PRELIMINARES

Una estructura de control iterativa ó ciclo, permite realizar varias veces el mismo conjunto de operaciones. El ciclo se repite mientras se cumple alguna condición lógica.

En Python existen dos tipos de estructuras de repetición (ó iterativas): **`while`** y **`for`**.

Ciclos con estructura de control `While`

La estructura **`While`** evalúa una condición lógica **antes** de comenzar cada iteración. Si la condición es verdadera, entonces se ejecutan las instrucciones que están dentro de la estructura **`While`**. En caso contrario, el ciclo termina y se sigue con la ejecución de las instrucciones escritas fuera del ciclo. Las instrucciones que se consideran dentro del ciclo son aquellas que están tabuladas de tal forma que se encuentran a la derecha de la palabra reservada **`while`**.

La sintaxis de la estructura **`while`** en Python es la siguiente:

```
while (condición):
   <instrucciones del while>
<primera instrucción fuera del while>
```

Es decir, el conjunto de instrucciones se repetirá **mientras** la condición sea verdadera. Si la condición es falsa ya desde un principio, el conjunto de instrucciones no se ejecuta nunca.

Por ejemplo, el siguiente código informa si el número ingresado por teclado por el usuario es positivo ó negativo, y se detiene cuando el usuario ingresa el número 0:

```
numeroUsuario=int(input("Ingresa un número, si quieres terminar ingresa el 0: "))
while (numeroUsuario!=0):
    if (numeroUsuario>0):
        print("Es positivo")
    else:
        print("Es negativo")
    numeroUsuario=int(input("Ingresa otro número (0 para terminar): "))
print("Fin del ciclo Mientras")
```

En este ejemplo, si se ingresa 0 la primera vez, la condición (numero!=0) será falsa. Por lo tanto, ni siquiera entra al bloque del Mientras, saliendo del ciclo y continuando con la siguiente instrucción que es:
`print(“Fin del ciclo Mientras”);`

Repetición con instrucción for

Una estructura de repetición for, permite iterar una ó más instrucciones. El número de veces que se repiten las instrucciones está definido por una variable de control (normalmente llamada contador). La sintaxis es la siguiente:

```
for nombreVariable in range(valorInicial,valorFinal,ModificaciónRango):
       <instrucciones>
```

Donde `nombreVariable` es el nombre de la variable de control (normalmente se le llama contador), **range(valorInicial,valorFinal,ModificaciónContador)** genera los números sobre los cuales iterará la variable `nombreVariable`. Los números generado por la función range van desde `<valorInicial>` hasta `<valorFinal>-1`, con un incremento ó decremento de `<ModificaciónRango>`.

Por ejemplo, para la siguiente estructura for:

```
for contador in range(1,6,1):
      print("#")
```

la función range(1,6,1) genera los números 1, 2, 3, 4, 5. Por lo cual, contador tomará cada uno de esos valores. La siguiente tabla muestra cómo va cambiando el valor de la variable contador (columna izquierda) junto con lo que va apareciendo en pantalla (columna derecha, entre paréntesis se muestran los símbolos que fueron impresos por pantalla en iteraciones anteriores):

Contador	Despliegue por pantalla
1	#
2	(#)#
3	(##)#
4	(###)#
5	(####)#

Al finalizar la ejecución del ciclo, en pantalla quedan desplegados 5 símbolos #.

Nota que se podría realizar lo mismo, con un ciclo **while de la siguiente manera:**

Ciclo implementado con `Mientras`	**Equivalencia al ciclo** `Para`
`Contador=1`	`for contador in range(1,6,1)`
`while (contador<6):`	`    print("#")`
`    print("#")`	
`    Contador=contador+1`	

Sin embargo, el uso de for es más compacto (2 líneas en vez de 4 líneas para ciclo while).

PROBLEMAS PROPUESTOS

1. Escribe un algoritmo que resuelva el siguiente problema: mostrar por pantalla los primeros 10 números pares. Resuelve este problema de 3 maneras distintas: usando ciclo while y ciclo for.

Una solución posible con estructura de control while
Esta solución usa una variable para contar el número de iteraciones y otra para almacenar el número de desplegar por pantalla.

```
contador=1
par=2
while (contador<=10):
    print(par)
    contador=contador+1
    par=par+2
```

Otra solución posible (con menos variables) con estructura de control for
Esta solución aprovecha la misma variable contador para contar el número de iteraciones y desplegar lo solicitado por pantalla.

```
for contador in range(1,11,1):
    print(contador*2)
```

2. Escribe un algoritmo que resuelva el siguiente problema: mostrar por pantalla las primeras 4 potencias de 2. Resuelve este problema de 3 maneras distintas: usando ciclo while, ciclo for.

Una solución posible con estructura de control while
Esta solución usa una variable para contar el número de iteraciones y otra para almacenar el número de desplegar por pantalla.

```
contador=1
potencia=2**1
while (contador<=4):
    print(potencia)
    contador=contador+1
    potencia=2**contador
```

Una solución posible con estructura de control for
Esta solución aprovecha la misma variable contador para contar el número de iteraciones y desplegar lo solicitado por pantalla.

```
for contador in range(1,5,1):
    print(2**contador)
```

3. Escribe un algoritmo que resuelva el siguiente problema: mostrar por pantalla los primeros N números pares. N es un valor ingresado por el usuario. Resuelve este problema de 2 maneras distintas: usando ciclo while y ciclo for.

Una solución posible con estructura de control while

```
N=int(input("¿Cuántos números pares quieres desplegar? "))
contador=2
while(contador<=2*N):
    print(contador)
    contador=contador+2
```

Una solución posible con estructura de control for

```
N=int(input("¿Cuántos números pares quieres desplegar? "))
contador=2
for contador in range(2,2*N+1,2):
    print(contador)
```

4. Escribe un algoritmo que resuelva el siguiente problema: desplegar por pantalla todos los números enteros que existen entre los números A y B (ambos inclusive) que ingresa el usuario. Nota que si A es menor que B, se despliegan los números en orden creciente. Pero si A es mayor que B, los números se despliegan en orden decreciente. A continuación se muestran 2 ejemplos de la ejecución del algoritmo:

Ejemplo 1
Ingresa el primer número
>3
Ingresa el segundo número
>7
3
4
5
6
7

Ejemplo 2
Ingresa el primer número
>8
Ingresa el segundo número
>5
8
7
6
5

Una solución posible

Por restricciones de espacio, a partir de ahora solo se muestra una de tantas soluciones posibles. Siempre ten presente que para un mismo problema existen varios algoritmos capaces de resolverlo.

```
N1=int(input("ingrese el primer número "))
N2=int(input("ingrese el segundo número "))
if (N1<=N2):
    for contador in range(N1,N2+1,1):
        print(contador)
else:
    for contador in range(N1,N2-1,-1):
        print(contador)
```

5. Escribe un algoritmo en Python que emule la solicitud de clave en una cajero automático: si el usuario equivoca la clave 3 veces consecutivas, el cajero muestra el mensaje "Tarjeta retenida", se "traga" la tarjeta (esta última acción no la puedes emular en POython ya que la instrucción "Tragar tarjeta" no existe) y termina la ejecución del algoritmo. Si el usuario logra ingresar la clave válida en uno de los 3 primeros intentos, entonces aparece el mensaje "Bienvenid@ al servicio". Supón que la clave válida se encuentra almacenada en una variable.

```
claveValida=711
numeroIntentos=0
while (numeroIntentos<3):
    claveIngresada=int(input("Ingresa tu clave: "))
    numeroIntentos=numeroIntentos+1

    if (claveIngresada==claveValida):
        numeroIntentos=3

if (claveIngresada==claveValida):
    print("Bienvenid@ al servicio")
else:
    print("Tarjeta retenida")
```

6. Escribe un algoritmo en Python que solicite números enteros al usuario, y vaya calculando la suma de todos ellos. La solicitud de números al usuario termina cuando se ingresa un **-1**, en cuyo caso se muestra por pantalla el resultado de la suma. Está permitido que el usuario ingrese como primer valor un -1, en cuyo caso el algoritmo termina inmediatamente,

```
suma=0
numIngresado = int(input("Ingrese un número entero o -1 para terminar "))

while (numIngresado!=-1):
    suma=suma+numIngresado
    numIngresado=int(input("Ingrese un número entero o -1 para terminar "))
print("La suma total de números es ", suma)
```

7. Escribe el algoritmo que permite desplegar por pantalla los números pares que se encuentran entre dos números ingresados. El menor valor se almacena en la variable numInferior y el mayor enl la variable numSuperior.

```
numInferior=int(input("Ingrese un número "))
numSuperior=int(input("Ingrese otro número "))
if (numInferior>numSuperior):
    temp=numInferior
    numInferior=numSuperior
    numSuperior=temp
for cont in range(numInferior,numSuperior+1,1):
    if (cont%2==0):
        print("Número par: ",cont)
```

8. Escribe en código un algoritmo que solicite al usuario el número de empleados de una empresa y luego calcule el salario a pagar a cada uno de ellos. Para ello, para cada empleado, debe solicitar los siguientes datos: nombre, número de horas semanales trabajadas por el empleador (NHT), valor de la hora trabajada (VHC). Luego, el salario se calcula como NHT*VHC.

```
numEmpleados=int(input("Ingrese el número de empleados "))
for cont in range(1,numEmpleados+1,1)
    nombreEmpleado=input("Ingrese nombre del empleado ")
    NHT=int(input("Ingrese número de horas trabajadas en la semana "))
    VHT=int(input("Ingrese valor por hora trabajada "))
    print("El salario de ", nombreEmpleado, " es: ", NHT*VHT)
```

9. Escribe el código de un algoritmo que calcule el factorial de un número ingresado. El factorial de un número **n** (denominado **n!**) se calcula como: **n*(n-1)*(n-2)*(n-3)*..1**.

```
N=int(input("Ingrese un número "))
fact=1
for cont in range(1,N+1,1):
    fact=fact*cont
print("El factorial de ", N, " es: ", fact)
```

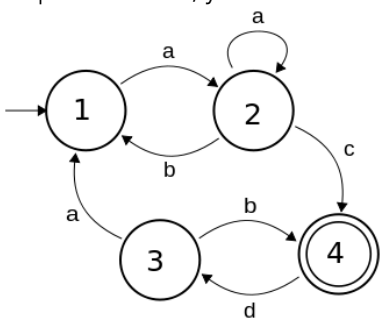
10. Escribe el pseudo-código de un algoritmo que solicita, de uno en uno, los datos de género de un grupo de personas y determina el porcentaje de hombres ,mujeres y personas que no desean entregar esta información. Para género masculino el usuario debe ingresar 'm' , '**f**' para el femenino y 'n' si no desea entregar esta información. El programa termina cuando se ingresa el caracter '**x**'.

```
numHombres=0
numMujeres=0
numSinInfo=0
genero="N"

while(genero!="x"):
    print("Ingrese genero de la persona (x para terminar)")
    genero = input("f:femenino, m:masculino,n:no deseo entregar esa información ")
    if(genero=="M" or genero=="m"):
        numHombres=numHombres+1
    if (genero=="F" or genero=="f"):
        numMujeres=numMujeres+1;
    if (genero=="N" or genero=="n"):
        numSinInfo=numSinInfo+1;

total=numHombres+numMujeres+numSinInfo
print("El porcentaje de mujeres es: ",numMujeres/total)
print("El porcentaje de hombres es: ",numHombres/total)
print("El porcentaje de personas sin información es: ",numSinInfo/total)
```

11. La figura de abajo representa las diferentes posiciones (identificadas con números dentro de círculos) por las que atraviesa una máquina de un casino muy especial, con la que se puede jugar con fichas del tipo **a, b, c** y **d**. Escribe el pseudo-código de un algoritmo que, comenzando por la posición inicial **1,** solicite una ficha a jugar en cada instante. Dependiendo del valor de la ficha, se va avanzando en las posiciones hasta intentar llegar a la posición final **4**. La siguiente posición de la máquina se determina a partir de la posición actual y la ficha que se inserta, y así sucesivamente.



Por ejemplo, si se está en la posición 1 y se ingresa una ficha a, se avanza a la posición 2 (se queda en 1 en otro caso).

El juego termina cuando el usuario presiona n. En ese caso, si la máquina se encuentra en la posición 4, el usuario **gana el juego**. Si se encuentra en cualquier otra posición, pierde.

```
posicion=1
ficha="a"
while (ficha!="n"):
    ficha=input("Ingrese una ficha (a,b,c,d) o n para terminar: ")
    if (posicion==1):
        if(ficha=="a"):
            posicion=2
    else:
        if (posicion==2):
            if (ficha=="b"):
                posicion=1
            if (ficha=="c"):
                posicion=4
        else:
            if (posicion==3):
                if (ficha=="a"):
                    posicion=1
                if (ficha=="b"):
                    posicion=4
            else:
                if (posicion==4):
                    if (ficha=="d"):
                        posicion=3
if (posicion==4):
    print("Felicitaciones, ganaste")
else:
    print("Perdiste, quedaste en la posición: ", posicion)
```

12. Escribe el código que calcula el resto de la división entre dos enteros positivos A y B, para A y B indicados por el usuario.

```
numA=int(input("Ingrese dividendo "))
numB=int(input("Ingrese divisor "))
resto = numA
while(resto >= numB):
    resto = resto - numB
print("El resto es ",resto)
```

13. Escribe el código que permite determinar si un número entero positivo P es primo. El valor de P se solicita al usuario.

```
numP=int(input("Ingrese el valor P: "))
esPrimo = 1
for posibleDivisor in range (2,numP,1):
    if (numP%posibleDivisor==0):
        esPrimo = 0
if (esPrimo == 0):
    print("El número ingresado no es primo")
else:
    print("El número ingresado es primo")
```

14. Escribe el código que permite resolver la siguiente serie: 1!+ 2! + 3! + …+ N!. El valor de N se solicita al usuario.

```
num=int(input("ingrese el valor N de la serie "))
fact=1
total=0
for i in range(1,num+1):
    fact=fact*i
    total=fact+total
print("el valor de la serie es ",total)
```

15. Realice un programa que determine si un número es palindrome. Un número palindrome es aquel que se lee de la misma manera de izquierda a derecha, como de derecha a izquierda, por ejemplo: 131 y 7887.

```
num=int(input("Ingrese un numero: "))
newNum=0
numOrig=num
while num>0:
    newNum=newNum*10+num%10
    num=int(num/10)
if (newNum == numOrig):
    print("numero palindrome")
else:
    print("numero NO palindrome")
```

16. Realice un programa que solicite números al usuario hasta que ingrese un 0 y devuelva el menor de todos los números ingresados que sean distintos a 0.

```
num=int(input("Ingrese un numero: "))
num=1
minNum=0
while (num!=0):
    num=int(input("Ingrese un numero: "))
    if (minNum==0):
       minNum=num
    else:
       if (num!=0) and (num<minNum):
          minNum=num

if (minNum==0):
     print("numero no valido")
else:
     print("el menor numero es ",minNum)
```