Question 3 The overall purpose of Gradient Descent is to minimize an objective function in an iterative or repetitive manner; essentially, it is an optimization algorithm for finding where the lowest point of a function occurs. The gradient of a function is the slope of that function at a specific point. In this case, we can imagine our function to be in the shape of a hill with many bumps. We start at the beginning of the hill and want to move in the direction that will get us to a lower point. We do this by calculating the gradient or slope at a point close-by on the hill and moving in the direction of steepest descent (downhill). We also need to decide how big of a step we should take in this direction - this is done by choosing a step size, λ. This step size can be chosen in many ways: it can be fixed, it can be a sequence, or we can find an optimal value for it. Ideally, we want the step size to be larger at first, since we are further away from the lowest point, and then as we approach this lowest point we want it to become smaller so that we do not "jump over" it. We need to choose a step size in which we can get to the lowest point quickly, but also not overshoot it. If the step size is too small, it will take us a very long time to get to the lowest point on the hill. Once we have our gradient, the direction, and the step-size, we move towards a lower value in the function. We then make note of which point we are on the hill. This process is repeated until we reach the minimum point. We figure out that we have reached a minimum when we cannot find a slope that takes us any lower, we can only go higher. At the end, we return the point which yields the lowest part of the hill. An important thing to define is where we want to start on the hill as this can affect where the algorithm takes us. Suppose there are many low points on the hill, but only one where we are the lowest. If we start in a different area, the algorithm could take us to a point which is a minimum in a local area but is not the minimum point of the whole hill. When we define this algorithm, we can also set a maximum number of iterations or repetitions that we want to do before giving up. Sometimes, the absolute minimum of the function cannot be reached if the algorithm reaches its maximum iteration number. We set this number because we do not want our algorithm to go on for a very long time. All in all, this algorithm finds the gradient or slope at a point on the function and moves in a downwards direction until we reach a new point where we find the slope again and so on. We keep doing this until we reach the lowest point.

Although batch-sequential, batch-stochastic, and ordinary gradient descent are all optimization methods for minimizing an objective function, there are slight differences that can be contrasted. The ordinary gradient descent algorithm involves going through each unit in the population P and calculating the gradient at this unit for every iteration. This differs from batch-sequential and batch-stochastic gradient descent, as for these algorithms the gradient is calculated at a batch of units in the population for every iteration. This is possible because when we sum over the population, the gradient can be split into N (where N is the size of the population) smaller and independent gradient calculations which can be computed in any order. This is valuable when N is extremely large as the ordinary gradient descent algorithm would take a very long time to find a solution. With batch-sequential and batch-stochastic, the units are split into H non-overlapping batches of size M (such that $H * M = N$) and the gradient of each batch can be calculated in parallel. Thus, batch-sequential and batch-stochastic gradient descent have a shorter computation time. Since we are estimating the gradient using a subset (or batch) of the population, we usually use a fixed step size, λ instead of optimizing the step size using line search. Batch-sequential and batch-stochastic gradient descent differ in the way in which the subsets or batches are chosen. In the batch-sequential, we sequentially move through the H batches we created and update the estimated theta value after each batch rather than after all batches. If it takes us more than H iterations to converge to a solution, we iterate through the batches sequentially. In batch-stochastic gradient descent, we randomly select samples (or batches) from the population and compute the gradient at each iteration. It is similar to batch-sequential in that it also updates the theta value after each batch, but it differs in that the batches are randomly selected. When comparing each of these algorithms on a graph visually, the path of ordinary gradient descent is a smooth line towards the direction of the minimum of the function. The path of batch-sequential gradient descent is in a slightly more noisy line (which moves in different directions) until it reaches the minimum of the objective function. The path of stochastic gradient descent is a highly noisy line (going in many different directions) that moves in the general direction of the solution. The paths of batch-sequential and batch-stochastic are more noisy since we compute the gradient at a sample (batch) of the population rather than each and every point. Batch-stochastic is even more noisy since we randomly select the sample (batch) at which to calculate the gradient.