('y')
axes[0].legend(title="Eq")
axes[0].grid(True, linestyle='--', linewidth=0.5)

# Secondo sottoplot
axes[1].plot(x, y2,
             color='blue',
             linestyle='-',
             marker='s',
             label=r"$y_2 = \log(x) + 1$")
axes[1].set_title('Plot 2')
axes[1].set_xlabel('x')
axes[1].legend(title="Eq")
axes[1].grid(True, linestyle='--', linewidth=0.5)

# Terzo sottoplot
axes[2].plot(x, y3,
             color='green',
             linestyle='-',
             marker='+',
             label=r"$y_3 = x^2 - 1$")
axes[2].set_title('Plot 3')
axes[2].set_xlabel('x')
axes[2].legend(title="Eq")
```

```
axes[2].grid(True, linestyle='--', linewidth=0.5)

fig.tight_layout()
plt.show()
```



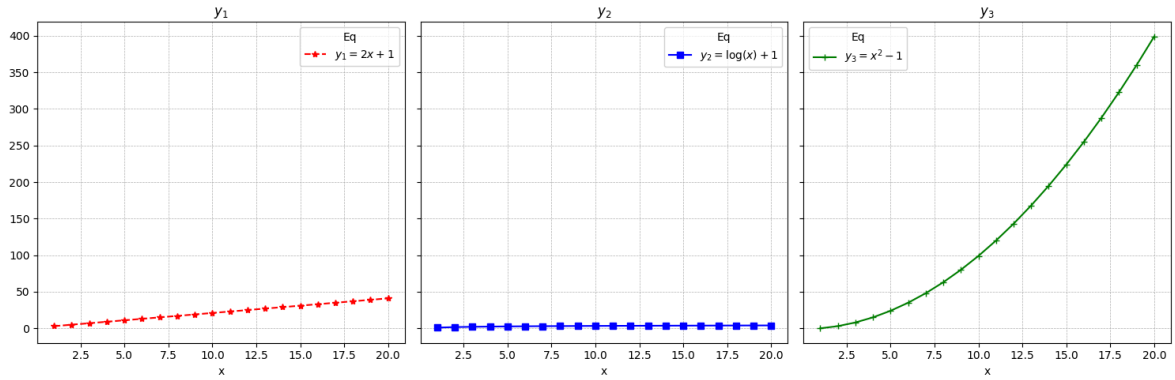
Replot the previous plot using the for loop, lists and dictionaries

`hint: functions = [...], {"title": ..., "y": ..., "label": ..., "linestyle": ...}, zip(functions, axes)`

In [ ]:

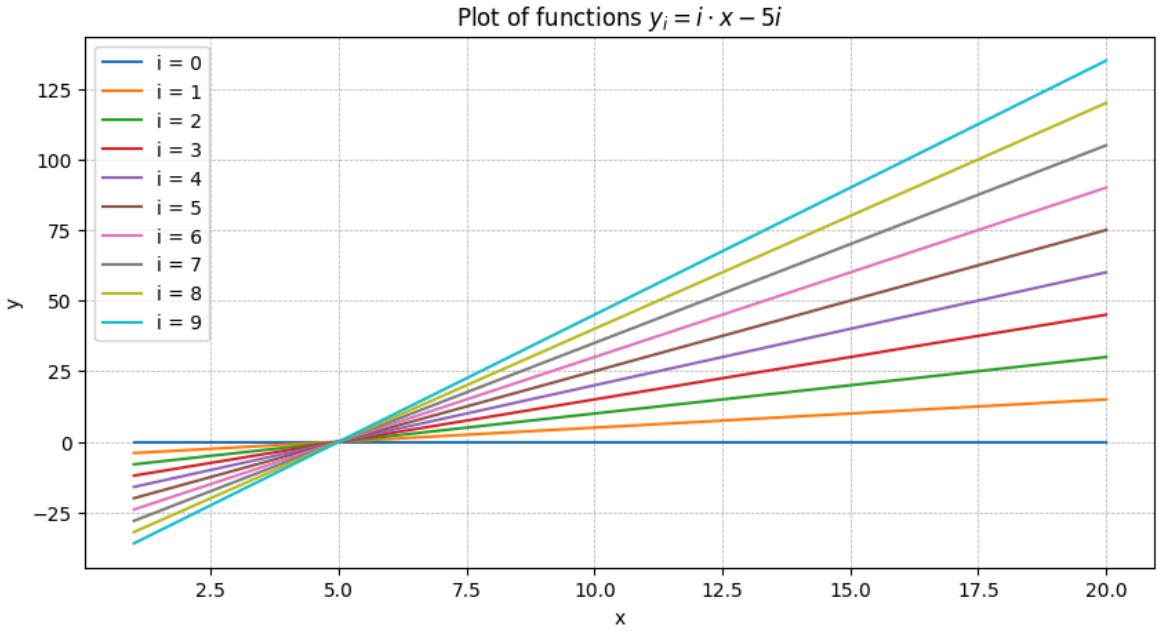
```
functions = [
    {"title": r"$y_1$",
     "y": y1,
     "label": r"$y_1 = 2x + 1$",
     "linestyle": '--',
     "color": 'red',
     "marker": '*'},
    {"title": r"$y_2$",
     "y": y2,
     "label": r"$y_2 = \log(x) + 1$",
     "linestyle": '-',
     "color": 'blue',
     "marker": 's'},
    {"title": r"$y_3$",
     "y": y3,
     "label": r"$y_3 = x^2 - 1$",
     "linestyle": '-',
     "color": 'green',
     "marker": '+'}
]

fig, axes = plt.subplots(1, 3, figsize=(15, 5), sharey=True)

for info, ax in zip(functions, axes):
    ax.plot(x, info["y"],
            color=info["color"],
            linestyle=info["linestyle"],
            marker=info["marker"],
            label=info["label"])
    ax.set_title(info["title"])
    ax.set_xlabel('x')
    ax.legend(title="Eq")
    ax.grid(True, linestyle='--', linewidth=0.5)
```

```
fig.tight_layout()
plt.show()
```



Plot functions $y_i = i \cdot x - 5i$ with i in range (0, 10) in the same plot with size (10,5) and a different color for each function

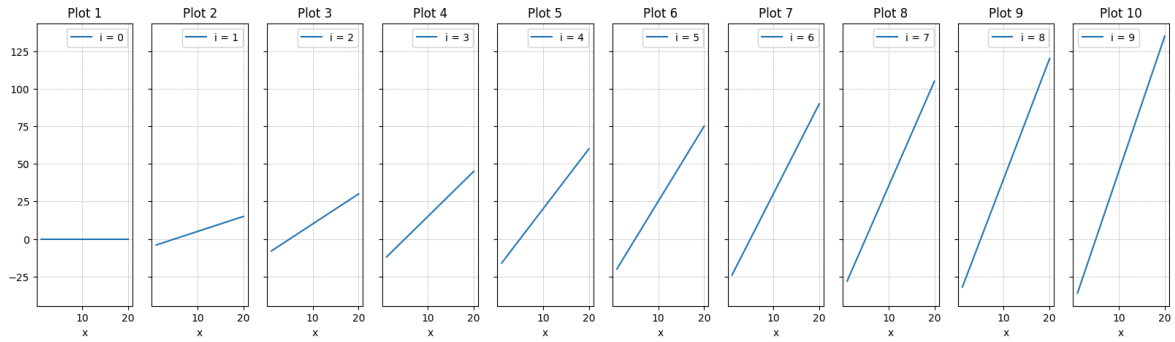
In [ ]:
```
plt.figure(figsize=(10, 5))
for i in range(0,10):
    y = i * x - 5 * i
    plt.plot(x, y, label=f"i = {i}")

plt.xlabel('x')
plt.ylabel('y')
plt.title('Plot of functions $y_i = i \cdot x - 5i$')
plt.legend()
plt.grid(True, linestyle='--', linewidth=0.5)
plt.show()
```



Plot functions $y_i = i \cdot x - 5i$ with i in range (0, 10) in different subplots with shared y and figure size (20,5)

In [ ]:
```
fig, axes = plt.subplots(1, 10, figsize=(20, 5), sharey=True)

for i, ax in enumerate(axes):
```

```
    y = i * x - 5 * i
    ax.plot(x, y, label=f"i = {i}")
    ax.set_xlabel('x')
    ax.set_title(f'Plot {i+1}')
    ax.grid(True, linestyle='--', linewidth=0.5)
    ax.legend()

plt.show()
```

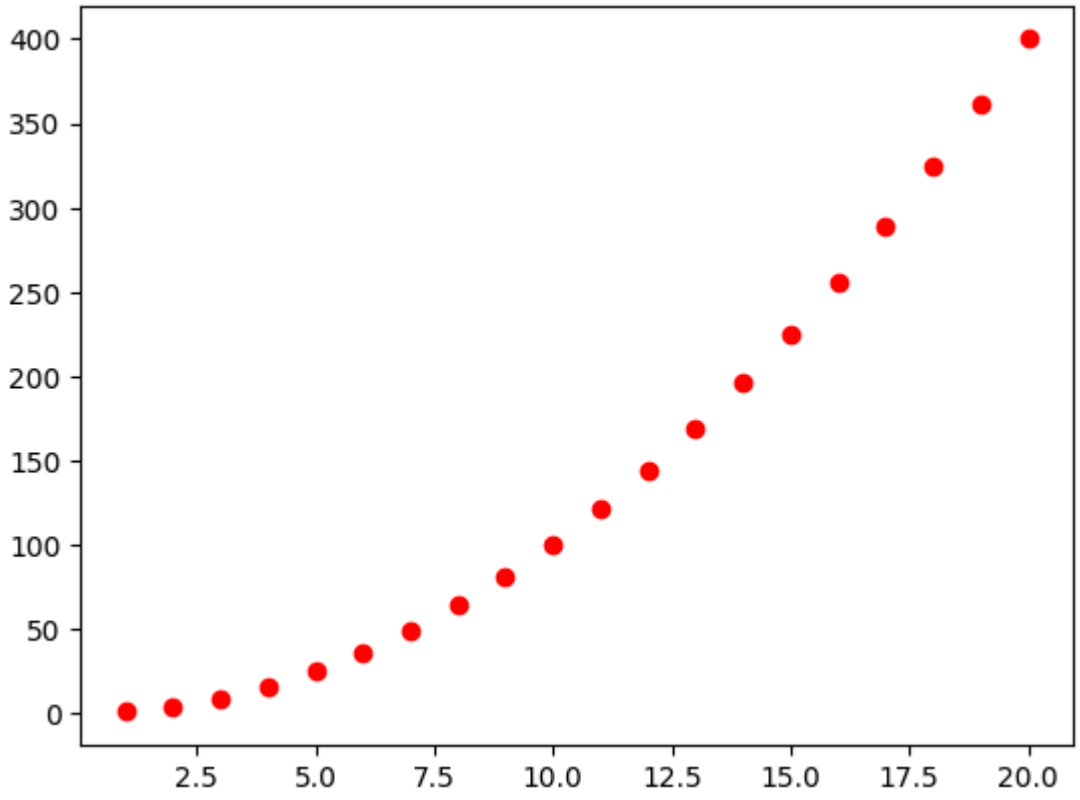


Intermediate

Plot a red scatterplot of x squared

`hint: plt.scatter`

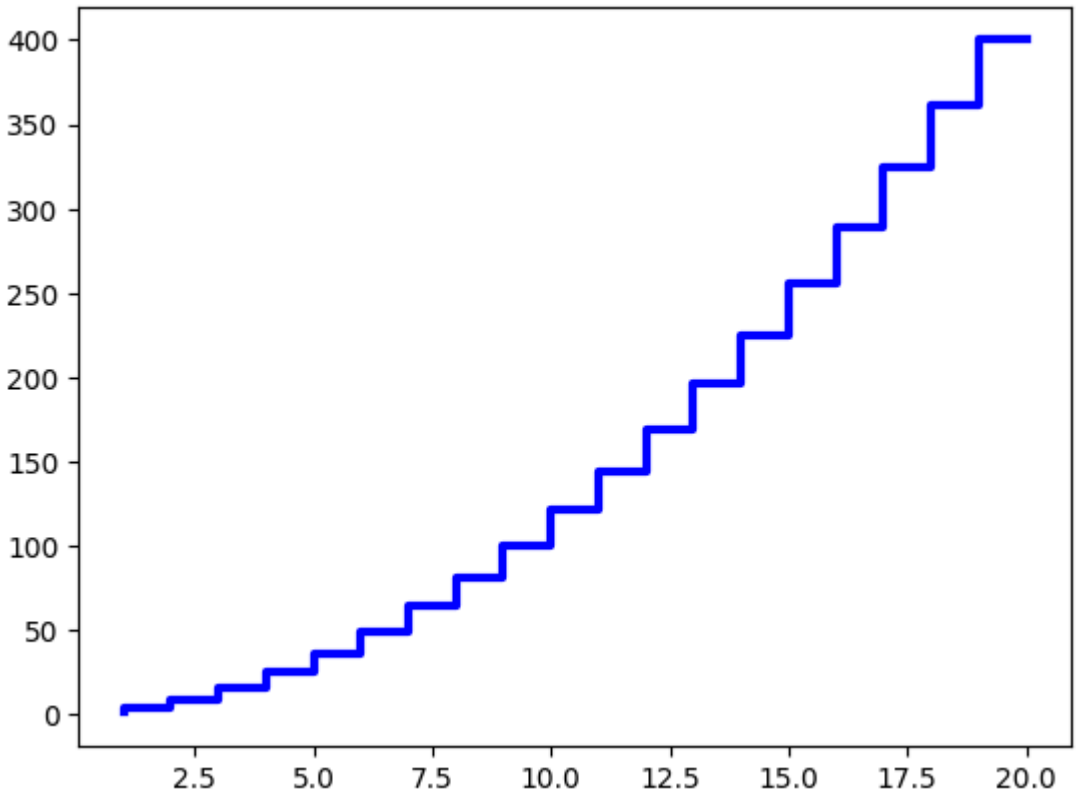
In [ ]:
```
y1 = x ** 2
plt.scatter(x,y1, color='red')
plt.show()
```



Plot a blue step plot of x squared with linewidth of 3

hint: plt.step

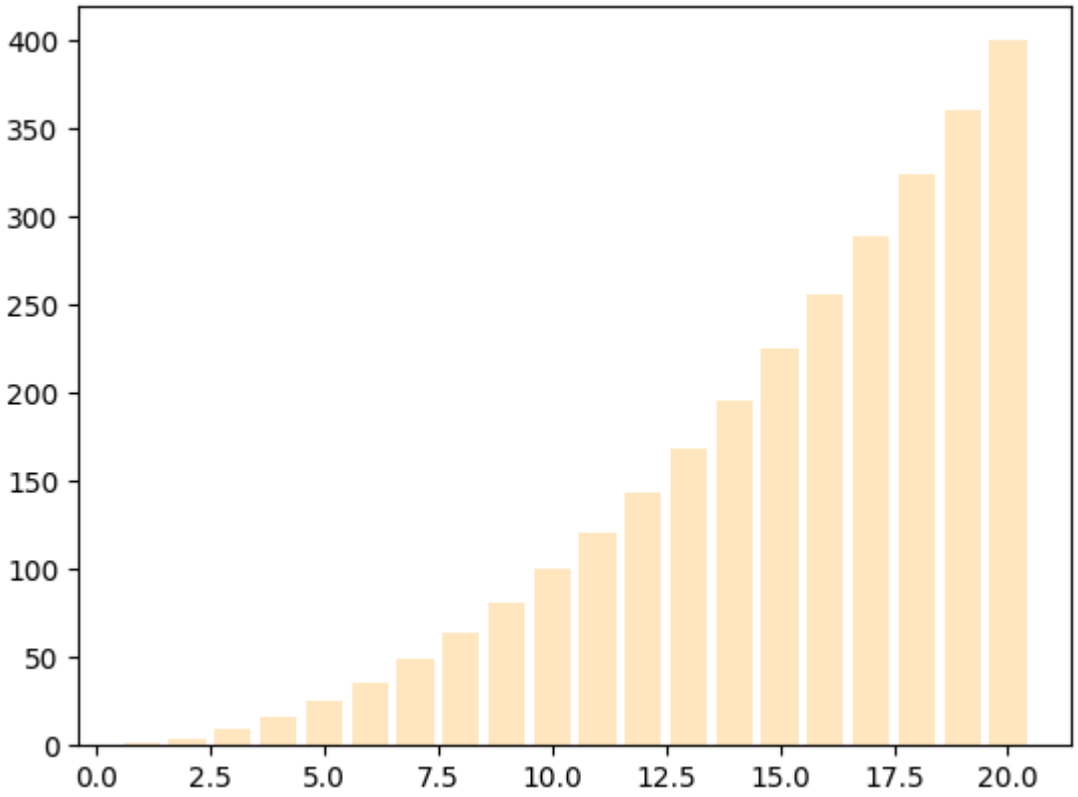
In [ ]:
```
plt.step(x,y1, color='blue', linewidth=3)
plt.show()
```



Plot an orange barplot plot of x squared with alpha of 0.25

hint: plt.bar

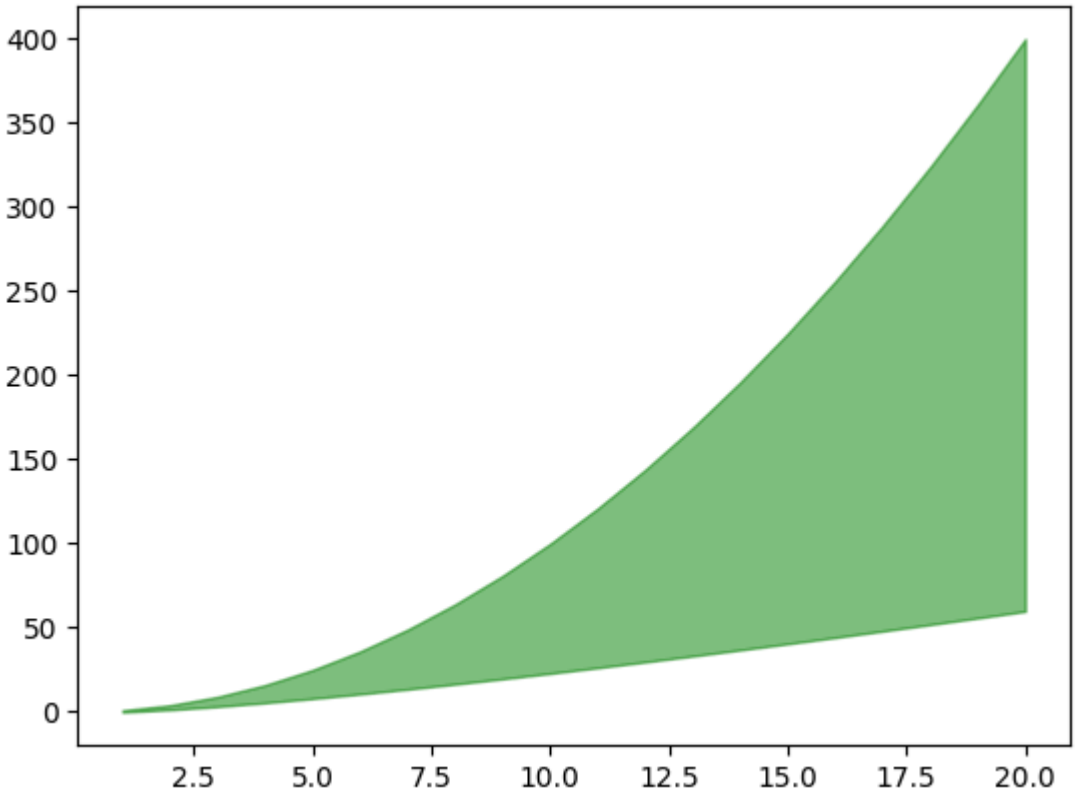
In [ ]:
```
plt.bar(x,y1, color='orange', alpha=0.25)
plt.show()
```



Plot the area between x squared and the $x \cdot log(x)$ function in green and alpha 0.5

`hint: plt.fill_between`

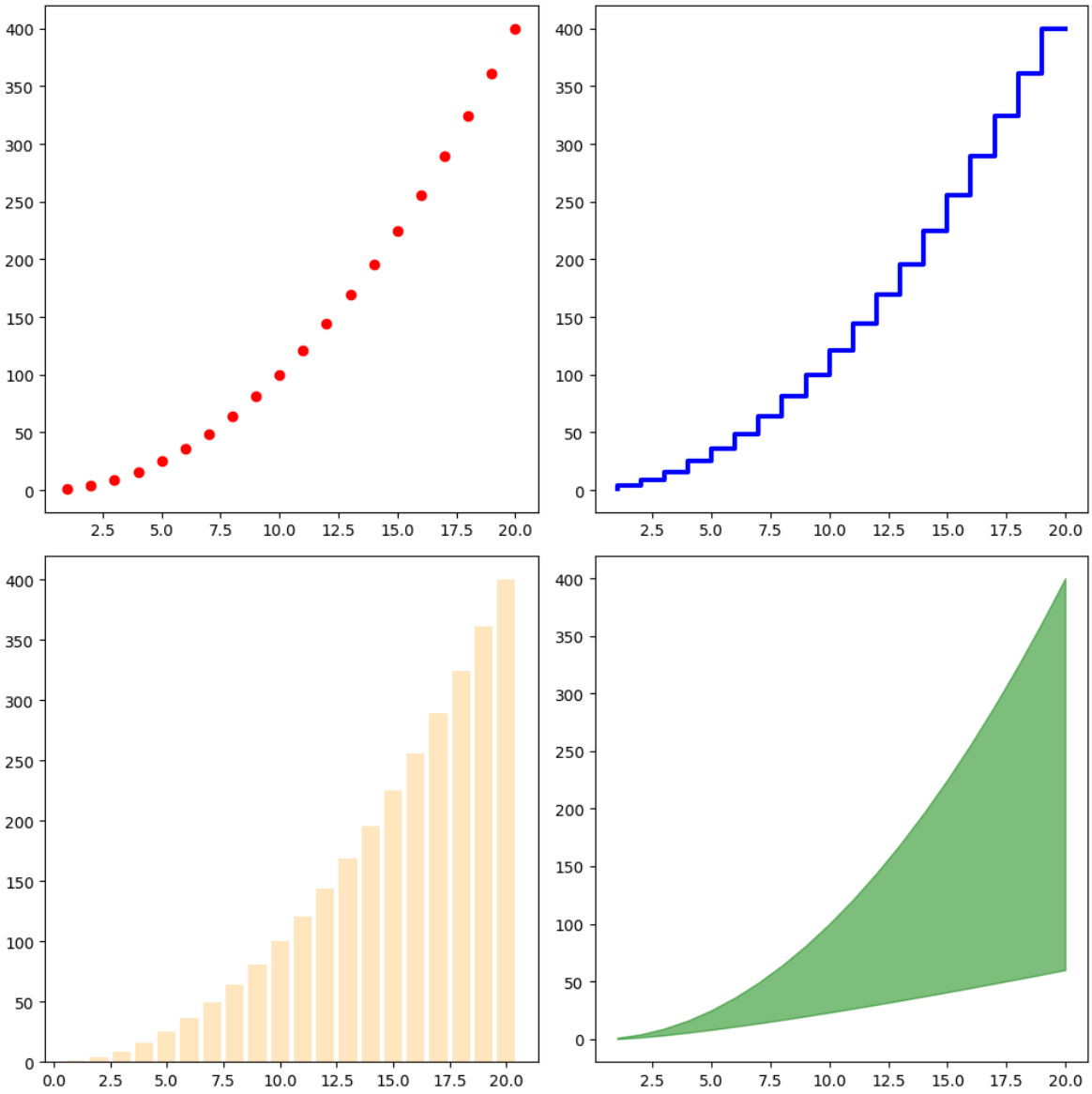
In [ ]:
```
y2 = x * np.log(x)
plt.fill_between(x,y1,y2,color='green', alpha=0.5)
plt.show()
```



Replot previous plots in square grid (2,2)

`hint: axes[][]`

In [ ]:
```
fig, axes = plt.subplots(2, 2, figsize=(10, 10))
axes[0, 0].scatter(x,y1, color='red')
axes[0, 1].step(x,y1, color='blue', linewidth=3)
axes[1, 0].bar(x,y1, color='orange', alpha=0.25)
axes[1, 1].fill_between(x,y1,y2,color='green', alpha=0.5)

plt.tight_layout()
plt.show()
```

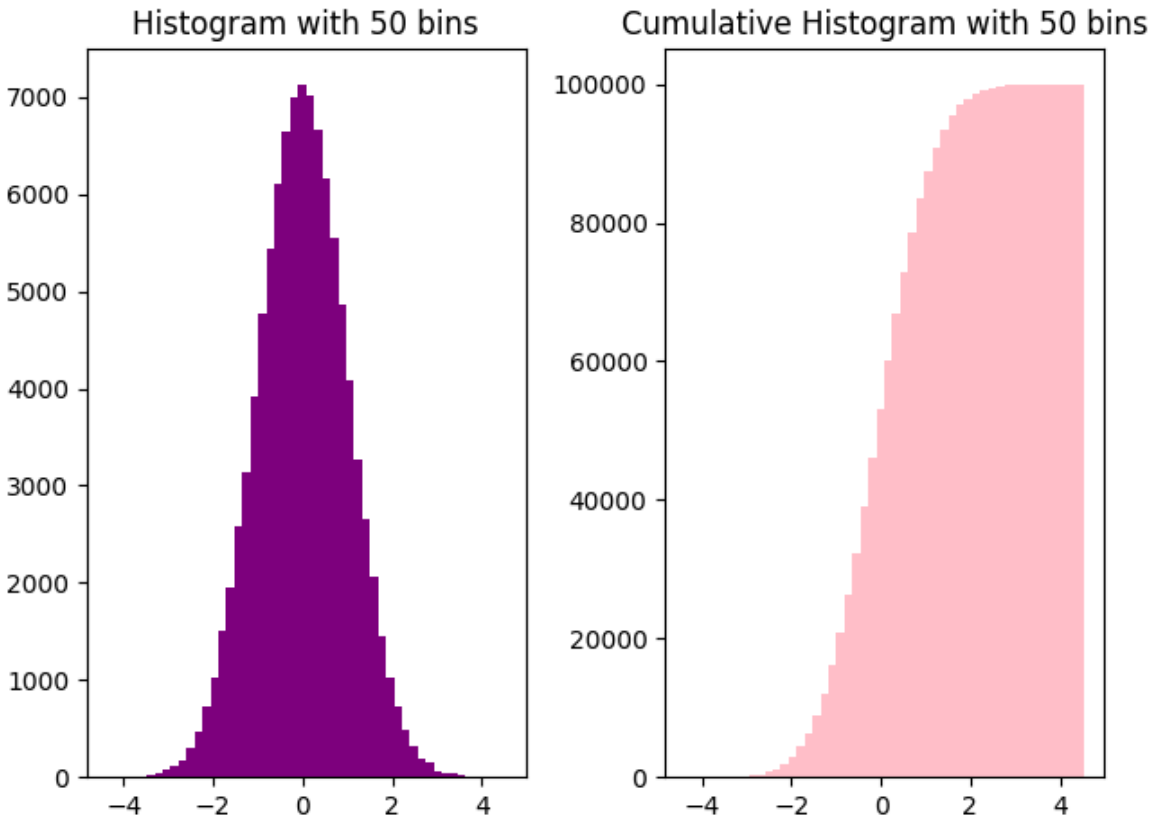


Plot purple histogram of 50 bins and pink cumulative instogram of 100K random samples in subplots

`hint: np.random.randn, axes[].hist(), cumulative=True, bins=50`

In [ ]:

```
random_samples = np.random.randn(100000)

fig, axes = plt.subplots(1, 2)

axes[0].hist(random_samples, bins=50, color='purple')
axes[0].set_title('Histogram with 50 bins')

axes[1].hist(random_samples, bins=50, cumulative=True, color='pink')
axes[1].set_title('Cumulative Histogram with 50 bins')

plt.tight_layout()
plt.show()
```


Histogram with 50 bins
Cumulative Histogram with 50 bins
7000
6000
5000
4000
3000
2000
1000
0
100000
80000
60000
40000
20000
0
−4
−2
0
2
4