科目名：プログラミング技法 II　　担当教員名：新田　直子
学籍番号：01Z18032　　名前：トゥチェク・トム

2019 年 04 月 24 日

平方根を求める

課題 1-1：a=0.25 の場合も答えが求められるように変更せよ

整数だけではなく、正数が入力された場合にも行けるように変更した。1 以上の正数は変更前にも行けたが、1 未満の数は問題だった。なぜなら、そうした数の平方根は入力された数の以上であるので、while ループの「x<a」という部分のせいで、ループは早すぎて終了した。

そのため、a は 1 未満と、a は 1 以上であるという場合に区別し、前者の条件の「x<a」を「x<1」に変更した。

課題 1-1：実行結果表

入力	出力	コメント
0	numGuesses = 1 0.0 is close to square root of 0.0	変更前と同じ
0.1	numGuesses = 3002 0.30009999999998327 is close to square root of 0.1	
0.25	numGuesses = 4900 0.48989999999996237 is close to square root of 0.25	求められた場合
0.9999	numGuesses = 9951 0.9949999999999067 is close to square root of 0.9999	
1	numGuesses = 9951 0.9949999999999067 is close to square root of 1.0	変更前と同じ 前件の結果は同じ
4.6789	numGuesses = 21609 2.1608000000001355 is close to square root of 4.6789	変更前と同じ
-1	numGuesses = 10002 Failed	期待された失敗

課題 1-2:下記のように 2 分法で探索するように変更し、調べた回数を変更前と比較せよ

下限と上限を使用し、前より速いアルゴリズムである。少しずつ上ることではなく、むしろ x は区間を毎度両分する。前回と同じように、入力は 1 未満か 1 以上か区別がある。限りは、a が 1 未満なら、a〜1 で始まり、さもないと 1〜a で始まる。

また、テスティングの時、ある入力でループが絶えないということが明らかになった。例えば、125678765432345678987654 を入力すると、ループが終了できない。おそらく、数が大きすぎて、浮動小数点数の計算の問題である。

それを治すために、mem というリストを使用し、ループの各回、変化があったかどうか確認する。なかったら、終了し、「失敗」を出力する。

課題 1-2：実行結果表

入力	1−1 出力	1−2 出力	コメント
0	numGuesses = 1 0.0	numGuesses = 1 0.0	変更前と同じ
0.1	numGuesses = 3002 0.30009999999998327	numGuesses = 3 0.32500000000000007	極めて速くなった
0.25	numGuesses = 4900 0.48989999999996237	numGuesses = 6 0.5078125	求められた場合 極めて速くなった
0.9999	numGuesses = 9951 0.9949999999999067	numGuesses = 2 0.99995	極めて速くなった
1	numGuesses = 9951 0.9949999999999067	numGuesses = 2 1.0	変更前と違って、正解
4.6789	numGuesses = 21609 2.1608000000001355	numGuesses = 9 2.164026953125	極めて速くなった
-1	numGuesses = 10002 Failed	numGuesses = 57 Failed	期待された失敗
100000000	（時間がかかりすぎる）	numGuesses = 47 10000.00000044575	大きい数も可能になった
125678765432 345678987654	（時間がかかりすぎる）	numGuesses = 94 Failed	大きすぎる数

課題 1-3:下記のようにニュートン法で探索するように変更し、調べた回数を変更前と比較せよ

ニュートン法のアルゴリズムを実装した。ループの各回で、f(x)と f'(x)という関数を使用し、新たな x を計算する。前と同じように、ある入力でループが終了しないので、 x の変化があったかどうか、今度も確認する。前回と違って、1 未満と 1 以上を区別する必要はない。また、 x は今回 0 ではなく、 1 で始まる。それが原因で、 0 などを入力すると、変更前と比べたら、結果は違う。

課題 1-3：実行結果表

入力	1−2 出力	1−3 出力	コメント
0	numGuesses = 1 0.0	numGuesses = 5 0.0625	前より劣る
0.1	numGuesses = 3 0.32500000000000007	numGuesses = 4 0.3196005081874647	
0.25	numGuesses = 6 0.5078125	numGuesses = 4 0.5001524390243902	求められた場合 前より的確
0.9999	numGuesses = 2 0.99995	numGuesses = 1 1.0	1 で始まるからすぐに終了
1	numGuesses = 2 1.0	numGuesses = 1 1.0	
4.6789	numGuesses = 9 2.164026953125	numGuesses = 4 2.1645227373433045	速くなった
-1	numGuesses = 57 Failed	ZeroDivisionError: float division by zero	負数は完全不可能になった（例外や無限ループ）
100000000	numGuesses = 47 10000.00000044575	numGuesses = 18 10000.000000082464	速くなった
125678765432345678987654	numGuesses = 94 Failed	numGuesses = 44 Failed	

課題 1-4: 課題 1-2 と 1-3 の考え方で平方根を求める関数をそれぞれ作成し、スライド 22 枚目のように main 関数を用いて、動作を確認するプログラムを作成せよ。また、他のプログラムからこのプログラムをモジュールとして用いて、各関数の動作を確認せよ。

main 関数を用いるプログラムにした。各関数は、成功ブーリアン、計算された平方根、調べた回数をリータンする。また、その関数を、他のプログラムからも使用できる。入力の必要はないテストプログラムの出力は下記である。

課題 1-4：実行結果

```
import kadai1_4

print(kadai1_4.k2(4,0.01))
print(kadai1_4.k3(4,0.01))

# further test cases
# さらなるテストケース
print("\nusing 1_2")
print(kadai1_4.k2(0,0.01))
print(kadai1_4.k2(1,0.01))
print(kadai1_4.k2(0.01,0.01))
print(kadai1_4.k2(0.25,0.01))
print(kadai1_4.k2(0.99,0.01))
print(kadai1_4.k2(1.01,0.01))
print(kadai1_4.k2(1000,0.01))
print(kadai1_4.k2(9999999999,0.01))
print(kadai1_4.k2(125678765432345678987654,0.01))

print("\nusing 1_3")
print(kadai1_4.k3(0,0.01))
print(kadai1_4.k3(1,0.01))
print(kadai1_4.k3(0.01,0.01))
print(kadai1_4.k3(0.25,0.01))
print(kadai1_4.k3(0.99,0.01))
print(kadai1_4.k3(1.01,0.01))
print(kadai1_4.k3(1000,0.01))
print(kadai1_4.k3(9999999999,0.01))
print(kadai1_4.k3(125678765432345678987654,0.01))
```

```
(True, 2.001953125, 10)
(True, 2.000609756097561, 4)


using 1_2
(True, 0.0, 1)
(True, 1.0, 2)
(True, 0.13375, 4)
(True, 0.5078125, 6)
(True, 0.995, 2)
(True, 1.005, 2)
(True, 31.622823238372803, 22)
(True, 99999.99999500546, 54)
(False, 354512010279.406, 94)

using 1_3
(True, 0.0625, 5)
(True, 1.0, 1)
(True, 0.10840434673026925, 5)
(True, 0.5001524390243902, 4)
(True, 0.995, 2)
(True, 1.005, 2)
(True, 31.622782450701045, 9)
(True, 99999.999995, 22)
(False, 354512010279.406, 44)
```