HW1 Programming Problem 4 (30 points)

Problem Description

In this problem you will implement gradient descent on the following function: $f(x) = x^2 + 3x + 6\sin(x)$. You will define your own gradient function $\texttt{fgrad}(x)$, and then using the provided learning rate $\eta = 0.15$ and initial guess $x_0 = 8$, you will print the value of x and $f(x)$ for the first 10 iterations.

Fill out the notebook as instructed, making the requested plots and printing necessary values.

Summary of deliverables:

Functions:

- `fgrad(x)`

Results:

- Printed values of x and $f(x)$ for the first 10 iterations of gradient descent

Discussion:

- Do your printed values appear to be converging towards the minimum of the function?

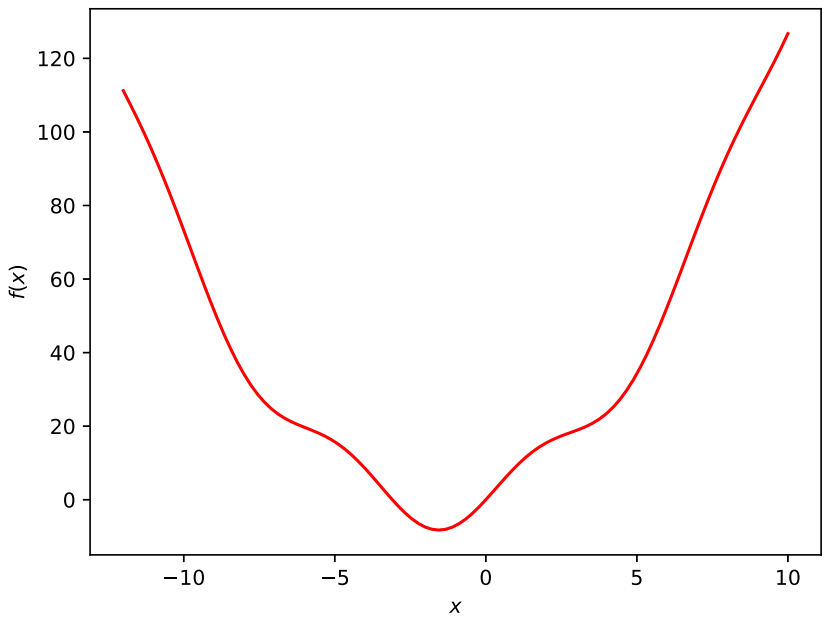
Imports and provided functions:

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return x**2 + 3*x + 6*np.sin(x)

def plotfx():
    # Sample function
    xs = np.linspace(-12,10,100)
    ys = f(xs)
    # Plot function
    plt.plot(xs,ys,'r-')
    plt.xlabel('$x$')
    plt.ylabel('$f(x)$')
    plt.show()

# Visualize the function
plotfx()
```



First define the function $\texttt{fgrad}(x)$

In [ ]:
```
# Your fgrad(x) function goes here
def fgrad(x):
    return 2*x + 3 + 6*np.cos(x)
```

Fill in the following code with the gradient descent update rule

For reference, your 10th iteration should have $x = -1.554$ and $f(x) = -8.246$

In [ ]:
```
iter = 10
eta = 0.15
x = 8
cur_df = 0
all_x = []
all_x.append(x)


for i in range(iter):
    # YOUR GRADIENT DESCENT CODE GOES HERE
    x = x - eta*fgrad(x)

    ### --- The following code is for plotting to dicuss the convergence --- ###
    #add each new x to the list of all x's
    all_x.append(x)

    print('Iteration %d, x = %.3f, f(x) = %.3f' %(i+1, x, f(x)))
```

```
Iteration 1, x = 5.281, f(x) = 38.675
Iteration 2, x = 2.762, f(x) = 18.138
Iteration 3, x = 2.319, f(x) = 16.734
Iteration 4, x = 1.786, f(x) = 14.410
Iteration 5, x = 0.993, f(x) = 8.988
Iteration 6, x = -0.247, f(x) = -2.147
Iteration 7, x = -1.496, f(x) = -8.233
Iteration 8, x = -1.565, f(x) = -8.246
Iteration 9, x = -1.551, f(x) = -8.246
Iteration 10, x = -1.554, f(x) = -8.246
```

Briefly discuss whether your printed values of x and $f(x)$ appear to have converged to the minimum of the function.

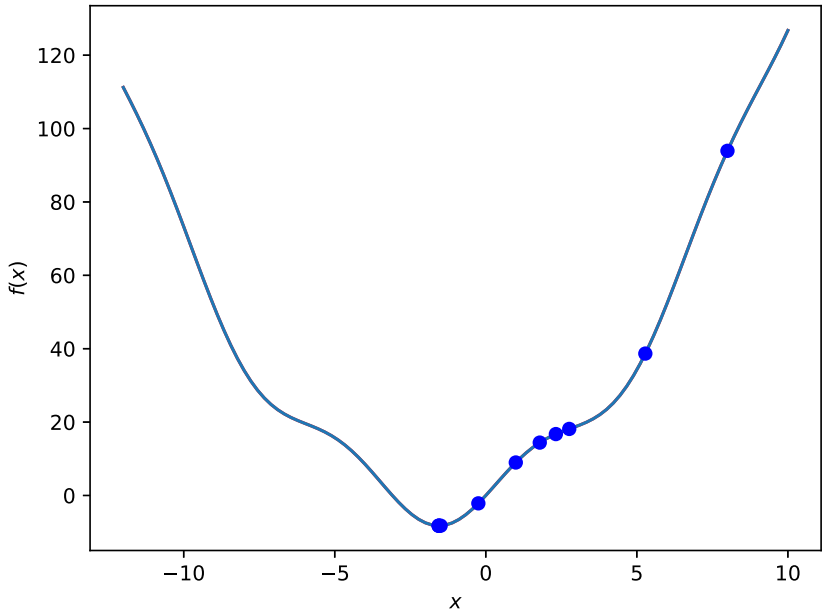
Feel free to refer to the provided plot of $f(x)$ above

Your response goes here

In [ ]:
```
def plot_gd(dg_x):
    xs = np.linspace(-12,10,100)
    ys = f(xs)
    # Plot function
    plt.plot(xs,ys,'r-')
    plt.xlabel('$x$')
    plt.ylabel('$f(x)$')
    plt.plot(xs, ys)

    # plot gradient descent path
    plt.plot(dg_x, f(dg_x), 'bo')

plot_gd(np.array(all_x))
```



From the plot of $f(x)$ and its gradient path above, we can see that the minimum of the function is at approximately $x = -1.5$. The printed

values of x and $f(x)$ appear to be converging to this value, so it appears that the gradient descent algorithm is working as expected. Also, this function is a convex function, so it guarantees that it will converge to global minimum.