Markdown Cell

- It is designed to allow for text explanations
- Can be used to create intricate layout including tables with extra packages
- Can create ordered and unordered lists
- Can create code snippets
- This is the *Markdown Guide*.

```
for i in range(5):
    print(i)
```

Before evaluating the cell think about these two questions.

1. What is going to be the result of evaluating this cell?
2. What will be the output?

```
a = 'pears'
b = 'bananas'
x = 8
y = -23
```

There was no output because all the cell does is assign values to variables; there are no print statements or return statements.

```
print(f"a = {a}")
print(f"b = {b}")
print(f"x = {x}")
print(f"y = {y}")
```

```
print(f"x * a = {a*x}")
```

```
import math

def get_sqr_root(x):
    return print(f"The square root of {x} is {math.sqrt(x)}")
```

```
get_sqr_root(8)
```

Now we are going to examine the changes made in previous cells. In the cell where variables were assigned change `a = "apples"` to `a = "pears"`. Don't forget to re-evaluate the cell.

Consider the following questions:

1. Why were the cells after this cell not changed?
2. What is the value of `a`?
3. What will happen if we re-evaluate the cell where be multiply `a` and `x`, but not the cell where we print the value of `a`?

```
get_sqr_root(16)
```

```
get_sqr_root(100)
```

```
# imports and matplotlib setup
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

# set x to be 0 to 4+pi in .1 increments
x = np.arange(0,4*np.pi,0.1)

# the sine and cosine values
y = np.sin(x)
z = np.cos(x)

# plot them
plt.plot(x,y,x,z)
plt.show()
```