HW #1:

Input:

def floatNum(s, c, f):

    return (-1)\*\*s\*2\*\*(c - 1023)\*(1 + f)

*# ================================================*

def SmallestFloating():

*# Define constants and generally low float number*

    i = 9

    num = floatNum(0, 1, i)

*# Iterate (and decrease the float value) until you can not decrease the float number by any more. Then return the value.*

    while i >= -1:

        temp = num

        num = floatNum(0, 1, i)

        i -= 1

        if num == 0:

            return temp

*# =================================================*

def LargestFloating():

*# Define constants and generally high float number*

    i = 30

    num = floatNum(0, 2046, 1 - 2\*\*(-52))

*# Iterate (and increase the float value) until you can not increase the float number by any more. Then return the value.*

    while i > 0:

        temp = num

        num = floatNum(0, 2046, 1 - 2\*\*(-i))

        i += 1

        if num == float('Inf'):

            return temp

*# =================================================*

def machineEpsilon():

*# Constants*

    epsilon = 1

    count = 0

*# Shift our epsilon value one bit over (by dividing by 2) at a time until it can no longer be reduced*

    while (1 + epsilon / 2 > 1):

        epsilon = epsilon / 2

        count = count + 1

*# Return epsilon and count size*

    return epsilon, count

*# ==================================================*

def main():

*# Use Functions to find largest and smallest floating point number along with macine epsilon*

    x = SmallestFloating()

    y = LargestFloating()

    z, count = machineEpsilon()

    print("Smallest Floating:", x,"\nLargest Floating:", y,"\nMachine Epsilon:", z)

*# ==================================================*

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Output:



Input:

import math

*# ====================================================*

def SqrtFinder(num, a, b):

*# Counter*

    n = 0

    N\_0 = 10

*# Equations for input*

    y1 = a\*\*2 - num

    y2 = b\*\*2 - num

*# Tolerance, 10 decimals*

    TOL = 1e-10

*# For loop via the bisection method*

    for i in range (0,N\_0):

        n = n + 1

        c = (a+b)/2

        y = c\*\*2 - num

        print("a\_" + str(n) + ":", a, "b\_" + str(n) + ":", b)

        if y == 0 or abs(y) <= TOL:

            break

        elif y\*y1 < 0:

            b = c

        else:

            a = c

    return c

*# ====================================================*

def error(p, ptrue):

*# Takes in true and measured values and returns absolute and relative error*

    abs\_error = abs(p - ptrue)

    rel\_error = abs\_error / ptrue

    return abs\_error, rel\_error

*# ====================================================*

def main():

*# Define interval and number to find*

    a1 = 1

    b1 = 2

    num = 2

    ans = SqrtFinder(num, a1, b1)

    print("\nThe root is approx. :", ans)

*# Use library to find true value*

    ptrue = math.sqrt(2)

*# Find Absolute and Relative error*

    abs\_error, rel\_error = error(ans, ptrue)

    print("Absolute Error:", abs\_error, "\nRelative Error:",rel\_error)

*# ====================================================*

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Output:

