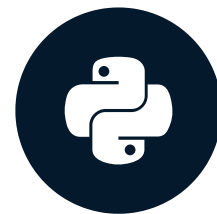


# Diving into NumPy arrays

PYTHON FOR MATLAB USERS



**Justin Kiggins**  
Product Manager

# Calculations on NumPy arrays

```
print(radius)
```

```
[[1, 2, 3]]
```

```
type(radius)
```

```
<class 'numpy.ndarray'>
```

```
area = np.pi * (radius ** 2)
print(area)
```

```
[3.14159265 12.56637061
 28.27433388]
```

| Operation      | Operator |
|----------------|----------|
| Addition       | +        |
| Subtraction    | -        |
| Multiplication | *        |
| Division       | /        |
| Exponentiation | **       |

# Math between NumPy arrays

```
type(revenue)
```

```
<class 'numpy.ndarray'>
```

```
print(revenue)
```

```
[[100, 114, 96, 120]]
```

```
type(cost)
```

```
<class 'numpy.ndarray'>
```

```
print(cost)
```

```
[102. 104. 101. 102.]
```

```
profit = revenue - cost  
print(profit)
```

```
[-2. 10. -5. 18.]
```

# Summarizing data in NumPy arrays

```
import numpy as np
```

```
np.mean(revenue)
```

```
107.5
```

Some useful NumPy functions:

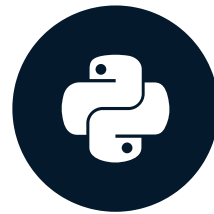
- `np.min()` returns smallest value in array
- `np.max()` returns largest value in array
- `np.sum()` returns sum of all values in array

# Let's practice!

PYTHON FOR MATLAB USERS

# Indexing NumPy arrays

PYTHON FOR MATLAB USERS



**Justin Kiggins**  
Product Manager

# Indexing with ranges

```
arr.shape
```

```
(100,)
```

```
slice = arr[10:15]
```

```
slice.shape
```

```
(5,)
```

# Indexing multidimensional arrays

```
image.shape
```

```
(1920, 1200)
```

```
window = image[100:150, 1000:1100]  
window.shape
```

```
(50, 100)
```

# Indexing with Boolean arrays

```
data.shape
```

```
(10000,)
```

```
is_valid.dtype
```

```
<np.bool>
```

```
np.sum(is_valid)
```

```
8732
```

```
valid_data = data[is_valid]  
valid_data.shape
```

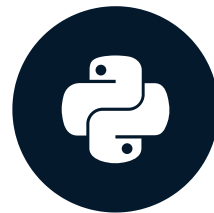
```
(8732,)
```

# Let's practice!

PYTHON FOR MATLAB USERS

# Lists

PYTHON FOR MATLAB USERS



**Justin Kiggins**  
Product Manager

# What is a list?

- Simple Python structure for storing data
- Lists can hold anything
- Similar to MATLAB's cell array
- But only one-dimensional
- Indexing like NumPy arrays

# Making lists

```
my_list = [8, 6, 7, 5, 3, 0, 9]  
print(my_list[2])
```

```
7
```

```
print(my_list[-3:])
```

```
[3, 0, 9]
```

# Making NumPy arrays from lists

```
my_list = [8, 6, 7, 5, 3, 0, 9]
```

```
import numpy as np
```

```
my_array = np.array(my_list)
```

```
type(my_array)
```

```
numpy.ndarray
```

# Multidimensional NumPy arrays from lists of lists

```
list_of_lists = [[2, 3], [9, 0], [1, 4]]  
import numpy as np  
arr = np.array(list_of_lists)  
print(arr)
```

```
[[2, 3]  
 [9, 0]  
 [1, 4]]
```

```
arr.shape
```

```
(3, 2)
```

# Differences between lists and NumPy arrays

| NumPy Arrays                                | Lists                               |
|---------------------------------------------|-------------------------------------|
| All elements must be same type              | Can mix types                       |
| <code>(+)</code> does element-wise addition | <code>(+)</code> concatenates lists |
| Multidimensional                            | Single dimension                    |
| Range & Boolean indexing                    | Only Range indexing                 |

# When to use each

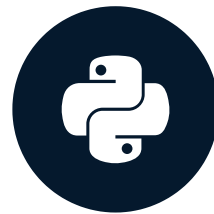
- Need to do math?
  - NumPy array
- Storing complex structures?
  - list
- Multidimensional data?
  - NumPy array

# Let's practice!

PYTHON FOR MATLAB USERS

# Customizing plots

PYTHON FOR MATLAB USERS



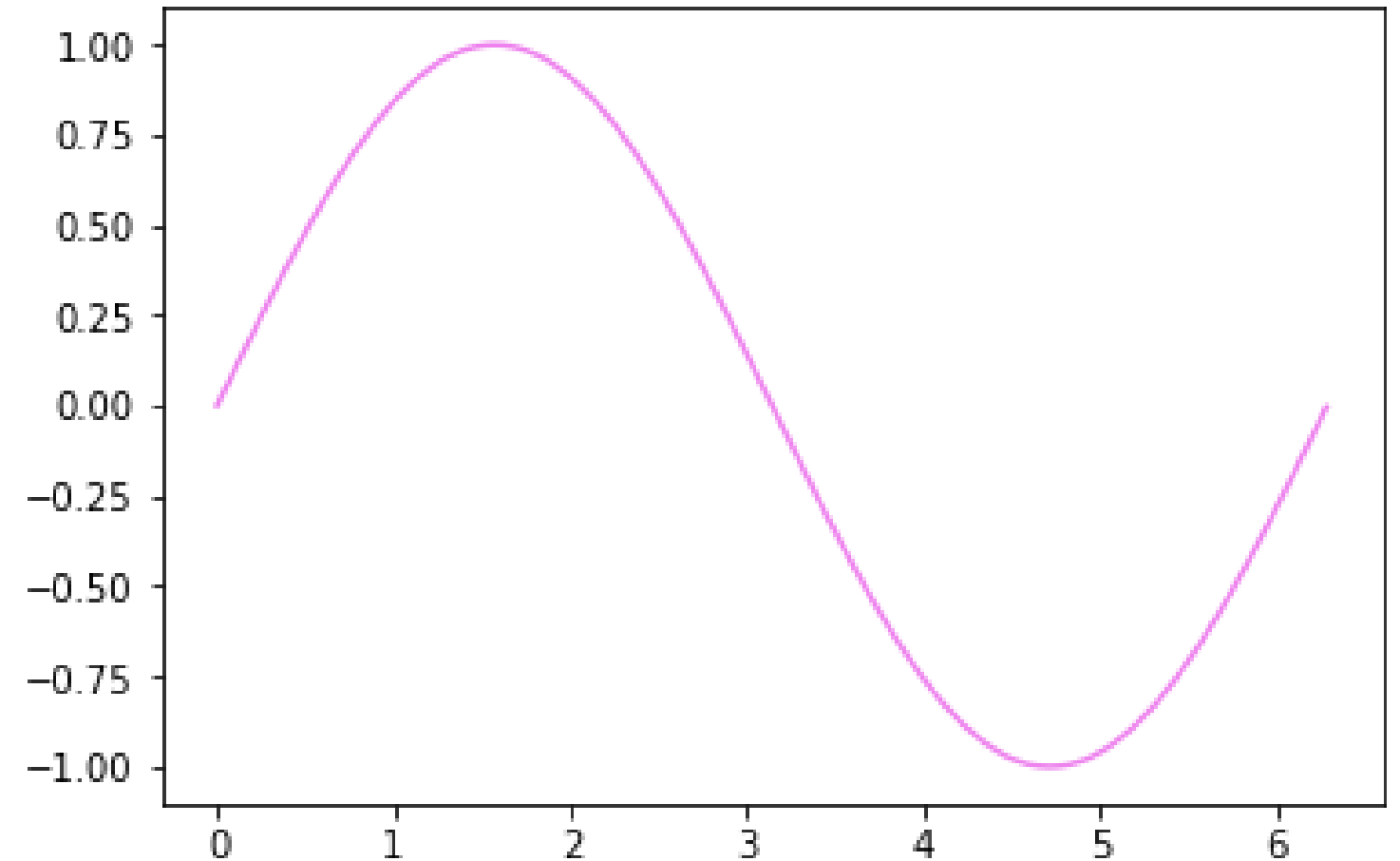
**Justin Kiggins**  
Product Manager

# Custom colors

```
import numpy as np
import matplotlib.pyplot as plt

# Generate data
x = np.arange(0, 2*np.pi, 0.01)
y = np.sin(x)

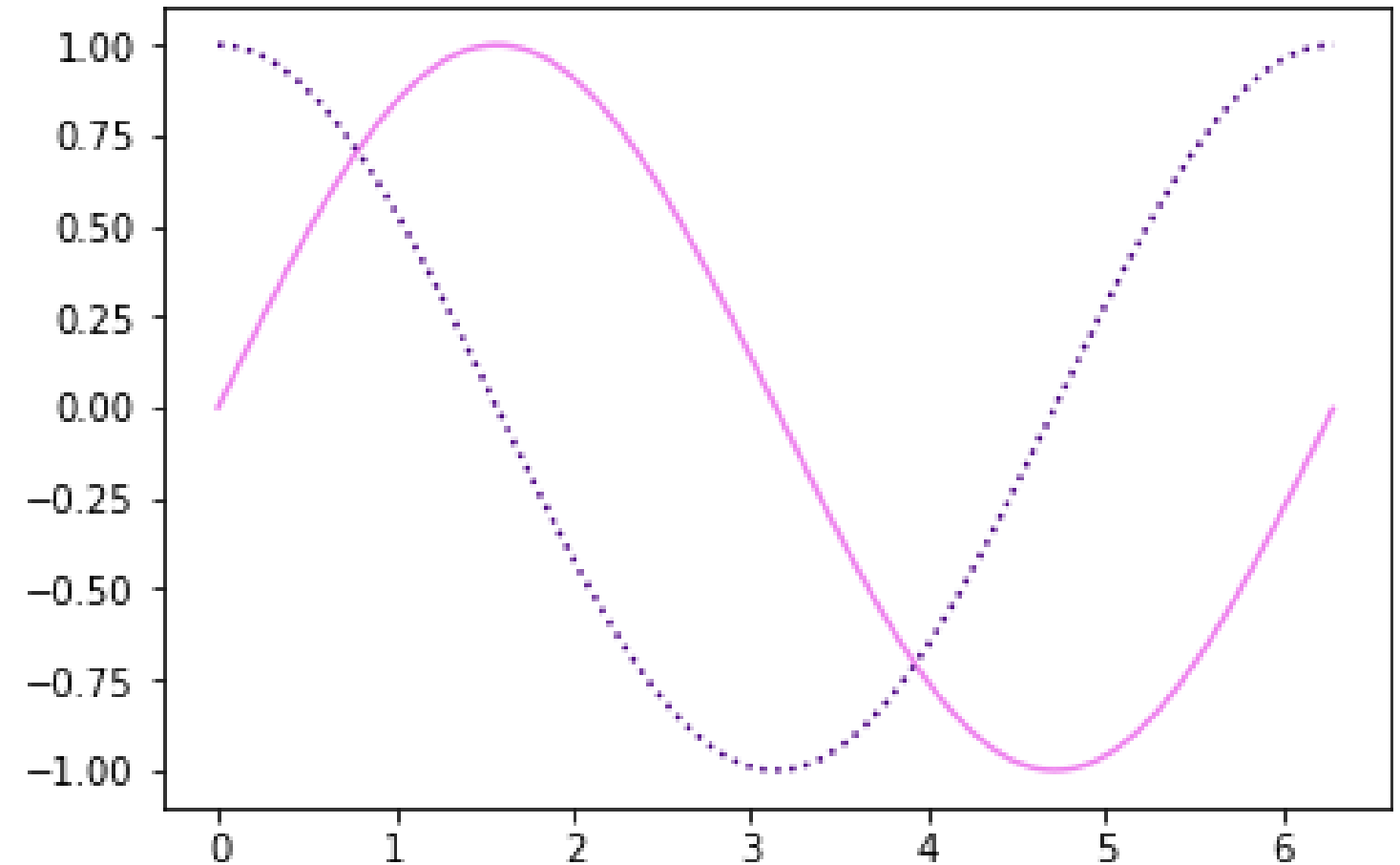
# Plot data
plt.plot(x, y, color='violet')
plt.show()
```



# Custom line styles

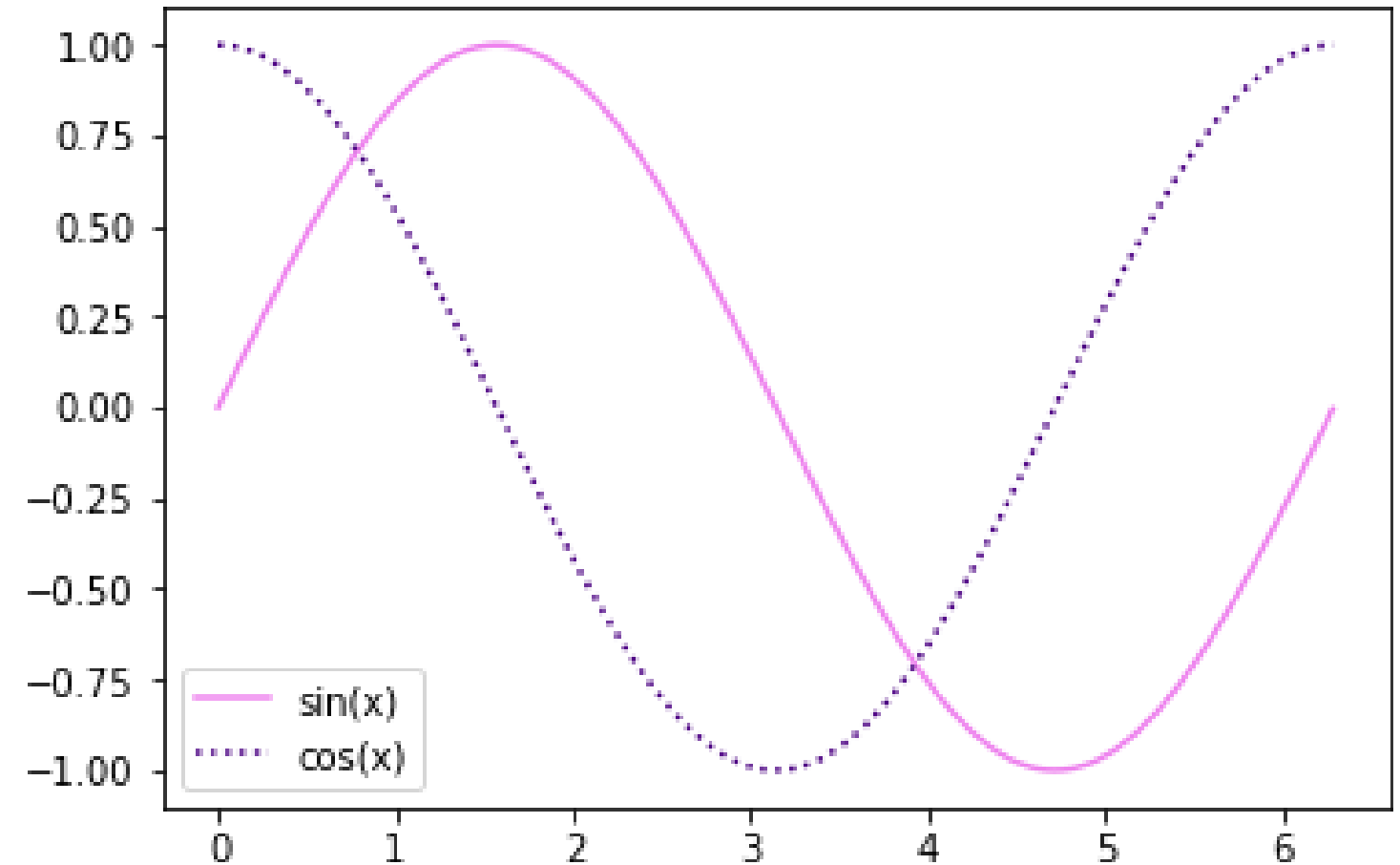
```
import numpy as np
import matplotlib.pyplot as plt
# Generate data
x = np.arange(0, 2*np.pi, 0.01)
y = np.sin(x)
y2 = np.cos(x)

# Plot data
plt.plot(x, y, color='violet')
plt.plot(x, y2, color='indigo',
         linestyle=':')
plt.show()
```



# Adding a legend

```
import numpy as np
import matplotlib.pyplot as plt
# Generate data
x = np.arange(0, 2*np.pi, 0.01)
y = np.sin(x)
y2 = np.cos(x)
# Plot data
plt.plot(x, y, color='violet',
         label='sin(x)')
plt.plot(x, y2, color='indigo',
         linestyle=':',
         label='cos(x)')
# Add a legend
plt.legend()
plt.show()
```

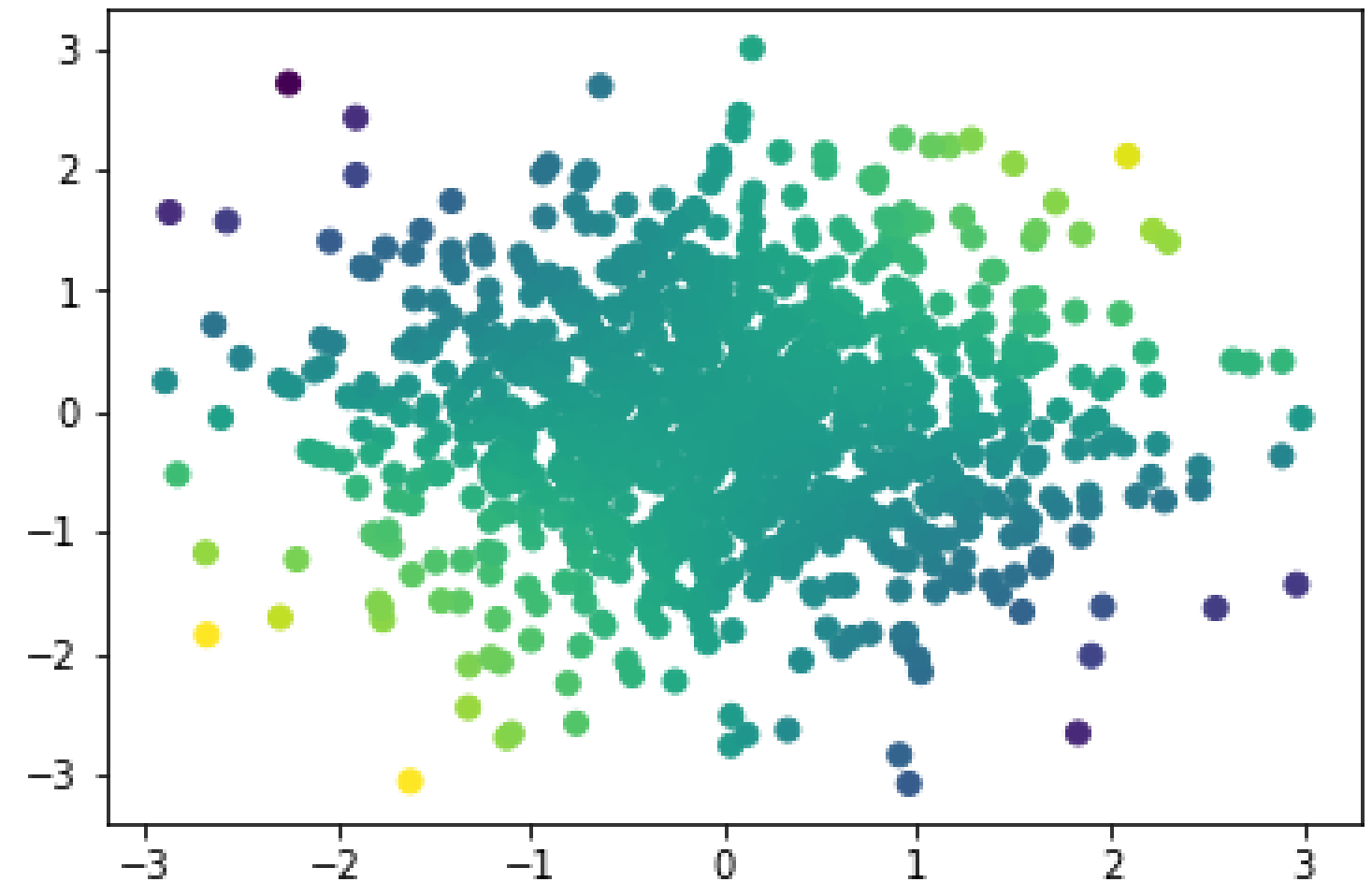


# Encoding data in marker color

```
import numpy as np
import matplotlib.pyplot as plt

# Generate data
x = np.random.randn(1000)
y = np.random.randn(1000)
z = x * y

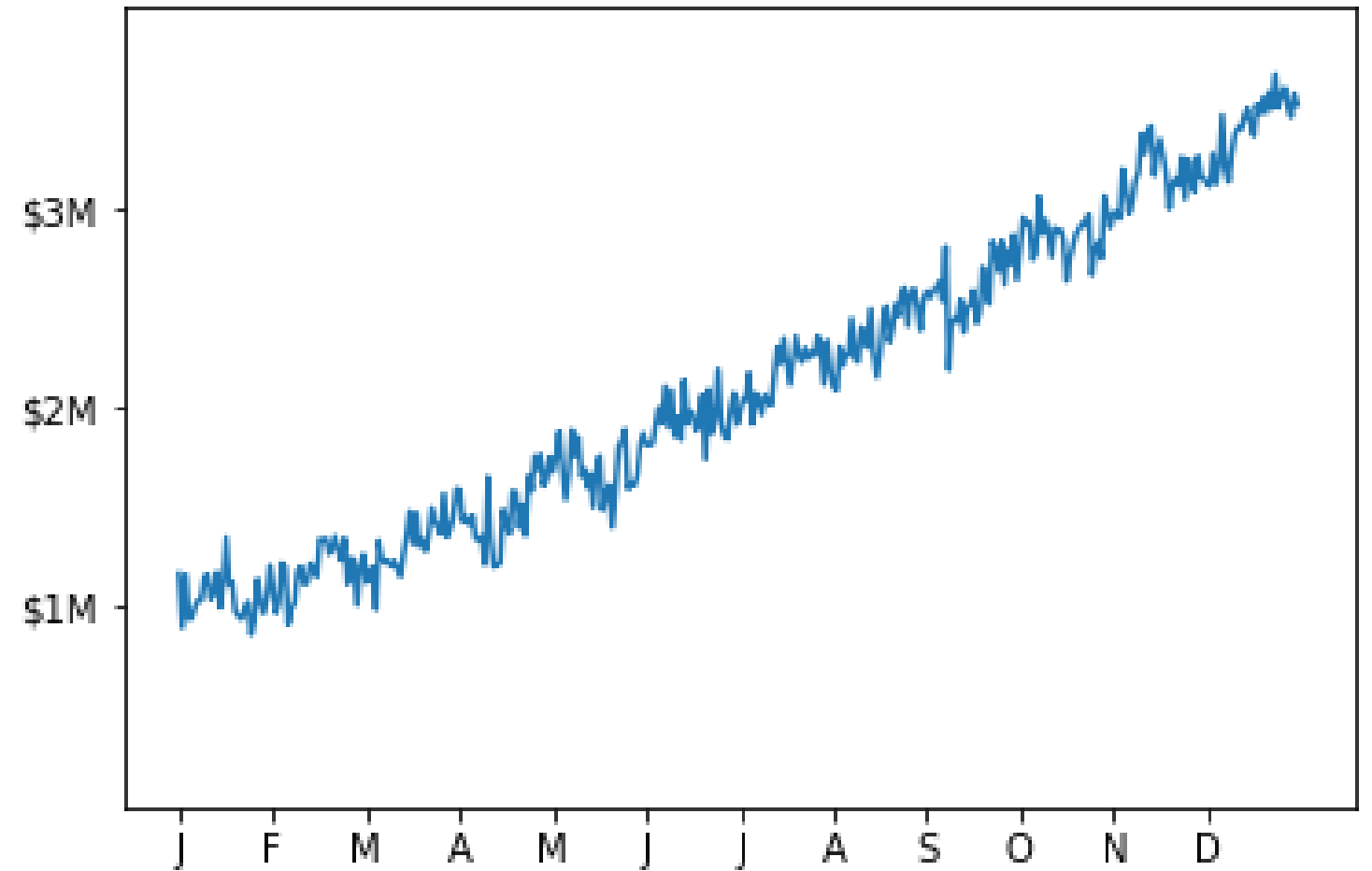
# Plot the data
plt.scatter(x, y, c=z)
plt.show()
```



# Custom tick labels

```
# Create the plot
plt.plot(months, revenue)

# Customize the ticks
plt.yticks(
    [1, 2, 3],
    ['$1M', '$2M', '$3M'])
plt.xticks(
    [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
    list('JFMAMJJASOND'))
```



# Let's plot some data!

PYTHON FOR MATLAB USERS