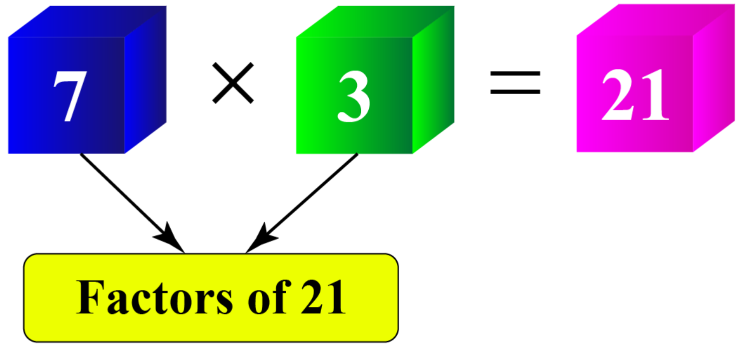
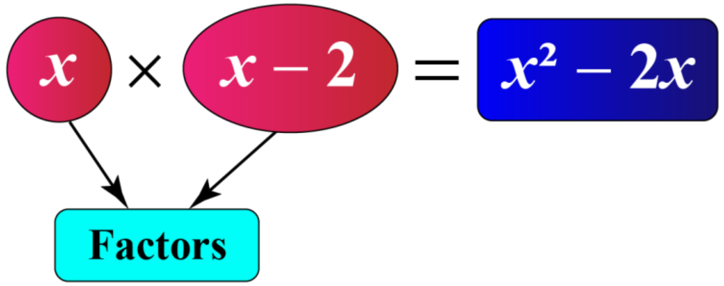
What Is Factoring?

- **The process of finding the factors is called factoring.**
- For example, 7 and 3 are the factors of 21.



Algebraic expressions also have factors. For example, x and x−2, are the factors of x^2−2x



Methods of Factoring

Method of Common Factors :

Consider a simple example: 3x+9

By factorizing each term we get, (3× x)+(3×3)

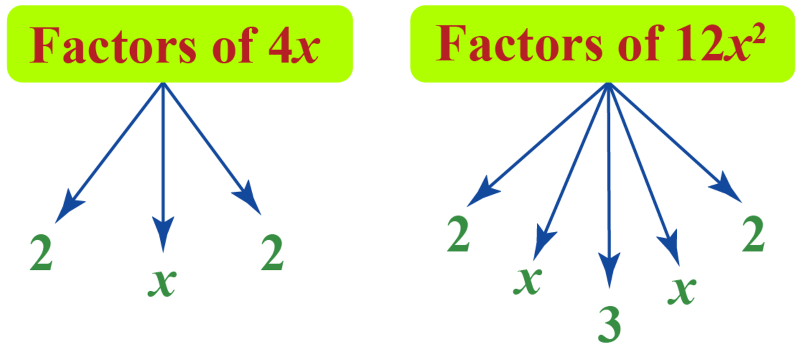
- By distributive law, 3x+9=(3×x)+(3×3)= 3(x+3)
-

$$\boxed{3x + 9} = \underbrace{\boxed{3(x + 3)}}$$

Did you notice that factor form has only one term?

Solving Quadratic Equations By Factoring

- 4x−12x2=0
-



We can observe that 4x4x is a common factor. Let’s take that common factor from the quadratic equation.

- 4x−12x2=0
- 4x(1−3x)=0

In other words, the quadratic equation can be factored as (4x)(1−3x)=0

George has a huge rectangular farm. He knows that the area of the farm is 528ft^2

He says "The length of the farm is one more than twice its breadth."
Can you use this information to write the factored form of 528?

Solving Quadratic Equations By Factoring

Let the breadth of the farm be x feet.

So, the length of the farm would be 2x+1 feet.

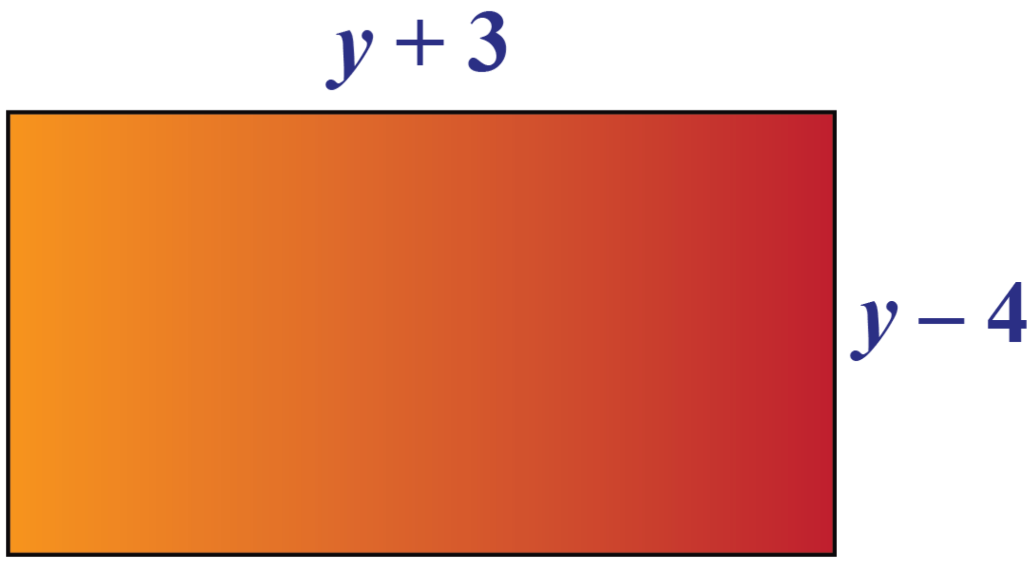
Area of Farm=Breadth×Length

528=   x ×(2x+1)

So, the factored form of 528 is x(2x+1)

Solving Quadratic Equations By Factoring

The dimensions of a rectangle are as shown in the figure.



Determine the area of this rectangle

Finding Square Root

Square root of a number is the value that returns the original number on multiplied by itself.

Finding square root by prime factorisation is an easy method. We need to factories the number under the root and pair them in two

For example, the square root of 9 is $\sqrt{9} = \sqrt{(3 \times 3)} = 3$.

Prime Factorisation

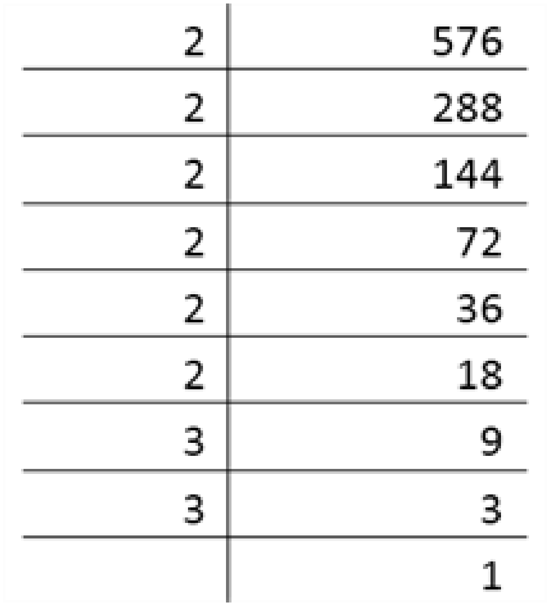
Let us consider the prime factors of a number and its square. For example 12 and 144,

$$12 = 2 \times 2 \times 3$$

$$144 = 2 \times 2 \times 3 \times 2 \times 2 \times 3$$

It can be observed that the prime factors in the prime factorisation of a square number, occur twice the number of times, it occurs in the number itself. For example, let us find the prime factors of 576.

Prime Factorisation



2	576
2	288
2	144
2	72
2	36
2	18
3	9
3	3
	1

For finding the square root, firstly we have to pair the common factors.

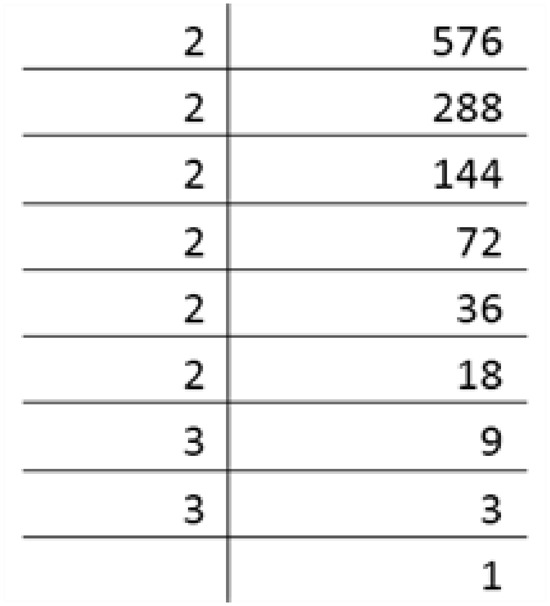
$$576 = \underline{2 \times 2} \times \underline{2 \times 2} \times \underline{2 \times 2} \times \underline{3 \times 3}$$

$$\Rightarrow 576 = 2^2 \times 2^2 \times 2^2 \times 3^2$$

The square root of 576 will be:

$$2 \times 2 \times 2 \times 3 = 24$$

Prime Factorisation



2	576
2	288
2	144
2	72
2	36
2	18
3	9
3	3
	1

For finding the square root, firstly we have to pair the common factors.

$$576 = \underline{2 \times 2} \times \underline{2 \times 2} \times \underline{2 \times 2} \times \underline{3 \times 3}$$

$$\Rightarrow \ 576 = 2^2 \times 2^2 \times 2^2 \times 3^2$$

The square root of 576 will be:

$$2 \times 2 \times 2 \times 3 = 24$$

Application of square root

Area of a square is the product of its sides. In a square, all the sides have the same length. Hence, if one of the sides is known then the area can be easily calculated. Area of square = Side × Side = Side^2.

For example, the area of a square is 225. Its length will be:

$225 = a^2$

$\Rightarrow 15 \times 15 = a^2$

$\Rightarrow a = 15$

Therefore, the side of the square is 15 units.

Application of square root

Area of a square is the product of its sides. In a square, all the sides have the same length. Hence, if one of the sides is known then the area can be easily calculated. Area of square = Side × Side = Side^2.

For example, the area of a square is 225. Its length will be:

$225 = a^2$

$\Rightarrow 15 \times 15 = a^2$

$\Rightarrow a = 15$

Therefore, the side of the square is 15 units.

Algorithm:

1. Create a variable (counter) *i* and take care of some base cases, i.e when the given number is 0 or 1.

2. Run a loop until *i*i <= n* , where n is the given number. Increment i by 1.

3. The floor of the square root of the number is *i – 1*

Python code

```
def floorSqrt(x):

        # Base cases
        if (x == 0 or x == 1):
                return x

        # Starting from 1, try all numbers until
        # i*i is greater than or equal to x.
        i = 1; result = 1
        while (result <= x):

                i += 1
                result = i * i

        return i - 1


x = 16
print(floorSqrt(x))
```

Smallest divisor of a number

1. The number *‘y’* is called the divisor of a number *‘x’* if *‘x/y’* is *0*.
2. If the number is *10*, then its *divisors* are *1,2,5 and 10*.
3. We will ignore *1* and consider *2* as the smallest divisor for the number.

Smallest divisor of a number

1. Ask the user to enter a *number*. Read it by using the *input()* function. It will read the user input data as a *string*. Convert it by wrapping it with the *int()* function.
2. Run one for loop from *2* to the *user input number*.
3. For each number, check if we can divide the user input number by this number or not. We are using an *if* condition here. If the current number can divide the user input number, this will be the *smallest divisor* for that number. Print that number.

Smallest divisor of a number

```
1.
num = int(input("Enter a number : "))

#2
for i in range(2, num+1):
    #3
    if num % i == 0:
        print ("The smallest divisor is”, i)
        break
```

Greatest common divisor (GCD)

- The **greatest common divisor** (**GCD**) of two or more integers, which are not all zero, is the largest positive integer that divides each of the integers.
- For two integers *x*, *y*, the greatest common divisor of *x* and *y* is denoted gcd(x,y)}.
- For example, the GCD of 8 and 12 is 4, that is, gcd(8,12)=4
- **Example**
- Thus the complete list of *divisors* of 54 is { 1,2,3,6,9,18,27,54}.
- Similarly, the divisors of 24 are {1,2,3,4,6,8,12,24}
- *common divisors* of 54 and 24, that is,
- {1,2,3,6}

Of these, the greatest is 6, so it is the *greatest common divisor*:

{gcd(54,24)=6.}

Greatest common divisor (GCD) Steps.

❖ **How to Find the Greatest Common Divisor?**

For a set of two positive integers (a, b) we use the below-given steps to find the greatest common divisor:

Step 1: Write the divisors of positive integer "a".

Step 2: Write the divisors of positive integer "b".

Step 3: Enlist the common divisors of "a" and "b".

Step 4: Now find the divisor which is the highest of both "a" and "b".

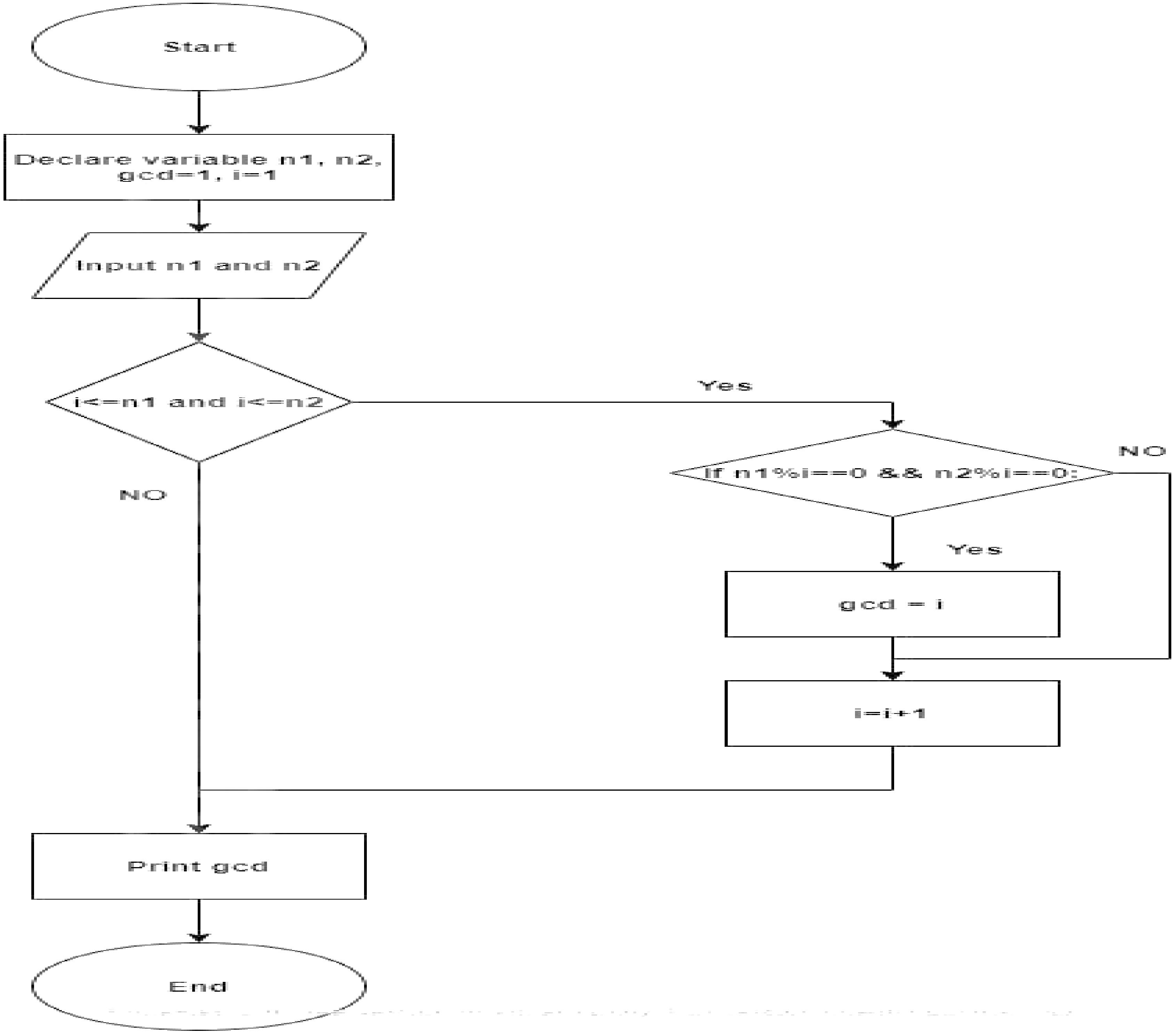
Greatest common divisor (GCD)

- **Using prime factorizations**
- Greatest common divisors can be computed by determining the prime factorizations of the two numbers and comparing factors.
- For example, to compute gcd(48, 180), we find the prime factorizations ,
- $48 = 2^4 \cdot 3^1$ and $180 = 2^2 \cdot 3^2 \cdot 5^1$; the GCD is then $2^{\min(4,2)} \cdot 3^{\min(1,2)} \cdot 5^{\min(0,1)}$
- $= 2^2 \cdot 3^1 \cdot 5^0 = 12$,

- The corresponding LCM is then $2^{\max(4,2)} \cdot 3^{\max(1,2)} \cdot 5^{\max(0,1)} = 2^4 \cdot 3^2 \cdot 5^1 = 720$.

Algorithm for Greatest Common Divisor

```
Step 1: Start
Step 2: Declare variable n1, n2, gcd=1, i=1
Step 3: Input n1 and n2
Step 4: Repeat until i<=n1 and i<=n2
        Step 4.1: If n1%i==0 && n2%i==0:
        Step 4.2: gcd = i
Step 5: Print gcd
Step 6: Stop
```


Start
Declare variable n1, n2, gcd=1, i=1
Input n1 and n2
i<=n1 and i<=n2
Yes
NO
If n1%i==0 && n2%i==0:
NO
Yes
gcd = i
i=i+1
Print gcd
End

Applications of Greatest common divisor

- A shopkeeper has 420 balls and 130 bats to pack in a day. She wants to pack them in such a way that each set has the same number in a box, and they take up the least area of the box. What is the number that can be placed in each set for this packing purpose?
- In the above problem, the greatest common divisor of 420 and 130 will be the required number.
- Other application like arranging students in rows and columns in equal number

The Euclidean Algorithm

The Euclidean Algorithm for finding GCD(A,B) is as follows:

If A = 0 then GCD(A,B)=B, since the GCD(0,B)=B, and we can stop.

If B = 0 then GCD(A,B)=A, since the GCD(A,0)=A, and we can stop.

Write A in quotient remainder form (A = B·Q + R)

Find GCD(B,R) using the Euclidean Algorithm since GCD(A,B) = GCD(B,R)

❖

Find the GCD of 270 and 192

A=270, B=192

A ≠0

B ≠0

Use long division to find that 270/192 = 1 with a remainder of 78. We can write this as:

270 = 192 * 1 +78

Find GCD(192,78), since GCD(270,192)=GCD(192,78)

A=192, B=78

A ≠0

B ≠0

Use long division to find that 192/78 = 2 with a remainder of 36. We can write this as:

192 = 78 * 2 + 36

Find GCD(78,36), since GCD(192,78)=GCD(78,36)

A=78, B=36

A ≠0

B ≠0

Use long division to find that 78/36 = 2 with a remainder of 6. We can write this as:

78 = 36 * 2 + 6

Find GCD(36,6), since GCD(78,36)=GCD(36,6)

A=36, B=6

A ≠0

B ≠0

Use long division to find that 36/6 = 6 with a remainder of 0. We can write this as:

36 = 6 * 6 + 0

Find GCD(6,0), since GCD(36,6)=GCD(6,0)

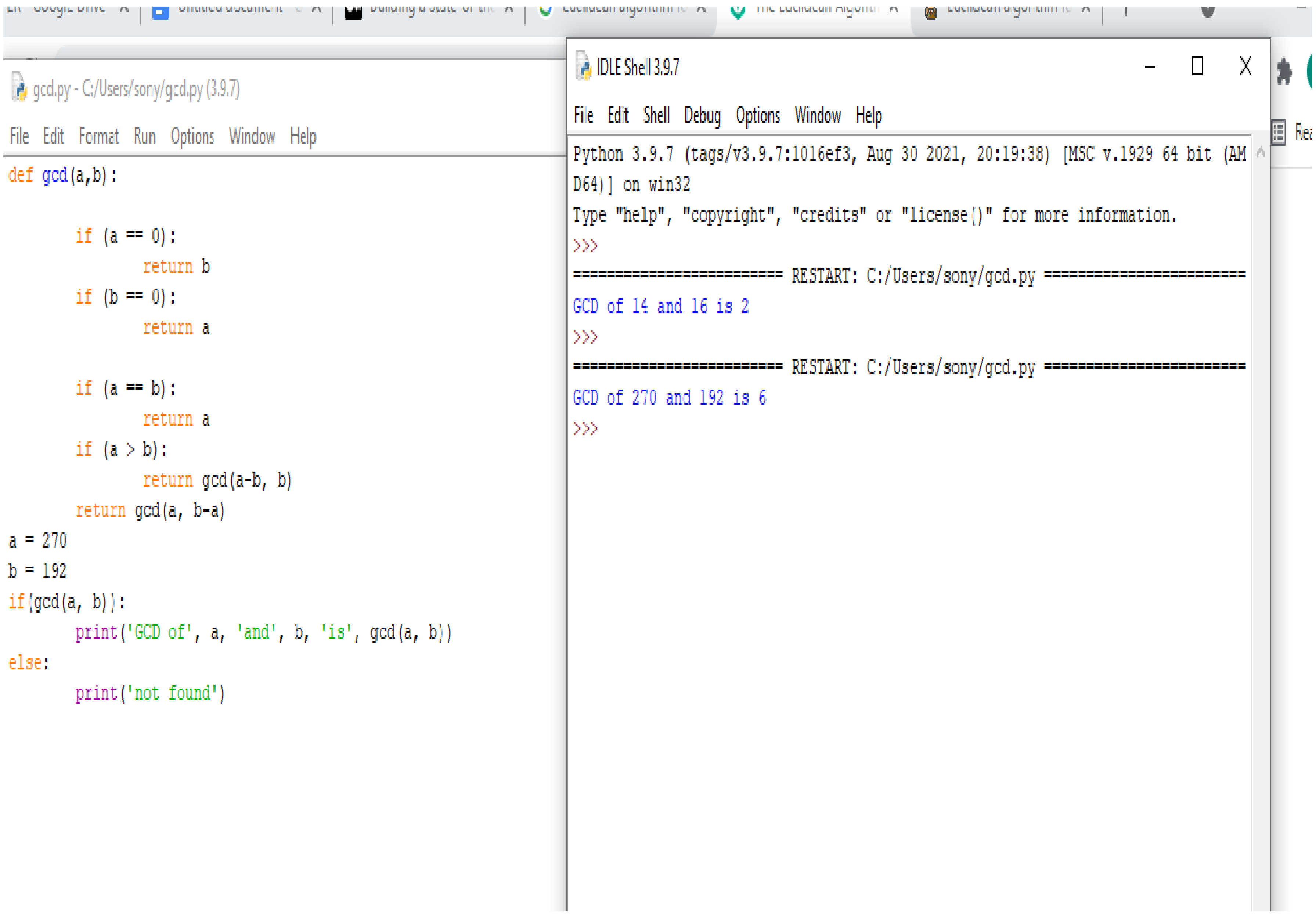
A=6, B=0

A $\neq$0

B =0, GCD(6,0)=6

Program to find GCD of two numbers

```
def gcd(a,b):

        if (a == 0):
                return b
        if (b == 0):
                return a

        if (a == b):
                return a
        if (a > b):
                return gcd(a-b, b)
        return gcd(a, b-a)
a = 14
b = 16
if(gcd(a, b)):
        print('GCD of', a, 'and', b, 'is', gcd(a, b))
else:
        print('not found')
```


gcd.py - C:/Users/sony/gcd.py (3.9.7)
File Edit Format Run Options Window Help
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help

```python
def gcd(a,b):

        if (a == 0):
                return b
        if (b == 0):
                return a

        if (a == b):
                return a
        if (a > b):
                return gcd(a-b, b)
        return gcd(a, b-a)
a = 270
b = 192
if(gcd(a, b)):
        print('GCD of', a, 'and', b, 'is', gcd(a, b))
else:
        print('not found')
```

```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======================== RESTART: C:/Users/sony/gcd.py ========================
GCD of 14 and 16 is 2
>>>
======================== RESTART: C:/Users/sony/gcd.py ========================
GCD of 270 and 192 is 6
>>>
```

Prime numbers:

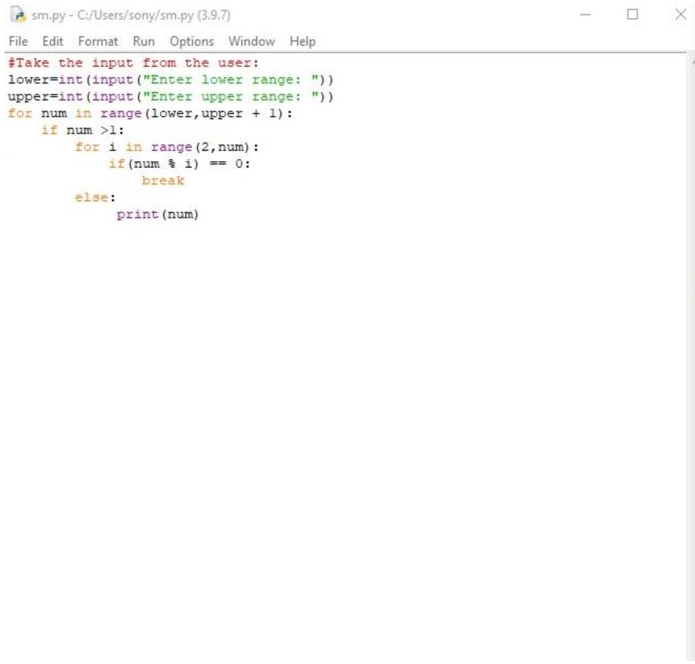
If the natural number is greater than 1 and having no positive divisors other than 1 and the number itself etc.

For example: 3, 7, 11 etc are prime numbers.

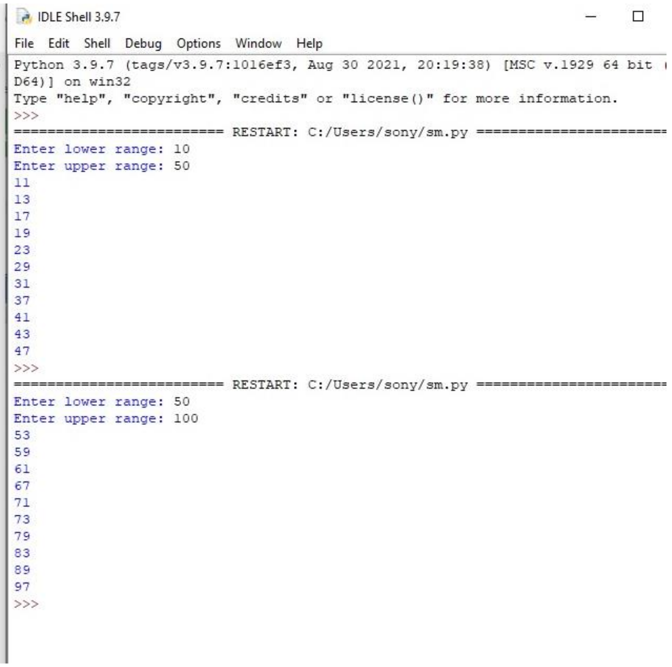
Program to Print all Prime Numbers between an Interval

```
#Take the input from the user:
lower = int(input("Enter lower range: "))
upper = int(input("Enter upper range: "))

for num in range(lower,upper + 1):
    if num > 1:
        for i in range(2,num):
            if (num % i) == 0:
            break
         else:
         print(num)
```

Program to Print all Prime Numbers between an Interval


sm.py - C:/Users/sony/sm.py (3.9.7)
File Edit Format Run Options Window Help

```
#Take the input from the user:
lower=int(input("Enter lower range: "))
upper=int(input("Enter upper range: "))
for num in range(lower,upper + 1):
    if num >1:
        for i in range(2,num):
            if(num % i) == 0:
                break
        else:
             print(num)
```



```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========================= RESTART: C:/Users/sony/sm.py ========================
Enter lower range: 10
Enter upper range: 50
11
13
17
19
23
29
31
37
41
43
47
>>>
========================= RESTART: C:/Users/sony/sm.py ========================
Enter lower range: 50
Enter upper range: 100
53
59
61
67
71
73
79
83
89
97
>>>
```

Prime Factorization

- Before going to learn the prime factorization formula, let us recall what is prime factorization. It is a way of expressing a number as a product of its prime factors.
- The prime factorization formula helps in finding the prime factorization of any number.
  The prime factorization formula of any number is given as:

$N = X^a \times Y^b \times Z^c$

where,

N = Any number

X, Y, and Z = Prime factors of number N

a, b, and c = exponents of prime factors X, Y, and Z respectively

How to find the prime factorization of a number?

- **1. Division Method**

A. Start dividing the number by the smallest prime number i.e., 2,

B. followed by 3, 5, and so on to find the smallest prime factor of the number.

C. Again, divide the quotient by the smallest prime number.

D. Repeat the process, until the quotient becomes 1 after repeated division.

E. Finally, represent the number as a product of all the prime factors.

How to find the prime factorization of a number?

- **1. Factor Tree Method**

A. Keep the number in the centered position as root.

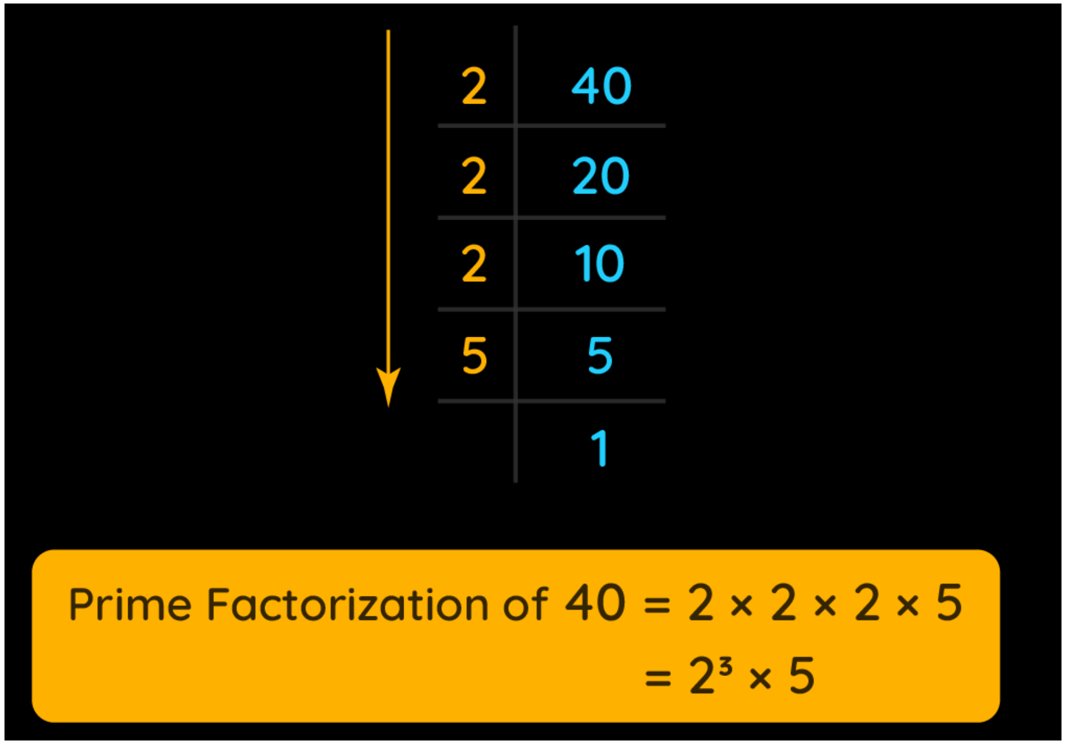
B. Divide the number with its smallest prime factor and represent the factor as a number in one branch.

C. Represent the quotient obtained in the other branch and repeat the above point for it, until you obtained 1 as the factor for the remaining number.

D. Each branch of the tree thus obtained will eventually end in a prime number.

How to find the prime factorization of a number?

- **Example 1:** Find the prime factorization of 40 using the prime factorization formula (the division method).



2	40
2	20
2	10
5	5
	1

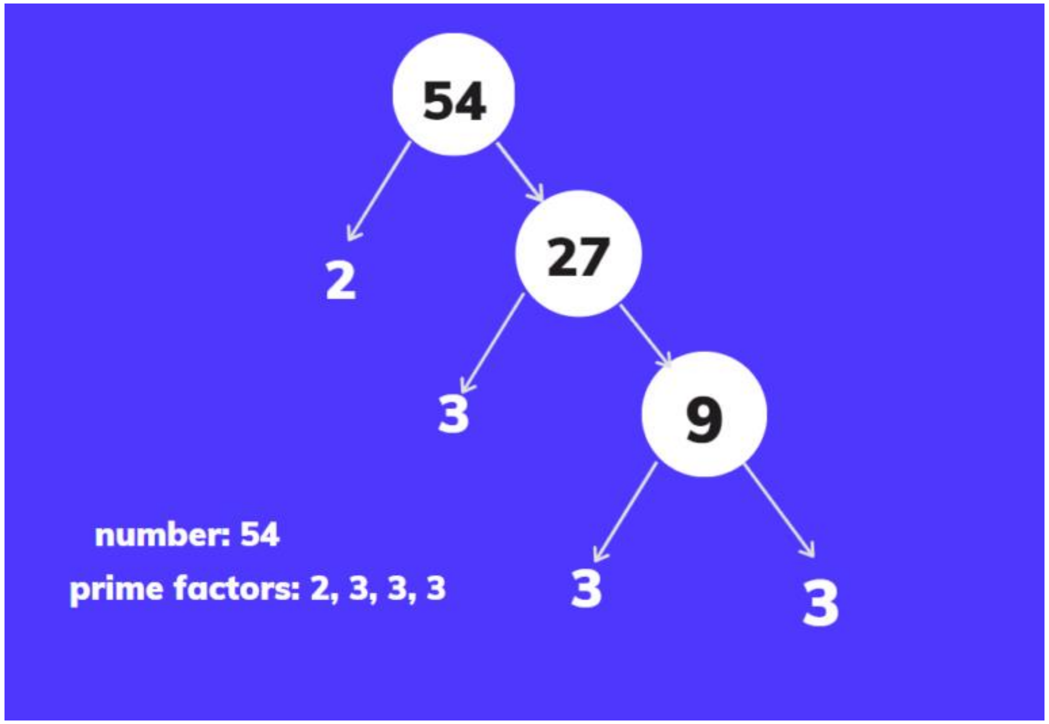
Prime Factorization of 40 = $2 \times 2 \times 2 \times 5$

$= 2^3 \times 5$

Prime factorization of 40 can thus be given as:
$40 = 2 \times 2 \times 2 \times 5$

How to find the prime factorization of a number?

- **Example 2:** Find the prime factorization of 54 using the prime factorization formula (the factor tree method).



Factors and Prime Factor Chart

Numbers	Factors	Prime Factor
1	1	
2	1, 2	2
3	1, 3	3
4	1, 2, 4	2 x 2
5	1, 5	5
6	1, 2, 3, 6	2 x 3
7	1, 7	7
8	1, 2, 4, 8	2 x 2 x 2
9	1, 3, 9	3 x 3
10	1, 2, 5, 10	2 x 5
11	1, 11	11
12	1, 2, 3, 4, 6, 12	2 x 2 x 3
13	1, 13	13
14	1, 2, 7, 14	2 x 7
15	1, 3, 5, 15	3 x 5
16	1, 2, 4, 8, 16	2 x 2 x 2 x 2
17	1, 17	17
18	1, 2, 3, 6, 9, 18	2 x 3 x 3
19	1, 19	19
20	1, 2, 4, 5, 10, 20	2 x 2 x 5
21	1, 3, 7, 21	3 x 7
22	1, 2, 11, 22	2 x 11
23	1, 23	23
24	1, 2, 3, 4, 6, 8, 12, 24	2 x 2 x 2 x 3
25	1, 5, 25	5 x 5
26	1, 2, 13, 26	2 x 13
27	1, 3, 9, 27	3 x 3 x 3
28	1, 2, 4, 7, 14, 28	2 x 2 x 7
29	1, 29	29
30	1, 2, 3, 5, 6, 10, 15, 30	2 x 3 x 5
31	1, 31	31
32	1, 2, 4, 8, 16, 32	2 x 2 x 2 x 2 x 2
33	1, 3, 11, 33	3 x 11
34	1, 2, 17, 34	2 x 17
35	1, 5, 7, 35	5 x 7

Numbers	Factors	Prime Factor
36	1, 2, 3, 4, 6, 9, 12, 18, 36	2 x 2 x 3 x 3
37	1, 37	37
38	1, 2, 19, 38	2 x 19
39	1, 3, 13, 39	3 x 13
40	1, 2, 4, 5, 8, 10, 20, 40	2 x 2 x 2 x 5
41	1, 41	41
42	1, 2, 3, 6, 7, 14, 21, 42	2 x 3 x 7
43	1, 43	43
44	1, 2, 4, 11, 22, 44	2 x 2 x 11
45	1, 3, 5, 9, 15, 45	3 x 3 x 5
46	1, 2, 23, 46	2 x 23
47	1, 47	47
48	1, 2, 3, 4, 6, 8, 12, 16, 24, 48	2 x 2 x 2 x 2 x 3
49	1, 7, 49	7 x 7
50	1, 2, 5, 10, 25, 50	2 x 5 x 5
51	1, 3, 17, 51	3 x 17
52	1, 2, 4, 13, 26, 52	2 x 2 x 13
53	1, 53	53
54	1, 2, 3, 6, 9, 18, 27, 54	2 x 3 x 3 x 3
55	1, 5, 11, 55	5 x 11
56	1, 2, 4, 7, 8, 14, 28, 56	2 x 2 x 2 x 7
57	1, 3, 19, 57	3 x 19
58	1, 2, 29, 58	2 x 29
59	1, 59	59
60	1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60	2 x 2 x 3 x 5
61	1, 61	61
62	1, 2, 31, 62	2 x 31
63	1, 3, 7, 9, 21, 63	3 x 3 x 7
64	1, 2, 4, 8, 16, 32, 64	2 x 2 x 2 x 2 x 2 x 2
65	1, 5, 13, 65	5 x 13
66	1, 2, 3, 6, 11, 22, 33, 66	2 x 3 x 11
67	1, 67	67
68	1, 2, 4, 17, 34, 68	2 x 2 x 17
69	1, 3, 23, 69	3 x 23
70	1, 2, 5, 7, 10, 14, 35, 70	2 x 5 x 7
71	1, 71	71

Numbers	Factors	Prime Factor
72	1, 2, 3, 4, 6, 8, 9, 12, 18, 24, 36, 72	2 x 2 x 2 x 3 x 3
73	1, 73	73
74	1, 2, 37, 74	2 x 37
75	1, 3, 5, 15, 25, 75	3 x 5 x 5
76	1, 2, 4, 19, 38, 76	2 x 2 x 19
77	1, 7, 11, 77	7 x 11
78	1, 2, 3, 6, 13, 26, 39, 78	2 x 3 x 13
79	1, 79	79
80	1, 2, 4, 5, 8, 10, 16, 20, 40, 80	2 x 2 x 2 x 2 x 5
81	1, 3, 9, 27, 81	3 x 3 x 3 x 3
82	1, 2, 41, 82	2 x 41
83	1, 83	83
84	1, 2, 3, 4, 6, 7, 12, 14, 21, 28, 42, 84	2 x 2 x 3 x 7
85	1, 5, 17, 85	5 x 17
86	1, 2, 43, 86	2 x 43
87	1, 3, 29, 87	3 x 29
88	1, 2, 4, 8, 11, 22, 44, 88	2 x 2 x 2 x 11
89	1, 89	89
90	1, 2, 3, 5, 6, 9, 10, 15, 18, 30, 45, 90	2 x 3 x 3 x 5
91	1, 7, 13, 91	7 x 13
92	1, 2, 4, 23, 46, 92	2 x 2 x 23
93	1, 3, 31, 93	3 x 31
94	1, 2, 47, 94	2 x 47
95	1, 5, 19, 95	5 x 19
96	1, 2, 3, 4, 6, 8, 12, 16, 24, 32, 48, 96	2 x 2 x 2 x 2 x 2 x 3
97	1, 97	97
98	1, 2, 7, 14, 49, 98	2 x 7 x 7
99	1, 3, 9, 11, 33, 99	3 x 3 x 11
100	1, 2, 4, 5, 10, 20, 25, 50, 100	2 x 2 x 5 x 5

How to find the prime factorization of a number?

Steps to find the prime factors of a number

1.) Let the number be denoted by num.

2.) while num is divisible by 2, we will print 2 and divide the num by 2.

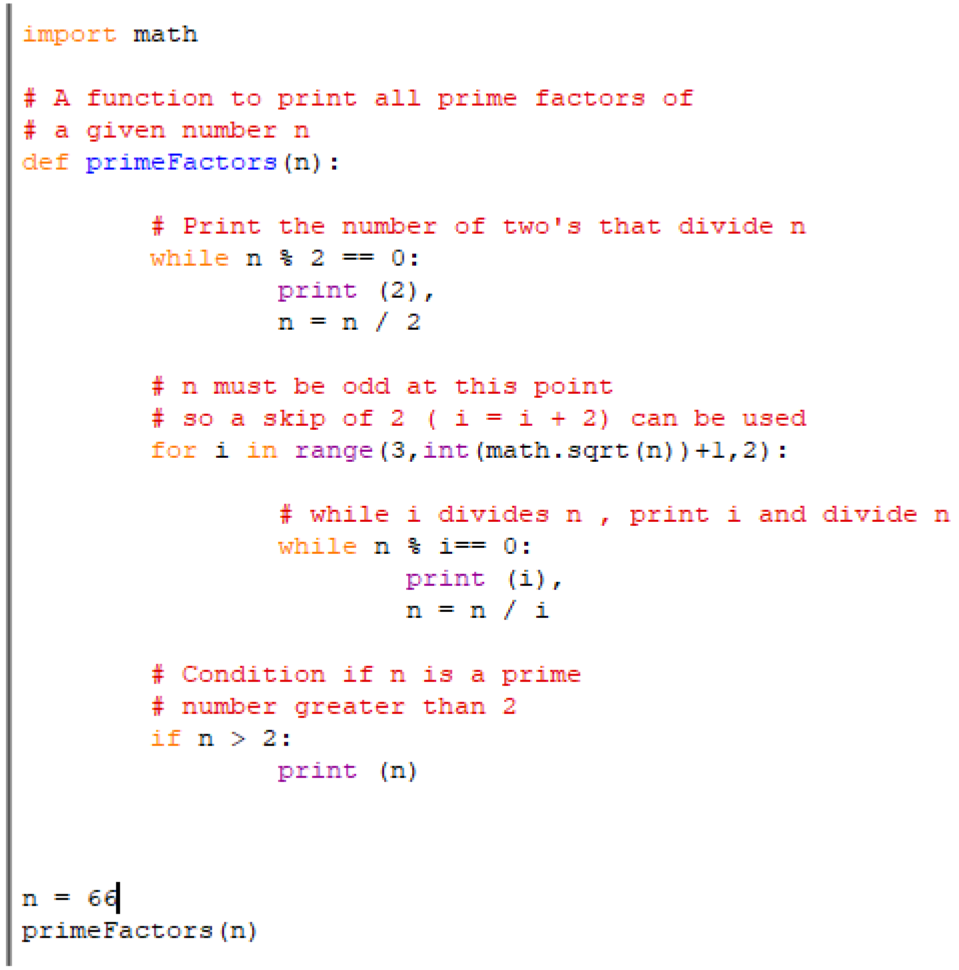
3). After step 2, num must be always odd.

4.) Start a loop from I = 3 to the square root of n. If i divide num, print i, and divide num by i. After i fail to divide num, increment the i value by 2 and continue.

5.) If num is a prime number and is greater than 2, then the num cannot become 1.

6.) So, print num if it is greater than 2.

How to find the prime factorization of a number?



```
import math

# A function to print all prime factors of
# a given number n
def primeFactors(n):

        # Print the number of two's that divide n
        while n % 2 == 0:
                print (2),
                n = n / 2

        # n must be odd at this point
        # so a skip of 2 ( i = i + 2) can be used
        for i in range(3,int(math.sqrt(n))+1,2):

                # while i divides n , print i and divide n
                while n % i== 0:
                        print (i),
                        n = n / i

        # Condition if n is a prime
        # number greater than 2
        if n > 2:
                print (n)


n = 66
primeFactors(n)
```

Pseudo random number generator

- **Pseudo Random Number Generator(PRNG)** refers to an algorithm that uses mathematical formulas to produce sequences of random numbers.
- PRNGs generate a sequence of numbers.
- A PRNG starts from an arbitrary starting state using a **seed state**.
- Many numbers are generated in a short time and can also be reproduced later, if the starting point in the sequence is known.

Applications of PRNG

- PRNGs are suitable for applications where many random numbers are required and where it is useful that the same sequence can be replayed easily.
- Popular examples of such applications are **simulation and modeling applications**.

Power function for large numbers

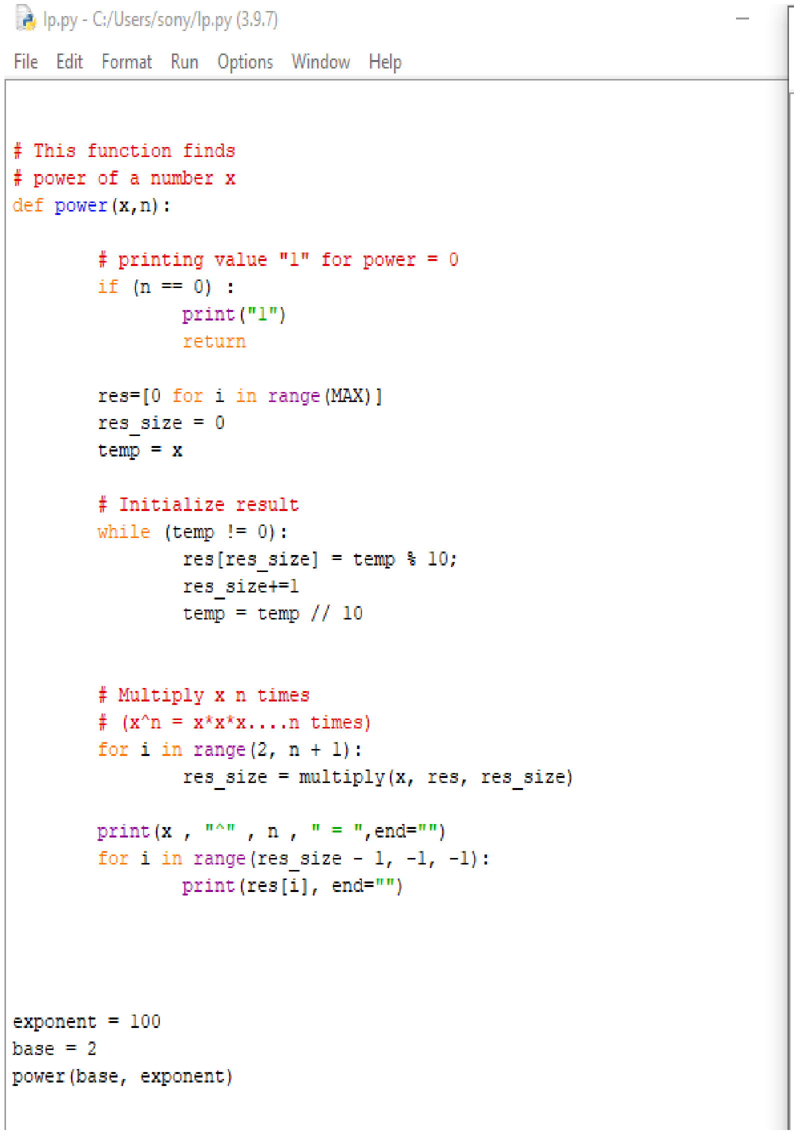
- Input : x = 2,
- n = 100
- Output : 1267650600228229401496703205376
- 2^100 has 31 digits
- if we use long long int which can store maximum 18 digits.

Lgorithm for finding power of a number.

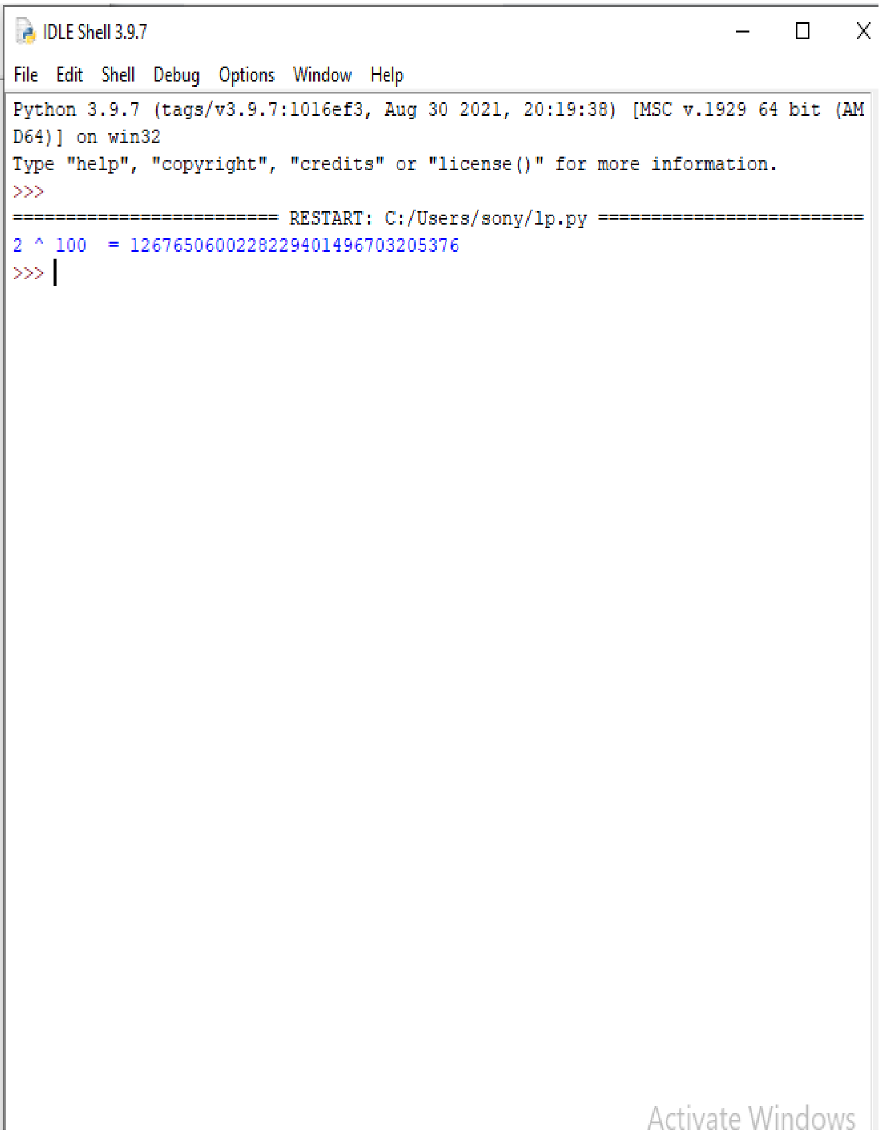
- Create an array res[] of MAX size and store x in res[] array and initialize res_size as the number of digits in x.
- Do following for all numbers from i=2 to n , Multiply x with res[] and update res[] and res_size to store the multiplication result.
- **Multiply(res[], x)**
- Initialize carry as 0.
- Do following for i=0 to res_size-1
    - Find prod = res[i]*x+carry
    - Store last digit of prod in res[i] and remaining digits in carry

  Store all digits of carry in res[] and increase res_size by number of digits.

finding power of a number.


lp.py - C:/Users/sony/lp.py (3.9.7)
File Edit Format Run Options Window Help

```python
# This function finds
# power of a number x
def power(x,n):

        # printing value "1" for power = 0
        if (n == 0) :
                print("1")
                return

        res=[0 for i in range(MAX)]
        res_size = 0
        temp = x

        # Initialize result
        while (temp != 0):
                res[res_size] = temp % 10;
                res_size+=1
                temp = temp // 10


        # Multiply x n times
        # (x^n = x*x*x....n times)
        for i in range(2, n + 1):
                res_size = multiply(x, res, res_size)

        print(x , "^" , n , " = ",end="")
        for i in range(res_size - 1, -1, -1):
                print(res[i], end="")




exponent = 100
base = 2
power(base, exponent)
```

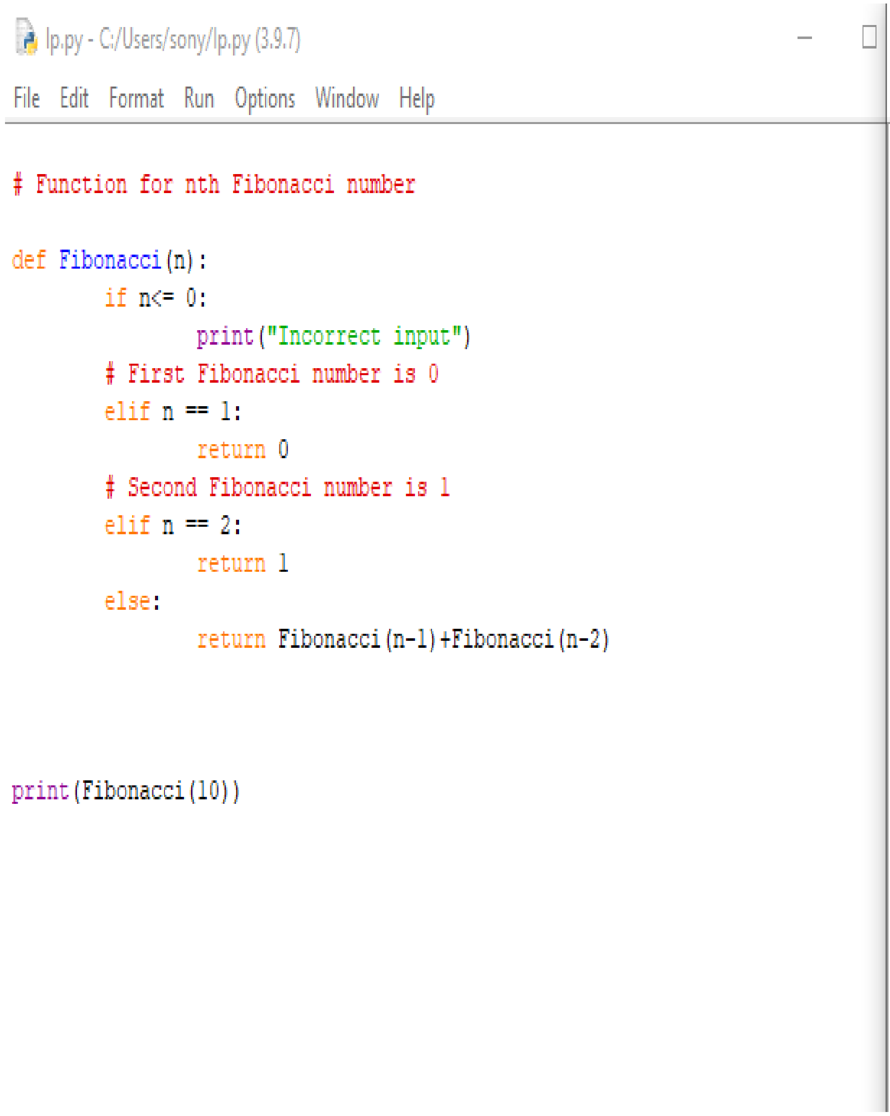


```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========================= RESTART: C:/Users/sony/lp.py =========================
2 ^ 100  = 1267650600228229401496703205376
>>> 
```

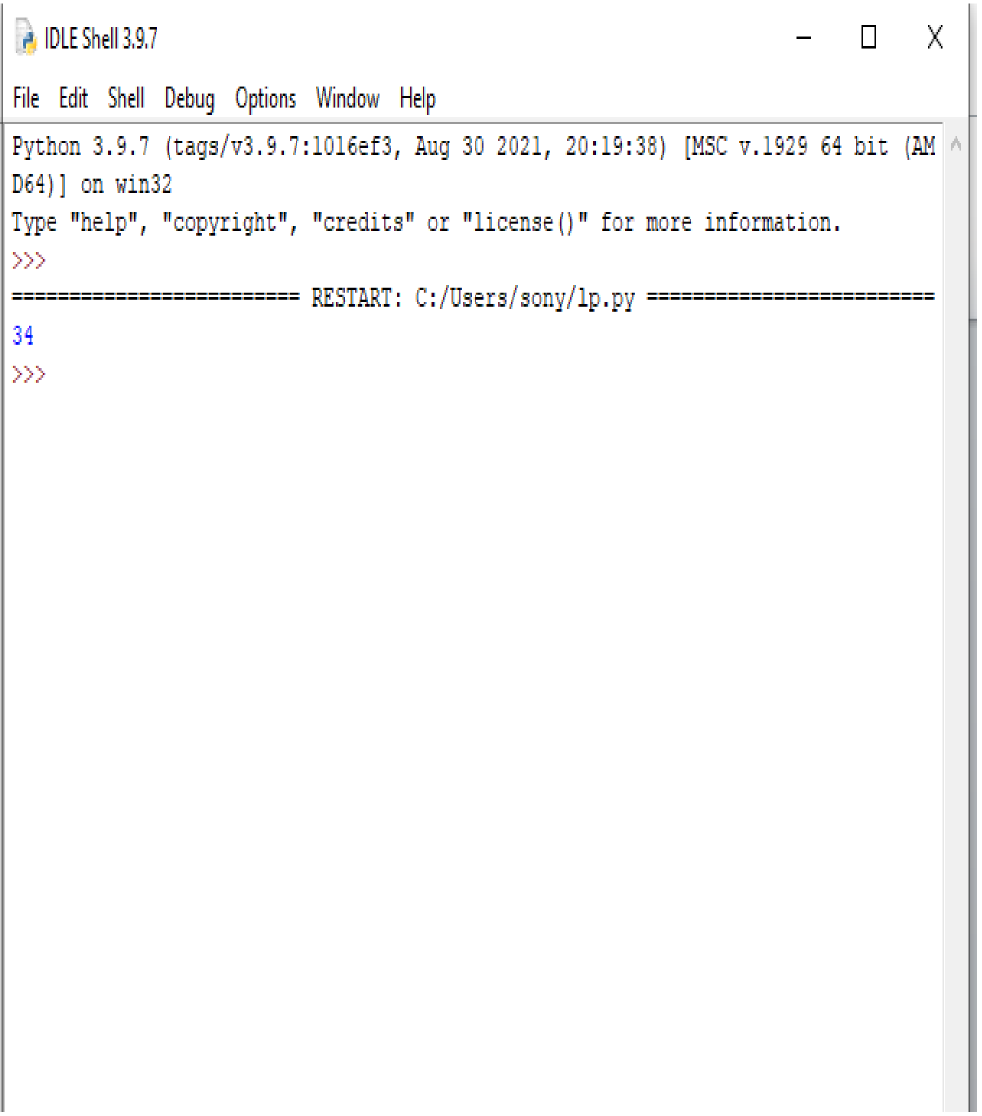
n-th Fibonacci number

- Recurrence relation .
- $F_n = F_{n-1} + F_{n-2}$
- $F_0 = 0$ and $F_1 = 1$.

n-th Fibonacci number


lp.py - C:/Users/sony/lp.py (3.9.7)
File Edit Format Run Options Window Help

```python
# Function for nth Fibonacci number

def Fibonacci(n):
        if n<= 0:
                print("Incorrect input")
        # First Fibonacci number is 0
        elif n == 1:
                return 0
        # Second Fibonacci number is 1
        elif n == 2:
                return 1
        else:
                return Fibonacci(n-1)+Fibonacci(n-2)



print(Fibonacci(10))
```



```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======================== RESTART: C:/Users/sony/lp.py ========================
34
>>>
```

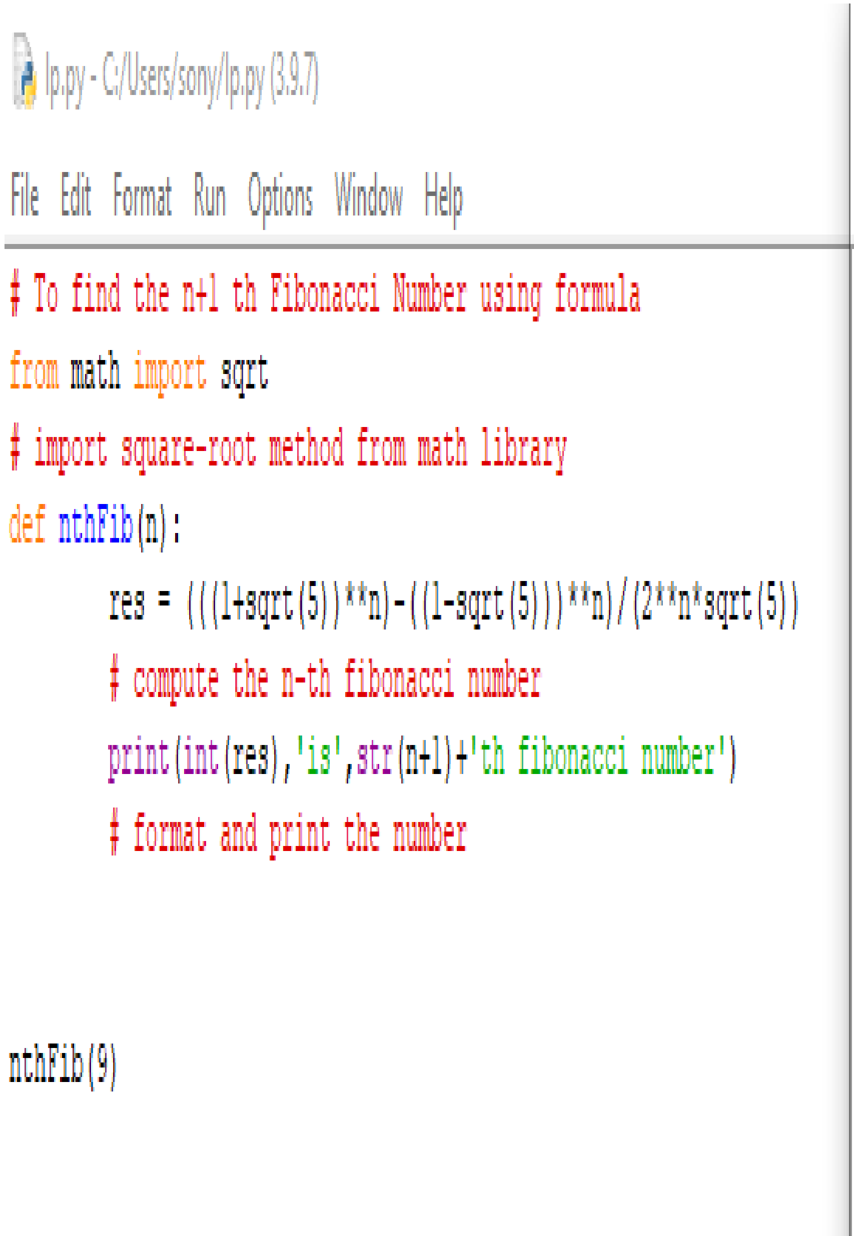
n-th Fibonacci number

$Fibonacci\ number\ F_n$

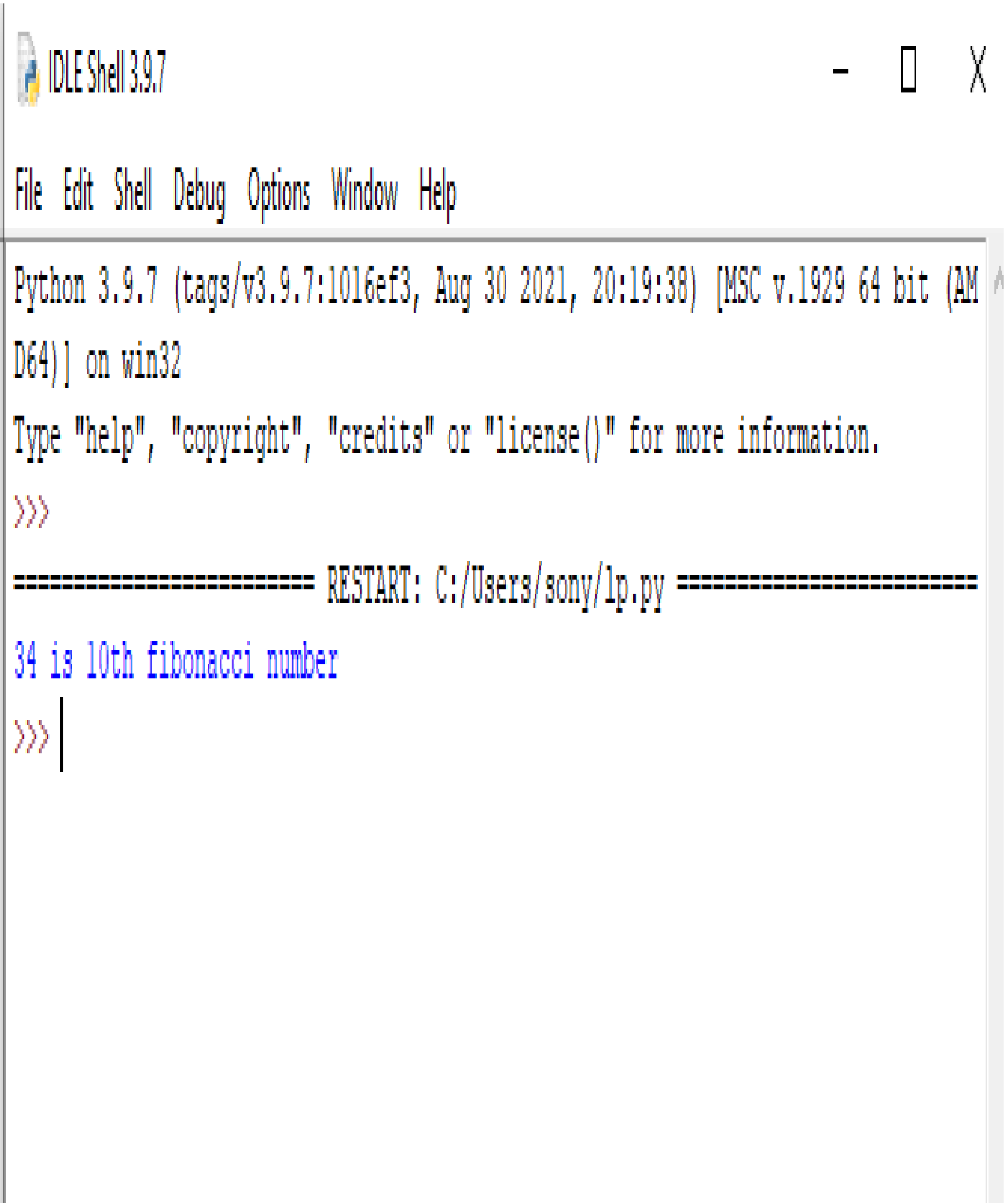
(1) $F_n = F_{n-1} + F_{n-2},\ \ F_1 = 1,\ F_2 = 1$

(2) $F_n = \frac{(1+\sqrt{5})^n - (1-\sqrt{5})^n}{2^n\sqrt{5}}$

n-th Fibonacci number


lp.py - C:/Users/sony/lp.py (3.9.7)
File Edit Format Run Options Window Help

```
# To find the n+1 th Fibonacci Number using formula
from math import sqrt
# import square-root method from math library
def nthFib(n):
        res = (((1+sqrt(5))**n)-((1-sqrt(5)))**n)/(2**n*sqrt(5))
        # compute the n-th fibonacci number
        print(int(res),'is',str(n+1)+'th fibonacci number')
        # format and print the number



nthFib(9)
```



```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======================= RESTART: C:/Users/sony/lp.py =======================
34 is 10th fibonacci number
>>>
```