Notebook Imports and Packages

In [1]:

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

%matplotlib inline
```

A simple cost function

$$f(x) = x^2 + x + 1$$

In [2]:

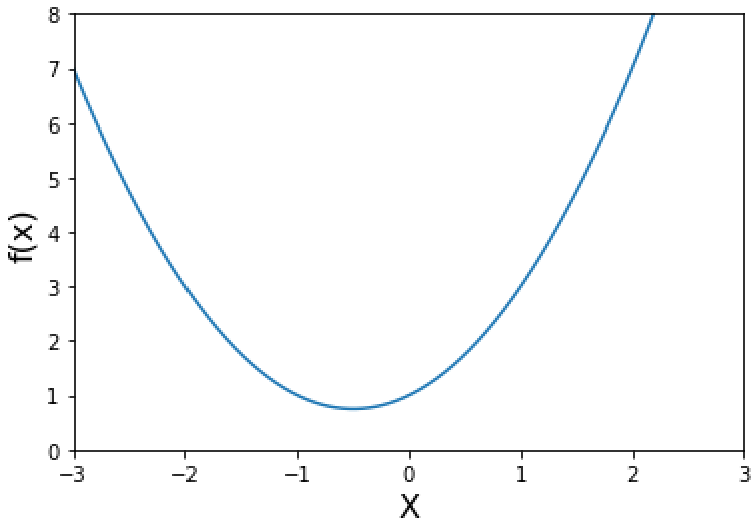
```
def f(x):
    return x**2 + x + 1
```

In [6]:

```
# Make Data
x_1 = np.linspace(start=-3, stop=3, num=500)
```

In [10]:

```
# Plot
plt.xlim([-3,3])
plt.ylim(0,8)
plt.xlabel('X', fontsize=16)
plt.ylabel('f(x)', fontsize=16)
plt.plot(x_1, f(x_1))
plt. show()
```



Slope & Derivatives

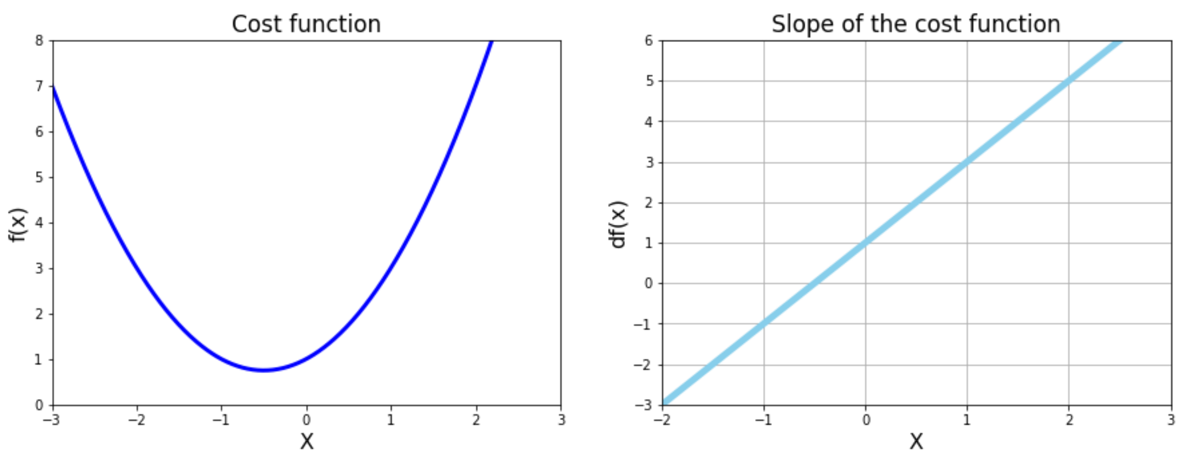
Create a python function for the derivative of f(x) called df(x)

In [11]:

```
def df(x):
    return 2*x +1
```

In [17]:

```
# Plot function and derivative function

plt.figure(figsize=[15,5])

# 1 Chart: Cost function
plt.subplot(1,2,1)

plt.xlim([-3,3])
plt.ylim(0,8)

plt.title('Cost function', fontsize=17)
plt.xlabel('X', fontsize=16)
plt.ylabel('f(x)', fontsize=16)
plt.plot(x_1, f(x_1), color ='blue', linewidth=3)

#2 Chart: Derivative
plt.subplot(1,2,2)

plt.title('Slope of the cost function', fontsize=17)
plt.xlabel('X', fontsize=16)
plt.ylabel('df(x)', fontsize=16)
plt.grid()

plt.xlim([-2,3])
plt.ylim(-3,6)

plt.plot(x_1, df(x_1), color='skyblue', linewidth=5)

plt. show()
```



Python Loops & Gradient Descent

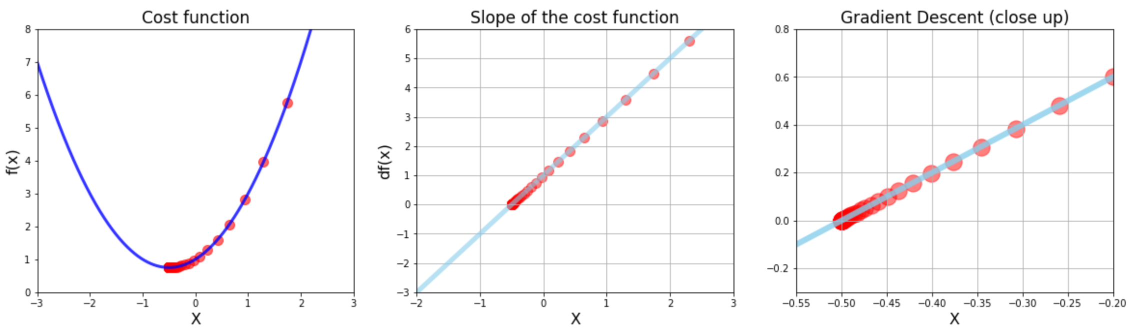
In [25]:

```
# Gradient Descent
new_x = 3
previous_x = 0
step_multiplier = 0.1
precision = 0.00001

x_list = [new_x]
slope_list = [df(new_x)]

for n in range(500):
    previous_x = new_x  #make a prediction
    gradient = df(previous_x)
    new_x = previous_x - step_multiplier * gradient

    step_size = abs(new_x - previous_x)
    # print(step_size)

    x_list.append(new_x)
    slope_list.append(df(new_x))

    if step_size < precision:
        print('Loop ran this many times: ', n)
        break

print('Local minimum occurs at: ', new_x)
print('Slope or df(x) at this point is: ', df(new_x))
print('f(x) value or cost at this point is: ', f(new_x))
```

```
Loop ran this many times:  50
Local minimum occurs at:  -0.49996003706460423
Slope or df(x) at this point is:  7.992587079153068e-05
f(x) value or cost at this point is:  0.7500000015970362
```

In [35]:

```
# Superimpose the gradient descent calculation on plot
plt.figure(figsize=[20,5])

# 1 Chart: Cost function
plt.subplot(1,3,1)

plt.xlim([-3,3])
plt.ylim(0,8)

plt.title('Cost function', fontsize=17)
plt.xlabel('X', fontsize=16)
plt.ylabel('f(x)', fontsize=16)

plt.plot(x_1, f(x_1), color ='blue', linewidth=3, alpha=0.8)

values = np.array(x_list)
plt.scatter(x_list, f(values), color = 'red', s=100, alpha=0.6)

#2 Chart: Derivative
plt.subplot(1,3,2)

plt.title('Slope of the cost function', fontsize=17)
plt.xlabel('X', fontsize=16)
plt.ylabel('df(x)', fontsize=16)
plt.grid()

plt.xlim([-2,3])
plt.ylim(-3,6)

plt.plot(x_1, df(x_1), color='skyblue', linewidth=5, alpha=0.6)

plt.scatter(x_list, slope_list, color='red', s=100, alpha=0.5)

#3 Chart: Derivative (Close Up)
plt.subplot(1,3,3)

plt.title('Gradient Descent (close up)', fontsize=17)
plt.xlabel('X', fontsize=16)
plt.grid()
plt.xlim([-0.55,-0.2])
plt.ylim(-0.3,0.8)

plt.plot(x_1, df(x_1), color='skyblue', linewidth=6, alpha=0.8)

plt.scatter(x_list, slope_list, color='red', s=300, alpha=0.5)


plt. show()
```



In [ ]:

```
```