# Some mathematical logics

**Netural number:** A **natural number** is a positive integer.depending on context **natural number**:

**Without 0**: natural numbers = {1, 2, 3, ...}
**With 0**: natural numbers = {0, 1, 2, 3, ...}

**Even number:** An **even number** is any integer that is exactly divisible by 2. In other words, when you divide an even number by 2, there is no remainder. An even number can be expressed as **2 × n**, where **n** is any integer.

n mod 2 = 0

**Odd number:** An **odd number** is an integer that **cannot be evenly divided by 2**. When divided by 2, it leaves a remainder of 1.

n mod 2 = 1

**Prime number:** A **prime number** is a natural number greater than 1 that has **exactly two distinct positive divisors**. 1 and itself.

**Key Properties:**

- It **cannot** be formed by multiplying two smaller natural numbers (except 1 and itself).
- The number **1 is *not* prime**.
- The **smallest prime number is 2**, which is also the **only even prime**.

**Factor of given number:** A **factor** (or divisor) of a number is an integer that **divides the number exactly** (with no remainder).

**For example:**

**Factors of 12** are:

- 1, 2, 3, 4, 6, 12
  (because all of these divide 12 without leaving a remainder)

**How to find factors of a number:**

1. Start from 1 and go up to the number itself.
2. Check which numbers divide the given number exactly (i.e., number % i == 0).

**Factorial :** A **factorial** (denoted by n!) is the product of all positive integers from 1 to n.

**Definition:**

$n! = n \times (n-1) \times (n-2) \times \ldots \times 1$

**Special case:**
0!=1(by definition)
**Examples:**

- 1!= 1
- 2!=2×1=2
- 3!=3×2×1=6
- 5!=5×4×3×2×1=120

**Leap year:** A **leap year** is a year that has **366 days** instead of the usual 365. The extra day is added to **February**, making it **29 days long** instead of 28.

**Rules to determine a leap year:**
A year is a leap year if:

- It is **divisible by 4**,
  **but not divisible by 100**,
  **unless it is also divisible by 400**.

**In short:**

- **Leap year:** 2000, 2016, 2020, 2024
- **Not a leap year:** 1900, 2100 (divisible by 100 but not by 400)

**Examples:**

- **2024** is a leap year → divisible by 4, not by 100
- **1900** is **not** a leap year → divisible by 100, but not by 400
- **2000** is a leap year → divisible by 400

| **Leep year-Concept** | | | | |
|---|---|---|---|---|
| **Year** | **Day** | **hour** | **minutes** | **seconds** |
| 1-Year | 365 | 5 | 48 | 47.5 |
| 2-Year | 365 | 5 | 48 | 47.5 |
| 3-Year | 365 | 5 | 48 | 47.5 |
| 4-Year | 365 | 5 | 48 | 47.5 |
| | **Remain times** | 20 | 192 | 190 |
| | | | 195/60=3h,15m | 190/6= 3min,10sec |
| | Approx 1-day which is added in every 4years | **23** | **15** | **10** |
| | | Add some time | | |
| leep year | 366 | | | |
| | | | | |

| | | | | |
|---|---|---|---|---|
| | negative time in leep year(-) | extra added time | 44 | 50 |
| | | | | |
| 100 year | | | 44*25=1100 | 50*25=1250 |
| | | | | 1250/60 |
| | | | 1120/60 | 20 minutes, 50 sec |
| | | | 18hours,40 minutes | |
| | | 18 | 40 | 50 |
| 100-years | 1-day remove in feb month | Not a Leep year | | |
| | | 5 | 19 | 10 |
| | | | | |
| | **( approx 6hour behind in 100th year)** | **5** | **19** | **10** |
| | **( approx 6hour behind in 100th year)** | **5** | **19** | **10** |
| | **( approx 6hour behind in 100th year)** | **5** | **19** | **10** |
| | | **20** | **76** | **40** |
| **400 years** | **Now add 1-day in 400th year** | **21** | **16** | **40** |
| | | **That's why it is leep year** | | |
| | | **2** | **44** | **20** |

**<u>Armstrongs:</u>** An **Armstrong number** (also known as a **narcissistic number**) is a number that is **equal to the sum of its own digits each raised to the power of the number of digits**.

**General Rule:**
**For an *n*-digit number:**
**Armstrong number⇒abcd…=$a^{number\_of\_digit}+b^{number\_of\_digit}+c^{number\_of\_digit}+d^{number\_of\_digit}+\ldots$ Examples:**

- **153**
  It's a 3-digit number:

13+53+33=1+125+27=1531^3 + 5^3 + 3^3 = 1 + 125 + 27 = 15313+53+33=1+125+27=153

- **9474**
  It's a 4-digit number:
  94+44+74+44=6561+256+2401+256=94749^4 + 4^4 + 7^4 + 4^4 = 6561 + 256 + 2401 + 256 = 947494+44+74+44=6561+256+2401+256=9474
- **370, 371, 407** are also 3-digit Armstrong numbers.

**Palindrome:** A **palindrome** is a number, word, phrase, or sequence that **reads the same forward and backward**.

**For numbers:**

A **palindromic number** stays the same when its digits are reversed.

- **121** → reversed is 121
- **1331** → reversed is 1331
- **123** → reversed is 321 (not a palindrome)

**For words:**

- **"madam"**, **"racecar"**, **"level"** are all palindromes.

For phrases (ignoring spaces and punctuation):

- **"A man, a plan, a canal, Panama"**

**LCM:** **LCM** stands for **Least Common Multiple** — the **smallest multiple** that two or more numbers **share in common**.

**Example:**

Find the LCM of **4** and **6**:

- Multiples of 4: 4, 8, 12, 16, ...
- Multiples of 6: 6, 12, 18, 24, ...
- **LCM = 12**

**How to find LCM:**

1. **Listing multiples** (as above) — good for small numbers.
2. **Prime factorization** — multiply highest powers of all primes involved.
3. **Using formula**:
   LCM(a,b)=|a×b| / GCD(a,b)

**HCF:** **HCF** stands for **Highest Common Factor**, also known as the **Greatest Common Divisor (GCD)**. It is the largest number that divides two or more numbers exactly (without leaving a remainder).

**Example:**

Find the HCF of **12** and **15**:

- Factors of 12: 1, 2, 3, 4, 6, 12
- Factors of 15: 1, 3, 5, 15
- **HCF = 3**

**How to find HCF:**

1. **Listing common factors**: Find the common factors of the numbers and pick the largest.
2. **Prime factorization**: Find the prime factorization of both numbers, then multiply the smallest powers of the common prime factors.
3. **Using the formula**:
   HCF(a,b)=|a×b| / LCM(a,b)

**Fibonacci:--**

The **Fibonacci series** is a sequence of numbers in which each number is the sum of the two preceding ones. It starts like this:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

**Harshad Numbers:**

A **Harshad number** (also known as a **Niven number**) is a number that is **divisible by the sum of its digits**. The term **Harshad** is derived from the Sanskrit words "Har" (joy) and "Shad" (give), meaning "giver of joy.

- **18** → Sum of digits = **1 + 8 = 9 → 18 is divisible by 9**
- **21** → Sum of digits = **2 + 1 = 3 → 21 is divisible by 3**
- **19** → Sum of digits = **1 + 9 = 10 → 19 is NOT divisible by 10**

**Anagrams Number:**

An **anagram** is a word or phrase formed by rearranging the letters of another word or phrase, using all the original letters exactly once

- **"listen" → "silent"**
- **"race" → "care"**
- **"evil" → "vile"**
- **"dormitory" → "dirty room"** (ignoring spaces)

**Neon Number:**

A **Neon Number** is a number where the sum of the digits of its square is equal to the original number.

- **9** → Square = **9 × 9 = 81**
    - Sum of digits of 81 → **8 + 1 = 9**
- **12** → Square = **12 × 12 = 144**
    - Sum of digits of 144 → **1 + 4 + 4 = 9**

**Peterson Numbers:**

A **Peterson number** is a number where the sum of the factorials of its digits equals the number itself.

1. **145**
    - Digits: **1, 4, 5**
    - Factorial Sum: **1! + 4! + 5! = 1 + 24 + 120 = 145**
2. **Other Peterson Numbers**: **1, 2, 145** (There are very few!)

**Spy Numbers**

A **Spy Number** is a number where the **sum of its digits** is equal to the **product of its digits**

1. **112**
    - Digits: **1, 1, 2**
    - Sum = **1 + 1 + 2 = 4**
    - Product = **1 × 1 × 2 = 4**
2. **123**
    - Digits: **1, 2, 3**
    - Sum = **1 + 2 + 3 = 6**
    - Product = **1 × 2 × 3 = 6**

**Sunny number**

A *sunny number* is a number that is one less than a perfect square. In other words, a number N is sunny if there exists an integer n such that:

$N+1=n^2$

For example:

- 3 is a sunny number because 3+1=4, and 4 is a perfect square (since 2^2=4).
- 8 is another sunny number because 8+1=9, and 9 is a perfect square (since 3^2 = 9)

# Control Flow Statements

In programming languages, flow control means the order in which the statements or instructions, that we write, get executed. In order to understand a program, we should be aware of what statements are being executed and in which order. So, understanding the flow of the program is very important. There are, generally, three ways in which the statements will be executed. They are,

1. **Sequential**
2. **Conditional**
3. **Looping**

**Sequential:** In this type of execution flow, the statements are executed one after the other sequentially. By using sequential statements, we can develop simple programs

**Example: Sequential statements:-**

```
print("Welcome")
print("to")
print("python class")

O/P:-
Welcome
to
python class
```

**Conditional:** Statements are executed based on the condition. As shown above in the flow graph, if the condition is true then one set of statements are executed, and if false then the other set. Conditional statements are used much in complex programs.

Conditional statements are also called decision-making statements. Let's discuss some conditions making statements in detail. There are three types of conditional statements in python. They are as follows:

1. **if statement**
2. **if-else statement**
3. **nested-if (if-elif-elif-else)**

**if-statement:-**
**syntax:-**

```
if condition:
    print("Block statement")
print("Out of block statement")
```

**Example:-**

```
num=int(input("Enter any no: "))
if num>=18:
    print("if block statment executed")
print("out of if block statements")

O/P:
Enter any no: 10
out of if block statements
PS E:\DataSciencePythonBatch> python control.py
Enter any no: 18
if block statment executed
out of if block statements
```

**Example:---**

```
# Example:---Checking if a number is positive, negative, or zero.

num = float(input("Enter a number: "))
if num > 0:
    print("The number is positive.")
elif num < 0:

    print("The number is negative.")
else:
```

```
    print("The number is zero.")
```

**if-else condition:-**
**syntax:-**

```
if condition:
    print("if block statement executed")
else:
    print("else block statement executed ")
```

**Example:---**

```
num=int(input("Enter any no: "))
if num>=18:
    print("if block statment executed")
else:
    print("else block statement executed")

O/P:-
PS E:\DataSciencePythonBatch> python control.py
Enter any no: 18
if block statment executed
PS E:\DataSciencePythonBatch> python control.py
Enter any no: 15
else block statement executed
```

**Example:---**

```
# Example:---Find grater no.

x=int(input("Enter first no."))
y=int(input("Enter second no."))
z=int(input("Enter third no."))
if x>y:
    if x>z:
        print("Greater ni is(x): ",x)
    else:
```

```
        print("Greater no is(z): ",z)
else:
    if y>z:
        print("Greater ni is(y): ",y)
    else:
        print("Greater no is(z): ",z)

O/P:-

Enter first no.10
Enter second no.10
Enter third no.20
Greater no is(z):  20
```

**Example:---**

```
# Example:-- Checking if a person is eligible to vote

age = int(input("Enter your age: "))
if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")

O/P:-
Enter your age: 35
You are eligible to vote.
```

**Example:---**

```
# Example:-- Checking if a year is a leap year
year = int(input("Enter a year: "))
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0 and year %100==0):
    print("It's a leap year.")
else:
    print("It's not a leap year.")

O/P:-
```

```
Enter a year: 2000
It's a leap year.
```

**Nested-If else:--**

```
# Example:-- Check your gread based on your own score

score = int(input("Enter your score: "))

if score >= 90:
    print("You got an A.")
else:
    if score >= 80:
        print("You got a B.")
    else:
        if score >= 70:
            print("You got a C.")
        else:
            if score >= 60:
                print("You got a D.")
            else:
                print("You got an F.")

O/P:-
Enter your score: 90
You got an A.
```

**Example:--**

```
# Example:-- Check given year is leep year or not.

year = int(input("Enter a year: "))
if year % 4 == 0:
    if year % 100 == 0:
        if year % 400 == 0:
            print("Leap year")
        else:
            print("Not a leap year")
    else:
```

```
        print("Leap year")
else:
    print("Not a leap year")
```

**if elif else statement in python:**
**Syntex:**

```
if (condition1):
    statement of if Block
elif(condition2):
    statment of elif Block
elif(condition3):
    statement if elif block
else:
    ststement of else block
```

Example:---

```
# example:-- Please choose value within range of o to 4.

print("Please enter the values from 0 to 4")
x=int(input("Enter a number: "))
if x==0:
        print("You entered:", x)
elif x==1:
        print("You entered:", x)
elif x==2:
        print("You entered:", x)
elif x==3:
        print("You entered:", x)
elif x==4:
        print("You entered:", x)
else:
        print("Beyond the range than specified")

O/P:-
Enter a number: 5
```

```
Beyond the range than specified
PS E:\DataSciencePythonBatch> python control.py
Please enter the values from 0 to 4
Enter a number: 4
You entered: 4
```

**Example:- Python Program to calculate the square root.**

```
# Example:-Python

num = float(input('Enter a number: '))
num_sqrt = num ** 0.5
print('The square root of Num :', num_sqrt)

O/P:-
Enter a number: 4
The square root of 4.000 is 2.000
PS E:\DataSciencePythonBatch> python control.py
Enter a number: 8
The square root of 8.000 is 2.828
```

**Example:-- Python Program to find the area of triangle.**

```
# Python Program to find the area of triangle
# s = (a+b+c)/2
# area = √(s(s-a)*(s-b)*(s-c))
a = float(input('Enter first side: '))
b = float(input('Enter second side: '))
c = float(input('Enter third side: '))
s = (a + b + c) / 2
area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
print('The area of the triangle is :', area)

O/P:---
Enter first side: 5
Enter second side: 6
Enter third side: 7
The area of the triangle is : 14.696938456699069
```

**Example:-- Python program to swap two variables.**

```
# Python program to swap two variables
x = input('Enter value of x: ')
y = input('Enter value of y: ')

# create a temporary variable and swap the values
temp = x
x = y
y = temp
print('The value of x after swapping: {}'.format(x))
print('The value of y after swapping: {}'.format(y))

O/P:---
Enter value of x: 5
Enter value of y: 8
The value of x after swapping: 8
The value of y after swapping: 5


# without using third variable
x = input('Enter value of x: ')
y = input('Enter value of y: ')
x, y = y, x
print('The value of x after swapping: {}'.format(x))
print('The value of y after swapping: {}'.format(y))

O/P:---
Enter value of x: 4
Enter value of y: 6
The value of x after swapping: 6
The value of y after swapping: 4

# By-using Addition and Subtraction.
x = int(input('Enter value of x: '))
y = int(input('Enter value of y: '))
x = x + y
y = x - y
x = x - y
```

```
print('The value of x after swapping: {}'.format(x))
print('The value of y after swapping: {}'.format(y))

O/P:---
Enter value of x: 4
Enter value of y: 6
The value of x after swapping: 6
The value of y after swapping: 4
# By-using Multiplication and division.
x = int(input('Enter value of x: '))
y = int(input('Enter value of y: '))
x = x * y
y = x / y
x = x / y
print('The value of x after swapping: {}'.format(x))
print('The value of y after swapping: {}'.format(y))

O/P:---
Enter value of x: 2
Enter value of y: 5
The value of x after swapping: 5.0
The value of y after swapping: 2.0


# By-using x-or(^) operator.
x = int(input('Enter value of x: '))
y = int(input('Enter value of y: '))
x = x ^ y
y = x ^ y
x = x ^ y
print('The value of x after swapping: {}'.format(x))
print('The value of y after swapping: {}'.format(y))

O/P:--
Enter value of x: 10
Enter value of y: 20
The value of x after swapping: 20
The value of y after swapping: 10
```

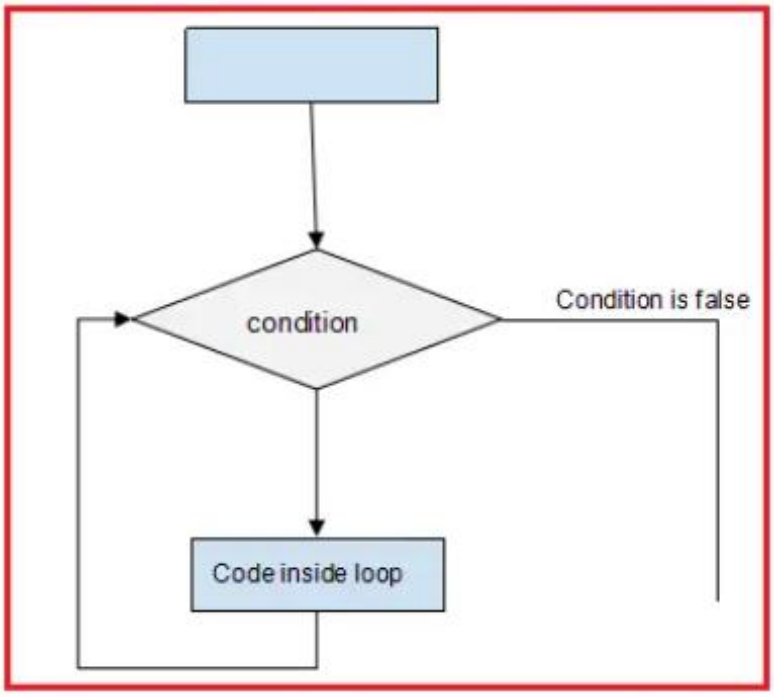
## ----:LOOPING Statement in Python (Iterations):----

If we want to execute a group of statements multiple times, then we should go for a looping kind of execution. There are two types of loops available in python.
They are:

1. **while loop**
2. **for loop**

**1. while loop:-** The while loop contains an expression/condition. As per the syntax colon (:) is mandatory otherwise it throws a syntax error. The condition gives the result as bool type, either True or False. The loop keeps on executing the statements until the condition becomes False. i.e. With the while loop we can execute a set of statements as long as a condition is true.

Flowchart for the while-loop:




**Parts of while loop in Python:**
**Initialization:**

This is the first part of the while loop. Before entering the condition section, some initialization is required.

**Condition:**
Once the initializations are done, then it will go for condition checking which is the heart of the while loop. The condition is checked and if it returns True, then execution enters the loop for executing the statements inside.

After executing the statements, the execution goes to the increment/decrement section to increment the iterator. Mostly, the condition will be based on this iterator value, which keeps on changing for each iteration. This completes the first iteration. In the second iteration, if the condition is False, then the execution comes out of the loop else it will proceed as explained in the above point.

**Increment/Decrement section:** This is the section where the iterator is increased or decreased. Generally, we use arithmetic operators in the loop for this section.

**Example: Printing numbers from 1 to 5 by using while loop**

1. The program is to print the number from 1 to 5
2. Before starting the loop, we have made some assignments( x = 1). This is called the Initialization section.
3. After initialization, we started the while loop with a condition x<=5. This condition returns True until x is less than 5.
4. Inside the loop, we are printing the value of x.
5. After printing the x value, we are incrementing it using the operator x+=1. This is called the increment/decrement section.
6. For each iteration, the value of x will increase and when the x value reaches 6, then the condition x<=5 returns False. At this iteration, the execution comes out of the loop without executing the statements inside. Hence in the output '6' is not printed.

```
x=1
while x<=5:
   print(x)
   x+=1

O/P:--
1
2
```

```
3
4
5
```

```
# Printing numbers from 1 to 5 by using while loop.
x=1
while x<=5:
    print(x)
    x+=1

# Printing numbers from 1 to 5 by using while loop.
x=1
while x<=5:
    if x<5:
        print(x,end=",")
    else:
        print(x,end="")
    x+=1

# Printing even numbers from 10 to 20 by using while loop.
x=10
while (x>=10) and (x<=20):
   print(x)
   x+=2
print("End")

# print sun of given n netural no
x=int(input("Enter any no : "))
sum=0
i=1
while i<=x:
    sum=sum+i
    if i<x:
        print(i,end="+")
    else:
        print(i,end="=")
    i=i+1

print(sum)
```

```
# print n even numbers
x= int(input("Enter how many even number you want :"))
n=1
while n<=x:
    print(2*n)
    n=n+1

# print n even numbers(1,2,3,4,5,6,--------)
x= int(input("Enter how many even numbers you want :"))
n=1
while n<=x:
    if n<x:
        print(2*n,end=",")
    else:
        print(2*n,end="")
    n=n+1

# Print sum of given even numbers.
x= int(input("Enter how many even numbers sum you want :"))
n=1
sum=0
while n<=x:
    sum=sum+2*n
    if n<x:
        print(2*n,end="+")
    else:
        print(2*n,end="=")
    n=n+1
print(sum)

# Print n odd numbers
x= int(input("Enter how many odd number you want :"))
n=1
while n<=x:
    if n<x:
        print((2*n-1),end=",")
    else:
        print((2*n-1),end="")
    n=n+1
```

```
# Print sum of n odd numbers
x= int(input("Enter how many odd number you want :"))
n=1
sum = 0
while n<=x:
    sum=sum+(2*n-1)
    if n<x:
        print((2*n-1),end="+")
    else:
        print((2*n-1),end="=")
    n=n+1
print(sum)

# WAP to print even no upto n netural no.
n=int(input("Enter no : "))
i=2
while i<=n:
    if i<=n-2:
        print(i,end=",")
    else:
        print(i)
    i+=2

# WAP to print odd no upto n netural no.
n=int(input("Enter no : "))
i=1
while i<=n:
    if i<=n-2:
        print(i,end=",")
    else:
        print(i)
    i+=2
```