International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 08 | August - 2022 Impact Factor: 7.185 ISSN: 2582-3930

A Brief Review on the Logical and Mathematical Aspects of Programming in Python

Archishman Dash

Abstract

Computers are excellent machines that are used for various purposes, calculation is one of them. Being a machine, a computer can not perform any task voluntarily, some specific instructions must be given by the user to get the computer to do the task. Computers can not make out languages of humans, that's why these instructions can be given to the computer using any programming language, python is amongst them.

Python is a beginner friendly and versatile programming language, it can be used to write a variety of different programs and its use is not limited to any specific type of problems. Web and mobile applications, search en- gines, games and animation software have python at their core. It's used in artificial intelligence and machine learning to perform data manipulation, analysis and visualisation. It is a high level language, meaning its closer to natural language like English, which makes it easier to understand and thus is beginner friendly.

Mathematics has always been shrouded in mystery, the more a person tries to make things out, the more and more mysteries of mathematics are surfaced. To perform any specific task on a small size of samples seems easier but as the size of the samples increase it becomes more and more difficult to perform the task manually. However computers being excellent calculators, can perform these specific tasks flawlessly and very fast. To do this the user needs to feed the instructions to the computer using any programming lan- guage. Python being a high level and versatile language can be of great use. To do any mathematical task with a computer, first the computer needs to be fed with a set of specific instructions, that are collectively called a pro- gram. These instructions can be fed to the computer using simple functions like 'input', logical statements such as 'if', 'else' and 'elif', data structures namely lists, dictionaries and tuples, arithmetic, assignment and compari- son operators and other predefined functions in python. After running the program, the computer prints the required output in just seconds. In this project several mathematical calculations are done using the com- puter. First the set of instructions are prepared by the user by doing the mathematics for a small sample, then the set of instructions are written in the form of a program using python and fed to the computer. Finally the program is run to do the task for a large sample and the output isprinted.

1 Visualisation

Understating the flow of the program is the key to write flawless programs that produce desired output. Sometimes when writing the program it may generate slightly different output than what the user requires, to avoid this, visualisation of the program is important. However high may be the level of the programming language, visualising the program may be a frustrating process to follow, when done manually. When done with the help of a com- puter the program is easily visualised and the flow of the program becomes apparent. This helps the user to rectify the program at several stages to get the desired output. The 'pythontutor.com' is an online python program visu- alisation website where a user can easily understand the flow of the program he or she has written and can make changes to his or her program to get the desired output. Many of the programs in this project have been visualised using the 'Python Tutor' website. This online python program visualisation website has been of great help to visualise the programs with different inputs and understand the flow of the programs. Through this website a user can understand the flow of the program step by step and can visualise what's happening in each step, so that changes can be made in the program at spe- cific stages to ensure that the desired output is achieved. (*Python Tutor: Learn Python, JavaScript, C, C++, and Java by visualizing code*, n.d.)

2 Finding the Largest and the Smallest Num- ber in a List

Program for Finding the Smallest Number in a List

```
n=input('Enter the number of numbers in which you want to find the smallest number')
a=int(n) lst=list()
for i in range(1,a+1): b=input('Enter a Number') if b=='Done':
break else:
c=int(b) lst.append(c)
print('The list of the numbers is',lst) smallest=None
for i in lst:
if smallest==None: smallest=i
elif i<smallest: smallest=i
print('The smallest number in the list is',smallest)
```

Output

Enter the number of numbers in which you want to find thesmallest number5
Enter a Number22 Enter a Number21 Enter a Number2 Enter a Number34 Enter a Number1
The list of the numbers is [22, 21, 2, 34, 1] The smallest number in the list is

Program for Finding the Largest Number in a List

```
n=input('Enter the number of numbers in which you want to find the largest number')
a=int(n) lst=list()
for i in range(1,a+1): b=input('Enter a Number') if b=='Done':
break else:
c=int(b) lst.append(c)
print('The list of the numbers is',lst) largest=None
for i in lst:
if largest==None: largest=i
elif i>largest: largest=i
print('The largest number in the list is',largest)
```

Output

Enter the number of numbers in which you want to find the largest number5
Enter a Number12 Enter a Number21 Enter a Number23 Enter a Number45 Enter a Number67
The list of the numbers is [12, 21, 23, 45, 67] The largest number in the list is 67

In the programs for finding the largest and the smallest number, first the user is asked to enter the number of numbers from which the largest and the smallest number is to be found. Then an empty list is filled with the numbers the user gives to the console. If the user decides to reduce the number of numbers in the list, he can enter 'Done' and the list will be created with a reduced number of numbers. After the list is created, the next step is to find the largest and the smallest number in the list. For which a variable largest and smallest is created with a value 'None', which is used to capture the first number of the list to be compared with the rest. Then the rest of the numbers are compared with the previous numbers and at the end the largest and the smallest value is stored in the respective variables, which is then printed. (Severance, 2020)

3 Prime Numbers, Twin Primes, Perfect Num- bers, Mersenne Primes

3.1 Prime Numbers

Program for Finding Prime Numbers upto a Number

```
n=input('Enter the number upto which you want to find primenumbers') N=int(n)
value=0
for i in range(2,N+1):
for a in range(2,(int(i//2)+1)): if i□a==0:
print(i,'Equals',a,'*',i//a) break%
else:
print(i,'is a prime number') value=value+1
print('Number of total prime numbers upto',N,'is',value)
```

Output

```
Enter the number upto which you want to find prime numbers100
2    is a prime number
3    is a prime number
4    Equals 2 * 2
5    is a prime number
6    Equals 2 * 3
7    is a prime number
8    Equals 2 * 4
9    Equals 3 * 3
10   Equals 2 * 5
11   is a prime number
12   Equals 2 * 6
13   is a prime number
14   Equals 2 * 7
15   Equals 3 * 5
16   Equals 2 * 8
17   is a prime number
```

18 Equals 2 * 9
19 is a prime number
20 Equals 2 * 10
21 Equals 3 * 7
22 Equals 2 * 11
23 is a prime number
24 Equals 2 * 12
25 Equals 5 * 5
26 Equals 2 * 13
27 Equals 3 * 9
28 Equals 2 * 14
29 is a prime number
30 Equals 2 * 15
31 is a prime number
32 Equals 2 * 16
33 Equals 3 * 11
34 Equals 2 * 17
35 Equals 5 * 7
36 Equals 2 * 18
37 is a prime number
38 Equals 2 * 19
39 Equals 3 * 13
40 Equals 2 * 20
41 is a prime number
42 Equals 2 * 21
43 is a prime number
44 Equals 2 * 22
45 Equals 3 * 15
46 Equals 2 * 2

47 is a prime number
48 Equals 2 * 24
49 Equals 7 * 7
50 Equals 2 * 25
51 Equals 3 * 17
52 Equals 2 * 26
53 is a prime number
54 Equals 2 * 27
55 Equals 5 * 11
56 Equals 2 * 28
57 Equals 3 * 19
58 Equals 2 * 29
59 is a prime number
60 Equals 2 * 30
61 is a prime number
62 Equals 2 * 31
63 Equals 3 * 21
64 Equals 2 * 32
65 Equals 5 * 13
66 Equals 2 * 33
67 is a prime number
68 Equals 2 * 34
69 Equals 3 * 23
70 Equals 2 * 35
71 is a prime number
72 Equals 2 * 36
73 is a prime number
74 Equals 2 * 37
75 Equals 3 * 25
76 Equals 2 * 38
77 Equals 7 * 11
78 Equals 2 * 39
79 is a prime number

80 Equals 2 * 40
81 Equals 3 * 27
82 Equals 2 * 41
83 is a prime number
84 Equals 2 * 42
85 Equals 5 * 17
86 Equals 2 * 43
87 Equals 3 * 29
88 Equals 2 * 44
89 is a prime number
90 Equals 2 * 45
91 Equals 7 * 13
92 Equals 2 * 46
93 Equals 3 * 31
94 Equals 2 * 47
95 Equals 5 * 19
96 Equals 2 * 48
97 is a prime number
98 Equals 2 * 49
99 Equals 3 * 33
100 Equals 2 * 50

Number of total prime numbers upto 100 is 25

Prime numbers are numbers which have only two divisors, ’1’ and itself.

In the program to find prime numbers upto a number which is entered by the user, starting from 2, a ’for’ loop is created which captures the numbers to be checked for being prime, in the next ’for’ loop the divisors of the number are found. The divisors are found from 2 to half of the number using the integer division (’//’) operator, because every number has 1 as its divisor and no divisors of a number can be greater than half of the number. The divisors are checked with an ’if’ statement, which when executed immediately prints the first set of divisors and also prints the statement that the number is not prime, and the ’else’ statement prints the statement that the number is prime when the ’for’ loop which finds its divisors is exhausted. An ’index’ variable is used to find how many primes are there in between ’2’ and the number the user entered. (van Rossum et al., 2018)

3.2 Twin Primes

Program to Find Twin Prime Pairs upto a Number

```
n=input('Enter the number upto which you want to find the twin prime pairs')
N=int(n) lst=list()
if n=='2' or n=='3':
print('There are no twin primes upto',N) else:
for i in range(2,N-2+1):
for x in range(2,((i//2)+1)): if i%x==0:
break
else:
p=i+2
for q in range(2,((p//2)+1)): if p%q==0:
break
else:
lst.append((i,p)) if len(lst)==0:
print('There are no twin primes upto',N) else:
print('The number of pairs of twin primes upto',N,'is'
,len(lst))
print('The twin prime pairs upto',N,'are') print (*lst,sep=', ')
```

Output

Enter the number upto which you want to find the twin primepairs1000 The number of pairs of twin primes upto 1000 is 35

The twin prime pairs upto 1000 are

(3, 5), (5, 7), (11, 13), (17, 19), (29, 31), (41, 43), (59, 61),

(71, 73), (101, 103), (107, 109), (137, 139), (149, 151), (179, 181),

(191, 193), (197, 199), (227, 229), (239, 241), (269, 271), (281,283),

(311, 313), (347, 349), (419, 421), (431, 433), (461, 463), (521,523),

(569, 571), (599, 601), (617, 619), (641, 643), (659, 661), (809, 811),
(821, 823), (827, 829), (857, 859), (881, 883)

Twin primes are pair of prime numbers with a difference of 2.

This program asks the user to enter the number upto which the pairs of twin primes are to be found and creates a blank list. If the user enters ’2’ or ’3’, the program prints that there are no twin prime pairs upto ’2’ or ’3’, using an ’if’ statement. If the user enters any number except ’2’ or ’3’, the program finds the prime numbers starting from 2 to the number which is 2 less than the user entered number, using a ’for’ loop. If a prime number is found, the program adds 2 to the number and checks, if that number is also prime, then a pair of twin prime is found and appended to the list in the form of a tuple. Then the rest of the numbers in the ’for’ loop are also checked for this condition. At the end of the program the number of twin prime pairs found are printed along with the twin prime pairs in the form of tuples.

3.3 Perfect Numbers

Program for Finding Perfect Numbers upto a Number

```
n=input(’Enter the number upto which you want to find the perfect numbers’)
N=int(n) index=0
for i in range(2,N+1): value=0
print(’Divisors of’,i,’are’) for a in range(1,(i//2)+1):
if i□a==0: %
print(a) value=value+a
print(’Total of divisors of’,i,’=’,value) if value==i:
print(i,’is a perfect number’) index=index+1
else:
print(i,’is not a perfect number’)
print(’Number of total perfect numbers upto’,N,’is’,index)
```

Output

Enter the number upto which you want to find the perfectnumbers30 Divisors of 2 are
1
Total of divisors of 2 = 1
2 is not a perfect number Divisors of 3 are
1
Total of divisors of 3 = 1
3 is not a perfect number Divisors of 4 are
1
2
Total of divisors of 4 = 3
4 is not a perfect number Divisors of 5 are
1
Total of divisors of 5 = 1
5 is not a perfect number Divisors of 6 are
1
2
3
Total of divisors of 6 = 6
6 is a perfect number Divisors of 7 are
1
Total of divisors of 7 = 1
7 is not a perfect number Divisors of 8 are
1
2
4
Total of divisors of 8 = 7
8 is not a perfect number Divisors of 9 are
1

3

Total of divisors of 9 = 4

9 is not a perfect number Divisors of 10 are

1

2

5

Total of divisors of 10 = 8

10 is not a perfect number Divisors of 11 are

1

Total of divisors of 11 = 1

11 is not a perfect number Divisors of 12 are

1

2

3

4

6

Total of divisors of 12 = 16

12 is not a perfect number Divisors of 13 are

1

Total of divisors of 13 = 1

13 is not a perfect number Divisors of 14 are

1

2

7

Total of divisors of 14 = 10

14 is not a perfect number Divisors of 15 are

1

3

5

Total of divisors of 15 = 9

1

2

4

8

Total of divisors of 16 = 15

15 is not a perfect number Divisors of 17 are

1

Total of divisors of 17 = 1

16 is not a perfect number Divisors of 18 are

1

2

3

6

9

Total of divisors of 18 = 21

17 is not a perfect number Divisors of 19 are

1

Total of divisors of 19 = 1

18 is not a perfect number Divisors of 20 are

1

2

4

5

10

Total of divisors of 20 = 22

19 is not a perfect number Divisors of 21 are

1

3

7

Total of divisors of 21 = 11

20 is not a perfect number Divisors of 22 are

1

2

11

Total of divisors of 22 = 14

21 is not a perfect number Divisors of 23 are

1

Total of divisors of 23 = 1

22 is not a perfect number Divisors of 24 are

1

2

3

4

6

8

12

Total of divisors of 24 = 36

23 is not a perfect number Divisors of 25 are

1

5

Total of divisors of 25 = 6

24 is not a perfect number Divisors of 26 are

1

2

13

Total of divisors of 26 = 16

25 is not a perfect number Divisors of 27 are

1

3

9

Total of divisors of 27 = 13

26 is not a perfect number Divisors of 28 are

1

2

4

7

14

Total of divisors of 28 = 28

27 is a perfect number Divisors of 29 are

1

Total of divisors of 29 = 1

28 is not a perfect number Divisors of 30 are

1

2

3

5

6

10

15

Total of divisors of 30 = 42

29 is not a perfect number

Number of total perfect numbers upto 30 is 2

Perfect numbers are numbers whose divisors except the number itself and including '1' adds up to the number itself. For example 28 is a perfect number. Its divisors except the number itself are 1,2,4,7,14, which adds up to 28.

In the program to find the perfect numbers between two numbers starting from 2 to a number entered by the user, a 'for' loop is created which captures the number to be checked for being perfect, a variable 'value' is created whose value is set to 0 in every iteration in the execution of the first 'for' loop every time. In the second 'for' loop the divisors of the number except the the number and including 1 are found, their sum is stored in the variable 'value', an 'if' statement decides whether the number is perfect or not. An 'index' variable is used to find the number of perfect numbers between '2' and the number the user entered.

3.4 Mersenne Primes

Program for Finding Mersenne Primes upto a Number

```
n=input('Enter a number upto which you want to find the mersenne primes')
N=int(n) index=0 i=2
while (2**i)<=N+1: a=(2**i)-1
for x in range(2,((a//2)+1)): if a%x==0:
print(a,'equals',x,'*',a//x) print(a,'is not a mersenne prime') i=i+1
break
else:
print(a,'is a mersenne prime') index=index+1
i=i+1
print('The number of mersenne primes upto',N,'is',index)
```

Output

```
Enter a number upto which you want to find the mersenne primes10
3 is a mersenne prime
7 is a mersenne prime
The number of mersenne primes upto 10 is 2
```

Mersenne numbers are numbers which are 1 less than a power of 2. If the Mersenne number is prime, its called a Mersenne prime.

In this program an user input decides upto which number the Mersenne primes are to be found. In a 'while' loop the Mersenne numbers are calculated and in the 'for' loop that number is checked whether its prime or not. If its prime its printed that the number is a Mersenne prime and if its not, its printed that the number is not a Mersenne prime. An 'index' variable is used to calculate how many Meresnne primes are there in between two numbers. (*Mersenne Primes: History, Theorems and Lists*, n.d.)

4 Palindromes

Program to Find if a Word is a Palindrome or Not

```
word=input('Enter a string') if len(word)==1:
print(word,'is a palindrome') elif len(word)%2==0:
for n in range(0,int(len(word)/2)): if word[n]==word[len(word)-(n+1)]:
if not n==int(len(word)/2)-1: continue
print(word,'is a palindrome') else:
print(word,'is not a palindrome') break
elif len(word)%2!=0:
for n in range(0,int((len(word)/2)-0.5)):
if word[n]==word[len(word)-(n+1)]:
if not n==int((len(word)/2)-0.5)-1: continue
print(word,'is a palindrome') else:
print(word,'is not a palindrome') break
```

Output

```
Enter a stringaboba aboba is a palindrome
```

Palindromes are word which are written the same starting from the beginning and the end. For example 'Noon' is written 'Noon' from the beginning and the end, and thus is a palindrome.

In the program to check whether a word is palindrome or not, first the user is asked to input a string. If the length of the string is '1', it is a palindrome.

If the length of the string is even, a 'for' loop iterates through every letter starting from the first letter upto the half of the string, and an 'if' statement

compares them with the rest of the string starting from the end to check whether they are same or not. If every time the 'if' statement is 'True' the string is a palindrome. If the 'if' statement becomes 'False' even once, the string is not a palindrome.

If the length of the string is odd, the middle letter is not compared with any other letter. The 'for' loop iterates through the first letter to the letter before the middle letter. The 'if' statement compares them with the rest of the letter starting from the end to check whether they are same or not. If the 'if' statement is 'True' every time, the string is a palindrome, otherwise not.

5 Time between two Dates

Program to Find Time between Two Dates

```
months={'Jan':1,'Feb':2,'Mar':3,'Apr':4,'May':5,'Jun':6,'Jul':7, 'Aug':8,'Sep':9,'Oct':10,'Nov':11,'Dec':12}
yyyymmdd2=input('Enter the date upto which you want to findthe total time')
yyyymmdd1=input('Enter the date from which you want to findthe total time')
dd2=int(yyyymmdd2[7:9]) dd1=int(yyyymmdd1[7:9]) yyyy2=int(yyyymmdd2[0:4])
yyyy1=int(yyyymmdd1[0:4]) mm2=months[yyyymmdd2[4:7]] mm1=months[yyyymmdd1[4:7]]
if yyyy1>yyyy2 or (yyyy1==yyyy2 and mm1>mm2) or (yyyy1==yyyy2 and mm1==mm2 and dd1>dd2):
print('The total time can not be found, please enter the date correctly')
elif (yyyy2 %4==0 and yyyy2 %100!=0 and mm2==months['Feb'] and dd2>29) or (yyyy2%4==0 and yyyy2%
100==0 and yyyy2%400==0 and mm2==months['Feb'] and dd2>29):
print('Feb in the year',yyyy2,'can not have more than 29days') elif (yyyy1%4==0 and yyyy1%100!=0 and
mm1==months['Feb'] and
dd1>29) or (yyyy1%4==0 and yyyy1%100==0 and yyyy1%400==0 and mm1==months['Feb']
and dd1>29):
print('Feb in the year',yyyy1,'can not have more than 29days') elif (yyyy2%4!=0 and mm2==months['Feb']
and dd2>28) or (yyyy2%4==0
and yyyy2%100==0 and yyyy2%400!=0 and mm2==months['Feb'] and dd2>28):
print('Feb in the year',yyyy2,'can not have more than 28days') elif (yyyy1%4!=0 and mm1==months['Feb']
and dd1>28) or (yyyy1%4==0
and yyyy1%100==0 and yyyy1%400!=0 and mm1==months['Feb'] and dd1>28):
print('Feb in the year',yyyy1,'can not have more than 28 days') elif dd2>=dd1:
```

```
dd=dd2-dd1 if mm2>=mm1:
mm=mm2-mm1 yyyy=yyyy2-yyyy1
print('The total time is',yyyy,'Years',mm,'Months',dd,'Days') else:
mm2=mm2+12 mm=mm2-mm1 yyyy2=yyyy2-1 yyyy=yyyy2-yyyy1
print('The total time is',yyyy,'Years',mm,'Months',dd,'Days')
else:
dd2=dd2+30 dd=dd2-dd1 mm2=mm2-1  if mm2>=mm1:
mm=mm2-mm1 yyyy=yyyy2-yyyy1
print('The total time is',yyyy,'Years',mm,'Months',dd,'Days') else:
mm2=mm2+12 mm=mm2-mm1 yyyy2=yyyy2-1 yyyy=yyyy2-yyyy1
print('The total time is',yyyy,'Years',mm,'Months',dd,'Days')
```

Output

Enter the date upto which you want to find the total time2001Jan31 Enter the date from which you want to find the totaltime2000Feb29 The total time is 0 Years 11 Months 2 Days

This program asks the user to enter two dates in the format yyyymmdd. Which is broken into two dates of type 'int', i.e. 'dd2', 'dd1', two years of type 'int', i.e. 'yyyy2', 'yyyy1', two months of type 'int', i.e. 'mm2', 'mm1', using indexing and the dictionary, which is defined at the top of the program. The '2' in these variables denotes the day, month and year upto which the time is to be calculated, the '1' denotes the day, month and year from which the time is to be calculated.

After this an 'if' statement checks that the day, month and year upto which the time is to be calculated comes after the day, month and year from which the time is to be calculated.
Four 'elif' statements are there, which check whether the user enters the month 'February' with the correct consideration of it having 29 days or 28 days. (*Leap Years*, n.d.)
Only after the above 'if' and 'elif' statements have become 'False', the rest of the program is executed which calculates the time between two dates and prints it.

6 Fibonacci Sequence

6.1 Fibonacci Sequence starting from ’0’

Program to Find Fibonacci Sequence upto a Position Starting from ’0’ and Finding the Number at a Specific Position

```
n=input(’Enter the position upto which you want to find theFibonacci sequence’)
N=int(n)
b=input(’Enter the position of the Fibonacci number you want tofind’) B=int(b)
if N==1:
print(’The Fibonacci sequence upto the first position starting from’,0,’is’)
elif N==2:
print(’The Fibonacci sequence upto the second position starting from’,0,’is’)
elif N==3:
print(’The Fibonacci sequence upto the third position starting from’,0,’is’)
else:
print(’The Fibonacci sequence upto the’,N,’th position starting from’,0,’is’)
lst=[0,1,2] if N<=3:
print(lst[0:N]) if B==1:
print(’First number of the Fibonacci sequence is’,lst[0]) elif B==2:
print(’Second number of the Fibonacci sequence is’,lst[1]) elif B==3:
print(’Third number of the Fibonacci sequence is’,lst[2])
else:
for i in range(4,N+1): lst.append(lst[-1]+lst[-2])
print(lst) if B==1:
print(’First number of the Fibonacci sequence is’,lst[0])
```

```
elif B==2:
print('Second number of the Fibonacci sequence is',lst[1]) elif B==3:
print('Third number of the Fibonacci sequence is',lst[2]) else:
print(B,'th number of Fibonacci sequence is',lst[B-1])
```

Output

```
Enter the position upto which you want to find the Fibonacci sequence10
Enter the position of the Fibonacci number you want to find8
The Fibonacci sequence upto the 10 th position starting from 0is [0, 1, 2, 3, 5, 8, 13, 21, 34, 55]
8 th number of Fibonacci sequence is 21
```

This program asks the user to enter the position upto which the Fibonacci Sequence is to be printed, and also asks the user to enter the position to find the specific number in the sequence.
The sequence is printed starting from 0 upto the position the user enters, and the specific number in the user entered position is also printed.
The program runs on the logic to add the current number in the sequence with the previous to calculate the next number in the sequence. The sequence 0,1,2 is considered as the 'definition' in the program.

6.2 Fibonacci Sequence starting from '1'

Program to Find Fibonacci Sequence upto a Position Starting from '1' and Finding the Number at a Specific Position

```
n=input('Enter the position upto which you want to find theFibonacci sequence')
N=int(n)
b=input('Enter the position of the Fibonacci number you want tofind') B=int(b)
if N==1:
print('The Fibonacci sequence upto the first position starting from',1,'is')
elif N==2:
print('The Fibonacci sequence upto the second position starting from',1,'is')
elif N==3:
print('The Fibonacci sequence upto the third position starting from',1,'is')
else:
print('The Fibonacci sequence upto the',N,'th position starting from',1,'is')
lst=[1,2,3] if N<=3:
```

```
print(lst[0:N]) if B==1:
print('First number of the Fibonacci sequence is',lst[0]) elif B==2:
print('Second number of the Fibonacci sequence is',lst[1]) elif B==3:
print('Third number of the Fibonacci sequence is',lst[2])
else:
for i in range(4,N+1): lst.append(lst[-1]+lst[-2])
print(lst) if B==1:
print('First number of the Fibonacci sequence is',lst[0]) elif B==2:
print('Second number of the Fibonacci sequence is',lst[1])


elif B==3:
print('Third number of the Fibonacci sequence is',lst[2]) else:
print(B,'th number of Fibonacci sequence is',lst[B-1])
```

Output

```
Enter the position upto which you want to find the Fibonacci sequence40
Enter the position of the Fibonacci number you want to find38 The Fibonacci sequence upto the 40 th position starting from 1is
[1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597,
2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393,196418,
317811, 514229, 832040, 1346269, 2178309, 3524578, 5702887,9227465,
14930352, 24157817, 39088169, 63245986, 102334155, 165580141]
38 th number of Fibonacci sequence is 63245986
```

This program is the same as the above program with the only exception that it starts from 1, and the 'definition' is considered to be 1,2,3.

The rest of the program runs on the logic to add the current number in the sequence with the previous to calculate the next number in the sequence.

7 Decimal,Binary,Octal and Hexadecimal Numbers

7.1 Converting Decimal to Binary, Decimal to Octal, Decimal to Hexadecimal

Program to Convert Decimal Number to its Binary Equivalent

```
n=input('Enter the number you want to convert to binary') N=int(n)
result=True lst=list() while result:
q=N//2 r=N%2
lst.append(r)
N=q
if q==0:
result=False
print('The remainders are',lst) lst.reverse()
print('The binary equivalent of the number ',n,' is ',
*lst,sep='')
```

Output

Enter the number you want to convert to binary4545
The remainders are [1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1] The binary equivalent of the number 4545 is 1000111000001

Program to Convert Decimal Number to its Octal Equivalent

```
n=input('Enter the number you want to convert to octal') N=int(n)
result=True lst=list() while result:
q=N//8 r=N%8
lst.append(r)
N=q
if q==0:
result=False
print('The remainders are',lst) lst.reverse()
print('The octal equivalent of the number ',n,' is ',
*lst,sep='')
```

Output

Enter the number you want to convert to octal966 The remainders are [6, 0, 7, 1]
The octal equivalent of the number 966 is 1706

Program to Convert Decimal Number to its Hexadecimal Equiv- alent

```
extra={10:'A',11:'B',12:'C',13:'D',14:'E',15:'F'}
n=input('Enter the number you want to convert to hexadecimal') N=int(n)
result=True lst=list() while result:
q=N//16 r=N%16
if r>=10 and r<16: r=extra[r]
lst.append(r)
N=q
if q==0:
result=False
print('The remainders are',lst) lst.reverse()
print('The hexadecimal equivalent of the number ',n,' is ',
*lst,sep='')
```

Output

Enter the number you want to convert to hexadecimal1123 The remainders are [3, 6, 4]
The hexadecimal equivalent of the number 1123 is 463

These programs ask the user to enter a decimal number which is converted to binary or octal or hexadecimal equivalent of that number.

These programs run on the logic to divide the number entered by the user by the required base, storing the remainder and again dividing the quo- tient by the required base and keep repeating the process until the quotient becomes 0. The remainders are stored in a list and are printed in reverse without space.

In the program to convert decimal numbers to its hexadecimal equivalent, a dictionary is used to store the value of 10, 11, 12,13, 14 and 15, indexing is used to access their values from the dictionary. The rest of the program is similar to the program of converting decimal to binary and decimal to octal equivalent.

7.2 Converting Binary, Octal and Hexadecimal num- bers to Decimal

7.2.1 Positional Notation Method

Program to Convert Binary Number to its Decimal Equivalent

```
n=input('Enter the binary number you want to convert to decimal') d=0
for i in range(0,len(n)): a=(2**(len(n)-(i+1)))*int(n[i]) d=d+a
print('The decimal equivalent of',n,'is',d)
```

Output

Enter the binary number you want to convert todecimal1000111000001 The decimal equivalent of 1000111000001 is 4545

Program to Convert Octal Number to its Decimal Equivalent

```
n=input('Enter the octal number you want to convert to decimal') d=0
for i in range(0,len(n)): a=(8**(len(n)-(i+1)))*int(n[i]) d=d+a
print('The decimal equivalent of',n,'is',d)
```

Output

Enter the octal number you want to convert to decimal1706 The decimal equivalent of 1706 is 966

Program to Convert Hexadecimal Number to its Decimal Equiv- alent

```
extra={'A':10,'B':11,'C':12,'D':13,'E':14,'F':15}
n=input('Enter the hexadecimal number you want to convert todecimal') d=0
lst=list() for N in n:
lst.append(N) print('The bits are',lst) for x in range(0,len(lst)):
if lst[x]=='A' or lst[x]=='B' or lst[x]=='C' or lst[x]=='D' or lst[x]=='E' or lst[x]=='F':
lst[x]=extra[lst[x]] if x!=len(lst)-1:
continue else:
print('The simplified bits',lst) for i in range(0,len(lst)):
a=(16**(len(lst)-(i+1)))*int(lst[i]) d=d+a
print('The decimal equivalent of',n,'is',d)
```

Output

Enter the hexadecimal number you want to convert to decimal463 The bits are ['4', '6', '3']
The decimal equivalent of 463 is 1123

These programs add together the value of multiplication of the bits of the required numbers starting from right with their weights. For a number having 'n' digits the right most bit has a weight of (base in which the number is in)^{0} and the left most bit has a weight of (base in which the number is in)$^{n-1}$.

In the program to convert hexadecimal numbers to its decimal equivalent a dictionary is used which stores the value of 'A','B','C', 'D', 'E', 'F', and indexing is used to access their value from the dictionary.

7.2.2 Doubling Method

Program to Convert Binary Number to its Decimal Equivalent

```
n=input('Enter the binary number you want to convert to decimal') a=(0*2)+int(n[0])
for i in range(1,len(n)): b=(a*2)+int(n[i]) a=b
print('The decimal equivalent of of the number',n,'is',a)
```

Output

Enter the binary number you want to convert todecimal1000111000001 The decimal equivalent of of the number 1000111000001 is 4545

Program to Convert Octal Number to its Decimal Equivalent

```
n=input('Enter the octal number you want to convert to decimal') a=(0*8)+int(n[0])
for i in range(1,len(n)): b=(a*8)+int(n[i]) a=b
print('The decimal equivalent of of the number',n,'is',a)
```

Output

Enter the octal number you want to convert to decimal1706 The decimal equivalent of of the number 1706 is 966

Program to Convert Hexadecimal Number to its Decimal Equiv- alent

```
extra={'A':10,'B':11,'C':12,'D':13,'E':14,'F':15}
n=input('Enter the hexadecimal number you want to convert todecimal') d=0
lst=list() for N in n:
lst.append(N) print('The bits are',lst) for x in range(0,len(lst)):
if lst[x]=='A' or lst[x]=='B' or lst[x]=='C' or lst[x]=='D' or lst[x]=='E' or lst[x]=='F':
lst[x]=extra[lst[x]] if x!=len(lst)-1:
continue else:
print('The simplified bits',lst) a=(0*16)+int(lst[0])
for i in range(1,len(lst)): b=(a*16)+int(lst[i]) a=b
print('The decimal equivalent of of the number',n,'is',a)
```

Output

Enter the hexadecimal number you want to convert to decimal463 The bits are ['4', '6', '3']

The decimal equivalent of of the number 463 is 1123

These programs multiply the bit with the base in which the number is and adds them to the next bit. In the first step since there is no bits to the left of the first bit the product is zero which is added to the first bit. This value is multiplied with the base in which the number is and added to the next bit and the process continues till the programs reach the right most bit. The value the programs get at last is the decimal equivalent and is printed.

In the program to convert hexadecimal numbers to its decimal equivalent a dictionary is used which stores the value of 'A','B','C', 'D', 'E', 'F', and indexing is used to access their value from the dictionary. (*Binary to Decimal*

- *Conversion, Formula, Conversion Chart, Examples*, n.d.)

7.3 Generalised Program to Convert a Decimal Num- ber to any Base Equivalent from Base 2 to 9

Program to Convert a Decimal Number to any Base Equivalent from Base 2 to 9

```
n=input('Enter the decimal number you want to convert to any base from 2 to 9')
m=input('Enter the base you want to convert the number to') M=int(m)
N=int(n) result=True lst=list() while result:
q=N//M r=N%M
lst.append(r)
N=q
if q==0:
result=False
print('The remainders are',lst) lst.reverse()
print('The decimal number ',n,' when converted to base '
,m,' is ',*lst,sep='')
```

Output

Enter the decimal number you want to convert to any base from 2 to912 Enter the base you want to convert the number to4

The remainders are [0, 3]

The decimal number 12 when converted to base 4 is 30

This program asks the user to enter the base to which the number is to be converted, along with the number. The rest of the program runs on the same logic of dividing the number with the entered base, storing the remainder, dividing the quotient with the entered base again and repeat the process until the quotient becomes '0'. The remainders are printed in reverse without space.

7.4 Generalised Programs to Convert a Number inany Base from Base 2 to 9 to its Decimal Equivalent

7.4.1 Positional Notation Method

Program to Convert a Number in any Base from Base 2 to 9 to its Decimal Equivalent

```
m=input('Enter the base from 2 to 9 in which the number is in') n=input('Enter the number you want to convert to decimal') M=int(m)
d=0
for i in range(0,len(n)): a=(M**(len(n)-(i+1)))*int(n[i]) d=d+a
print('The decimal equivalent of',n,'is',d)
```

Output

```
Enter the base from 2 to 9 in which the number is in4 Enter the number you want to convert to decimal30
The decimal equivalent of 30 is 12
```

The user is asked to enter the base in which the number is in, along with the number. The rest of the program runs on the same logic of multiplying every bit with their weight and adding them together.

7.4.2 Doubling Method

Program to Convert a Number in any Base from Base 2 to 9 to its Decimal Equivalent

```
m=input('Enter the base from 2 to 9 in which the number is in') n=input('Enter the number you want to convert to decimal') M=int(m)
a=(0*M)+int(n[0])
for i in range(1,len(n)): b=(a*M)+int(n[i]) a=b
print('The decimal equivalent of of the number',n,'is',a)
```

Output

Enter the base from 2 to 9 in which the number is in4 Enter the number you want to convert to decimal30 The decimal equivalent of of the number 30 is 12

The user enters the base in which the number is in, along with the number. The program starts from the left most bit, multiplying every bit with the base in which the number is in, adds them to the next bit, repeats the process until the program reaches the right most bit. In the first step since there is no bit to the left of the left most bit the product '0' is added to the left most bit. (*Binary to Decimal - Conversion, Formula, Conversion Chart, Examples*, n.d.)

7.5 Converting Binary Number to its Octal Equivalent

Program to Convert Binary Number to its Octal Equivalent

```
bo={'000':0,'001':1,'010':2,'011':3,'100':4,'101':5,'110':6,'111':7}
n=input('Enter the binary number you want to convert to octal')
N=None
if len(n)%3==1:
N='00'+n
print('The binary number is',N) elif len(n)%3==2:
N='0'+n
print('The binary number is',N) else:
N=n
print('The binary number is',N) lst=list()
for a in range(0,len(N),3): lst.append(N[a:a+3])
print(lst)
for key,val in bo.items():
for e in range(0,len(lst)): if lst[e]==key:
lst[e]=val
print(lst)
print('The octal equivalent of the number ',n,' is ',*lst,sep='')
```

Output

Enter the binary number you want to convert to octal1011011 The binary number is 001011011
['001', '011', '011']

[1, 3, 3]

The octal equivalent of the number 1011011 is 133

This program asks the user to enter the binary number. The program checks the length of the binary number and converts the length to be divisible by 3, by adding ’0’ to the left of the binary number. Then it uses a dictionary and indexing to convert every 3 bits of the binary number to a single bit of octal number, and prints it. (*Binary to Octal Conversion - Methods, Definition, Rules, Examples & Practice Questions*, n.d.)

7.6 Converting Octal Number to its Binary Equivalent

7.6.1 Program-1

Program to Convert Octal Number to its Binary Equivalent

```
bo={’0’:’000’,’1’:’001’,’2’:’010’,’3’:’011’,’4’:’100’,’5’:’101’, ’6’:’110’,’7’:’111’}
n=input(’Enter the octal number you want to convert to binary’) lst=list()
for a in range(0,len(n)): lst.append(n[a:a+1])
print(lst)
for key,val in bo.items():
for e in range(0,len(lst)): if lst[e]==key:
lst[e]=val
print(lst)
print(’The binary equivalent of the number ’,n,’ is ’,*lst,sep=’’)
```

Output

```
Enter the octal number you want to convert to binary133 [’1’, ’3’, ’3’]
[’001’, ’011’, ’011’]
The binary equivalent of the number 133 is 001011011
```

The program asks the user to enter the octal number. A dictionary and indexing is used by the program to convert every bit of the octal number to 3 bits of the binary number and prints it. (*Octal to Binary - Conversion, Table, Examples*, n.d.)

7.6.2 Program-2

Program to Convert Octal Number to its Binary Equivalent

```
bo={'0':'000','1':'001','2':'010','3':'011','4':'100','5':'101', '6':'110','7':'111'}
n=input('Enter the octal number you want to convert to binary') print('The octal number is',n)
lst=list()
for a in range(0,len(n)): lst.append(n[a:a+1])
print(lst)
for key,val in bo.items():
for e in range(0,len(lst)): if lst[e]==key:
lst[e]=val
print(lst)
if lst[0]=='000':
lst[0]='' print(lst)
elif lst[0]=='001': lst[0]='1' print(lst)
elif lst[0]=='010': lst[0]='10'
print(lst)  elif lst[0]=='011':
lst[0]='11'
print(lst)
print('The binary equivalent of the number ',n,' is ',*lst,sep='')
```

Output

```
The octal number is 1236 ['1', '2', '3', '6']
['001', '010', '011', '110']
['1', '010', '011', '110']
The binary equivalent of the number 1236 is 1010011110
```

The program asks the user to enter the octal number. A dictionary and indexing is used by the program to convert every bit of the octal number to 3 bits of the binary number, 'if' statements are used to remove any '0' that appear to the left of the binary number while converting the first bit of the octal number. (*Octal to Binary - Conversion, Table, Examples*, n.d.)

7.7 Converting Binary Number to its HexadecimalEquiv- alent

Program to Convert Binary Number to its Hexadecimal Equivalent

```
bh={'0000':0,'0001':1,'0010':2,'0011':3,'0100':4,'0101':5,'0110':6,
'0111':7,'1000':8,'1001':9,'1010':'A','1011':'B','1100':'C', '1101':'D','1110':'E','1111':'F'}
n=input('Enter the binary number you want to convert to hexadecimal')
N=None
if len(n)%4==1:
N='000'+n
print('The binary number is',N) elif len(n)%4==2:
N='00'+n
print('The binary number is',N) elif len(n)%4==3:
N='0'+n
print('The binary number is',N) else:
N=n
print('The binary number is',N) lst=list()
for a in range(0,len(N),4): lst.append(N[a:a+4])
print(lst)
for key,val in bh.items():
for e in range(0,len(lst)): if lst[e]==key:
lst[e]=val
print(lst)
print('The hexadecimal equivalent of the number ',n,' is ',*lst, sep='')
```

Output

```
Enter the binary number you want to convert tohexadecimal1011010 The binary number is 01011010
['0101', '1010']
[5, 'A']
The hexadecimal equivalent of the number 1011010 is 5A
```

This program asks the user to enter the binary number. The program checks the length of the binary number and converts the length to be divisible by 4, by adding '0' to the left of the binary number. Then it uses a dictionary and indexing to convert every 4 bits of the binary number to a single bit of hexadecimal number, and prints it. (*Binary to Hexadecimal - Definition, Conversion Steps, Conversion with Decimal Point, Examples*, n.d.)

7.8 Converting Hexadecimal Number to its BinaryEquiv- alent

7.8.1 Program-1

Program to Convert Hexadecimal Number to its Binary Equivalent

```
bh={'0':'0000','1':'0001','2':'0010','3':'0011','4':'0100','5':'0101',
'6':'0110','7':'0111','8':'1000','9':'1001','A':'1010','B':'1011',
'C':'1100','D':'1101','E':'1110','F':'1111'}
n=input('Enter the hexadecimal number you want to convert tobinary') lst=list()
for a in range(0,len(n)): lst.append(n[a:a+1])
print(lst)
for key,val in bh.items():
for e in range(0,len(lst)): if lst[e]==key:
lst[e]=val
print(lst)
print('The binary equivalent of the number ',n,' is ',*lst,sep='')
```

Output

```
Enter the hexadecimal number you want to convert to binary5A ['5', 'A']
['0101', '1010']
The binary equivalent of the number 5A is 01011010
```

The program asks the user to enter the hexadecimal number. A dictionary and indexing is used by the program to convert every bit of the octal number to 4 bits of the binary number and prints it. (*Hexadecimal to Binary - Meaning, Conversion Table, Examples, FAQs*, n.d.)

7.8.2 Program-2

Program to Convert Hexadecimal Number to its Binary Equivalent

```
bh={'0':'0000','1':'0001','2':'0010','3':'0011','4':'0100','5':'0101',
'6':'0110','7':'0111','8':'1000','9':'1001','A':'1010','B':'1011',
'C':'1100','D':'1101','E':'1110','F':'1111'}
n=input('Enter the hexadecimal number you want to convert tobinary') lst=list()
for a in range(0,len(n)): lst.append(n[a:a+1])
print(lst)
for key,val in bh.items():
for e in range(0,len(lst)): if lst[e]==key:
lst[e]=val
print(lst)
if lst[0]=='0000':
lst[0]='' print(lst)
elif lst[0]=='0001': lst[0]='1' print(lst)
elif lst[0]=='0010': lst[0]='10'
print(lst)
elif lst[0]=='0011': lst[0]='11'
print(lst)
elif lst[0]=='0100': lst[0]='100'
print(lst)
elif lst[0]=='0101': lst[0]='101'
print(lst)
elif lst[0]=='0110': lst[0]='110'
print(lst)
elif lst[0]=='0111':
```

```
lst[0]='111'
print(lst)
print('The binary equivalent of the number ',n,' is ',*lst,sep='')
```

Output

```
Enter the hexadecimal number you want to convert to binary57AB ['5', '7', 'A', 'B']
['0101', '0111', '1010', '1011']
['101', '0111', '1010', '1011']
The binary equivalent of the number 57AB is 101011110101011
```

The program asks the user to enter the hexadecimal number. A dictionary and indexing is used by the program to convert every bit of the hexadecimal number to 4 bits of the binary number, 'if' statements are used to remove any '0' that appear to the left of the binary number while converting the first bit of the hexadecimal number. (*Hexadecimal to Binary - Meaning, Conversion Table, Examples, FAQs*, n.d.)

8 Finding the Repeating Pattern in a String

Program to Find the Repeating Pattern in a String

```
str=input('Enter a string')
for n in range(1,int(len(str)//2)+1): lst=list()
for i in range(0,len(str),n): lst.append(str[i:i+n])
print(lst)
if len(lst[-1])==len(lst[0]):
if lst.count(lst[0])==len(lst): print(lst[0],'is repeating') break
else:
if not n==len(str)//2: continue
else:
print('There is no repeating pattern found')
else:
lst.remove(lst[-1])
if lst.count(lst[0])==len(lst): print(lst[0],'is repeating') break
else:
if not n==len(str)//2: continue
else:
```

print('There is no repeating pattern found')

Output

Enter a stringabcbcabcabcbcabcabcbcabcab

['a', 'b', 'c', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c', 'b', 'c',

'a', 'b', 'c', 'a', 'b', 'c', 'b', 'c', 'a', 'b', 'c', 'a', 'b']

['ab', 'cb', 'ca', 'bc', 'ab', 'cb', 'ca', 'bc', 'ab', 'cb','ca', 'bc', 'ab']

['abc', 'bca', 'bca', 'bcb', 'cab', 'cab', 'cbc', 'abc', 'ab']

['abcb', 'cabc', 'abcb', 'cabc', 'abcb', 'cabc', 'ab']

['abcbc', 'abcab', 'cbcab', 'cabcb', 'cabca', 'b']

['abcbca', 'bcabcb', 'cabcab', 'cbcabc', 'ab']

['abcbcab', 'cabcbca', 'bcabcbc', 'abcab']

['abcbcabc', 'abcbcabc', 'abcbcabc', 'ab'] abcbcabc is repeating

To find the repeating pattern in a string, the string must be divided into sub strings of equal lengths (*Python Split String into Specific Length Chunks - Python Examples*, n.d.) and checked every time if they are the same or not. The length of the sub strings can be a minimum of 1. The maximum length of the sub strings can be half of the length of the string.

This program asks the user to enter a string. Which is then divided into sub strings of length which ranges from 1 to half of the length of the string. Every time after the string is divided into sub strings of equal lengths, the sub strings are printed in a list. An 'if' statement checks if the length of the last element in the list is same as the first element, if this statement is 'True' and the frequency of the first element in the list is same as the length of the list, then this substring is the repeating pattern in the string. If none of the list containing the sub strings satisfy this condition, there is no repeating pattern in the string.

If the length of the last element is not the same as the first element of the list, it is removed from the list and then the above conditions are checked again.

9 LCM and HCF

9.1 Least Common Multiple (LCM)

Program to Find LCM of a List of Numbers

```
n=input('Enter the number of numbers of which you want to find the lcm or least common multiple')
a=int(n) lst=list()
for i in range(1,a+1): b=input('Enter a Number') if b=='Done':
break else:
c=int(b) lst.append(c)
print('The list of the numbers is',lst) largest=None
for i in lst:
if largest==None: largest=i
elif i>largest: largest=i
print('The largest number in the list is',largest)
N=2
k=largest  j=k result=True while result:
for x in lst:
if j□x==0:                    %
if not x==lst[-1]: continue
print('The lcm of the list of the numbers is',j) result=False
else:
j=N*k N=N+1


x=lst[0] break
```

Output

```
Enter the number of numbers of which you want to find the lcmor least common multiple4
Enter a Number23 Enter a Number46 Enter a Number69 Enter a Number112
The list of the numbers is [23, 46, 69, 112] The largest number in the list is 112
The lcm of the list of the numbers is 7728
```

The lcm of a list of numbers must be greater than or equal to the largest number in the list. If it is greater than the largest number in the list, it must be the smallest multiple of the largest number in the list such that it is divided by every number in the list. (*Python Program to Find LCM* ,n.d.)

This program asks the user to enter the number of numbers in the list, and then asks to enter the numbers. The largest number in the list is found by the program. The program checks, if the largest number is divisible by all the numbers in the list, it is the lcm. If it is not divisible by all the numbers in the list, then it checks the same condition for the multiples of the largest number. The smallest multiple of the largest number in the list, divisible by all the numbers in the list is the lcm.

9.2 Highest Common Factor (HCF)

9.2.1 Program-1

Program to Find HCF of a List of Numbers

n=input('Enter the number of numbers of which you want to find the hcf or highest common factor')
a=int(n) lst=list()
for i in range(1,a+1): b=input('Enter a Number') if b=='Done':
break else:
c=int(b) lst.append(c)
print('The list of the numbers is',lst) smallest=None
for i in lst:
if smallest==None: smallest=i
elif i<smallest: smallest=i
print('The smallest number in the list is',smallest) k=smallest
result=True while result:
for x in lst:
if x□k==0: %
if not x==lst[-1]: continue
print('The hcf of the list of the numbers is',k) result=False
else:
k=k-1 x=lst[0] break

Output

Enter the number of numbers of which you want to find the hcfor highest common factor4
Enter a Number24 Enter a Number48 Enter a Number72 Enter a Number112
The list of the numbers is [24, 48, 72, 112] The smallest number in the list is 24
The hcf of the list of the numbers is 8

The hcf of a list of numbers must be less than or equal to the smallest number in the list. That means the hcf of a list of numbers must be lying between 1 and the smallest number in the list, including both. Whichever largest number coming between 1 and the smallest number in the list divides every number in the list is the hcf of the list of numbers. (*Python Program to Find HCF or GCD*, n.d.)
This program asks the user to enter the number of numbers in the list. Then asks to enter the numbers. The smallest number is found by the pro- gram. The program checks, if the smallest number in the list divides all the numbers in the list, it is the hcf. If it doesn't divide every number in the list, then the number coming before it will be checked for the same condition. Whichever largest number coming between the smallest number in the list and 1, including both, divides every number in the list is the hcf of the list of numbers.

9.2.2 Program-2

Program to Find HCF of a List of Numbers

```
n=input('Enter the number of numbers of which you want to find the hcf or highest common factor')
a=int(n) lst=list()
for i in range(1,a+1): b=input('Enter a Number') if b=='Done':
break else:
c=int(b) lst.append(c)
print('The list of the numbers is',lst) smallest=None
for i in lst:
if smallest==None: smallest=i
elif i<smallest: smallest=i
print('The smallest number in the list is',smallest) k=smallest
lt=list()
for j in range(k,0,-1): if k%j==0:
lt.append(j)
y=1
result=True while result:
```

```
for x in lst:
if x□k==0:                    %
if not x==lst[-1]: continue
print('The hcf of the list of the numbers is',k) result=False
else:
while y<len(lt):



k=lt[y] break
y=y+1 x=lst[0] break
```

Output

Enter the number of numbers of which you want to find the hcfor highest common factor4
Enter a Number12 Enter a Number24 Enter a Number28 Enter a Number48
The list of the numbers is [12, 24, 28, 48] The smallest number in the list is 12
The hcf of the list of the numbers is 4

The hcf of a list of numbers must be less than or equal to the smallest number in the list. That means the hcf of a list of numbers must be one of the factors of the smallest number in the list coming between 1 and the smallest number in the list, including both. Whichever largest factor of the smallest number in the list coming between 1 and the smallest number in the list divides every number in the list is the hcf of the list of numbers. (*Python Program to Find HCF or GCD*, n.d.)

This program asks the user to enter the number of numbers in the list, then asks to enter the numbers. The smallest number is found by the pro- gram. The program checks, if the smallest number in the list divides all the numbers in the list, it is the hcf. If it doesn't divide every number in the list, then the factor of the smallest number coming before it will be checked for the same condition. Whichever largest factor coming between the smallest number in the list and 1, including both, divides every number in the list is the hcf of the list of numbers.

10 Collatz Conjecture

This conjecture states that every positive integer eventually converges to 1 when two simple arithmetic operations are performed on them. These arithmetic operations are as followed:-

→ *Multiply the number by 3 and add 1 to the result, if the number is odd.*

→ *Divide the number by 2, if it is even.* (Das, 2022)

10.1 Finding the Sequence of a Number in Collatz Conjecture and Counting the Steps

Program to Find the Sequence of a Number in Collatz Conjecture and Counting the Steps

```
n=input('Enter a number of which you want to find the sequence in the Collatz conjecture')
N=int(n) a=None step=0 result=True
print('The sequence of',n,'in the Collatz conjecture is') print(N)
while result:
if N%2==1:
a=(3*N)+1
step=step+1 print(int(a))
N=a else:
a=N/2 step=step+1 print(int(a))
N=a
if N!=1:
continue else:
print('steps=',step) result=False
```

Output

```
Enter a number of which you want to find the sequence in the Collatz conjecture7
7
The sequence of 7 in the Collatz conjecture is 7
22
11
34
17
```

52
26
13
40
20
10
5
16
8
4
2
1
steps= 16

This program follows the above two algorithms through an ’if’ and ’elif’ statement inside a ’while’ loop, which executes until the number entered by the user reaches 1. A step variable counts the steps a number takes to reach 1.

10.2 Finding the Collatz Sequence of Every Number upto a Number and Counting the Steps for Every Number

Program to Find the Collatz Sequence of Every Number upto a Number and Counting the Steps for Every Number

```
n=input(’Enter a number upto which you want to find the sequence in the Collatz conjecture’)
N=int(n) a=None step=0 result=True
for i in range(1,N+1): step=0  result=True
print(’The sequence of’,i,’in the Collatz conjecture is’) print(i)
while result:
if i□2==1:                    %
a=(3*i)+1 step=step+1 print(int(a)) i=a
else:
a=i/2 step=step+1 print(int(a)) i=a
if i!=1:
```

```
continue else:
print('steps=',step) result=False
```

Output

```
Enter a number upto which you want to find the sequence in the Collatz conjecture10
The sequence of 1 in the Collatz conjecture is 1
4
2
1
steps= 3
The sequence of 2 in the Collatz conjecture is 2
1
steps= 1
The sequence of 3 in the Collatz conjecture is 3
10
5
16
8
4
2
1
steps= 7
The sequence of 4 in the Collatz conjecture is 4
2
1
steps= 2
The sequence of 5 in the Collatz conjecture is 5
16
8
4
2
1
steps= 5
The sequence of 6 in the Collatz conjecture is
```

6
3
10
5
16
8
4
2
1
steps= 8
The sequence of 7 in the Collatz conjecture is 7
22
11
34
17
52
26
13
40
20
10
5
16
8
4
2
1
steps= 16
The sequence of 8 in the Collatz conjecture is 8
4
2
1
steps= 3
The sequence of 9 in the Collatz conjecture is 9
28

14
7
22
11
34
17
52
26
13
40
20
10
5
16
8
4
2
1
steps= 19
The sequence of 10 in the Collatz conjecture is 10
5
16
8
4
2
1
steps= 6

This program asks the user to enter the number upto which the Collatz sequence for every number starting from ’1’ is to be printed. This program runs the above program inside a ’for’ loop to print the Collatz sequence for every number starting from ’1’ upto the user entered number and calculates the steps every number take to reach 1.

11 Prime Factorisation

Any number can be broken down to be the product of prime numbers, this method of breaking down a number into its prime factors is called prime factorisation of a number.

11.1 Finding the Prime Factorisation of a Number

Program to Find Prime Factorisation of a Number

```
n=input('Enter a number of which you want to find the prime factors') N=int(n)
print('The prime factorisation of the number') for a in range(2,(N//2)+1):
if N%a==0:
lst=list() result=True while result:
p=N
for i in range(2,int(p//2)+1): if p%i==0:
break lst.append(i)
N=N/i i=2
if N!=2:
for x in range(2,((int(N)//2)+1)): if int(N)%x==0:
result=True break
else:
result=False if N!=1:
lst.append(int(N)) else:
result=False break
else:


result=True print(int(n),'=',end=' ') print(*lst,sep=' ') break
else:
print(int(n),'=',1,'',N)
```

Output

Enter a number of which you want to find the prime factors966 The prime factorisation of the number 966 = 2 3 7 23

This program asks the user to enter the number of which the prime factori- sation is required. Then it checks

the factors of the number to check if it is prime or not. If it is prime only two factors ’1’ and the number itself are printed. If it is not prime. inside a ’while’ loop its divisibility is checked in a ’for’ loop, if a factor is found the ’for’ loop is broken, the first factor of the number is stored in a list, the number is divided by the first factor, the quotient becomes the number and the above process is continued until the quotient becomes a prime number except ’2’ or the quotient becomes ’1’. If the quotient becomes ’2’ the above process is continued until the quotient becomes ’1’. If the quotient becomes a prime number except ’2’ it is ap- pended to the list of factors and the ’while’ loop is broken. In this way all the prime factors of the number are stored in the list and with the help of ’print’ statements its prime factorisation is printed.

11.2 Finding the Prime Factorisation of Every Num- ber upto a Number Starting from ’1’

Program to Find Prime Factorisation of Every Number upto a Number Starting from ’1’

```
n=input(’Enter a number upto which you want to find the prime factors’) N=int(n)
print(’The prime factorisation of the number’) for y in range(1,N+1):
z=y
for a in range(2,(z//2)+1): if z□a==0:
lst=list() result=True while result:
p=z
for i in range(2,int(p//2)+1): if p%i==0:
break lst.append(i) z=z/i
i=2
if z!=2:
for x in range(2,((int(z)//2)+1)): if int(z)%x==0:
result=True break
else:
result=False if z!=1:
lst.append(int(z)) else:
result=False break
else:
result=True print(y,’=’ ,end=’ ’ )
```

print(z,'=',1,'ₒ',z)

Output

Enter a number upto which you want to find the prime factors200 The prime factorisation of the number

1 = 1ₒ 1

2 = 1ₒ 2

3 = 1ₒ 3

4 = 2ₒ 2

5 = 1ₒ 5

6 = 2ₒ 3

7 = 1ₒ 7

8 = 2ₒ 2 ₒ 2

9 = 3ₒ 3

10 = 2 ₒ 5

11 = 1 ₒ 11

12 = 2 ₒ 2 ₒ 3

13 = 1 ₒ 13

14 = 2 ₒ 7

15 = 3 ₒ 5

16 = 2 ₒ 2 ₒ 2 ₒ 2

17 = 1 ₒ 17

18 = 2 ₒ 3 ₒ 3

19 = 1 ₒ 19

20 = 2 ₒ 2 ₒ 5

21 = 3 ₒ 7

22 = 2 ₒ 11

23 = 1 ₒ 23

24 = 2 ₒ 2 ₒ 2 ₒ 3

25 = 5 ₒ 5

26 = 2 ₒ 13

27 = 3 ₀ 3 ₀ 3

28 = 2 ₀ 2 ₀ 7

29 = 1 ₀ 29

30 = 2 ₀ 3 ₀ 5

31 = 1 ₀ 31

32 = 2 ₀ 2 ₀ 2 ₀ 2 ₀2

33 = 3₀ 11

34 = 2₀ 17

35 = 5 ₀ 7

36 = 2 ₀ 2 ₀ 3 ₀ 3

37 = 1₀ 37

38 = 2₀ 19

39 = 3₀ 13

40 = 2 ₀ 2 ₀ 2 ₀ 5

41 = 1 ₀ 41

42 = 2 ₀ 3 ₀ 7

43 = 1 ₀ 43

44 = 2 ₀ 2 ₀ 11

45 = 3 ₀ 3 ₀ 5

46 = 2₀ 23

47 = 1₀ 47

48 = 2 ₀ 2 ₀ 2 ₀ 2 ₀3

49 = 7 ₀ 7

50 = 2 ₀ 5 ₀ 5

51 = 3 ₀ 17

52 = 2 ₀ 2 ₀ 13

53 = 1 ₀ 53

54 = 2 ₀ 3 ₀ 3 ₀ 3

55 = 5 ₀ 11

56 = 2 ₀ 2 ₀ 2 ₀ 7

57 = 3₀ 19

58 = 2₀ 29

59 = 1₀ 59

60 = 2 ₀ 2 ₀ 3 ₀ 5

61 = 1 ₒ 61

62 = 2 ₒ 31

63 = 3 ₒ 3 ₒ 7

64 = 2 ₒ 2 ₒ 2 ₒ 2 ₒ 2 ₒ2

65 = 5 ₒ 13

66 = 2 ₒ 3 ₒ 11

67 = 1 ₒ 67

68 = 2 ₒ 2 ₒ 17

69 = 3 ₒ 23

70 = 2 ₒ 5 ₒ 7

71 = 1 ₒ 71

72 = 2 ₒ 2 ₒ 2 ₒ 3 ₒ3

73 = 1 ₒ 73

74 = 2 ₒ 37

75 = 3 ₒ 5 ₒ 5

76 = 2 ₒ 2 ₒ 19

77 = 7 ₒ 11

78 = 2 ₒ 3 ₒ 13

79 = 1 ₒ 79

80 = 2 ₒ 2 ₒ 2 ₒ 2 ₒ5

81 = 3 ₒ 3 ₒ 3 ₒ 3

82 = 2 ₒ 41

83 = 1 ₒ 83

84 = 2 ₒ 2 ₒ 3 ₒ 7

85 = 5 ₒ 17

86 = 2 ₒ 43

87 = 3 ₒ 29

88 = 2 ₒ 2 ₒ 2 ₒ 11

89 = 1 ₒ 89

90 = 2 ₒ 3 ₒ 3 ₒ 5

91 = 7 ₒ 13

92 = 2 ₒ 2 ₒ 23

93 = 3 ₒ 31

94 = 2 ₒ 47

95 = 5 ₒ 19

96 = 2 ₒ 2 ₒ 2 ₒ 2 ₒ 2 ₒ3

97 = 1 ₒ 97

98 = 2 ₒ 7 ₒ 7

99 = 3 ₒ 3 ₒ 11

100 = 2 ₒ 2 ₒ 5 ₒ 5

101 = 1 ₒ 101

102 = 2 ₒ 3 ₒ 17

103 = 1 ₒ 103

104 = 2 ₒ 2 ₒ 2 ₒ 13

105 = 3 ₒ 5 ₒ 7

106 = 2 ₒ 53

107 = 1 ₒ 107

108 = 2 ₒ 2 ₒ 3 ₒ 3 ₒ 3

109 = 1 ₒ 109

110 = 2 ₒ 5 ₒ 11

111 = 3 ₒ 37

112 = 2 ₒ 2 ₒ 2 ₒ 2 ₒ 7

113 = 1 ₒ 113

114 = 2 ₒ 3 ₒ 19

115 = 5 ₒ 23

116 = 2 ₒ 2 ₒ 29

117 = 3 ₒ 3 ₒ 13

118 = 2 ₒ 59

119 = 7 ₒ 17

120 = 2 ₒ 2 ₒ 2 ₒ 3 ₒ 5

121 = 11 ₒ 11

122 = 2 ₒ 61

123 = 3 ₒ 41

124 = 2 ₒ 2 ₒ 31

125 = 5 ₒ 5 ₒ 5

126 = 2 ₒ 3 ₒ 3 ₒ 7

127 = 1 ₒ 127

128 = 2 ₒ 2 ₒ 2 ₒ 2 ₒ 2 ₒ 2 ₒ2

129 = 3 ₒ 43

130 = 2 ₒ 5 ₒ 13

131 = 1 ₒ 131

132 = 2 ₒ 2 ₒ 3 ₒ 11

133 = 7 ₒ 19

134 = 2 ₒ 67

135 = 3 ₒ 3 ₒ 3 ₒ 5

136 = 2 ₒ 2 ₒ 2 ₒ 17

137 = 1 ₒ 137

138 = 2 ₒ 3 ₒ 23

139 = 1 ₒ 139

140 = 2 ₒ 2 ₒ 5 ₒ 7

141 = 3 ₒ 47

142 = 2 ₒ 71

143 = 11 ₒ 13

144 = 2 ₒ 2 ₒ 2 ₒ 2 ₒ 3 ₒ3

145 = 5 ₒ 29

146 = 2 ₒ 73

147 = 3 ₒ 7 ₒ 7

148 = 2 ₒ 2 ₒ 37

149 = 1 ₒ 149

150 = 2 ₒ 3 ₒ 5 ₒ 5

151 = 1 ₒ 151

152 = 2 ₒ 2 ₒ 2 ₒ 19

153 = 3 ₒ 3 ₒ 17

154 = 2 ₒ 7 ₒ 11

155 = 5 ₒ 31

156 = 2 ₒ 2 ₒ 3 ₒ 13

157 = 1 ₒ 157

158 = 2 ₒ 79

159 = 3 ₒ 53

160 = 2 ₒ 2 ₒ 2 ₒ 2 ₒ 2 ₒ5

161 = 7 ₒ 23

162 = 2 ₒ 3 ₒ 3 ₒ 3 ₒ 3

163 = 1 ₒ 163

164 = 2 ₒ 2 ₒ 41

165 = 3 ₒ 5 ₒ 11

166 = 2 ₀ 83

167 = 1 ₀ 167

168 = 2 ₀ 2 ₀ 2 ₀ 3 ₀ 7

169 = 13 ₀ 13

170 = 2 ₀ 5 ₀ 17

171 = 3 ₀ 3 ₀ 19

172 = 2 ₀ 2 ₀ 43

173 = 1 ₀ 173

174 = 2 ₀ 3 ₀ 29

175 = 5 ₀ 5 ₀ 7

176 = 2 ₀ 2 ₀ 2 ₀ 2 ₀ 11

177 = 3 ₀ 59

178 = 2 ₀ 89

179 = 1 ₀ 179

180 = 2 ₀ 2 ₀ 3 ₀ 3 ₀ 5

181 = 1 ₀ 181

182 = 2 ₀ 7 ₀ 13

183 = 3 ₀ 61

184 = 2 ₀ 2 ₀ 2 ₀ 23

185 = 5 ₀ 37

186 = 2 ₀ 3 ₀ 31

187 = 11 ₀ 17

188 = 2 ₀ 2 ₀ 47

189 = 3 ₀ 3 ₀ 3 ₀ 7

190 = 2 ₀ 5 ₀ 19

191 = 1 ₀ 191

192 = 2 ₀ 2 ₀ 2 ₀ 2 ₀ 2 ₀ 2 ₀3

193 = 1 ₀ 193

194 = 2 ₀ 97

195 = 3 ₀ 5 ₀ 13

196 = 2 ₀ 2 ₀ 7 ₀ 7

197 = 1 ₀ 197

198 = 2 ₀ 3 ₀ 3 ₀ 11

199 = 1 ₀ 199

200 = 2 ₀ 2 ₀ 2 ₀ 5 ₀ 5

In this program the above program is executed inside a 'for' loop which breaks down every number starting from '1' upto the user entered number to its prime factors and with the help of 'print' statements prints them.

11.3 Finding the Prime Factorisation of Every Num- ber upto a Number Starting from a User Entered Number

Program to Find Prime Factorisation of Every Number upto a Number Starting from a User Entered Number

```
n=input('Enter a number upto which you want to find the prime factors') m=input('Enter the number from which you want to find the primefactors') N=int(n)
M=int(m) if N<M:
print('The prime factorisation can not be found, please enter the numbers correctly')
else:
print('The prime factorisation of the number') for y in range(M,N+1):
z=y
for a in range(2,(z//2)+1): if z%a==0:
lst=list() result=True while result:
p=z
for i in range(2,int(p//2)+1): if p□i==0:
break lst.append(i) z=z/i                                    %
i=2
if z!=2:
for x in range(2,((int(z)//2)+1)): if int(z)%x==0:
result=True break
else:
result=False if z!=1:
```

```
                    lst.append(int(z)) else:
                    result=False break

                else:

            else:
          esult=True
    print(y,'=',end='')
    print(*lst,sep='ه')
            break
print(z,'=',1,'ه',z)
```

Output

Enter a number upto which you want to find the prime factors400 Enter the number from which you want to find the primefactors301 The prime factorisation of the number

301 = 7 ₀ 43

302 = 2 ₀ 151

303 = 3 ₀ 101

304 = 2 ₀ 2 ₀ 2 ₀ 2 ₀ 19

305 = 5 ₀ 61

306 = 2 ₀ 3 ₀ 3 ₀ 17

307 = 1 ₀ 307

308 = 2 ₀ 2 ₀ 7 ₀ 11

309 = 3 ₀ 103

310 = 2 ₀ 5 ₀ 31

311 = 1 ₀ 311

312 = 2 ₀ 2 ₀ 2 ₀ 3 ₀ 13

313 = 1 ₀ 313

314 = 2 ₀ 157

315 = 3 ₀ 3 ₀ 5 ₀ 7

316 = 2 ₀ 2 ₀ 79

317 = 1 ₀ 317

318 = 2 ₀ 3 ₀ 53

319 = 11 ₀ 29

320 = 2 ₀ 2 ₀ 2 ₀ 2 ₀ 2 ₀ 2 ₀5

321 = 3 ₀ 107

322 = 2 ₀ 7 ₀ 23

323 = 17 ₀ 19

324 = 2 ₀ 2 ₀ 3 ₀ 3 ₀ 3 ₀3

325 = 5 ₀ 5 ₀ 13

326 = 2 ₀ 163

327 = 3 ₀ 109

328 = 2 ₀ 2 ₀ 2 ₀ 41

329 = 7 ₀ 47

330 = 2 ₀ 3 ₀ 5 ₀ 11

331 = 1 ₀ 331

332 = 2 ₀ 2 ₀ 83

333 = 3 ₒ 3 ₒ 37

334 = 2 ₒ 167

335 = 5 ₒ 67

336 = 2 ₒ 2 ₒ 2 ₒ 2 ₒ 3 ₒ7

337 = 1 ₒ 337

338 = 2 ₒ 13 ₒ 13

339 = 3 ₒ 113

340 = 2 ₒ 2 ₒ 5 ₒ 17

341 = 11 ₒ 31

342 = 2 ₒ 3 ₒ 3 ₒ 19

343 = 7 ₒ 7 ₒ 7

344 = 2 ₒ 2 ₒ 2 ₒ 43

345 = 3 ₒ 5 ₒ 23

346 = 2 ₒ 173

347 = 1 ₒ 347

348 = 2 ₒ 2 ₒ 3 ₒ 29

349 = 1 ₒ 349

350 = 2 ₒ 5 ₒ 5 ₒ 7

351 = 3 ₒ 3 ₒ 3 ₒ 13

352 = 2 ₒ 2 ₒ 2 ₒ 2 ₒ 2 ₒ11

353 = 1 ₒ 353

354 = 2 ₒ 3 ₒ 59

355 = 5 ₒ 71

356 = 2 ₒ 2 ₒ 89

357 = 3 ₒ 7 ₒ 17

358 = 2 ₒ 179

359 = 1 ₒ 359

360 = 2 ₒ 2 ₒ 2 ₒ 3 ₒ 3 ₒ5

361 = 19 ₒ 19

362 = 2 ₒ 181

363 = 3 ₒ 11 ₒ 11

364 = 2 ₒ 2 ₒ 7 ₒ 13

365 = 5 ₒ 73

366 = 2 ₒ 3 ₒ 61

367 = 1 ₒ 367

368 = 2 ₒ 2 ₒ 2 ₒ 2 ₒ 23

369 = 3 ₒ 3 ₒ 41

370 = 2 ₒ 5 ₒ 37

371 = 7 ₒ 53

372 = 2 ₒ 2 ₒ 3 ₒ 31

373 = 1 ₒ 373

374 = 2 ₒ 11 ₒ 17

375 = 3 ₒ 5 ₒ 5 ₒ 5

376 = 2 ₒ 2 ₒ 2 ₒ 47

377 = 13 ₒ 29

378 = 2 ₒ 3 ₒ 3 ₒ 3 ₒ 7

379 = 1 ₒ 379

380 = 2 ₒ 2 ₒ 5 ₒ 19

381 = 3 ₒ 127

382 = 2 ₒ 191

383 = 1 ₒ 383

384 = 2 ₒ 2 ₒ 2 ₒ 2 ₒ 2 ₒ 2 ₒ 2 ₒ 3

385 = 5 ₒ 7 ₒ 11

386 = 2 ₒ 193

387 = 3 ₒ 3 ₒ 43

388 = 2 ₒ 2 ₒ 97

389 = 1 ₒ 389

390 = 2 ₒ 3 ₒ 5 ₒ 13

391 = 17 ₒ 23

392 = 2 ₒ 2 ₒ 2 ₒ 7 ₒ 7

393 = 3 ₒ 131

394 = 2 ₒ 197

395 = 5 ₒ 79

396 = 2 ₒ 2 ₒ 3 ₒ 3 ₒ 11

397 = 1 ₒ 397

398 = 2 ₒ 199

399 = 3 ₒ 7 ₒ 19

400 = 2 ₒ 2 ₒ 2 ₒ 2 ₒ 5 ₒ5

This program is similar to the above program, with the exception of also asking the user to enter the number from which the prime factorisation is required. Then it prints the prime factorisation of every number from that user entered number upto the user entered number.

12 Persistence of Numbers

Persistence is the number of times an arithmetic operation can be performed on an integer until the integer reaches a point where the operation can no longer be performed on the integer. Persistence is of two types:-

Multiplicative Persistence The number of times multiplicationcan be done on the digits of an integer until it reaches a single digit.

Additive Persistence The number of times addition can be doneon the digits of an integer until it reaches a single digit. (*Persistence of a number - Academic Kids*, n.d.)

12.1 Finding the Multiplicative Persistence of a Num- ber

Program to Find Multiplicative Persistence of a Number

```
n=input('Enter a number') product=1
steps=0 result=1
print('The products are') if len(n)!=1:
for i in range(0,len(n)): product=product*int(n[i])
print(product) steps=steps+1
while len(str(product))!=1: a=str(product)
for x in range(0,len(a)): product=result*int(a[x]) result=product
print(product) result=1 steps=steps+1
print('steps=',steps) else:
print(int(n)) print('steps=',0)
```

Output

Enter a number2777 The products are 686
288
128
16
6
steps= 5

First the user is asked to enter a number. Using a 'for' loop and indexing the product of all digits of the number is found. A 'while' loop executes the same process on the above product until the products of the digits reach a single digit. A step variable is used to count the steps to reach single digit.

12.2 Finding the Additive Persistence of a Number

Program to Find Additive Persistence of a Number

```
n=input('Enter a number') addition=0
steps=0 result=0
print('The additions are') if len(n)!=1:
for i in range(0,len(n)): addition=addition+int(n[i])
print(addition) steps=steps+1
while len(str(addition))!=1: a=str(addition)
for x in range(0,len(a)): addition=result+int(a[x]) result=addition
print(addition) result=0 steps=steps+1
print('steps=',steps) else:
print(int(n)) print('steps=',0)
```

Output

Enter a number1386 The additions are 18
9
steps= 2

First the user is asked to enter a number. Using a 'for' loop and indexing the sum of all digits of the number is found. A 'while' loop executes the same process on the above sum until the sum of the digits reach a single digit. A step variable is used to count the steps to reach single digit.

13 Factorial of a Number

Factorial of a positive integer is defined as the product of the number with all the numbers smaller than it.

13.1 Finding the Factorial of a Number

Program to Find Factorial of a Number

```
n=input('Enter the number of which you want to find the factorial') i=int(n)
value=i
if int(n)<0:
print('Factorial domain is defined for positive whole numbersonly') elif int(n)==1:
print('The factorial of the number',1,'is',1) elif int(n)==0:
print('The factorial of the number',0,'is',1) else:
while i>0:
value=value*(i-1) i=i-1
if i==1:
break
print('The factorial of the number',int(n),'is',value)
```

Output

Enter the number of which you want to find the factorial10 The factorial of the number 10 is 3628800

The user is asked to enter a number. The program checks if the number is non negative. The factorial of '0' and '1' is 1, is considered the definition statements for this program. If the user enters a number which is not '0' or '1', the program uses a 'while' loop to multiply the number with all the numbers coming before it. The 'while' loop executes until it reaches '1'. Finally the factorial of the number is printed.

13.2 Finding the Factorial of Every Number upto a Number

Program to Find Factorial of Every Number upto a Number

```
n=input('Enter the number upto which you want to find the factorial') if int(n)==0:
print('The factorial of the number',0,'is',1) elif int(n)==1:
print('The factorial of the number',0,'is',1) print('The factorial of the number',1,'is',1)
else:
print('The factorial of the number',0,'is',1) print('The factorial of the number',1,'is',1) for i in range(2,(int(n)+1)):
x=i value=x while x>0:
value=value*(x-1) x=x-1
if x==1:
break
print('The factorial of the number',i,'is',value)
```

Output

```
Enter the number upto which you want to find the factorial10
    The factorial of the number 0 is 1
    The factorial of the number 1 is 1
    The factorial of the number 2 is 2
    The factorial of the number 3 is 6
    The factorial of the number 4 is 24
    The factorial of the number 5 is 120
    The factorial of the number 6 is 720
    The factorial of the number 7 is 5040
    The factorial of the number 8 is 40320
    The factorial of the number 9 is 362880
The factorial of the number 10 is 3628800
```

This program runs the above program in a 'for' loop. The 'for' loop iterates from '0' to the user entered number, thus printing the factorial of every number from '0' to the user entered number.

14 Finding if a Year is Leap Year or Not

Program to Find if a Year is Leap Year or Not

```
Year=input('Enter a year') yr=int(Year)
if yr %4==0 and yr %100!=0: print(yr,'is a leap year')
elif yr %4==0 and yr %100==0: if yr%400==0:
print(yr,'is a leap year') else:
print(yr,'is not a leap year')
else:
print(yr,'is not a leap year')
```

Output

```
Enter a year1700
1700 is not a leap year
```

For a year to be a leap year some conditions are there:-

1 If the year is divisible by 4 and not divisible by 100, then the year is a leap year.

2 If a year is divisible by 4 and it is also divisible by 100, for it to be a leap year, it must also be divisible by 400.

3 If a year is divisible by 4 and 100 and it is not divisible by 400, it is not a leap year.

4 If a year is not divisible by 4, it is not a leap year. (*Leap Years*, n.d.)

This program asks the user to enter the year and converts it to an integer. Then using 'if', 'elif' and 'else' statements the program checks the above conditions and prints if the year the user entered is leap year or not.

15 Conclusion

Computers being excellent calculators are able to perform large mathematical tasks with ease. This is achieved with the help of flawless program that can be run to get the desired output. This project is aimed at, writing python programs that can be used to perform mathematical tasks with large size of sample. These programs are designed to be understood by both a beginner and an expert. All of the programs were written in several steps. First the user performs the mathematics manually on a small sample several times to prepare the set of instructions, the instructions were written in the form of python programs using simple fundamental concepts such as 'input' functions, logical statements like 'if', 'else', 'elif', data structures namely lists, dictionaries and tuples, arithmetic, assignment and comparison operators and other predefined functions, then the program is visualised with the help of the online python program visualisation website with several inputs to make sure that the desired output is obtained every time.

In this project it becomes clear that to get the computer to perform any mathematical task it is important to understand the concepts of mathematics and apply it to a small sample to prepare a set of instructions. After prepar- ing the set of instructions it is necessary to have a complete understanding of the fundamental concepts of the programming language to convert the set of instructions to a program, so that it can be used to make the computer perform the mathematics on a large sample.

References

Binary to decimal - conversion, formula, conver- sion chart, examples. (n.d.). Retrieved from https://www.cuemath.com/numbers/binary-to-decimal/

Binary to hexadecimal - definition, conversion steps, conver- sion with decimal point, examples. (n.d.). Retrieved from https://www.cuemath.com/numbers/binary-to-hexadecimal/

Binary to octal conversion - methods, definition, rules, ex- amples & practice questions. (n.d.). Retrieved from https://www.cuemath.com/numbers/binary-to-octal-conversion/

Das, A. (2022, 01). *The Collatz Conjecture: Beauty or Conundrum of Mathematics?* Retrieved from https://www.cantorsparadise.com/the-collatz-conjecture- beauty-or-conundrum-of-mathematics-6589e45babf7

Hexadecimal to binary - meaning, conversion ta- ble, examples, faqs. (n.d.). Retrieved from https://www.cuemath.com/numbers/hexadecimal-to-binary/

Leap Years. (n.d.). Retrieved from https://www.mathsisfun.com/leap-years.html

Mersenne primes: History, theorems and lists. (n.d.). Retrieved from https://primes.utm.edu/mersenne/

Octal to binary - conversion, table, examples. (n.d.). Retrieved from https://www.cuemath.com/numbers/octal-to-binary/

Online python compiler (interpreter). (n.d.). Retrieved from https://www.programiz.com/python-programming/online-compiler/

Persistence of a number - academic kids. (n.d.). Retrieved from https://academickids.com/encyclopedia/index.php/Persistence_of_a_number

Python program to find hcf or gcd. (n.d.). Retrieved from https://www.programiz.com/python-programming/examples/hcf

Python program to find lcm. (n.d.). Retrieved from https://www.programiz.com/python-programming/examples/lcm

Python split string into specific length chunks - python examples. (n.d.). Retrieved from https://pythonexamples.org/python-split-string-into-specific-length-chunks/

Python tutor: Learn python, javascript, c, c++, and java by visualizing code. (n.d.). Retrieved from https://pythontutor.com/

Severance, C. R. (2020). *Python for everybody: Exploring data in python 3.*

van Rossum, G., et al. (2018). Python tutorial release 3.7. 0. *Python Software Foundation*.