CptS 111, Spring 2023
Lect. #24, Apr. 19, 2023
Class Notes

Today's Agenda:

1. Plotting basics
2. Styling plots
3. Adding text to plots

Ch. 10

Plotting

In PA #6 we used some rudimentary data visualization techniques to display the relative strengths of some Pokemon characters. However, *plotting* sets of data generated by others or by ourselves provides better data visualization and often leads to insights and discoveries. The `matplotlib` module for Python can be used to plot data in numerous ways.

- `matplotlib` is short for MATLAB Plotting Library
- It's Python's way of implementing the plotting package used in MATLAB, a proprietary software package popular in engineering because it's very useful for solving matrix problems.
- It isn't a standard Python library, i.e., you have to download it separately.
- For our purposes, we only need the `pyplot` module, one of the modules within the very powerful `matplotlib` module; the `matplotlib` module is very large and has a lot of separate modules (see https://matplotlib.org/3.3.2/py-modindex.html (https://matplotlib.org/3.3.2/py-modindex.html)).

1. Plotting Basics

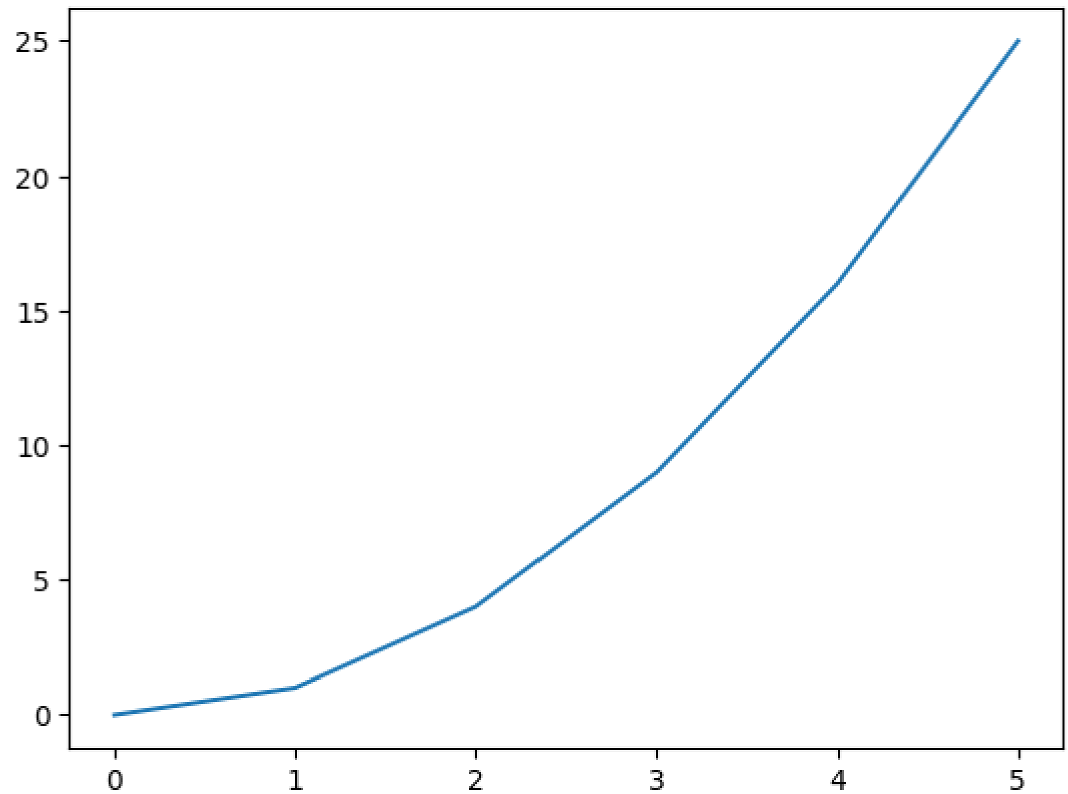
After installing `matplotlib` , we have to import the `pyplot` module in order to create plots.

In [1]:
```
# Import matplotlib.pyplot; using the plt alias is standard practice

import matplotlib.pyplot as plt
```

Note that `plt` is used as the default alias by many Python programmers. `matplotlib` functions often use lists as arguments. Let's consider the simplest example for creating a plot.

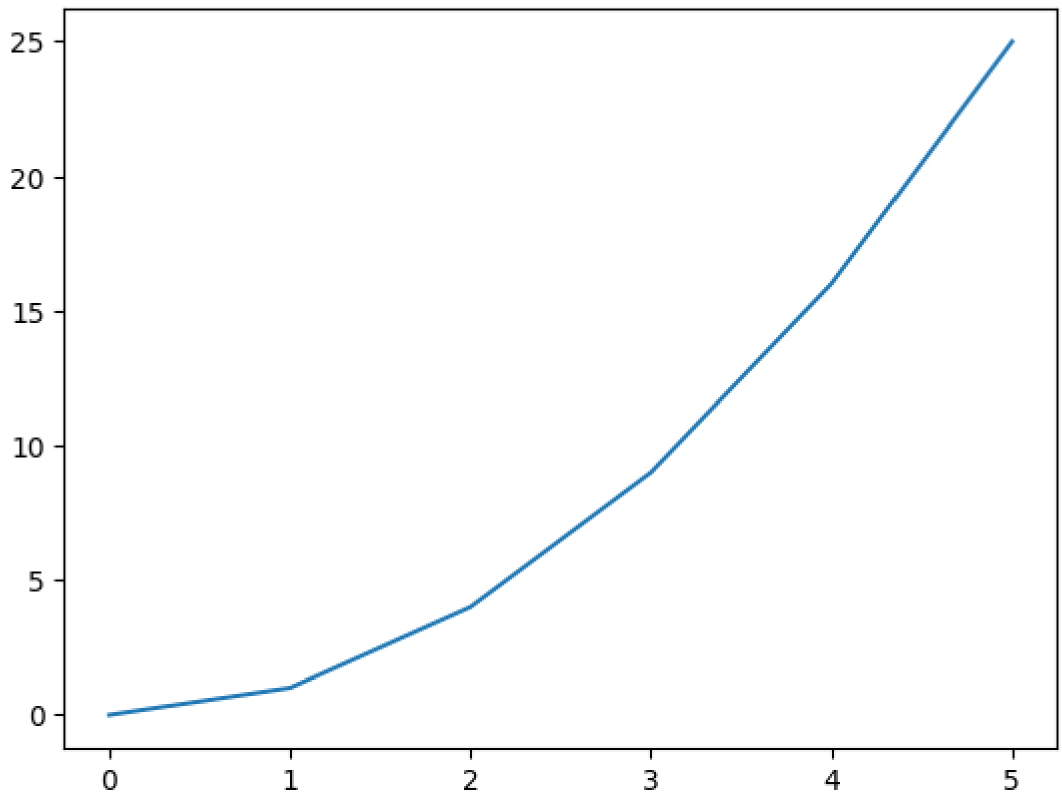
In [2]:

```
# Simplest plot example using lists as arguments
# plt.show() is always required to draw a plot

plt.plot([0, 1, 2, 3, 4, 5], [0, 1, 4, 9, 16, 25]) # plots data
plt.show()                                           # draws plot--statemen
```



`plt.plot()` is used to plot the data. Note that `plt.show()` is always required as the last statement. If you forget to use it, the plot won't be drawn. The list arguments we used above represent the `x` and `y` points. Thus, we could have also used the following.

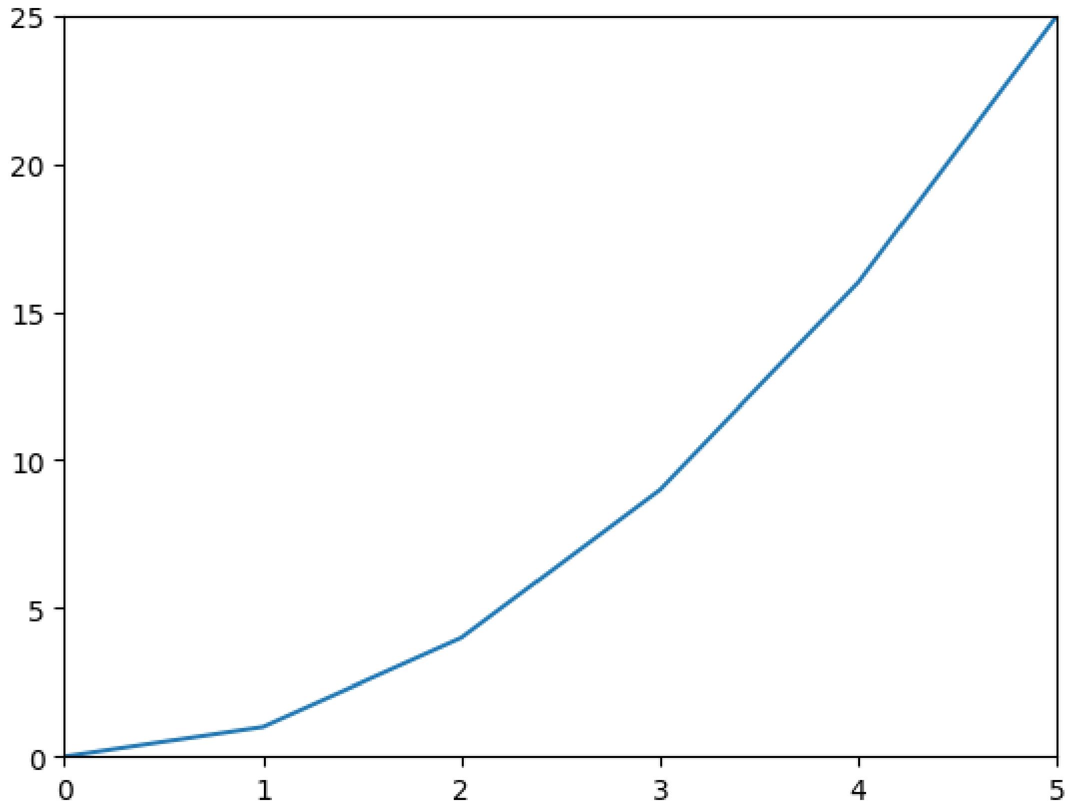
In [3]:
```
# Define lists and use them with plot()

x = [0, 1, 2, 3, 4, 5]       # values of x
y = [0, 1, 4, 9, 16, 25]     # values of y
plt.plot(x, y)
plt.show()
```



We can also use a list to change the values of the axis limits.

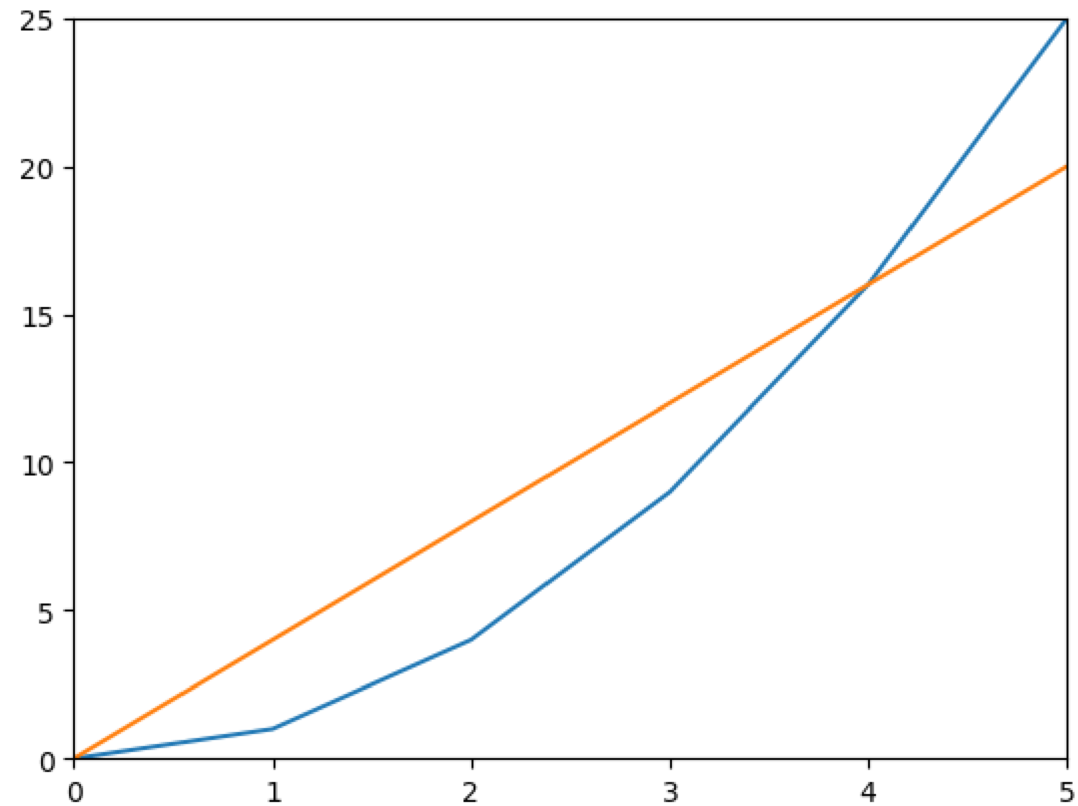
In [4]:

```
# Change axis values using axis() with a list argument

plt.plot(x, y)
plt.axis([0, 5, 0, 25])    # [x1, x2, y1, y2] list of axis limits
plt.show()
```



We can add more than one line to our plot simply by calling the `plot()` function again.

In [5]:
```
# We can draw as many lines as we want by calling plot() multiple times

plt.plot(x, y)
plt.plot(x, [0, 4, 8, 12, 16, 20])
plt.axis([0, 5, 0, 25])    # [x1, x2, y1, y2] list of axis limits
plt.show()
```



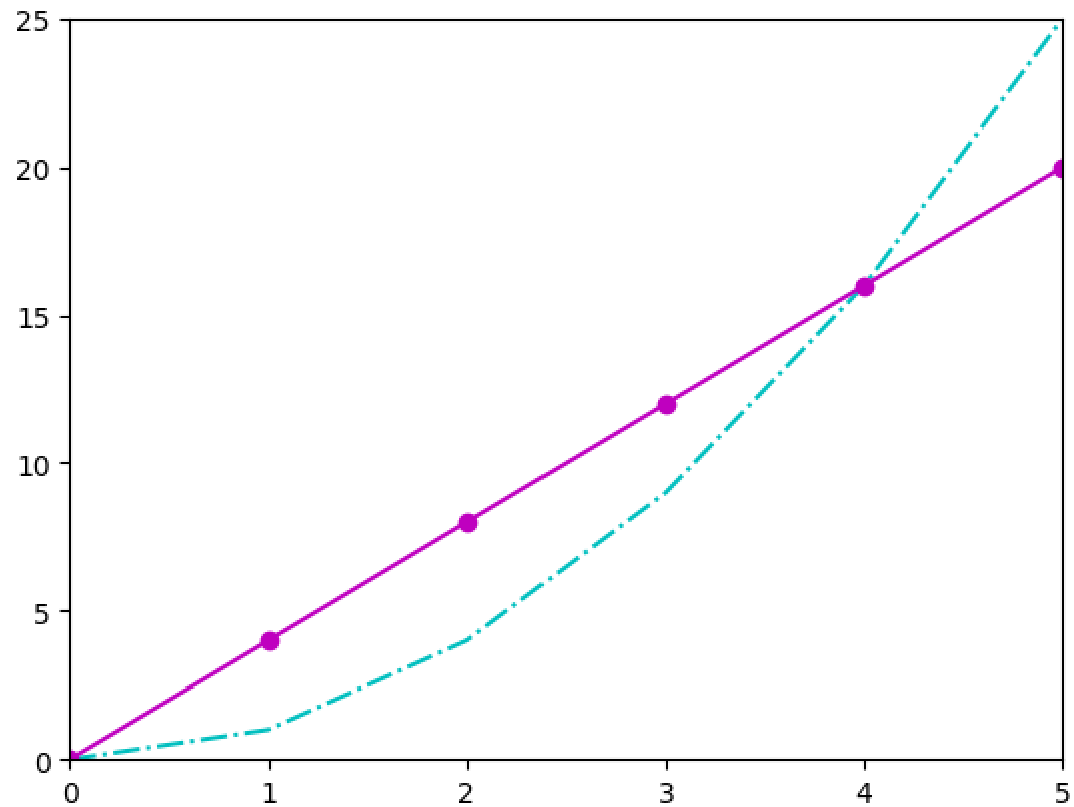
2. Styling Plots

As explained in zyBooks, we can use both format strings and keywords to change the look of our plots. For example,

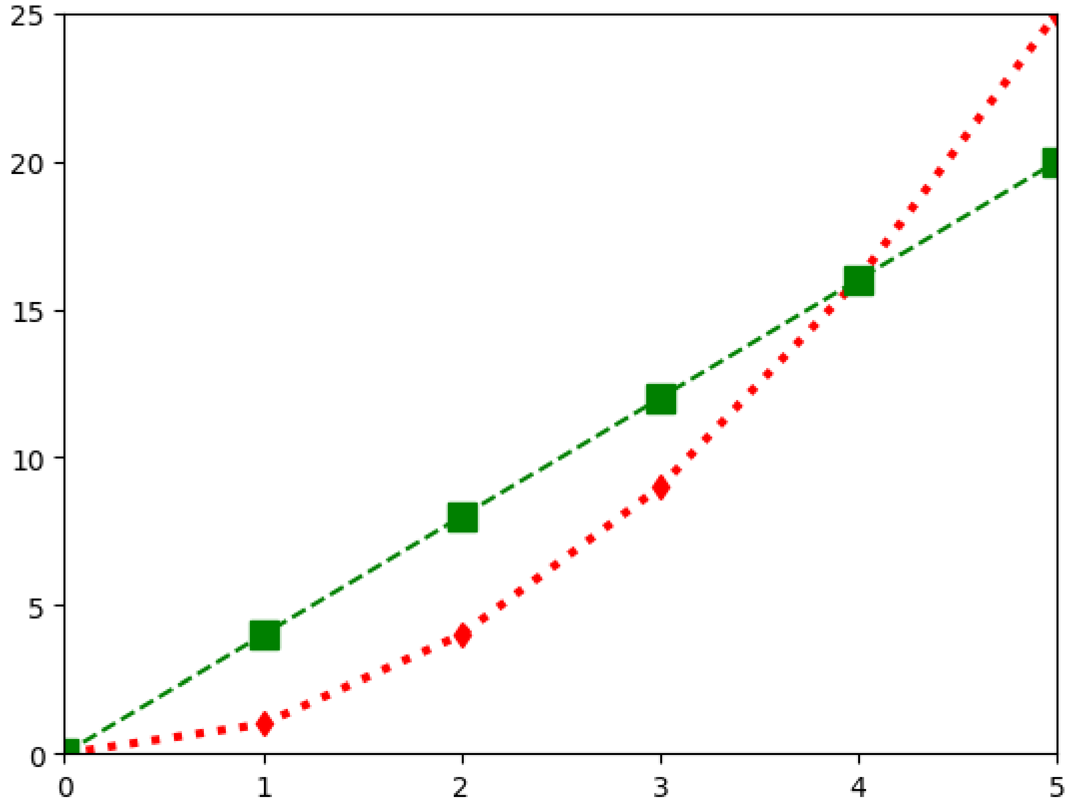
- `'ro-'` : solid red line with circle markers (color, marker, line style)
- `'g^'` : green triangle markers
- `'m--'` : dashed maroon line
- `'co-.'` : dashed-dot cyan line with circle markers
- linewidth: width of the line (keyword)
- markersize: size of the markers (keyword)
- label: label to use for a line in a legend (keyword)

and there are many more. Note that in the format strings, the color comes first, then marker-or-line or marker-and-line. Let's look at a few examples.

In [6]:
```
# We can change line colors and add markers to plots very easily
# This plot uses the default values for line widths and marker sizes

z = [0, 4, 8, 12, 16, 20]
plt.plot(x, y, 'c-.')        # dashed-dot cyan line
plt.plot(x, z, 'mo-')        # solid maroon line with circle markers
plt.axis([0, 5, 0, 25])
plt.show()
```



In [7]:
```
# We can also change the width of lines and the size of the markers

plt.plot(x, y, 'rd:', linewidth=3)     # dotted red line w/ diamond marker
plt.plot(x, z, 'gs--', markersize=10) # dashed line w/ big, green square
plt.axis([0, 5, 0, 25])
plt.show()
```



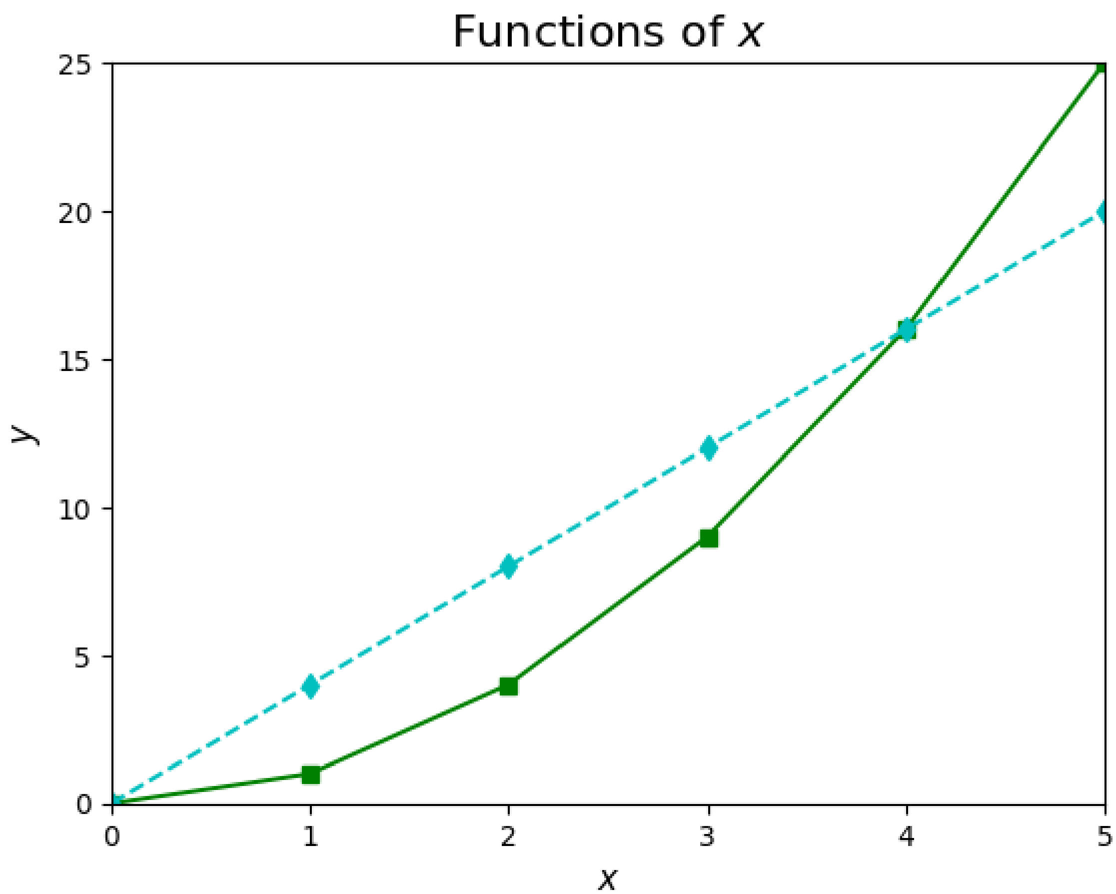
3. Adding Text to Plots

We can add text to plots including:

- axis labels
- plot title
- line labels for a legend
- annotations

We'll first add axis labels and a title to the previous figure.

In [8]:
```
# Add axis labels and title to our plot

plt.plot(x, y, 'gs-')
plt.plot(x, z, 'cd--')
plt.axis([0, 5, 0, 25])
plt.xlabel('$x$', fontsize=12)       # $'s delimit math font and operation
plt.ylabel('$y$', fontsize=12)       # fontsize controls font size
plt.title('Functions of $x$', fontsize=16)
plt.show()
```



Now let's add a legend.

In [9]:
```
# Plot with axes labels, title, and legend

plt.plot(x, y, 'gs-', label='$y=x^2$')
plt.plot(x, z, 'cd--', label='$y=4x$')
plt.axis([0, 5, 0, 25])
plt.xlabel('$x$', fontsize=12)       # $'s delimit math font and operation
plt.ylabel('$y$', fontsize=12)       # fontsize controls font size
plt.title('Functions of $x$', fontsize=16)
plt.legend(loc='upper left')         # left, center, right
plt.show()
```