**Part 1**

**Encapsulate the following Python code from Section 7.5 in a function named my\_sqrt that takes a as a parameter, chooses a starting value for x, and returns an estimate of the square root of a.**

**while True:  
     y = (x + a/x) / 2.0  
     if y == x:  
          break  
     x = y**

def my\_sqrt (a): #function of my\_sqrt with parameter 'a'

x = 1 #initial variable 'x' is 1

while True: #iteration for below expression being True

y = (x + a/x) / 2.0 #Newton's method for calculating square root

if y == x: #boolean, 'y' equals 'x' is True, if not is False

break #statement to exit the loop if 'y' equals 'x' is True

x = y #variable 'x' equals variable 'y'

return x #return statement, to return the value of variable 'x'

print() #print statement, placeholder

print (my\_sqrt(1)) #print statement, to print estimated square root of 'a'

**Part 2**

**Write a function named test\_sqrt that prints a table like the following using a while loop, where "diff" is the absolute value of the difference between my\_sqrt(a) and math.sqrt(a).**

a = 1 | my\_sqrt(a) = 1 | math.sqrt(a) = 1.0 | diff = 0.0  
a = 2 | my\_sqrt(a) = 1.41421356237 | math.sqrt(a) = 1.41421356237 | diff = 2.22044604925e-16  
a = 3 | my\_sqrt(a) = 1.73205080757 | math.sqrt(a) = 1.73205080757 | diff = 0.0  
a = 4 | my\_sqrt(a) = 2.0 | math.sqrt(a) = 2.0 | diff = 0.0  
a = 5 | my\_sqrt(a) = 2.2360679775 | math.sqrt(a) = 2.2360679775 | diff = 0.0  
a = 6 | my\_sqrt(a) = 2.44948974278 | math.sqrt(a) = 2.44948974278 | diff = 0.0  
a = 7 | my\_sqrt(a) = 2.64575131106 | math.sqrt(a) = 2.64575131106 | diff = 0.0  
a = 8 | my\_sqrt(a) = 2.82842712475 | math.sqrt(a) = 2.82842712475 | diff = 4.4408920985e-16  
a = 9 | my\_sqrt(a) = 3.0 | math.sqrt(a) = 3.0 | diff = 0.0

**Modify your program so that it outputs lines for a values from 1 to 25 instead of just 1 to 9.**

def test\_sqrt (): #function of test\_sqrt with no parameter

a = 1 #initial variable 'a' is 1

while a <=9: #iteration for the variable 'a' from 1 to 9

print ('a =', a,'| my\_sqrt (a) =', my\_sqrt (a),'| math.sqrt (a) =', math.sqrt (a),'| diff =', abs(math.sqrt (a) - my\_sqrt (a))) #print statement, to print required output

a = a + 1 #increments 'a' by 1; 'a' is equal to 'a' + 1

test\_sqrt() #calling function 'test\_sqrt'

**Modified:**

def test\_sqrt (): #function of test\_sqrt with no parameter

a = 1 #initial variable 'a' is 1

while a <=25: #iteration for the variable 'a' from 1 to 25

print ('a =', a,'| my\_sqrt (a) =', my\_sqrt (a),'| math.sqrt (a) =', math.sqrt (a),'| diff =', abs(math.sqrt (a) - my\_sqrt (a))) #print statement, to print required output

a = a + 1 #increments 'a' by 1; 'a' is equal to 'a' + 1

test\_sqrt() #calling function 'test\_sqrt'

1. **Does the submission include a my\_sqrt function that takes a single argument and includes the while loop from the instructions?**

def my\_sqrt (a): #function of my\_sqrt with parameter 'a'

x = 1 #initial variable 'x' is 1

while True: #iteration for below expression being True

y = (x + a/x) / 2.0 #Newton's method for calculating square root

if y == x: #boolean, 'y' equals 'x' is True, if not is False

break #statement to exit the loop if 'y' equals 'x' is True

x = y #variable 'x' equals variable 'y'

return x #return statement, to return the value of variable 'x'

print() #print statement, placeholder

print (my\_sqrt(1)) #print statement, to print estimated square root of 'a'

1. **Does the my\_sqrt function initialize x and return its final value?**

def my\_sqrt (a): #function of my\_sqrt with parameter 'a'

x = 1 #initial variable 'x' is 1

while True: #iteration for below expression being True

y = (x + a/x) / 2.0 #Newton's method for calculating square root

if y == x: #boolean, 'y' equals 'x' is True, if not is False

break #statement to exit the loop if 'y' equals 'x' is True

x = y #variable 'x' equals variable 'y'

return x #return statement, to return the value of variable 'x'

print() #print statement, placeholder

print (my\_sqrt(1)) #print statement, to print estimated square root of 'a'

**Output**

1

1. **Does the test\_sqrt function print a values from 1 to 25?**
2. **Does the test\_sqrt function print the values returned by my\_sqrt for each value of a?**
3. **Does the test\_sqrt function print correct values from math.sqrt for each value of a?**
4. **Does the test\_sqrt function print the absolute value of the differences between my\_sqrt and math.sqrt for each value of a?**

a = 1 | my\_sqrt (a) = 1 | math.sqrt (a) = 1.0 | diff = 0.0  
a = 2 | my\_sqrt (a) = 1.414213562373095 | math.sqrt (a) = 1.4142135623730951 | diff =

2.220446049250313e-16  
a = 3 | my\_sqrt (a) = 1.7320508075688772 | math.sqrt (a) = 1.7320508075688772 | diff =

0.0  
a = 4 | my\_sqrt (a) = 2.0 | math.sqrt (a) = 2.0 | diff = 0.0  
a = 5 | my\_sqrt (a) = 2.23606797749979 | math.sqrt (a) = 2.23606797749979 | diff = 0.0  
a = 6 | my\_sqrt (a) = 2.449489742783178 | math.sqrt (a) = 2.449489742783178 | diff =

0.0  
a = 7 | my\_sqrt (a) = 2.6457513110645907 | math.sqrt (a) = 2.6457513110645907 | diff =

0.0  
a = 8 | my\_sqrt (a) = 2.82842712474619 | math.sqrt (a) = 2.8284271247461903 | diff =

4.440892098500626e-16  
a = 9 | my\_sqrt (a) = 3.0 | math.sqrt (a) = 3.0 | diff = 0.0  
a = 10 | my\_sqrt (a) = 3.162277660168379 | math.sqrt (a) = 3.1622776601683795 | diff =

4.440892098500626e-16  
a = 11 | my\_sqrt (a) = 3.3166247903554 | math.sqrt (a) = 3.3166247903554 | diff = 0.0  
a = 12 | my\_sqrt (a) = 3.4641016151377544 | math.sqrt (a) = 3.4641016151377544 | diff =

0.0  
a = 13 | my\_sqrt (a) = 3.6055512754639896 | math.sqrt (a) = 3.605551275463989 | diff =

4.440892098500626e-16  
a = 14 | my\_sqrt (a) = 3.7416573867739413 | math.sqrt (a) = 3.7416573867739413 | diff =

0.0  
a = 15 | my\_sqrt (a) = 3.872983346207417 | math.sqrt (a) = 3.872983346207417 | diff =

0.0  
a = 16 | my\_sqrt (a) = 4.0 | math.sqrt (a) = 4.0 | diff = 0.0  
a = 17 | my\_sqrt (a) = 4.123105625617661 | math.sqrt (a) = 4.123105625617661 | diff =

0.0  
a = 18 | my\_sqrt (a) = 4.242640687119286 | math.sqrt (a) = 4.242640687119285 | diff =

8.881784197001252e-16  
a = 19 | my\_sqrt (a) = 4.358898943540673 | math.sqrt (a) = 4.358898943540674 | diff =

8.881784197001252e-16  
a = 20 | my\_sqrt (a) = 4.47213595499958 | math.sqrt (a) = 4.47213595499958 | diff =

0.0  
a = 21 | my\_sqrt (a) = 4.58257569495584 | math.sqrt (a) = 4.58257569495584 | diff =

0.0  
a = 22 | my\_sqrt (a) = 4.69041575982343 | math.sqrt (a) = 4.69041575982343 | diff =

0.0  
a = 23 | my\_sqrt (a) = 4.795831523312719 | math.sqrt (a) = 4.795831523312719 | diff =

0.0  
a = 24 | my\_sqrt (a) = 4.898979485566356 | math.sqrt (a) = 4.898979485566356 | diff =

0.0  
a = 25 | my\_sqrt (a) = 5.0 | math.sqrt (a) = 5.0 | diff = 0.0

1. **Does the my\_sqrt function compute values that are almost identical to math.sqrt ("diff" less than 1e-14)?**

Yes, most of them have no difference at all.

**COMPLETE SCRIPT INPUT & OUTPUT ON NEXT 3 PAGES**

**COMPLETE SCRIPT INPUT**

import math #imports math module

print ("Printing estimated Square Root of 'a'") #print statement, heading

def my\_sqrt (a): #function of my\_sqrt with parameter 'a'

x = 1 #initial variable 'x' is 1

while True: #iteration for below expression being True

y = (x + a/x) / 2.0 #Newton's method for calculating square root

if y == x: #boolean, 'y' equals 'x' is True, if not is False

break #statement to exit the loop if 'y' equals 'x' is True

x = y #variable 'x' equals variable 'y'

return x #return statement, to return the value of variable 'x'

print() #print statement, placeholder

print (my\_sqrt(1)) #print statement, to print estimated square root of 'a'

print ()

print ('Printing 1 through 9:') #print statement, heading

print () #print statement, placeholder

def test\_sqrt (): #function of test\_sqrt with no parameter

a = 1 #initial variable 'a' is 1

while a <=9: #iteration for the variable 'a' from 1 to 9

print ('a =', a,'| my\_sqrt (a) =', my\_sqrt (a),'| math.sqrt (a) =', math.sqrt (a),'| diff =', abs(math.sqrt (a) - my\_sqrt (a))) #print statement, to print required output

a = a + 1 #increments 'a' by 1; 'a' is equal to 'a' + 1

test\_sqrt() #calling function 'test\_sqrt'

print () #print statement, placeholder

print ('Printing 1 through 25:') #print statement, heading

print() #print statement, placeholder

def test\_sqrt (): #function of test\_sqrt with no parameter

a = 1 #initial variable 'a' is 1

while a <=25: #iteration for the variable 'a' from 1 to 25

print ('a =', a,'| my\_sqrt (a) =', my\_sqrt (a),'| math.sqrt (a) =', math.sqrt (a),'| diff =', abs(math.sqrt (a) - my\_sqrt (a))) #print statement, to print required output

a = a + 1 #increments 'a' by 1; 'a' is equal to 'a' + 1

test\_sqrt() #calling function 'test\_sqrt'

**COMPLETE OUTPUT**

Printing estimated Square Root of 'a'

1

Printing 1 through 9:

a = 1 | my\_sqrt (a) = 1 | math.sqrt (a) = 1.0 | diff = 0.0

a = 2 | my\_sqrt (a) = 1.414213562373095 | math.sqrt (a) = 1.4142135623730951 | diff = 2.220446049250313e-16

a = 3 | my\_sqrt (a) = 1.7320508075688772 | math.sqrt (a) = 1.7320508075688772 | diff = 0.0

a = 4 | my\_sqrt (a) = 2.0 | math.sqrt (a) = 2.0 | diff = 0.0

a = 5 | my\_sqrt (a) = 2.23606797749979 | math.sqrt (a) = 2.23606797749979 | diff = 0.0

a = 6 | my\_sqrt (a) = 2.449489742783178 | math.sqrt (a) = 2.449489742783178 | diff = 0.0

a = 7 | my\_sqrt (a) = 2.6457513110645907 | math.sqrt (a) = 2.6457513110645907 | diff = 0.0

a = 8 | my\_sqrt (a) = 2.82842712474619 | math.sqrt (a) = 2.8284271247461903 | diff = 4.440892098500626e-16

a = 9 | my\_sqrt (a) = 3.0 | math.sqrt (a) = 3.0 | diff = 0.0

Printing 1 through 25:

a = 1 | my\_sqrt (a) = 1 | math.sqrt (a) = 1.0 | diff = 0.0

a = 2 | my\_sqrt (a) = 1.414213562373095 | math.sqrt (a) = 1.4142135623730951 | diff = 2.220446049250313e-16

a = 3 | my\_sqrt (a) = 1.7320508075688772 | math.sqrt (a) = 1.7320508075688772 | diff = 0.0

a = 4 | my\_sqrt (a) = 2.0 | math.sqrt (a) = 2.0 | diff = 0.0

a = 5 | my\_sqrt (a) = 2.23606797749979 | math.sqrt (a) = 2.23606797749979 | diff = 0.0

a = 6 | my\_sqrt (a) = 2.449489742783178 | math.sqrt (a) = 2.449489742783178 | diff = 0.0

a = 7 | my\_sqrt (a) = 2.6457513110645907 | math.sqrt (a) = 2.6457513110645907 | diff = 0.0

a = 8 | my\_sqrt (a) = 2.82842712474619 | math.sqrt (a) = 2.8284271247461903 | diff = 4.440892098500626e-16

a = 9 | my\_sqrt (a) = 3.0 | math.sqrt (a) = 3.0 | diff = 0.0

a = 10 | my\_sqrt (a) = 3.162277660168379 | math.sqrt (a) = 3.1622776601683795 | diff = 4.440892098500626e-16

a = 11 | my\_sqrt (a) = 3.3166247903554 | math.sqrt (a) = 3.3166247903554 | diff = 0.0

a = 12 | my\_sqrt (a) = 3.4641016151377544 | math.sqrt (a) = 3.4641016151377544 | diff = 0.0

a = 13 | my\_sqrt (a) = 3.6055512754639896 | math.sqrt (a) = 3.605551275463989 | diff = 4.440892098500626e-16

a = 14 | my\_sqrt (a) = 3.7416573867739413 | math.sqrt (a) = 3.7416573867739413 | diff = 0.0

a = 15 | my\_sqrt (a) = 3.872983346207417 | math.sqrt (a) = 3.872983346207417 | diff = 0.0

a = 16 | my\_sqrt (a) = 4.0 | math.sqrt (a) = 4.0 | diff = 0.0

a = 17 | my\_sqrt (a) = 4.123105625617661 | math.sqrt (a) = 4.123105625617661 | diff = 0.0

a = 18 | my\_sqrt (a) = 4.242640687119286 | math.sqrt (a) = 4.242640687119285 | diff = 8.881784197001252e-16

a = 19 | my\_sqrt (a) = 4.358898943540673 | math.sqrt (a) = 4.358898943540674 | diff = 8.881784197001252e-16

a = 20 | my\_sqrt (a) = 4.47213595499958 | math.sqrt (a) = 4.47213595499958 | diff = 0.0

a = 21 | my\_sqrt (a) = 4.58257569495584 | math.sqrt (a) = 4.58257569495584 | diff = 0.0

a = 22 | my\_sqrt (a) = 4.69041575982343 | math.sqrt (a) = 4.69041575982343 | diff = 0.0

a = 23 | my\_sqrt (a) = 4.795831523312719 | math.sqrt (a) = 4.795831523312719 | diff = 0.0

a = 24 | my\_sqrt (a) = 4.898979485566356 | math.sqrt (a) = 4.898979485566356 | diff = 0.0

a = 25 | my\_sqrt (a) = 5.0 | math.sqrt (a) = 5.0 | diff = 0.0

**Reference**

Downey, A. (2015). *Think Python: How to Think Like a Computer Scientist* (2nd ed., Version 2.2.23).Green Tea Press.