

# Some Example on Loops

# Square root with Newton's method

Newton's method of computing a square root of a number  $N$  is to make a first guess, then to improve that guess to get a better guess, then to improve the better guess to get an even better guess, and so on. The guess improvement process is given by a formula:

$$\text{newGuess} = N / (2 * \text{oldGuess}) + \text{oldGuess} / 2$$

The reason this works is in your calculus book, but all you need to do in this program is know how to use the formula.

# Square root with Newton's method

$$\text{newGuess} = N/(2*\text{oldGuess}) + \text{oldGuess}/2$$

For example, say that you want the square root of  $N = 3.00$ . The first guess can be nearly any value. 1.0 is good enough.

oldGuess	$N/(2*\text{oldGuess})$	oldGuess/2	newGuess
1.0	1.5	0.5	2.0
2.0	0.75	1.0	1.75
1.75	0.85714	0.875	1.73214
1.73214	0.86598	0.86607	1.73205

The fourth new guess is pretty close:

$$1.73205 * 1.73205 = 2.99999$$

You can't always count on getting good results in four steps, however. A program should keep improving the guess and stop when the results are nearly correct, however long that takes.

# Square root with Newton's method

## Ending Condition for the Loop

To determine when the answer is close to correct, compute its square. When `oldGuess*oldGuess` is very close to `N`, then

$$N/(oldGuess*oldGuess) == almost\ 1.00$$

or

$$N/(oldGuess*oldGuess) - 1.00 == almost\ 0.00$$

Unfortunately, we don't know if "almost 0.00" will be negative or positive, so we need to take the absolute value of it to make sure that it is positive. The **absolute value** of a number is the number with its negative sign (if any) removed. In math books the absolute value of  $x$  is shown as  $|x|$ . So the loop will end when:

$$| N/(oldGuess*oldGuess) - 1.00 | == almost\ 0.00$$

Now we need to decide what "almost 0.00" means. If the variables are all of type double precision, then they will have about 15 decimal places of precision. To be safe, assume that 14 places of precision can be reached. "Almost zero" to 14 places of precision means "less than 0.00000000000001"

```

sqrt.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\7-1\sqrt.py (3.4.4)
File Edit Format Run Options Window Help

def sqrt(N):
    smallValue = 1.0E-14
    guess = 1.00    # the same first guess works for any N
    while ( abs( N/(guess*guess) - 1.0 ) > smallValue ) :
        guess = N/(2*guess) + guess/2    # calculate a new guess
    return(guess)

N=float(input("Enter a number:"))
if ( N >= 0.0 ):
    guess=sqrt(N)
    print("The square root of " , N , " is " , guess )
else:
    print("Please enter a positive number")
print("Program End")

#>>>
#Output
# Enter a number:144
# The square root of  144.0  is  12.0
# Program End
#>>>
# Enter a number:-12
# Please enter a positive number
# Program End
#>>>
# Enter a number:0.000000000000000000003
# The square root of  3e-20  is  1.7320508075688775e-10
# Program End
#>>>

```

```

sqrt.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\19-1\sqr.py (3.4.4)
File Edit Format Run Options Window Help

def sqrt(N):
    smallValue = 1.0E-14
    guess = 1.00    # the same first guess works for any N
    while ( abs( N/(guess*guess) - 1.0 ) > smallValue ) :
        guess = N/(2*guess) + guess/2    # calculate a new guess
    return(guess)

N=float(input("Enter a number:"))
if ( N >= 0.0 ):
    guess=sqrt(N)
    print("The square root of " , N , " is " , guess )
else:
    print("Please enter a positive number")
print("Program End")

#>>>
N=25, smallValue = 1.0E-14

abs( N/(guess*guess) - 1.0)          guess = N/(2*guess) + guess/2
-----
25 / 1.0 * 1.0 -1 = 24.0              25/ 2* 1.0 + 1.0 /2 = 12.5                + 0.5                = 13
25 / 13.0 * 13.0 -1 = 0.8520710059171598    25/ 2*13.0 + 13.0 /2 = 0.9615384615384616 + 6.5                = 7.461538461538462
25 / 7.46 * 7.46 -1 = 0.5509618450419811    25/ 2* 7.46 + 7.46 /2 = 1.675257731958763 + 3.730769230769231    = 5.406026962727994
25 / 5.40 * 5.40 -1 = 0.14457173945943513    25/ 2* 5.40 + 5.40 /2 = 2.3122341205809005 + 2.703013481363997    = 5.015247601944898
25 / 5.01 * 5.01 -1 = 0.006071255060286607    25/ 2* 5.01 + 5.01 /2 = 2.4923993772815 + 2.507623800972449    = 5.000023178253949
25 / 5.00 * 5.00 -1 = 9.27123711225164e-06    25/ 2* 5.00 + 5.00 /2 = 2.4999884109267483 + 2.5000115891269745    = 5.000000000053722
25 / 5.00 * 5.00 -1 = 2.1488921753132217e-11    25/ 2* 5.00 + 5.00 /2 = 2.49999999973139 + 2.500000000026861    = 5.000000000000000

```



counter1.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\17-1\counter1.py (3.4.4)

File Edit Format Run Options Window Help

```
def Counter(n):
    A, B, C= 0, 0, 0
    while (A < n):
        print("result is :", A , B , C)
        C=C + 1
        if (C==n) :
            C=0
            B=B+1
        if (B==n) :
            B=0
            A=A+1
    return

n=int(input("Enter an integer:"))
Counter(n) ;
print("Program Terminate")

#>>>
#Output
#>>>
```

1 ≤ 3

A	B	C	
0	0	0	←
0	0	1	←
0	0	2	←
0	0	3	←
0	1	0	←
0	1	1	←
0	1	2	←
0	1	3	←
0	2	0	←
0	2	1	←
0	2	2	←
0	2	3	←
1	0	0	←
1	0	1	←
1	0	2	←
1	0	3	←
2	0	0	←
2	0	1	←
2	0	2	←
2	0	3	←
3	0	0	←

Enter an integer:3

result is : 0 0 0

result is : 0 0 1

result is : 0 0 2

result is : 0 1 0

result is : 0 1 1

result is : 0 1 2

result is : 0 2 0

result is : 0 2 1

result is : 0 2 2

result is : 1 0 0

result is : 1 0 1

result is : 1 0 2

result is : 1 1 0

result is : 1 1 1

result is : 1 1 2

result is : 1 2 0

result is : 1 2 1

result is : 1 2 2

result is : 2 0 0

result is : 2 0 1

result is : 2 0 2

result is : 2 1 0

result is : 2 1 1

result is : 2 1 2

result is : 2 2 0

result is : 2 2 1

result is : 2 2 2

Program Terminate

3

0 2 3



# Loop Version

$n \leq 3$

```
def Counter(n):
    A= 0
    while (A < n):
        B=0
        while (B<n) :
            C=0
            while (C<n) :
                print("result is :", A, B, C) ;
                C=C+1
            B=B+1
        A=A+1
    return

n=int(input("Enter an integer:"))
Counter(n) ;
print("Program Terminate")
```

A	B	C	
0	0	0	←
0	0	1	←
0	0	2	←
0	0	3	
0	1	0	←
0	1	1	
0	1	2	←
0	1	3	
0	2	0	←
0	2	1	
0	2	2	←
1	3	3	
1	0	0	←
2	2	2	←
3	3	3	←

```
Enter an integer:3
result is : 0 0 0
result is : 0 0 1
result is : 0 0 2
result is : 0 1 0
result is : 0 1 1
result is : 0 1 2
result is : 0 2 0
result is : 0 2 1
result is : 0 2 2
result is : 1 0 0
result is : 1 0 1
result is : 1 0 2
result is : 1 1 0
result is : 1 1 1
result is : 1 1 2
result is : 1 2 0
result is : 1 2 1
result is : 1 2 2
result is : 2 0 0
result is : 2 0 1
result is : 2 0 2
result is : 2 1 0
result is : 2 1 1
result is : 2 1 2
result is : 2 2 0
result is : 2 2 1
result is : 2 2 2
Program Terminate
```

# Power

```
power.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\17-1\power.py (3.4)
File Edit Format Run Options Window Help
def power(x,y):
    p=1
    while (y >=1) :
        p= p * x
        y = y -1
    return(p) ;

x=int(input("Enter first integer:"))
y=int(input("Enter second integer:"))
result=power(x,y)
print("result is :", result)

|
```

x	y	P
2	3	1
	2	2
	1	4
	0	8

8 ✓

# Power

```

power1.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\17-1\power1.py (3.4.4)
File Edit Format Run Options Window Help

def power(x,y):
    p=1
    while (y >=1) :
        if ((y%2)==1) : p=p*x
        x= x * x
        y = y //2
    return(p)

x=int(input("Enter first integer:"))
y=int(input("Enter second integer:"))
result=power(x,y)
print("result is :", result)

```

~~x y p~~

x	y	p
2	6	1
4	3	4
16	1	64
256	0	

$x^y = \begin{cases} x^{\frac{y}{2}} \cdot x^{\frac{y}{2}} & y \text{ is even} \\ x^{\frac{y-1}{2}} \cdot x^{\frac{y-1}{2}} \cdot x & y \text{ is odd} \end{cases}$

$2^6 = 2^3 \cdot 2^3 = 8 \cdot 8 = 64$   
 $2^3 = 2^1 \cdot 2^1 \cdot 2 = 8$

# Sumuatiom

Our goal is to write a program that computes the following sum:

$$\text{sum} = 1/1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6$$

(This may look like a pointless thing to do, but you will see such sums in calculus, where they are called harmonic series.)

$$\text{sum} = \sum_{i=1}^{\text{limit}} \frac{1}{i} = \sum_{i=1}^6 \frac{1}{i} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6}$$

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
def value(limit):  
    term=1  
    lastTerm=limit  
    sum=0.0  
    while(term<=lastTerm):  
        sum=sum + 1.0 / term  
        term=term+1  
    return sum
```

```
limit=int(input("Enter the limit of the series: "))  
result=value(limit)  
print("sum of ", limit, "terms: ", result)
```

limit = 10    terms = 10    limit = 10

$$\text{sum} = 0 + \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots$$

1.5

$\Sigma$

>>>

RESTART: C:\Documents and Settings\admin\Desktop\:

ple.py

Enter the limit of the series: 10

sum of 10 terms: 2.9289682539682538



# Sin

جاءت في ١٠

$$\sin(x) = \frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots$$

$$J = 1$$

$$\frac{x^J}{J!}$$

$$J = 3$$

$$J = 5$$

$$- \frac{x^J}{J!}$$

$$+ \frac{x^J}{J!}$$

$$\begin{array}{c} 3 \\ \frac{x^3}{3!} \end{array} \xrightarrow{x \cdot x \cdot x} \begin{array}{c} 5 \\ \frac{x^5}{5 \times 4 \times 3!} \end{array}$$



```
#>>>
#output
#>>>
Enter a degree:30
Enter the precision:3
Corresponding radian is: 0.523598775598298860
result is : 0.5000021325887924

Enter a degree:60
Enter the precision:10
Corresponding radian is: 1.0471975511965976
result is : 0.8660254037844385

Enter a degree:90
Enter the precision:3
Corresponding radian is: 1.5707963267948966
result is : 1.0045248555348174

Enter a degree:90
Enter the precision:50
Corresponding radian is: 1.5707963267948966
result is : 1.000000000000000002
|
```

# Prime

```

prim.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/prim.py (3.4.4)
File Edit Format Run Options Window Help

def prime(N):
    i = 2
    while(i < N):
        j = 2
        while(j <= (i/j)):
            if not(i%j): break
            j = j + 1
        if (j > i/j) : print (i, " is prime")
        i = i + 1
    return

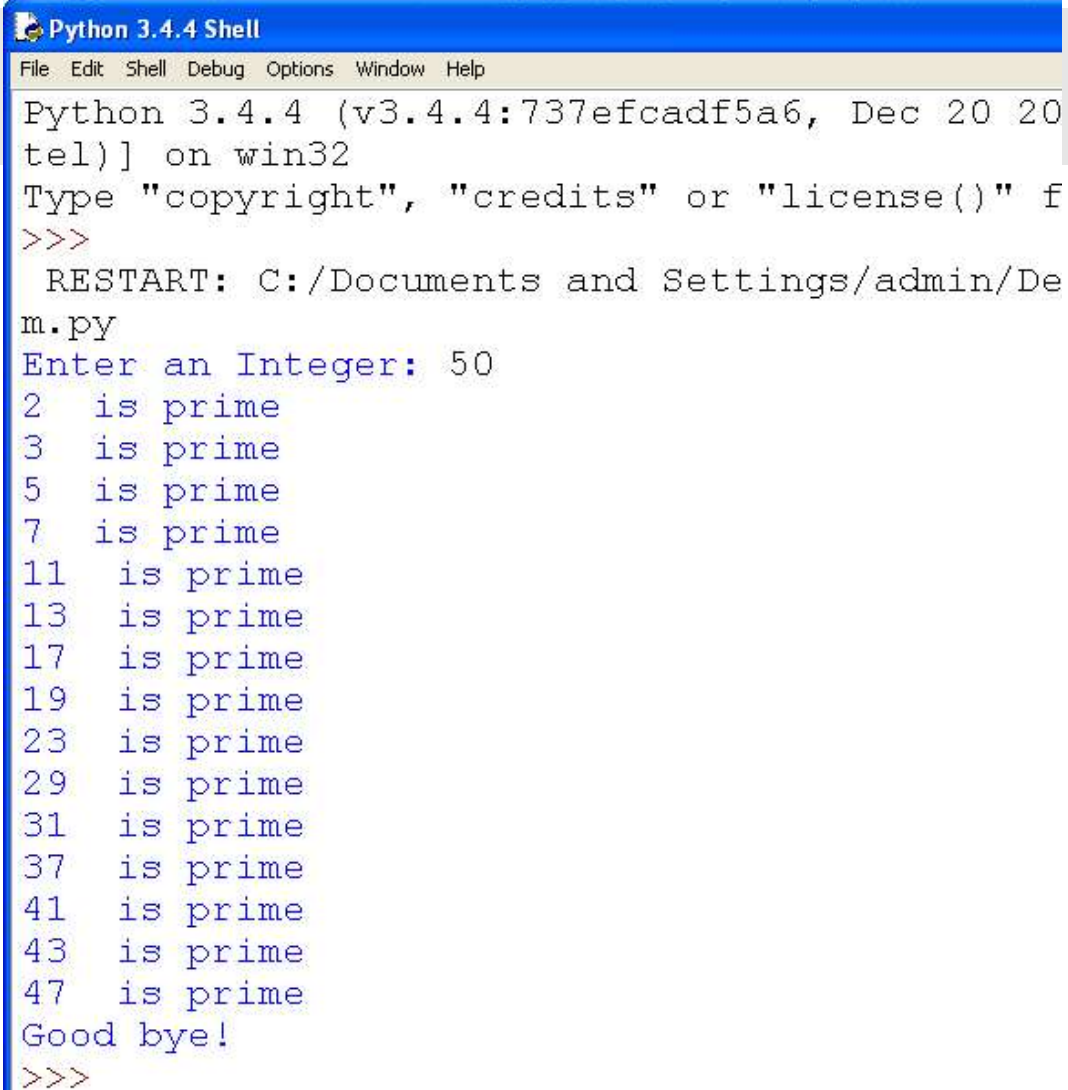
N=int(input("Enter an Integer: "))
prime(N)
print ("Good bye!")

```

Handwritten notes and table:

$i$        $j < 2, 3, 4, \dots, \sqrt{i}$   
 $j \leq \sqrt{i}$   
 $j \times j \leq i$   
 $j \leq \frac{i}{j}$

$i$	$j$	$i/j$
→ 11	2	5
	3	3
	4	2
12	2	6
13		



A screenshot of a Python 3.4.4 Shell window. The window has a blue title bar with the text "Python 3.4.4 Shell". Below the title bar is a menu bar with the options "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following output:

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 20
tel)] on win32
Type "copyright", "credits" or "license()" f
>>>
  RESTART: C:/Documents and Settings/admin/De
m.py
Enter an Integer: 50
2  is prime
3  is prime
5  is prime
7  is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
Good bye!
>>>
```

# Prime

```
*prim1.py - /home/nowzari/Desktop/python/python-my/python/examples/11-loop-ex/prim
File Edit Format Run Options Window Help

def prime(N):
    i = 2
    while(i < N):
        j = 2
        while(j <= (i/j)):
            if not(i%j): break
            j = j + 1
        else: print (i, " is prime")
        i = i + 1
    return

N=int(input("Enter an Integer: "))
prime(N)
print ("Good bye!")
```

i	j	i/j
→ 11	2	5
	3	3
	4	2
12	2	6
13	2	



**End**