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`**. En esta guía veremos solamente la primera estructura.

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. La instrucciones que se consideran dentro del ciclo son aquellas que están entre la palabra reservada **`While`** y **`la última instrucción indentada`**.

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

```
while (condición) :
   <instrucciones del bloque while>
<instrucciones después 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 o negativo, y se detiene cuando el usuario ingresa el número 0:

```
#código en Python para determinar si número es positivo o negativo
print( "Ingresa un número, si quieres terminar ingresa el 0" )
numero_usuario = int( input() )

while ( numero_usuario != 0 ) :
        if ( numero_usuario > 0 ):
                print( "Es positivo" )
        else :
                print( "Es negativo" )

        print( "Ingresa otro número (0 para terminar)" )
        numero_usuario = int( input() )

print( "Fin del ciclo while" )
```

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 **while**, saliendo del ciclo y continuando con la siguiente instrucción que es: `print("Fin del ciclo while")`

PROBLEMAS PROPUESTOS

1. Escribe un algoritmo que resuelva el siguiente problema: mostrar por pantalla los primeros 10 números pares. Resuelve este usando ciclo While.

Una solución posible

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 While

Esta solución aprovecha la misma variable contador para contar el número de iteraciones y desplegar lo solicitado por pantalla.

```
contador = 2
while ( contador<=20 ) :
    print( contador )
    contador = contador + 2
```

2. Escribe un algoritmo que resuelva el siguiente problema: mostrar por pantalla las primeras 4 potencias de 2. Resuelve este problema usando ciclo While.

Una solución posible

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
```

Otra solución posible (con menos variables) con estructura de control While

Esta solución aprovecha la misma variable contador para contar el número de iteraciones y desplegar lo solicitado por pantalla.

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

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 usando ciclo While.

Una solución posible

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

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

```
print("Ingresa el primero número" )
Num1 = int( input() )
print("Ingresa el segundo número" )
Num2 = int( input() )
if ( Num1 == Num2 ):
  print( Num1 )
else:
  if ( Num1 < Num2 ):
      while ( Num1 <= Num2 ):
        print ( Num1 )
        Num1 = Num1 + 1
  else:
      while ( Num1 >= Num2 ):
        print ( Num1 )
        Num1 = Num1 - 1
```

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 Python 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 "Bienvenido al servicio". Supón que la clave válida se encuentra almacenada en una variable.

```
clave_valida = 711
numero_intentos = 0
while ( True ):
  print( "Ingresa tu clave" )
  clave_ingresada = int( input() )
  numero_intentos = numero_intentos + 1
  if ( clave_ingresada == clave_valida) or (numero_intentos == 3):
    break
if ( numero_intentos < 3 ):
  print( "Bienvenido 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
print( "Ingrese un número entero o -1 para terminar " )
num_ingresado = int( input() )
while ( num_ingresado != -1 ):
  suma = suma + num_ingresado
  print( "Ingrese un número entero o -1 para terminar " )
  num_ingresado = int( input() )
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 num_inferior y el mayor en la variable num_superior.

```
print( "Ingrese dos números enteros" )
num_inferior = int( input() )
num_superior = int( input() )
if ( num_inferior > num_superior ):
  temp=num_inferior
  num_inferior=num_superior
  num_superior=temp

contador = num_inferior
while ( contador < num_superior ):
  if ( contador%2 == 0 ):
    print( "Número par: ", contador )
  contador = contador + 1
```

8. Escribe el código de 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.

```
print( "Ingrese el número de empleados " )
num_empleados = int( input() )
contador = 0
while (contador < num_empleados):
  print( "Ingrese nombre del empleado " )
  nombre_empleado = input()
  print( "Ingrese número de horas trabajadas en la semana" )
  NHT = int( input() )
  print( "Ingrese valor por hora trabajada " )
  VHT = int( input() )
  print( "El salario de ", nombre_empleado, " es: ", NHT*VHT )
  contador = contador + 1
```

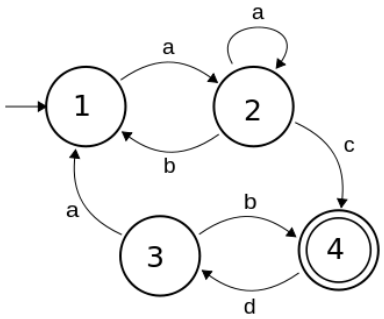
9. Escribe el código de un algoritmo que calcule el factorial de un número ingresado. El factorial de un número **n** (denominado **n!**) se calcula como: **n*(n-1)*(n-2)*(n-3)*..1**.

```
print( "Ingrese un número " )
num = int( input() )
total  = 1
cont = 1
while(cont<=num):
  total = total * cont
  cont = cont + 1
print( "El factorial de ", num, " es: ", total )
```

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

```
num_mujeres = 0
num_hombres = 0
num_sin_info = 0
while( True ):
  print( "Ingrese género de la persona (x para terminar)" )
  print( "f:femenino, m:masculino,n:no deseo entregar esa información" )
  genero = input()
  if (genero=="F" or genero=="f"):
    num_mujeres=num_mujeres+1;
  elif (genero=="M" or genero=="m"):
    num_hombres=num_hombres+1;
  elif (genero=="N" or genero=="n"):
    num_sin_info=num_sin_info+1;
  elif genero == 'x':
    break
total=num_hombres+num_mujeres+num_sin_info
print( "El porcentaje de mujeres es: ",num_mujeres/total )
print( "El porcentaje de hombres es: ",num_hombres/total  )
print( "El porcentaje de personas sin información es: ",num_sin_info/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 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
while( True ):
  print( "Ingrese una ficha (a,b,c,d) o n para terminar" )
  ficha = input()
  if (posicion==1):
    if (ficha=="a"):
      posicion=2
  else:
    if (posicion==2):
      if (ficha=="b"):
        posicion=1
      elif (ficha=="c"):
        posicion=4
    else:
      if (posicion==3):
        if (ficha=="a"):
          posicion=1
        elif (ficha=="b"):
          posicion=4
      else:
        if (posicion==4):
          if (ficha=="d"):
            posicion=3
  if (ficha=="n"):
    break
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.

```
print( "Ingrese dividendo y divisor" )
numA = int( input() )
numB = int( input() )
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.

```
print( "Ingrese el valor P" )
numP = int( input() )
es_primo = 1
posible_divisor = 2
while( posible_divisor < numP ):
  if ( numP % posible_divisor == 0 ):
    es_primo = 0
  posible_divisor = posible_divisor + 1
if ( es_primo == 0 ):
  print( "El número ingresado no es primo" )
else:
  print( "El número ingresado es primo" )
```

14. Escribe el código que escriba los N primeros números primos. El valor de N se solicita al usuario.

```
print( "Ingrese el valor N" )
numN = int( input() )

posible_primo = 1
while(numN > 0):
  es_primo = 1
  posible_divisor = 2
  while( posible_divisor < posible_primo ):
    if ( posible_primo % posible_divisor == 0 ):
      es_primo = 0
    posible_divisor = posible_divisor + 1
  if ( es_primo == 1 ):
    print( posible_primo )
    numN = numN - 1
  posible_primo = posible_primo + 1
```