Homework 3: This HW is based on the code for Lecture 7 (Gradient Descent).

Instructions:

Place the answer to your code only in the area specified. Also, make sure to run all your code, meaning, press >> to "Restart Kernel and Run All Cells". This should plot all outputs including your answers to homework questions. After this, go to file (top left) and select "Print". Save your file as a PDF and upload the PDF to Canvas.

Question:

Try to a build a simple optimizer to minimize:

```
f(w) = a[0] + a[1]*w + a[2]*w^2 + ... + a[d]*w^d
```

for the coefficients `a = [0,0.5,-2,0,1]`.

- Plot the function f(w) (2 points)
- Can you see where the minima is? (1 point)
- Write a function that outputs f(w) and its gradient (3 points).
- Run the optimizer on the function to see if it finds the minima (3 poimts).
- Print the funciton value and number of iterations (3 points).
- Instead of writing the function for a specific coefficient vector `a`, create a class that works for an arbitrary vector `a` (3 points).

You may wish to use the `poly.polyval(w,a)` method to evaluate the polynomial.

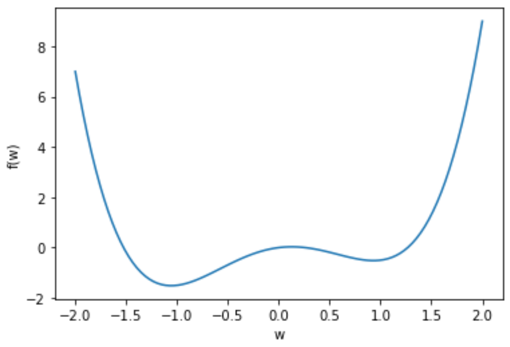
```
import numpy as np
import numpy.polynomial.polynomial as poly
import matplotlib.pyplot as plt

a = [0, 0.5, -2, 0, 1]

def f(w,a):
  return poly.polyval(w,a)
```

Plot the function f(w)

```
w_val = np.linspace(-2, 2, 100)
f_val = f(w_val, a)
plt.plot(w_val, f_val)
plt.xlabel('w')
plt.ylabel('f(w)')
plt.show()
```



Find the minimum of the function f(w)

```
min_val = np.min(f_val)
min_w = w_val[np.argmin(f_val)]
print(f'The minimum value of f(w) is {min_val:4f} at w = {min_w:.4f}.')

     The minimum value of f(w) is -1.513917 at w = -1.0707.
```

Write a function that outputs f(w) and its gradient

```
def f_grad(w, a):
    f_w = poly.polyval(w, a)
    df_da = poly.polyder(a)
    df_dw = poly.polyval(w,df_da)
    return f_w, df_dw
```

Optimizer on the function using the gradient descent algorithm:

```
def gradient_descent(f, f_grad, init_x, learning_rate=0.1, tol=1e-6, max_iter=1000):
    x = init_x
    for i in range(max_iter):
        fx, grad = f_grad(x, a)
        x_new = x - learning_rate * grad
        if abs(fx - f(x_new, a)) < tol:
            break
        x = x_new
    return x, fx, i

x0 = 0.0
x_opt, f_opt, num_iter = gradient_descent(f, f_grad, x0, learning_rate=0.1, tol=1e-6, max_iter=1000)
```

Print the function value and number of iterations:

```
print(f'Function value: {f_opt}')
print(f'Number of iterations: {num_iter}')
```

```
     Function value: -1.5147534273007441
     Number of iterations: 10
```

create a class that works for an arbitrary vector a

```
class PolynomialOptimizer:
    def __init__(self, a):
        self.a = a

    def f(self, w):
        return poly.polyval(w, self.a)

    def f_grad(self, w):
        df_da = poly.polyder(self.a)
        df_dw = poly.polyval(w, df_da)
        return self.f(w), df_dw

    def gradient_descent(self, init_w, learning_rate=0.1, tol=1e-6, max_iter=1000):
        w = init_w
        for i in range(max_iter):
            fx, grad = self.f_grad(w)
            w_new = w - learning_rate * grad
            if abs(fx - self.f(w_new)) < tol:
                break
            w = w_new
        return w, fx, i+1
```

Test the Optimizer

```
optimizer = PolynomialOptimizer(a)
initial_w = 0.0
min_w, min_val, num_iterations = optimizer.gradient_descent(initial_w)
print("Optimal value of w: ", min_w)
print("Minimum value of f(w): ", min_val)
print("Number of iterations: ", num_iterations)
```

```
     Optimal value of w:  -1.0572405904006204
     Minimum value of f(w):  -1.5147534273007441
     Number of iterations:  11
```

✓ 0s completed at 9:44 PM