

SUBINDICADOR # 7

Implementa las estructuras de bucle for y while bajo la sintaxis de Python, en el desarrollo de programas que requieren el manejo de grandes cantidades de datos

1. Introducción:

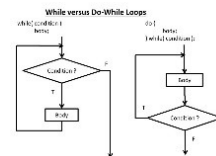
En los temas propuestos el alumno reconocerá las estructuras repetitivas, denominadas también bucles, que les van a permitir el control del ingreso de grandes cantidades de datos.

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. Estructuras Repetitivas

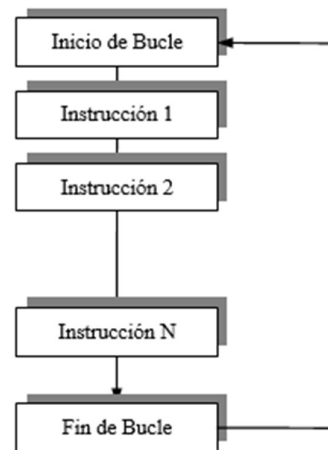
a. Definiciones previas: bucle, contador y acumulador

Los algoritmos que manejan estructuras repetitivas, se caracteriza por que entre sus instrucciones se muestran estructuras secuenciales, estructuras selectivas y estructuras que forman bucles, es decir un conjunto de instrucciones que se ejecutan un número determinado de veces, dependiendo una condición o un límite definido.



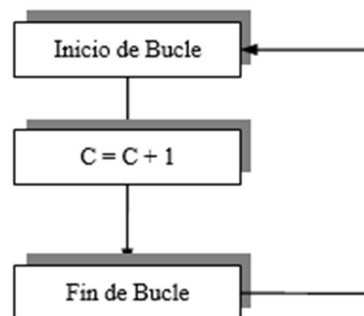
Se manejan algunos conceptos asociados:

- **Bucle:** es denominado también lazo (Loop), es una parte de un algoritmo o programa, cuyas instrucciones se repiten un número determinado de veces. En todo algoritmo se debe establecer cuáles son las sentencias que formaran las tareas repetitivas.

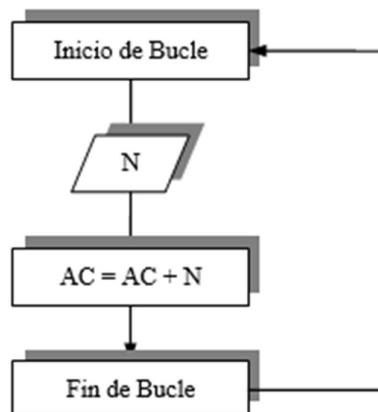


- **Contador:** los procesos repetitivos por lo general lo utilizan, ya que necesitan contar los sucesos o acciones internas del bucle. Una de las formas de controlar un bucle es mediante un contador.

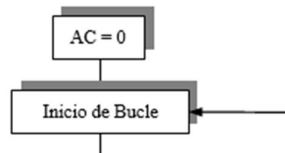
Un contador es una variable cuyo valor se incrementa o decrementa en una cantidad constante por cada interacción que se genera en el bucle.



- **Acumulador:** es denominado también totalizador, es una variable cuya misión es almacenar cantidades o valores resultantes de sumas (restas) sucesivas. Realiza la misma función que un contador con la diferencia de que el incremento o decremento de cada suma es variable en lugar de constante como en el caso del contador.



Se debe tener en consideración que todo contador y acumulador debe ser inicializado generalmente; el valor por lo general es cero, sin embargo, no es una norma depende de qué actividad se desarrolle, del mismo modo el lugar donde se inicializa debe ubicar fuera de un bucle que es el lugar donde finalmente se incrementa o decrementa sus valores.



3. Ciclos en Python

a. Sentencia repetitiva While

Esta sentencia repetitiva evalúa una condición al inicio del bucle, si esta es verdadera se ejecutarán una serie de acciones, las cuales seguirán realizándose hasta que la condición se convierta en falsa, momento en el que se dejarán de evaluar las acciones y se procederá a continuar con la siguiente línea de sentencias fuera del bucle.

Se debe tener en consideración lo siguiente:

- Si deseas ejecutar más de una sentencia / instrucción dentro de un while, se debe indentar, es decir, poner sangría a todas las instrucciones de la misma manera.
- Una instrucción o conjunto de instrucciones ejecutadas dentro del while se llama el cuerpo del ciclo.
- Las instrucciones internas deben poder cambiar el valor de la condición de ingreso (True), porque si sino ocurre ello, se ingresa en un bucle infinito.
- La instrucción While es muy usada en las validaciones de datos.

En general debemos considerar que la sentencia While, solo permite la evaluación de las acciones en su interior siempre en cuando la condición de entrada sea verdadera; sino nunca podrá ingresar a la estructura.

Observemos el siguiente código:

```
print("Bucle While")
R=0
while (R<5) :
    R=R+1
    print("Vuelta nro:",R)
```

Al ejecutar se mostrará lo siguiente:

```
Bucle While
Vuelta nro: 1
Vuelta nro: 2
Vuelta nro: 3
Vuelta nro: 4
Vuelta nro: 5
```

Se indica que la variable R (contador), se está inicializando fuera del bucle en 0 y luego en la condicional del While se traduce: "Mientras R sea Menor que 5 entonces..."

De esa manera se ingresa al bucle, se incrementa el contador R y se imprime valor por valor (R es quien decide cuándo termina el bucle).

b. Sentencia repetitiva For

Esta es una sentencia repetitiva, se caracteriza por tener un inicio y un fin que serán establecidos vía programa y no tiene condicionales (evaluaciones lógicas) para la salida o ingreso al bucle; esta sentencia repetitiva es una de las más fáciles de aplicar y utilizar; sin embargo, permite el recorrido de estructuras mucho más complejas.

La estructura for en el Python, contiene algunos elementos que permiten el control del bucle:

- Cualquier variable después de la palabra reservada for es la variable de control del bucle; cuenta los giros y lo hace automáticamente.
- Se tiene la palabra reservada **in** introduce un elemento de sintaxis que describe el rango de valores posibles que se asignan a la variable de control.

- La función **range()**, es responsable de generar todos los valores deseados de la variable de control, este comienza su trabajo desde 0 y finaliza un valor antes del argumento final indicado.
- La palabra clave **pass** dentro del bucle - no hace nada en absoluto; es una instrucción vacía, se coloca porque la sintaxis del ciclo for exige al menos una instrucción, es la misma que usamos en otras estructuras.

A una estructura for, siempre se ingresará, ya que esta tiene definido un inicio y un fin de recorrido; cuando en un ejercicio o problema se conoce el límite de trabajo, es preferible el uso de for.

Observemos el siguiente código:

```
for i in range(5):  
    print("Vuelta nro:",i)
```

Al ejecutar se mostrará lo siguiente:

```
Vuelta nro: 0  
Vuelta nro: 1  
Vuelta nro: 2  
Vuelta nro: 3  
Vuelta nro: 4
```

Si deseamos que se observe el número de vueltas desde 1 y no en 0, simplemente agregar lo siguiente al código:

```
for i in range(5):  
    print("Vuelta nro:",i+1)
```

c. La función range() y sus argumentos.

La función range(), se utiliza para representar una secuencia inmutable de números enteros, es decir que no se puede modificar. Este contiene 3 variantes bien definidas al momento de trabajar:

- Con un solo parámetro:

```
for i in range(3):  
    print("Vuelta nro:",i)
```

Observe que solo existe un solo parámetro dentro de la función range() y es quien indica el número de vueltas que debe generarse, el resultado será:

```
Vuelta nro: 0
Vuelta nro: 1
Vuelta nro: 2
```

- Con dos parámetros:

```
for i in range(2,7):
    print("Vuelta nro:",i)
```

Observe que ahora existen dos parámetros dentro de la función range(), la primera indica de donde parte y el valor final de llegada (no olvidar que nunca llega al último valor), el resultado será:

```
Vuelta nro: 2
Vuelta nro: 3
Vuelta nro: 4
Vuelta nro: 5
Vuelta nro: 6
```

- Con tres parámetros:

```
for i in range(1,10,2):
    print("Vuelta nro:",i)
```

Observe que ahora existen tres parámetros dentro de la función range(), la primera indica de donde parte, la segunda el último el valor de llegada y el tercer valor el número de saltos que debe generarse desde el parámetro indicado como uno, el resultado será:

```
Vuelta nro: 1
Vuelta nro: 3
Vuelta nro: 5
Vuelta nro: 7
Vuelta nro: 9
```

d. Control de ciclos, declaraciones break y continue

Las instrucciones break y continue, tienen cada una su propia finalidad dentro del código y se aplican a cualquiera de las instrucciones vistas anteriormente, es decir, for o while.

Break, nos permite forzar la finalización de un bucle, desde el lugar donde sea ubicado.

Continue, nos permite obviar alguna parte del controlador de un bucle, de modo que genera una zona ciega para el bucle.

Caso **break**:

```
print("Instrucción break")
for i in range(8):
    if (i==5):
        break
    print("Vuelta Nro", i)
```

Al ejecutar, observar el resultado de la impresión.

```
Instrucción break
Vuelta Nro 0
Vuelta Nro 1
Vuelta Nro 2
Vuelta Nro 3
Vuelta Nro 4
```

Caso **continue**:

```
print("Instrucción continue")
for i in range(8):
    if (i>3) and (i<=5):
        continue
    print("Vuelta Nro", i)
```

Al ejecutar, observar el resultado de la impresión.

```
Instrucción continue
Vuelta Nro 0
Vuelta Nro 1
Vuelta Nro 2
Vuelta Nro 3
Vuelta Nro 6
Vuelta Nro 7
```

Como podrá haber observado, cada una tiene su ventaja al momento de programar, solo debe considerar en que instante usarlas.

e. Manejo de la opción else en los ciclos

La instrucción else de los bucles, es propia de Python e indica que, si algo no ocurre en la instrucción, puede mostrarse lo indicado dentro de esa instrucción.

En el caso propuesto veremos cómo opera la instrucción:

```
Op=False
while(Op):
    print("Dentro...!!!")
    Op=False
else:
    print("Fuera...!!!")
```

Al ejecutar, observará lo siguiente:

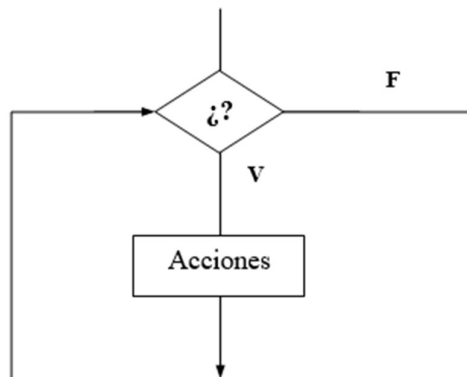
Fuera...!!!

Ello, está indicando que nunca ingresó al while, ya que la variable Op tenía el valor inicial de False y por consiguiente solo quedaba pasar a la siguiente línea fuera del bucle, que en este caso es el else del While.

f. Diagrama de Flujo de la sentencia for y while.

El diagrama de estas estructuras varía mucho en cuanto al gráfico, ya que cada autor inclusive propone sus propios símbolos, sin embargo, usaremos los que manipula el PSeInt actualmente.

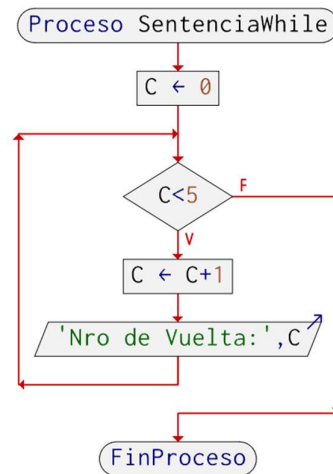
Para el caso de While:



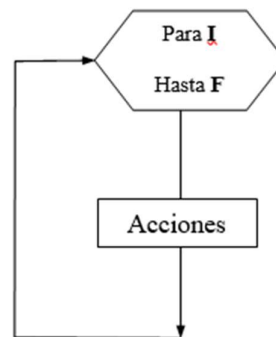
El rombo implica la condicional de ingreso al bucle, si esta resulta verdadera, se ingresa, caso contrario siempre estaremos fuera de la estructura, observar el ejemplo planteado a continuación:

En este caso se está generando 5 vueltas, mediante el control de la variable C, la cual es inicializada fuera del bucle en 0.

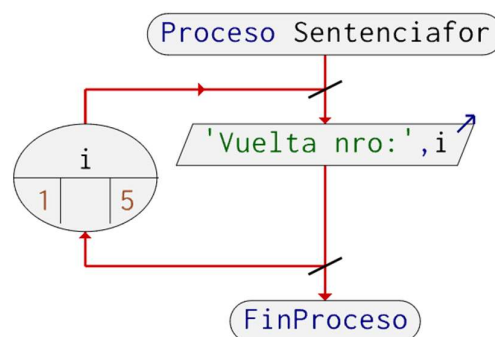
Para guiarse debe observar las líneas de conexión.



Para el caso de For:



El símbolo mostrado, indica la secuencia de inicio y fin, del bucle for, sin embargo, este si tiene variaciones en el siguiente diagrama, pero la idea es la misma al momento de controlar le bucle.



En el ejemplo, el bucle está controlando 5 vueltas, partiendo desde 1, se debe tener en consideración que en las estructuras de otros lenguajes de programación el inicio como el final son fijos; mientras que, en Python, ocurre que el control es por medio de la función range() que inicia en 0 y termina en

un valor antes del indicado; a no ser que sea forzado a iniciar en un número (ver casos de range()).

4. Operadores Lógicos y bits en Python

a. Los Operadores lógicos

Los operadores lógicos los hemos tratado en el capítulo anterior, por ser de necesidad al momento de crear condicionales, estos son: and, or y not.

b. Generar expresiones lógicas

Las expresiones lógicas se diseñan usando los operadores lógicos y se debe considerar en su uso la manipulación de las tablas de verdad.

c. Operadores bitwise

En los operadores lógicos aplicados a los bits en Python, existen cuatro que le permiten manipular bits de datos individuales, a estos se les denominan operadores bitwise.

Interactúan con los operadores lógicos y se agrega uno adicional que es el operador xor, estos operadores con sus símbolos son los siguientes:

&	(ampersand) - conjunción a nivel de bits.
	(barra vertical) - disyunción a nivel de bits.
~	(tilde) - negación a nivel de bits.
^	(signo de intercalación) - exclusivo a nivel de bits o (xor).

“La diferencia en el funcionamiento de los operadores lógicos y de bits es que los operadores lógicos no penetran en el nivel de bits de su argumento. Solo les interesa el valor entero final.”; ejecutar y observe el resultado:

```
print("Datos iniciales:")
x = 4
y = 1
## Operaciones
a = x & y
b = x | y
c = ~ x
d = x ^ 5
e = x >> 2
f = x << 2
print("Salida de valores:")
print(a, b, c, d, e, f)
```

5. Desarrollo de ejercicios tipo

Los ejercicios que veremos a continuación muestran la aplicación de las instrucciones for y while para el control de bucles; a partir de ahora la solución de los problemas será de manera directa, es decir ya no se considera el análisis ya que es algo indirecto que debemos considerar desarrollar siempre, es decir las preguntas *¿Qué te piden que realices?* y *¿Qué datos necesito conocer?* serán de uso intrínseco.

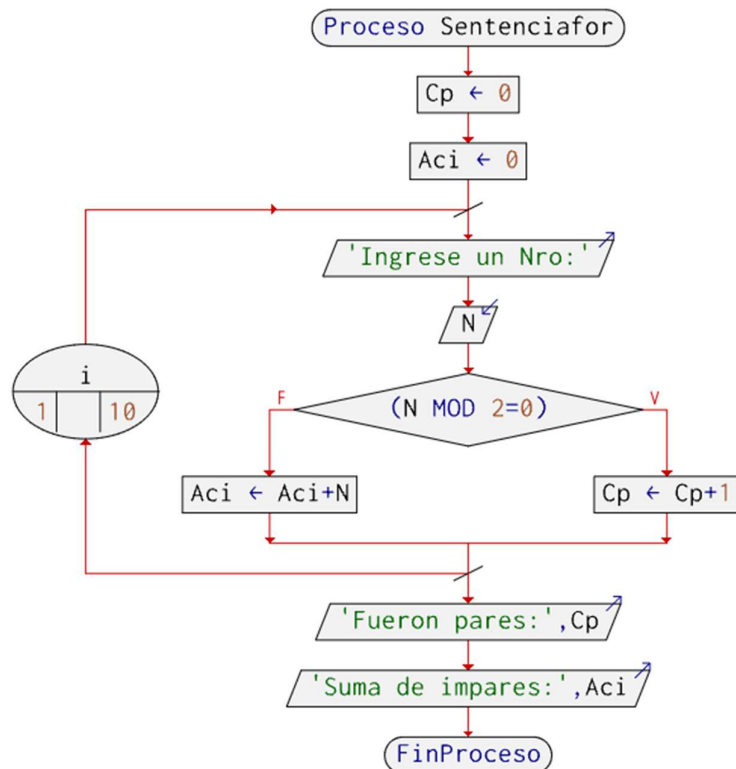
1. Problema Prg1

Diseñar un algoritmo que permita ingresar 10 número, y luego indicar cuantos fueron pares y la suma de los impares.

Solución:

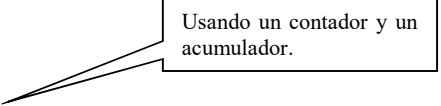
En el ejercicio nos están indicando que son 10 los números a evaluar, por lo que debemos usar el for de manera directa y dentro de esta estructura desarrollar lo solicitado, contar y acumular pares e impares respectivamente; respecto a cómo definir los números pares e impares ya se ha tratado en ejercicios anteriores,

Diagrama de Flujo:



Lenguaje de Programación

```
Cp = 0
Aci = 0
for i in range(10):
    print("Ingrese un Nro:")
    n = int(input())
    if (n%2==0):
        Cp = Cp+1
    else:
        Aci = Aci+n
print("Fueron pares:",Cp)
print("Suma de impares:",Aci)
```

Usando un contador y un acumulador.

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese un Nro:
5
Ingrese un Nro:
7
Ingrese un Nro:
8
Ingrese un Nro:
8
Ingrese un Nro:
8
Ingrese un Nro:
3
Ingrese un Nro:
4
Ingrese un Nro:
7
Ingrese un Nro:
3
Ingrese un Nro:
2
Fueron pares: 5
Suma de impares: 25
```

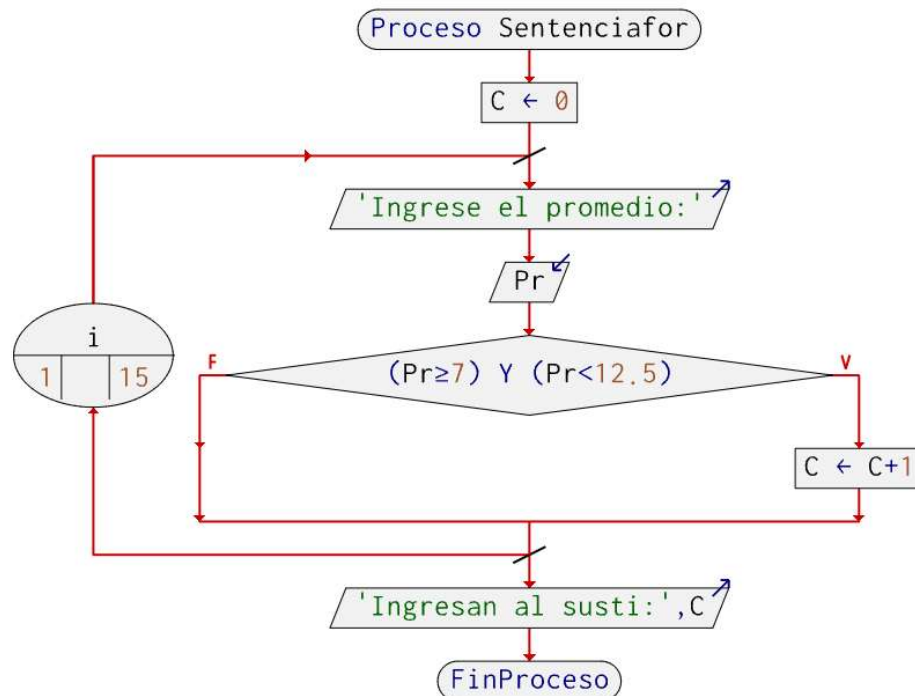
2. Problema Prg2

Diseñar un algoritmo que permita ingresar 15 promedios de notas, pertenecientes a los alumnos del curso de Base de Datos, luego indicar cuantos de los alumnos podrán acceder al examen sustitutorio, si sabemos que ingresan los alumnos que tienen como promedio de 7 a más y menos de 12.5.

Solución:

Tenemos que evaluar los promedios de 15 alumnos, conocemos el total por lo que debemos usar for directamente, luego trabajar con el rango entregado para contarlos y resolver el problema.

Diagrama de Flujo:



Lenguaje de Programación

```
c=0
for i in range(15):
    print("Ingrese el promedio:")
    pr = float(input())
    if (pr>=7) and (pr<12.5):
        c=c+1
    print("Ingresan al susti:",c)
```

Al ejecutar el programa, se muestra lo siguiente:

Ingrese el promedio:	Ingrese el promedio:
12	6
Ingrese el promedio:	Ingrese el promedio:
16	8
Ingrese el promedio:	Ingrese el promedio:
18	9
Ingrese el promedio:	Ingrese el promedio:
9	3
Ingrese el promedio:	Ingrese el promedio:
10	4
Ingrese el promedio:	Ingrese el promedio:
4	5
Ingrese el promedio:	Ingresan al susti: 7
17	
Ingrese el promedio:	
11	
Ingrese el promedio:	
10	

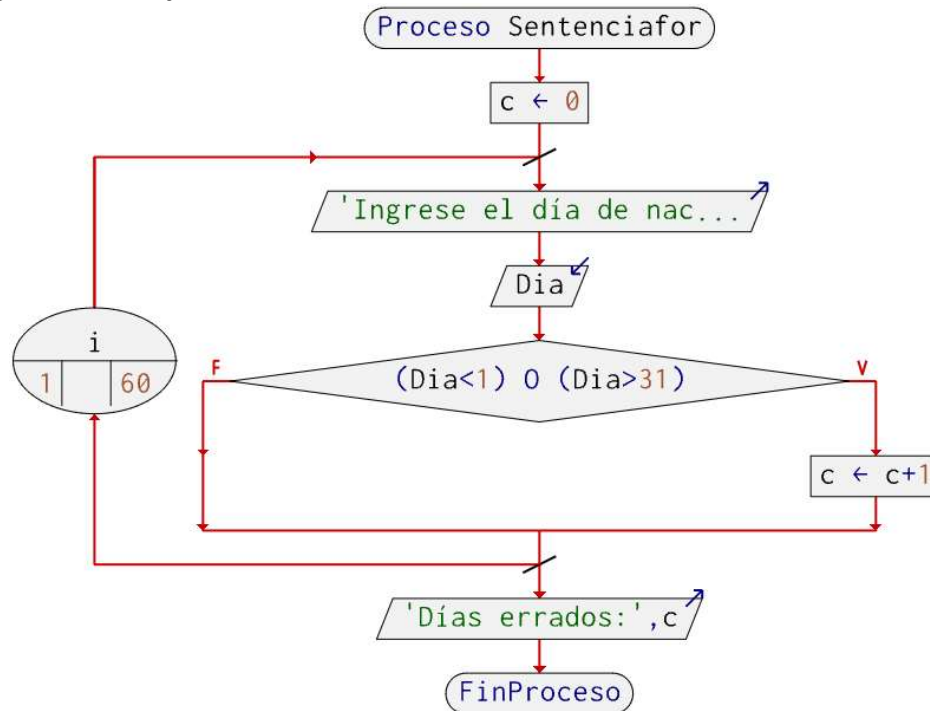
3. Problema Prg3

Se ingresará el día del nacimiento de 60 personas, mediante un algoritmo indicar cuántos días que se ingresaron fueron errados.

Solución:

En el ingreso del día que representa al día de nacimiento, se debe considerar que estos pueden variar entre el día 1 y 31, por el resto es solo usar el contador adecuado.

Diagrama de Flujo:



Lenguaje de Programación

```
c=0
for i in range(6):
    print("Ingrese el día de nacimiento:")
    dia = int(input())
    if (dia<1) or (dia>31):
        c = c+1
print("Días errados:",c)
```

Se está ejecutando solo
para 6

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese el día de nacimiento:
5
Ingrese el día de nacimiento:
90
Ingrese el día de nacimiento:
-4
Ingrese el día de nacimiento:
5
Ingrese el día de nacimiento:
13
Ingrese el día de nacimiento:
17
Días errados: 2
```

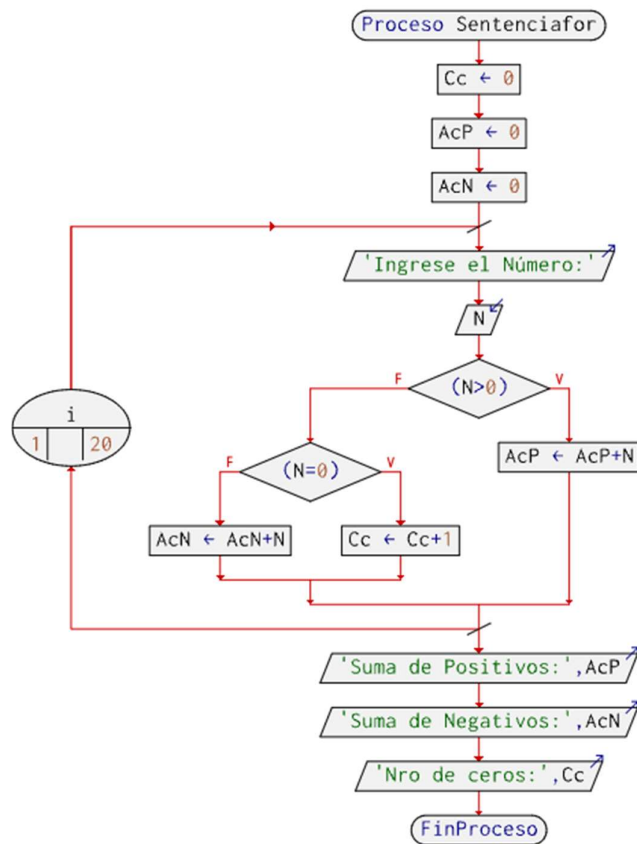
4. Problema Prg4

Desarrolle un algoritmo que permita ingresar 20 números y luego indicar la suma de los que fueron positivos, la suma de los que fueron negativos y cuantos fueron cero.

Solución:

Este problema, nos muestra una estructura más compleja, pero simple en su desarrollo, ya que hemos tratado ejemplos similares antes.

Diagrama de Flujo:



Lenguaje de Programación

```

cc = 0
acp = 0
acn = 0
for i in range(6):
    print("Ingrese el Número:")
    n = int(input())
    if (n > 0):
        acp = acp + n
    else:
        if (n == 0):
            cc = cc + 1
        else:
            acn = acn + n
print("Suma de Positivos:", acp)
print("Suma de Negativos:", acn)
print("Nro de ceros:", cc)

```

Al ejecutar el programa, se muestra lo siguiente:


```
Ingrese el Número:  
5  
Ingrese el Número:  
-3  
Ingrese el Número:  
0  
Ingrese el Número:  
0  
Ingrese el Número:  
4  
Ingrese el Número:  
90  
Suma de Positivos: 99  
Suma de Negativos: -3  
Nro de ceros: 2
```

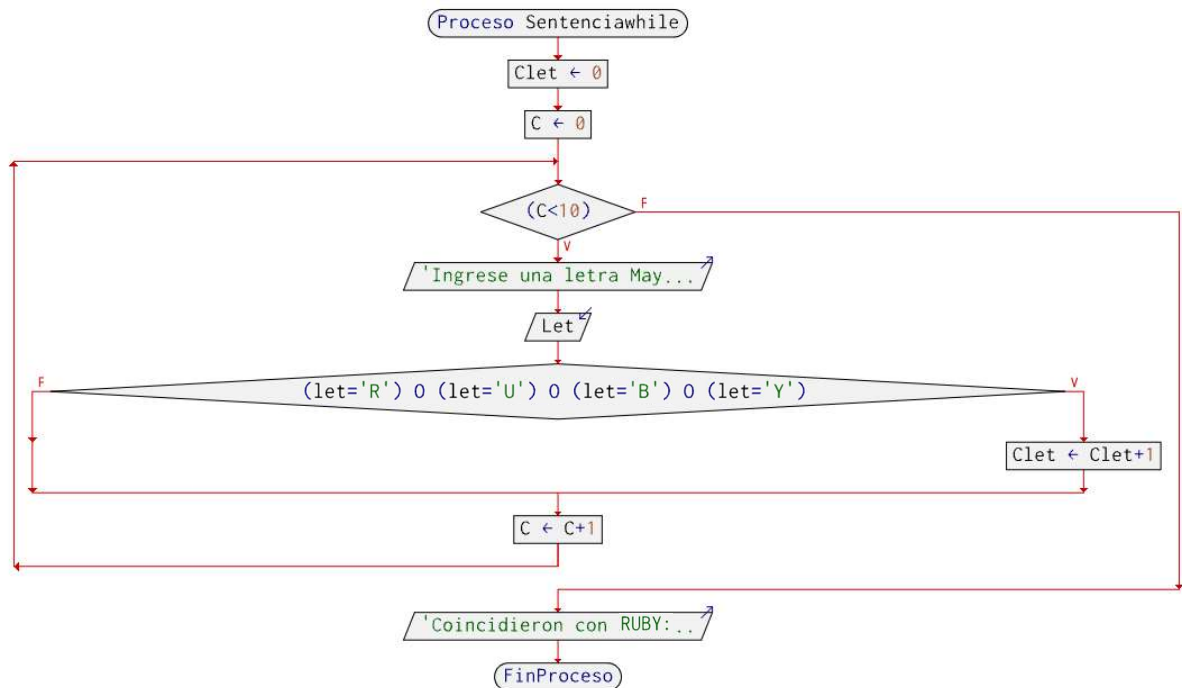
5. Problema Prg5

Desarrolle un algoritmo que permita ingresar 10 letras Mayúsculas, luego indicar cuantas fueron letras ingresadas están registradas dentro de la palabra RUBY; usar la sentencia while para la solución.

Solución:

Debemos ingresar letras en este problema, se debe evaluar si la letra ingresada pertenece a la palabra RUBY, veremos 2 soluciones a continuación, la primera tratando todo de manera lógica y la segunda usando operadores/instrucciones de Python.

Diagrama de Flujo:



Lenguaje de Programación

```

Clet = 0
c = 0
while (c<5):
    print("Ingrese una letra Mayúscula:")
    let = input()
    if (let=="R") or (let=="U") or (let=="B") or (let=="Y"):
        Clet = Clet+1
    c = c+1
print("Coincidieron con RUBY:",Clet," Letras")

```

Al ejecutar el programa, se muestra lo siguiente:

```

Ingrese una letra Mayúscula:
E
Ingrese una letra Mayúscula:
R
Ingrese una letra Mayúscula:
H
Ingrese una letra Mayúscula:
Y
Ingrese una letra Mayúscula:
Q
Coincidieron con RUBY: 2 Letras

```

Usando algunas instrucciones de Python, podemos obtener otra solución al problema, observar:

```
Clet = 0
c = 0
while (c<5):
    print("Ingrese una letra Mayúscula:")
    let = input()
    for i in "RUBY":
        if(i==let):
            Clet = Clet+1
            break
    c = c+1
print("Coincidieron con RUBY:",Clet," Letras")
```

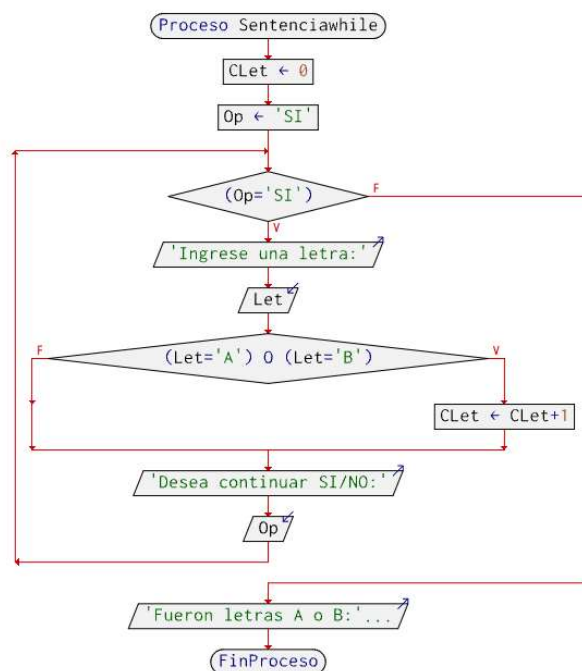
6. Problema Prg6

Desarrolle un algoritmo que permita ingresar letras y luego indicar cuantas fueron la letra A o B, el problema debe finalizar cuando se responde a la pregunta: ¿Si desea continuar SI o NO?, usar la sentencia while.

Solución:

Este problema es un clásico de la sentencia while, ya que no te indican el número de letras a evaluar en el caso, por lo tanto, se debe buscar la forma de poder finalizar el programa, y en esta ocasión nos indican que se debe preguntar ¿Si desea continuar SI o NO?

Diagrama de Flujo:



Lenguaje de Programación

```
clet = 0
op = "SI"
while (op=="SI"):
    print("Ingrese una letra:")
    let = input()
    if (let=="A" or (let=="B")):
        clet = clet+1
    print("Desea continuar SI/NO:")
    op = input()
print("Fueron letras A o B:",clet)
```

Se control el bucle con una pregunta que se tiene que captura al final

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese una letra:
A
Desea continuar SI/NO:
SI
Ingrese una letra:
O
Desea continuar SI/NO:
SI
Ingrese una letra:
P
Desea continuar SI/NO:
SI
Ingrese una letra:
B
Desea continuar SI/NO:
NO
Fueron letras A o B: 2
```

7. Problema Prg7

Desarrolle un algoritmo que permita mostrar la suma de los elementos de la siguiente secuencia:

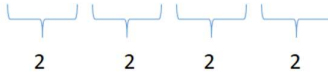
$$S = 1 + 3 + 5 + 7 + 9 + 11 + \dots$$

Para 80 números.

Solución:

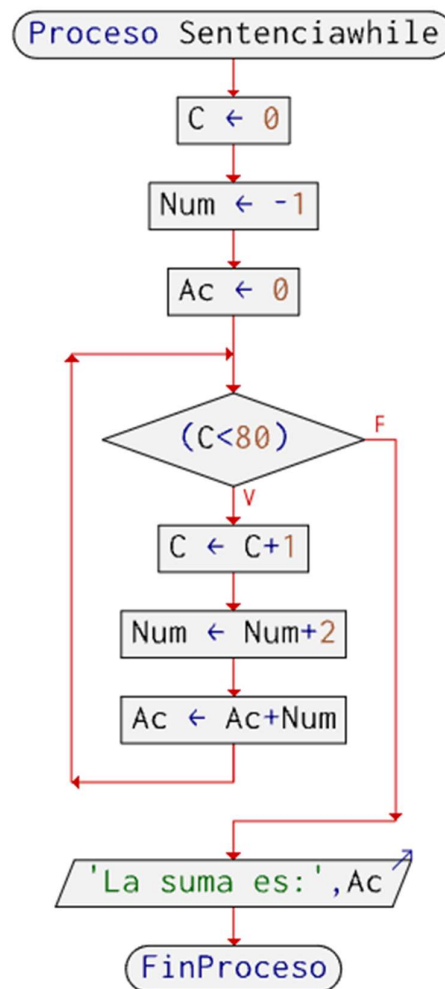
En este problema, debemos considerar la generación de cada cifra por cada vuelta que se genere, hasta llegar a los 80 dígitos, vamos a desarrollarlo con la instrucción while, para poder manipular una secuencia debe ubicar si existe alguna relación entre los números y luego crear el algoritmo.

$$S = 1 + 3 + 5 + 7 + 9 + 11 + \dots$$



Si observamos la secuencia, existe una razón aritmética entre cada número, por lo que debemos manejar ello al momento de generar cada número y luego acumularlo.

Diagrama de Flujo:



Observe que la estructura tiene tres partes bien definidas, la variable **C** que es quien controla el bucle, la variable **Num** que es quien genera cada número en cada vuelta y la variable **Ac** que es el acumulador de número generados.

Lenguaje de Programación

```
C = 0
Num = -1
Ac = 0
while (C<80):
    C = C+1
    Num = Num+2
    Ac = Ac+Num
print("La suma es:",Ac)
```

Se construyen el número
con un contador

Al ejecutar el programa, se muestra lo siguiente:

La suma es: 6400

8. Problema Prg8

Desarrolle un algoritmo que permita mostrar la suma de los elementos de la siguiente secuencia:

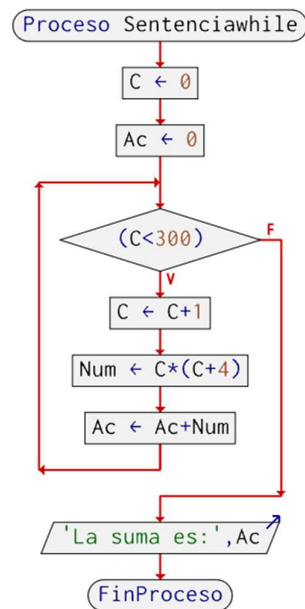
$$S = 1*5 + 2*6 + 3*7 + 4*8 +$$

Para 300 números.

Solución:

En este problema, debemos considerar el construir el número por cada vuelta en base a otras variables existentes, no siempre uno o dos contadores nos generan el número. Observe el código y como se construye el número ahora.

Diagrama de Flujo:



Observe que el número se está construyendo de la siguiente manera: $C*(C+4)$; es decir se está usando el controlador de vueltas C para la construcción.

Lenguaje de Programación

```
c = 0
ac = 0
while (c < 300) :
    c = c + 1
    num = c * (c + 4)
    ac = ac + num
print("La suma es:", ac)
```

Al ejecutar el programa, se muestra lo siguiente:

La suma es: 9225650

6. Actividad

A) Ejercicios en General

Desarrolle los siguientes ejercicios usando for o while dependiendo el caso, para que practique lo desarrollado:

- Desarrolle un algoritmo que permita ingresar 15 números de más de 2 cifras, luego indicar la suma de los que fueron mayores a 100.

- b) Desarrolle un algoritmo que permita ingresar letras, luego indicar cuantas representan a las letras B, el problema termina cuando se ingresa la letra C.
- c) Se ingresan 50 números que representan a un mes del año, luego indicar cuántos de esos números representaron a verano e invierno.
- d) Desarrolle un algoritmo que permita ingresar 15 números y luego indicar la suma de los que fueron mayores a 100.
- e) Desarrolle un algoritmo que permita ingresar 50 números y luego indicar la suma de los que fueron múltiplos de 2 y la resta de los que fueron múltiplos de 7; no considerar a los que son múltiplos de ambos.
- f) Se ingresan los montos de las ventas de una tienda (100 ventas), indicar cuántos de esos montos fueron menores a 50 soles, cuantos están en el rango de 50 a 100, cuantos están en el rango de 100 a 200 y cuantos sobrepasaron de 200.
- g) Se va ingresar el nombre de 100 países, mediante un programa indicar cuantos pertenecen a:
- América del Norte: (USA, Canadá)
 - América Central: (México, El Salvador, Cuba, Panamá)
 - América del Sur: (Perú, Chile, Colombia, Brasil, Argentina)
 - Considerar SOLO lo indicado anteriormente.
- h) Se ingresan 200 números de 3 cifras indicar la suma de todos los que se consideran capicúas
- i) Se ingresa la edad de 100 personas, indicar que cantidad de edades ingresadas pertenecen a cada etapa del desarrollo cognitivo, queda a criterio el colocar si toma o no algún extremo, eso sí, no se deben perder valores.

○ [**Etapa Sensomotora 0-2 años**] ○

○ [**Etapa Preoperacional 2-7 años**] ○

○ [**Etapa de las Operaciones
Concretas 7-11 años**] ○

○ [**Etapa de las Operaciones
Formales 11-15 años**] ○

- j) Indique la suma de las siguientes sucesiones:

$$S = 3 + 6 + 9 + 12 + 15 + 18 + \dots$$

Para 100 números.

$$S = 2/3 + 4/4 + 6/5 + 8/6 + 10/7 + \dots$$

Para 100 números.

7. 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.