SML - Exercise 2

Maximilian Nothnagel, Stefanie Martin

1   Task 1: Optimization

1.1   1a) Numerical Optimization

```python
#Function for calculating f'(x)
def solveDerivative(x):
#Calculate Derivative f'(x)
result = 400 * math.pow(x, 3) + (2-400*(x+1)) * x-2
#print('f\'(', x,') =', result) #print result
return result

n = 20
#needs to be surprisingly small for decent results.
learningRate = 0.000001
#Randomly determine starting x
curX = random.randint(1, n-1)
for curIteration in range(1, 10000):
curResult = solveDerivative(curX)
print(curIteration, " : f\'(", curX,") =", curResult)
#negate to always move towards negative
diff = learningRate * -curResult
#apply to curX for next iteration
curX += diff
if curIteration % 2000 == 0:
input("Press Enter to continue...")

print("final X is", curX, "; with f\'(",curX,")= ",
                            solveDerivative(curX) )
```

Learning rate impacts the size of the "steps" we take in each iteration. Since we are approximating, we generally step over the lowest point at some point, and then start stepping back and forth over the lowest point.

A smaller learning rate will lead to smaller steps while moving towards the lowest point, but to a larger gap created by stepping back and forth over the lowest point once it has been found.

Larger learning rates lead to fast, unaccurate conclusions, smaller rarely come to a conclusion within 10k iterations.

Choose the learning rate just right to get a accurate conclusion, while still arriving at the conclusion in most cases. (conclusion meaning the lowest point)
Samples taken from one running of the script:

$$\#0001 f'(18,0) = 2196034,0$$
$$\#1000 f'(1,8132) = 345,853$$
$$\#2000 f'(1,6567) = 59,694$$
$$\#3000 f'(1,6262) = 13,164$$
$$\#4000 f'(1,6193) = 3,050$$
$$\#5000 f'(1,6177) = 0,715$$
$$\#6000 f'(1,6173) = 0,1679$$
$$\#7000 f'(1,61721) = 0,0395$$
$$\#8000 f'(1,617187) = 0,00923$$
$$\#9000 f'(1,61718799) = 0,0021822$$
$$\#10000 f'(1,6171806) = 0,0005131$$