Sesion 27

1. Introducción:

En este tema los alumnos aprenderán a validar el ingreso de datos, para que los procesos se ejecuten de forma correcta; de ese modo ya no se tendrá que asumir que el ingreso de los datos por parte del usuario es el correcto y evitar errores por ese motivo al ejecutar el código; desde este tema en adelante todo lo que se desarrolle deberá contar con su validación respectiva.

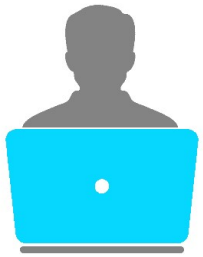
MÓDULO 3: Valores booleanos, Instrucciones if-elif-else, Bucles while y for, Control de flujo, Operaciones lógicas y bit a bit, Listas y arreglos.

2. Validación de datos

a. Definición y finalidad

Cuando los usuarios hacen uso de cualquier programa, estos deben contar con las validaciones respectivas, de modo que los datos que se ingresen sean los correctos y no generen errores al momento de realizarse los procesos respectivos.

En ese sentido el usuario, siempre puede cometer errores al momento del registro de datos ya que es un proceso manual. Sin embargo, el sistema/programa debe estar en la capacidad de evitar que esos errores cometido de manera involuntaria en su gran mayoría se registren o generen mensajes de error por validaciones de integridad que pueden estar codificados en la misma base de datos.



Para ello, veremos cómo se codifica una validación de datos en Python a nivel de consola, para luego más adelante usted pueda migrar la misma idea a otro nivel; la idea en general es comprender el concepto y la técnica de validación.

Se valida para que el sistema tenga los datos exactos que la aplicación requiere y se genere información correcta para la toma de decisiones o se genere nuevos datos de entrada.

b. Tipos de código para validar datos, uso del while

Cuando se procede a generar códigos de validación podemos crearlos de diversas formas, todo depende como se maneje la lógica de programación, las instrucciones de un lenguaje de programación, los operadores lógicos y las funciones que provee el lenguaje.

Observemos el 1er código, donde se implementa una validación para el ingreso de números positivos:

```
N=0
C=0
while(N<=0):
    print("Ingrese el número:")
    N=float(input())
    C=C+1
if(C!=1):
    print(f"Se cometieron {C-1} errores")
```

El código indica que mientras que el valor de N sea <= que 0 (se inicializa en un valor que aplique esa regla), deberá estar en el bucle while dando vueltas hasta que se ingrese un valor contrario a la condicional, que sería lo válido ya que indica que deben ser número positivos los ingresos.
La variable C lo que hace es contar el número de intentos fallidos y al final del código se muestran.

Al ejecutar se mostrará lo siguiente:

```
Ingrese el número:
0
Ingrese el número:
-2
Ingrese el número:
-56
Ingrese el número:
7
Se cometieron 3 errores
```

Observación: En algunos códigos publicados y que tratan respecto a este punto, pueden encontrar algo similar a lo siguiente:

```
print("Ingrese el número:")
N=float(input())
while(N<=0):
    print("Ingrese el número:")
    N=float(input())
```

El detalle en esta validación es que se genera redundancia de código al momento de la doble lectura de datos, algo no ideal.

Observemos el 2do código, donde se implementa la misma validación que el ejemplo anterior, ingreso de números positivos:

```
C=0
Sw=True
while(Sw):
    print("Ingrese el número:")
    N=float(input())
    if(N>0):
        Sw=False
    C=C+1
if(C!=1):
    print(f"Se cometieron {C-1} errores")
```

El código indica que para controlar el bucle while se usa un indicador (variable Sw) lógico que está en valor True, al momento de ingresar los valores se condiciona el ingreso, si este resulta ser un número válido(positivo) cambia dicho indicador a False; con lo que finaliza el bucle.
La variable C lo que hace es contar el número de intentos fallidos y al final del código se muestran.

Al ejecutar se mostrará lo siguiente:

```
Ingrese el número:
0
Ingrese el número:
-2
Ingrese el número:
-8
Ingrese el número:
4
Se cometieron 3 errores
```

Este código puede tener una leve variante, e implica el uso de la instrucción break, observar:

```
C=0
Sw=True
while(Sw):
    C=C+1
    print("Ingrese el número:")
    N=float(input())
    if(N>0):
        break
if(C!=1):
    print(f"Se cometieron {C-1} errores")
```

Podrá observar dos cambios, la posición del contador y el break,

3. Desarrollo de ejercicios tipo

Los ejercicios que veremos a continuación muestran la aplicación de las instrucciones for y while para el control de bucles; incluyendo el tema de validaciones en general. A partir de este tema en adelante todo ingreso en código deberá validarse.

1. Problema Prg1

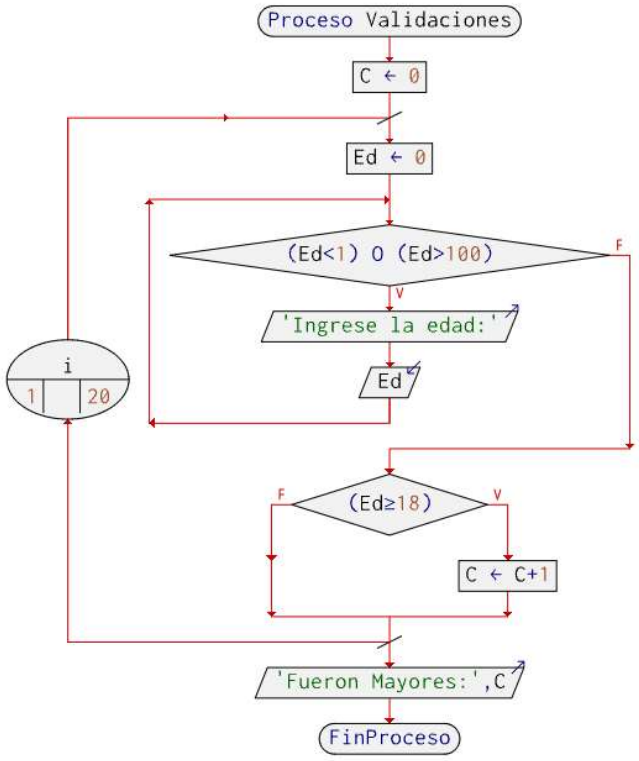
Desarrolle un algoritmo que permita ingresar las edades de 20 alumnos, luego indicar cuantos fueron mayores de edad.

Solución:

Ahora debemos validar el ingreso de las edades antes de operar, pero el problema en ese sentido no nos entrega datos adicionales así que debemos asumir algunos límites para definir una edad válida, en todo caso podemos indicar que la mínima edad debe ser 1 y la máxima podría ser 100; con esos valores trabajamos.

Es obvio que, en un caso real, vamos a requerir que nos indiquen los límites de las edades con que se registran a los clientes, usuarios, etc.

Diagrama de Flujo:



Lenguaje de Programación

```
c = 0
for i in range(5):
    ed = 0
    while (ed<1) or (ed>100):
        print("Ingrese la edad:")
        ed = float(input())
    if (ed>=18):
        c = c+1
print("Fueron Mayores:",c)
```

Validando la edad

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese la edad:
4
Ingrese la edad:
19
Ingrese la edad:
400
Ingrese la edad:
50
Ingrese la edad:
0
Ingrese la edad:
14
Ingrese la edad:
18
Fueron Mayores: 3
```

Debe considerar que cuando se validan datos, el ingreso de los valores puede resultar exacto a lo solicitado o mucho mayor, ya que se está aceptando solo valores válidos para su evaluación; en el ejemplo puede ver que se está trabajando en base a 5 datos, sin embargo, se han ingresado 7, 5 válidos y 2 inválidos.

Por otro lado, el código se puede alterar, para que, al momento de existir un error, se muestre un mensaje al usuario indicando ello, observar:

```
c = 0
for i in range(5):
    ed = 0
    while (ed<1) or (ed>100):
        print("Ingrese la edad:")
        ed = float(input())
        if (ed<1) or (ed>100):
            print("Error al ingresar la EDAD...!!!")
    if (ed>=18):
        c = c+1
print("Fueron Mayores:",c)
```

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese la edad:
17
Ingrese la edad:
102
Error al ingresar la EDAD...!!!
Ingrese la edad:
60
Ingrese la edad:
4
Ingrese la edad:
0
Error al ingresar la EDAD...!!!
Ingrese la edad:
15
Ingrese la edad:
19
Fueron Mayores: 2
```

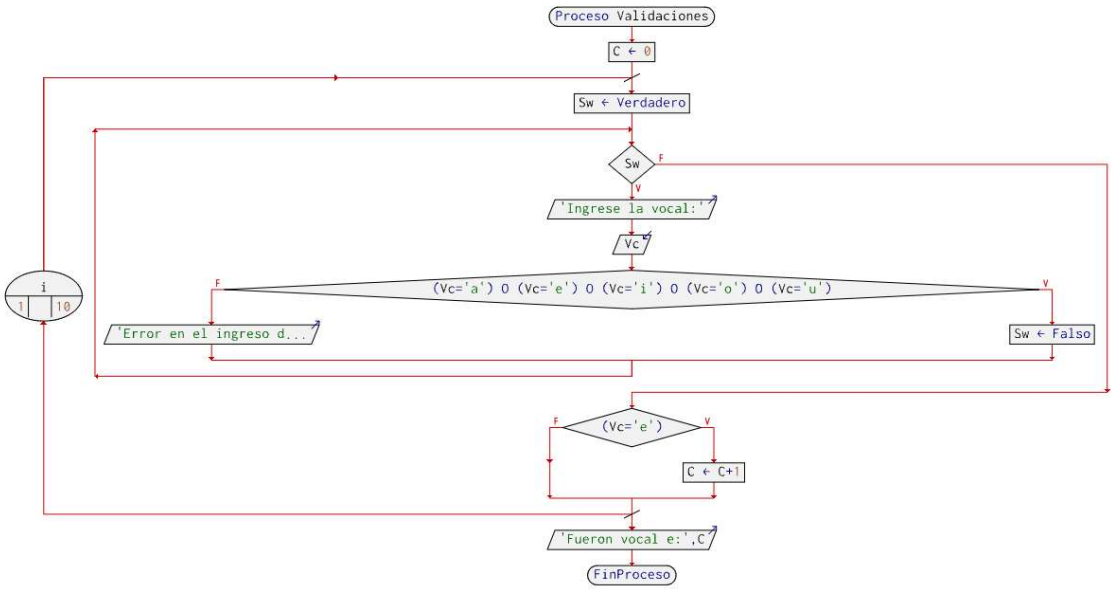
Ante un error, se envía un mensaje al usuario para que ingrese de nuevo el valor solicitado.

2. Problema Prg2

Desarrolle un algoritmo que permita ingresar 10 vocales y luego indicar cuantas fueron la letra E.

Solución:
En este problema nos están solicitando ingresar vocales en este punto se tiene que validar, el resto es solo usar un contador.

Diagrama de Flujo:



Lenguaje de Programación

```
c = 0
for i in range(5):
    sw = True
    while sw:
        print("Ingrese la vocal:")
        vc = input()
        if (vc=="a")or(vc=="e")or(vc=="i")or(vc=="o")or(vc=="u"):
            sw = False
        else:
            print("Error en el ingreso de la vocal...!!!")
    if (vc=="e"):
        c = c+1
print("Fueron vocal e:",c)
```

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese la vocal:
o
Ingrese la vocal:
t
Error en el ingreso de la vocal...!!!
Ingrese la vocal:
r
Error en el ingreso de la vocal...!!!
Ingrese la vocal:
e
Ingrese la vocal:
a
Ingrese la vocal:
a
Ingrese la vocal:
i
Fueron vocal e: 1
```

En la solución propuesta, podemos observar que se está validando por medio de indicadores lógicos (que llamaremos switch en adelante), con lo que la condicional del bucle que valida se hace menos engorrosa.

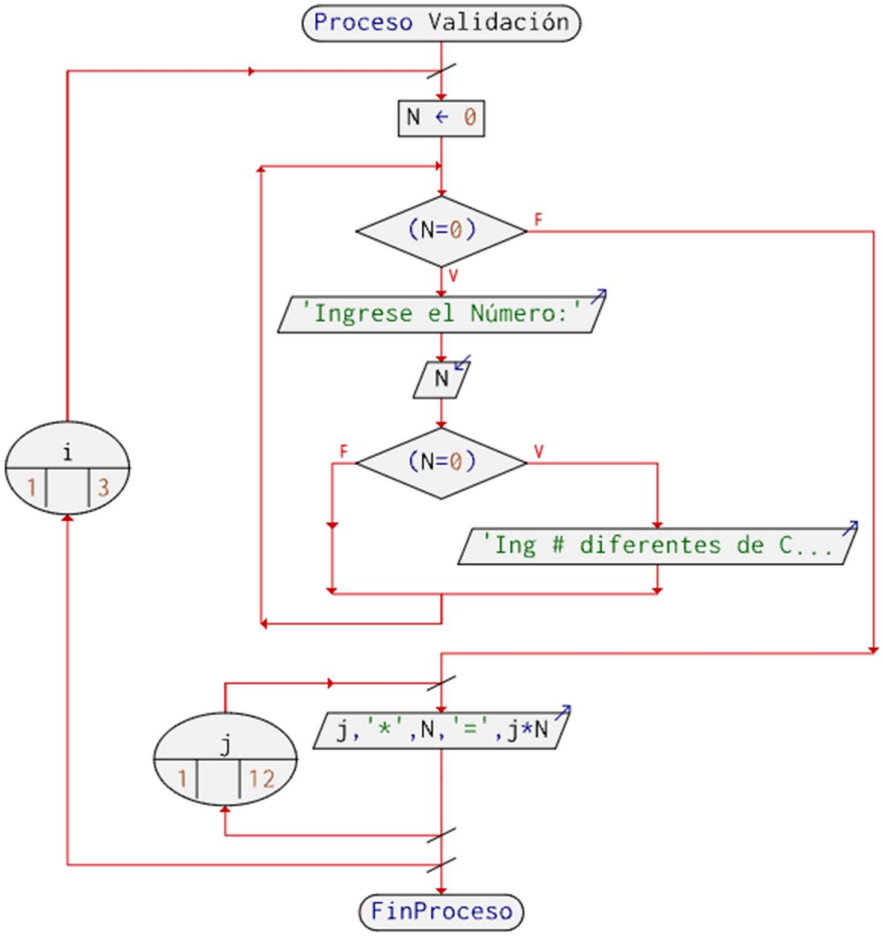
3. Problema Prg3

Desarrolle un algoritmo que permita ingresar 100 números diferentes de cero, luego deberá mostrar la tabla de multiplicar del 1 al 12 solo para los números ingresados que fueron pares.

Solución:

En este caso la validación es fácil de implementar, ya que los números ingresados deben ser diferentes de cero; el proceso de mostrar la tabla de multiplicar implica el uso de un for de 1 a 12.

Diagrama de Flujo:



Lenguaje de Programación

Se construye la tabla de multiplicar.

```
for i in range(100):
    n = 0
    while (n==0):
        print("Ingrese el Número:")
        n = float(input())
        if (n==0):
            print("Ing # diferentes de CERO...!!!")
    for j in range(12):
            print(j,"*",n,"=",j*n)
```

Al ejecutar el programa, se muestra lo siguiente, si no existe error:

```
Ingrese el Número:
5
0 * 5.0 = 0.0
1 * 5.0 = 5.0
2 * 5.0 = 10.0
3 * 5.0 = 15.0
4 * 5.0 = 20.0
5 * 5.0 = 25.0
6 * 5.0 = 30.0
7 * 5.0 = 35.0
8 * 5.0 = 40.0
9 * 5.0 = 45.0
10 * 5.0 = 50.0
11 * 5.0 = 55.0
```

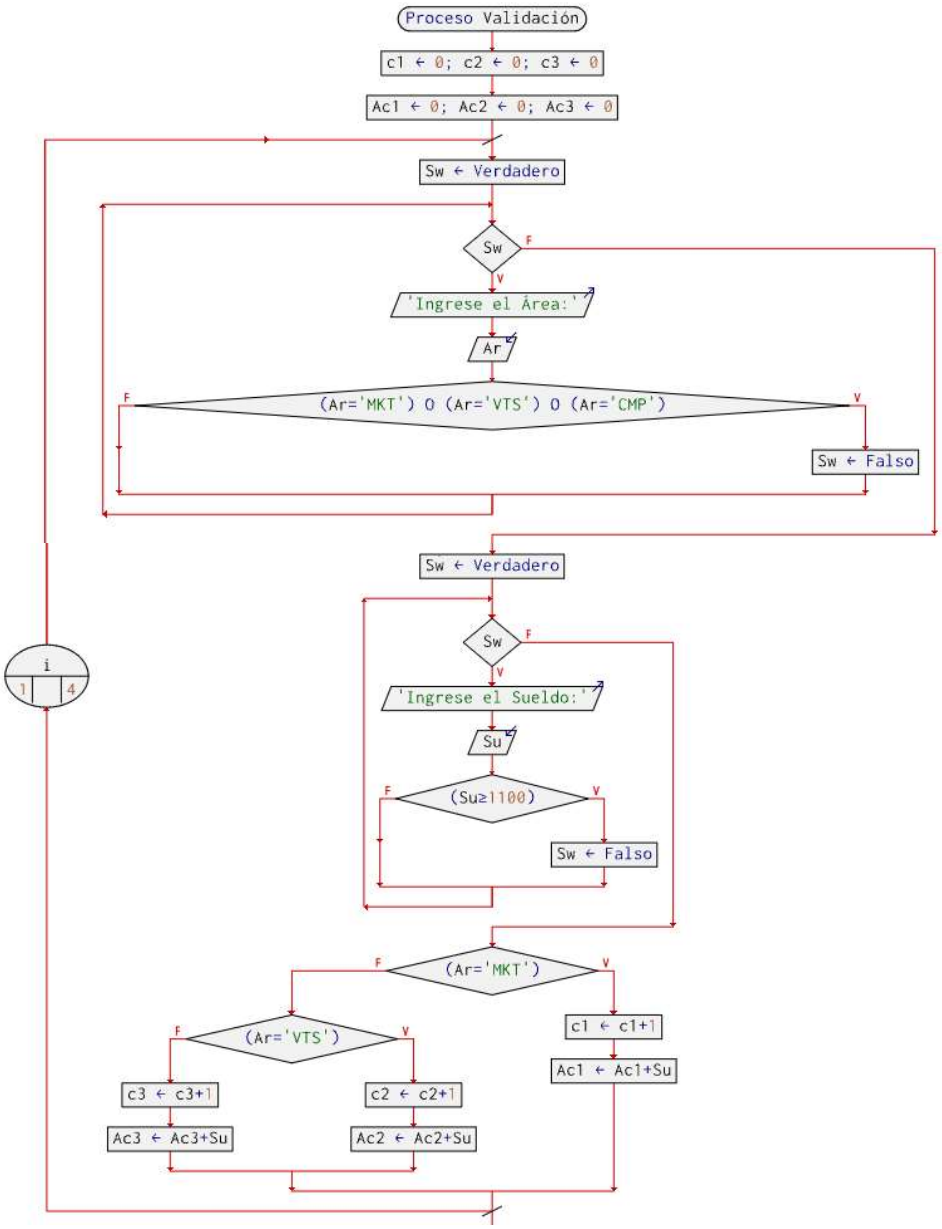
Y ante la ocurrencia de un error en el ingreso del número, se mostrará de la siguiente manera, para luego continuar:

```
Ingrese el Número:
0
Ing # diferentes de CERO...!!!
Ingrese el Número:
19
0 * 19.0 = 0.0
1 * 19.0 = 19.0
2 * 19.0 = 38.0
3 * 19.0 = 57.0
4 * 19.0 = 76.0
5 * 19.0 = 95.0
6 * 19.0 = 114.0
7 * 19.0 = 133.0
8 * 19.0 = 152.0
9 * 19.0 = 171.0
10 * 19.0 = 190.0
11 * 19.0 = 209.0
```

4. Problema Prg4

Se va a ingresar los datos de 100 empleados, deberá ingresar por teclado su área de trabajo: Marketing, Ventas o Compras y el sueldo de cada uno (ninguno puede estar por debajo de 1100 soles), se pide realizar un algoritmo que permita saber cuántos empleados existen por cada categoría y el promedio de sus sueldos por cada categoría.

Solución:
El problema propuesto es de seguimiento ordenado al texto entregado, tiene un diagrama estructurado que tiene que funcionar correctamente para evitar errores en el cálculo de lo solicitado; no es un problema complejo en su solución.
Diagrama de Flujo:



Lenguaje de Programación

```
c1 = 0;c2 = 0;c3 = 0
ac1 = 0;ac2 = 0;ac3 = 0
for i in range(100):
    sw = True
    while sw:
        print("Ingrese el Área:")
        ar = input()
        if (ar=="MKT") or (ar=="VTS") or (ar=="CMP"):
            sw = False
    sw = True
    while sw:
        print("Ingrese el Sueldo:")
        su = float(input())
        if (su>=1100):
            sw = False
    if (ar=="MKT"):
        c1 = c1+1
        ac1 = ac1+su
    else:
        if (ar=="VTS"):
            c2 = c2+1
            ac2 = ac2+su
        else:
            c3 = c3+1
            ac3 = ac3+su
print("MKT, Nro de empleados:",c1)
print("MKT, Promedio sueldos:",ac1/c1)
print("VTS, Nro de empleados:",c2)
print("VTS, Promedio sueldos:",ac2/c2)
print("CMP, Nro de empleados:",c3)
print("CMP, Promedio sueldos:",ac3/c3)
```

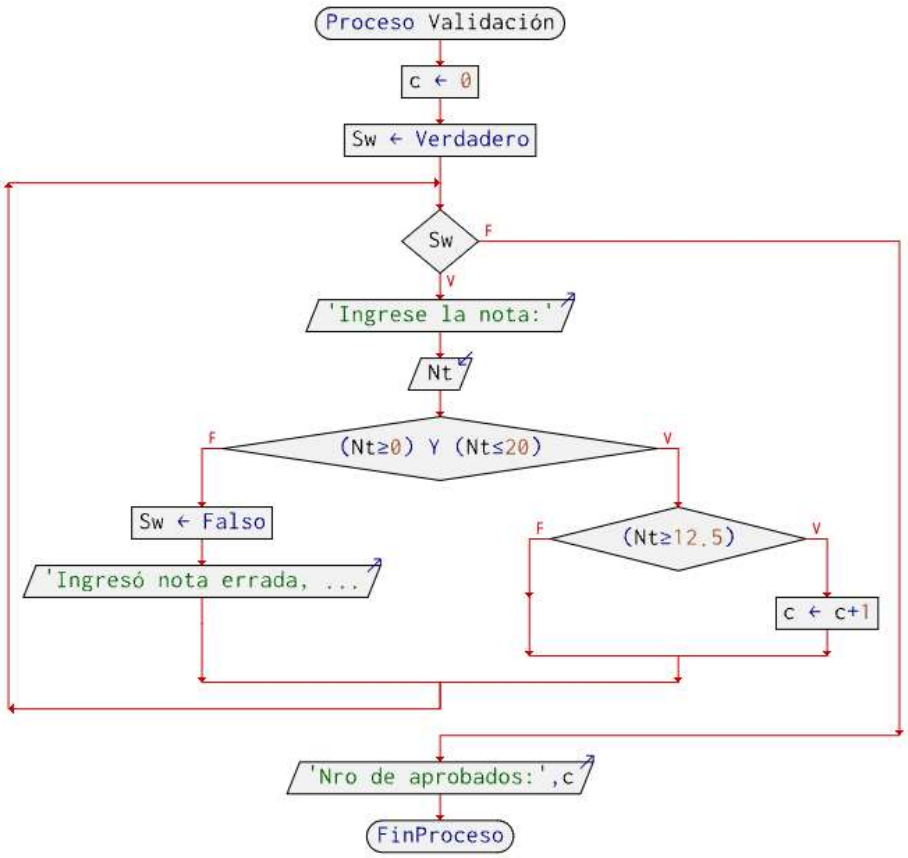
Al ejecutar el programa, se mostrará al finalizar los contadores de cada área y sus respectivos promedios.

5. Problema Prg5

Se ingresan notas, desarrolle un algoritmo que permita mostrar cuantos aprobaron el curso, considerar que el programa se termina automáticamente si se ingresa una nota errada.

Solución:
Para este caso, las notas tienen un rango fijo y es el de 0 a 20, los aprobados son los que tienen de 12.5 hacia arriba, el control del bucle debe realizarse con while, porque no nos indica la cantidad de notas a ingresar.

Diagrama de Flujo:



Lenguaje de Programación

```
c = 0
sw = True
while sw:
    print("Ingrese la nota:")
    nt = int(input())
    if (nt>=0) and (nt<=20):
        if (nt>=12.5):
            c = c+1
    else:
        print("Ingresó nota errada, finaliza todo")
        break
print("Nro de aprobados:",c)
```

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese la nota:
12
Ingrese la nota:
19
Ingrese la nota:
20
Ingrese la nota:
80
Ingresó nota errada, finaliza todo
Nro de aprobados: 2
```

4. Actividad

A) Ejercicios en general

Desarrolle los siguientes ejercicios usando for o while dependiendo el caso, todos los ejercicios deberán estar validados.

a) Se ingresan 50 números que representan a un mes del año (validar que estén entre 1 y 12), luego indicar cuántos de esos números representaron a otoño o verano.

b) Desarrolle un algoritmo que permita ingresar 15 números negativos, luego indicar la suma de los que fueron mayores a -10.

c) Realizar un algoritmo que permita el ingreso 200 números de tres cifras cada uno, descomponerlos y determinar cuántos de los dígitos que componen cada número fueron pares y cuántos impares.

d) Diseñar un algoritmo que permita mostrar datos estadísticos sobre las preferencias para las próximas elecciones para presidente en general; las alternativas son: PP, APRA, SI CUMPLE (validar el ingreso), al finalizar

el ingreso de datos (100 000 votantes), se deberá observar el total de preferencia por cada partido, quien es el posible ganador y en caso de haber empate con el primer puesto, mostrar que partidos pierden y no pasan a la segunda vuelta (no pasan a 2da vuelta el 3er y 4to puesto).

e) Desarrollar un algoritmo que permita ingresar la sección (A, B o C) y el sexo (M o F) para los alumnos de un colegio (validar el ingreso), al finalizar se deberá mostrar:

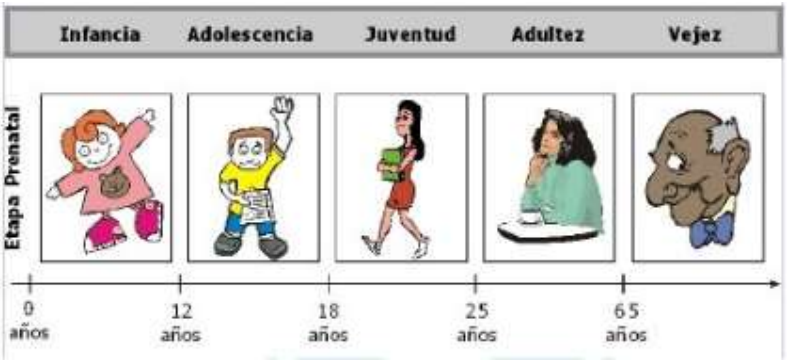
- Total, de hombres por cada sección
- Total, de mujeres por cada sección
- Total, general de hombres
- Total, general de mujeres

El ingreso de datos debe finalizar cuando se responda de manera negativa a la pregunta si desea continuar SI o NO (validar este ingreso).

f) Se ingresan 20 números positivos, muestre la factorial de cada uno de ellos.

g) Desarrollar un algoritmo que permita ingresar 100 números positivos, luego indique el número de divisores que tiene cada uno.

h) Se ingresa la edad de 100 personas (mínimo 0 años, máximo 150 años), indicar que cantidad de edades ingresadas pertenecen a cada etapa del desarrollo humano, queda a criterio el colocar si toma o no el extremo (NO considerar la etapa Prenatal).



5. Fuentes consultadas:

A) https://edube.org/learn/programming-essentials-in-python-part-1-spanish
B) Edison Zavaleta C. (2005). *Fundamentos de Programación*. Perú, Editorial Abaco-Lima.