El futuro digital es de todos

MinTIC

Hechos QUE CONECTAN

CICLO 1

EJE TEMÁTICO 3

VARIABLES DE CONTROL Y CICLOS ITERATIVOS

Universidad Industrial de Santander

UIS

MisionTIC2022

Variables de control y ciclos iterativos

Ruta de aprendizaje

Variables de control y ciclos iterativos

Este recurso busca dar a entender qué son las variables de control en programación y cómo haciendo uso de estas, son estructurados los ciclos iterativos. Las variables de control y los ciclos iterativos son elementos que le dan lógica a la funcionalidad de los algoritmos o códigos.

Palabras claves: Ciclo, Estructuras, Iterativo, Variables

Contextualización

En la vida diaria hay muchos eventos que suceden cada cierto tiempo, haciéndolo ya sea a un periodo constante o no. Entre estos eventos se encuentran el ciclo del agua, las rutas del transporte público, la nómina mensual, entre otros. Si un domiciliario decide ir al supermercado con una lista de compras y se acerca a la caja, la cajera debe tomar cada uno de los productos que piensa llevar y escanearlos para incluirlos en la factura. A estos proceso se le conoce como **proceso iterativo**.

Ahora, si el domiciliario requiere realizar la compra de los productos que se necesitan en un restaurante de lunes a sábado, este sería un proceso iterativo adicional al que realiza la cajera. Sería un proceso iterativo dentro de otro. En donde el proceso iterativo externo tiene una condición que exige que los únicos días en los cuales el domiciliario hará la compra serán los lunes, los martes, los miércoles, los jueves, los viernes y los sábados, exceptuando el día domingo. En programación, nos encontramos muy a menudo con procesos de este estilo, los cuales son optimizados haciendo uso de variables de control y estructuras de código conocidas como **ciclos iterativos**.

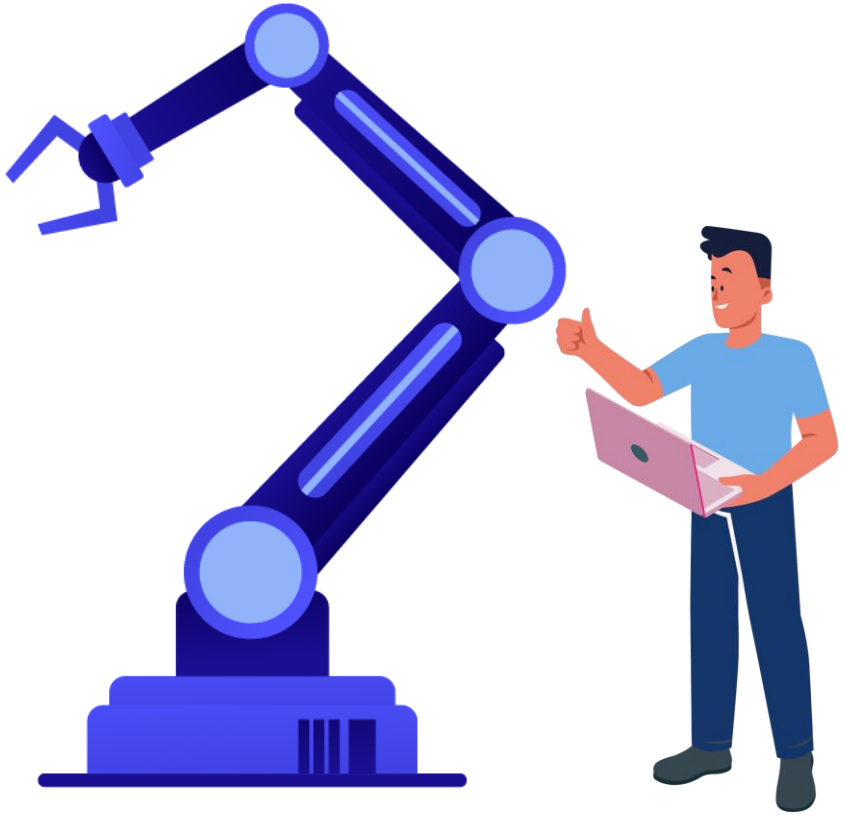
Introducción

¡Hola! De seguro has estado practicando un montón con Python! En este punto tendrás muy claro los motivos y la manera de utilizar variables, condicionales y los distintos operadores aritméticos y lógicos. Incluso, ya podrías generar uno que otro algoritmo de inteligencia artificial, haciendo uso de las estructuras de control condicionales. Por lo tanto, ya te encuentras preparado para aprender y entender las estructuras de control iterativas; estoy seguro de que lo disfrutarás.

```
for item in data:
    result = []
    new_data = process(item)
    value = calculate(new_data)
    if value < 0:
        result.append(value-1)
    elif value > 0:
        result.append(value+1)
    else:
        result.append(0)
```

Introducción

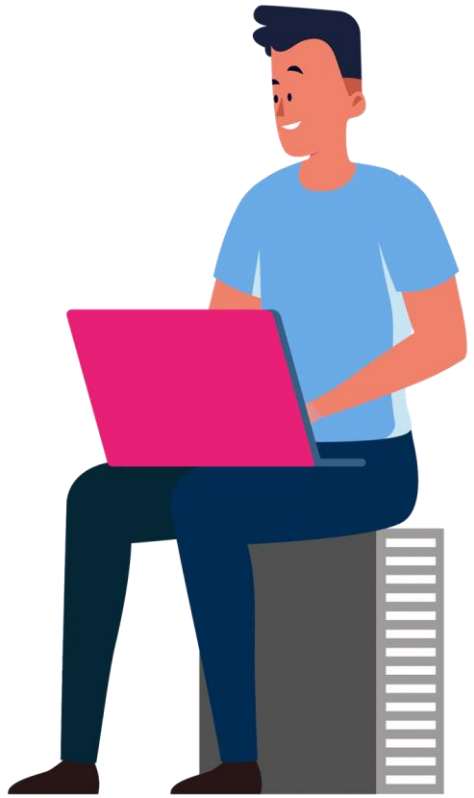
Comencemos por lo importante, ¿Por qué son necesarias las estructuras de control iterativas? De seguro has visto en documentales y películas brazos robóticos o máquinas de llenado ejecutando una misma tarea una y otra vez, así como lo harías si tuvieras que ir todos los días una y otra vez al mercado. En la antigüedad, estas tareas eran realizadas por humanos; sin embargo, dadas las capacidades de las máquinas, las pueden hacer de manera rápida y **repetitiva**. Déjame aclarar que la automatización es un área de la ciencia muy amplia, pero estás a punto de conocer un elemento básico para que la automatización en su nivel más simple sea posible.



Introducción

En este punto notarás que hemos mencionado mucho las palabras “repetición”, “iteración”, y ahora apareció un concepto nuevo que es la “**automatización**”, puesto que es la base para entender las estructuras de control iterativas. Por lo tanto, vamos a dejar a un lado la palabra “repetición” y la vamos a reemplazar por “iteración” ¿De acuerdo?

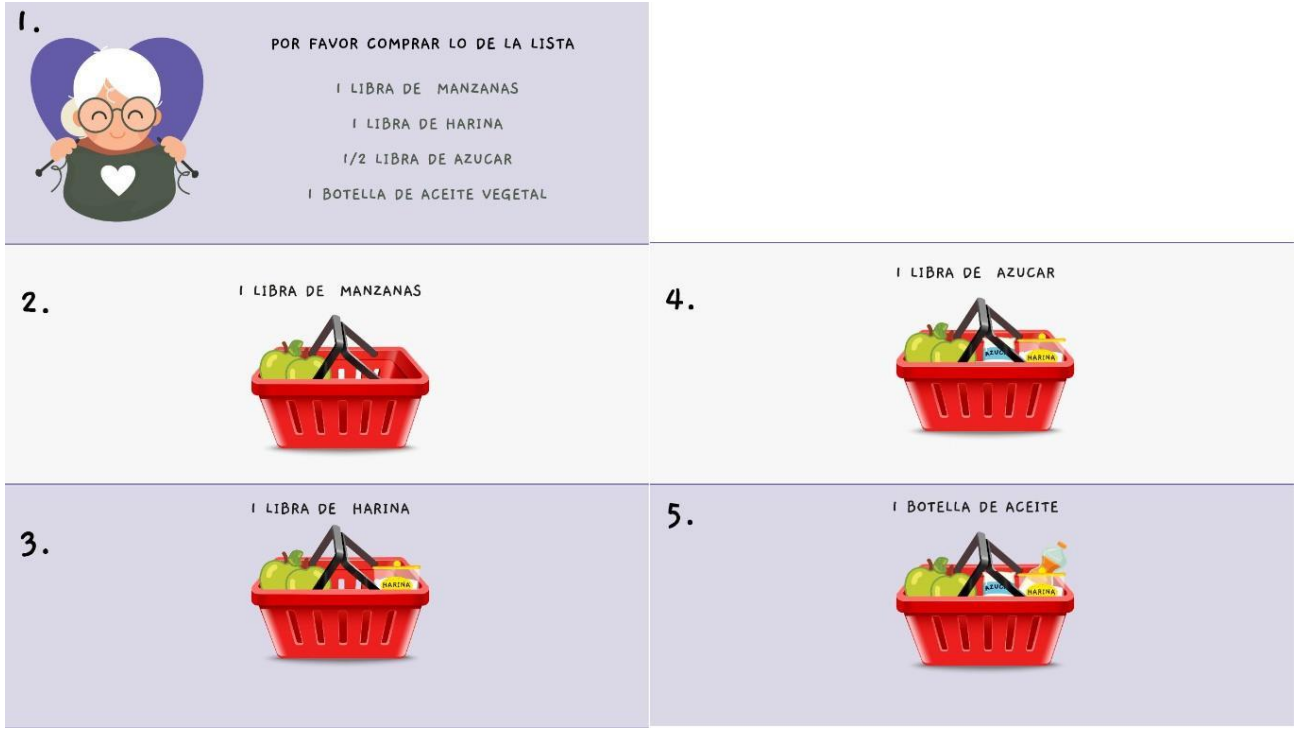
Entonces, las estructuras de control iterativas son nuestra herramienta para ejecutar en los programas bloques de código de manera iterativa hasta que una o más condiciones se cumplan; ya vamos a ir entendiendo un poco más las condiciones y las distintas estructuras, pero por el momento ten claro el objetivo principal de las estructuras de control iterativas y su relación con la automatización.



Variables de control

Las **variables** son claves en las **estructuras de control iterativas,** puesto que son el puente entre las iteraciones y la condición para que esta se ejecute. Normalmente y a lo largo de tu rol como programador, encontrarás tres tipos de variables de control principales: las **banderas**, los **acumuladores** y los **contadores**.

Variables de control

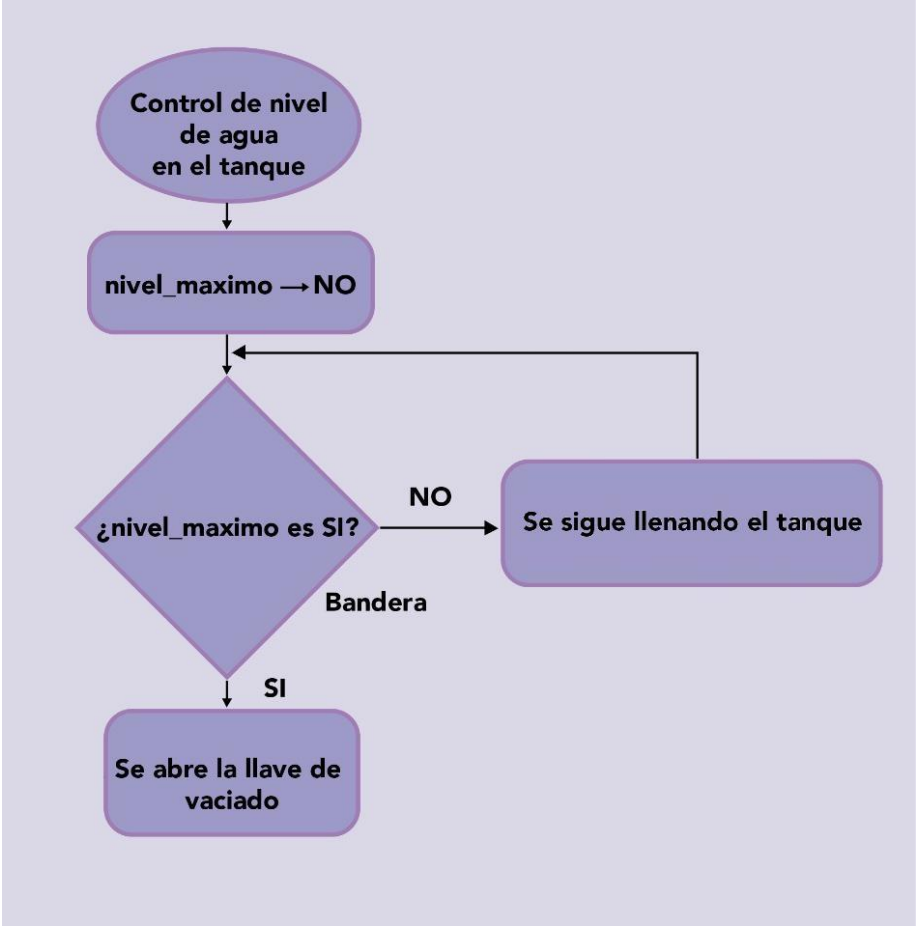


El proceso de compra en un supermercado tiene mucha similitud con las **variables de control**, ya que por ejemplo si no encontramos un producto no lo llevamos o lo reemplazamos (**banderas**). Al hacer la compra vamos almacenando los ítems en el carrito de compras (**acumuladores**). Al pensar en todos los elementos de la lista, chequeamos la cantidad de ítems que debemos comprar (**contadores**).

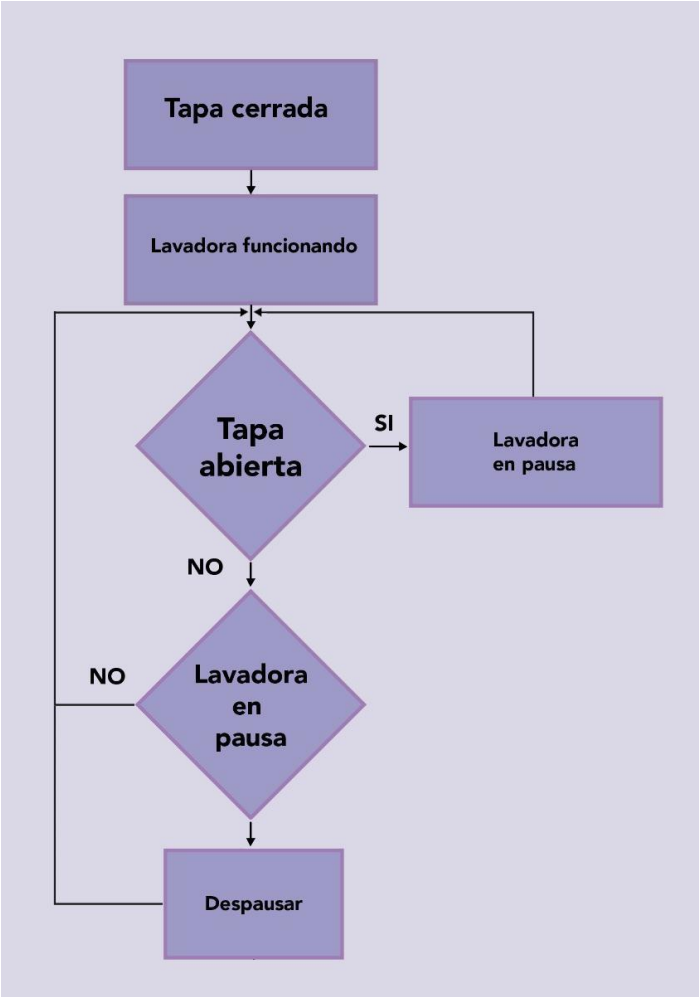
Variables de control. Banderas

Las **banderas** hacen referencia a variables que toman un valor preferiblemente **binario**, booleano e indican un estado; su valor y el cambio del mismo, definen el estado en el que se encuentra el programa.

Por ejemplo, se requiere llevar el **control del nivel de agua de un tanque**. Esta tarea sería análoga al diagrama de flujo de la derecha.



Variables de control. Banderas



Un proceso en el cual vemos el uso de variables tipo **banderas** es el del **sistema de seguridad de la tapa de una lavadora**.

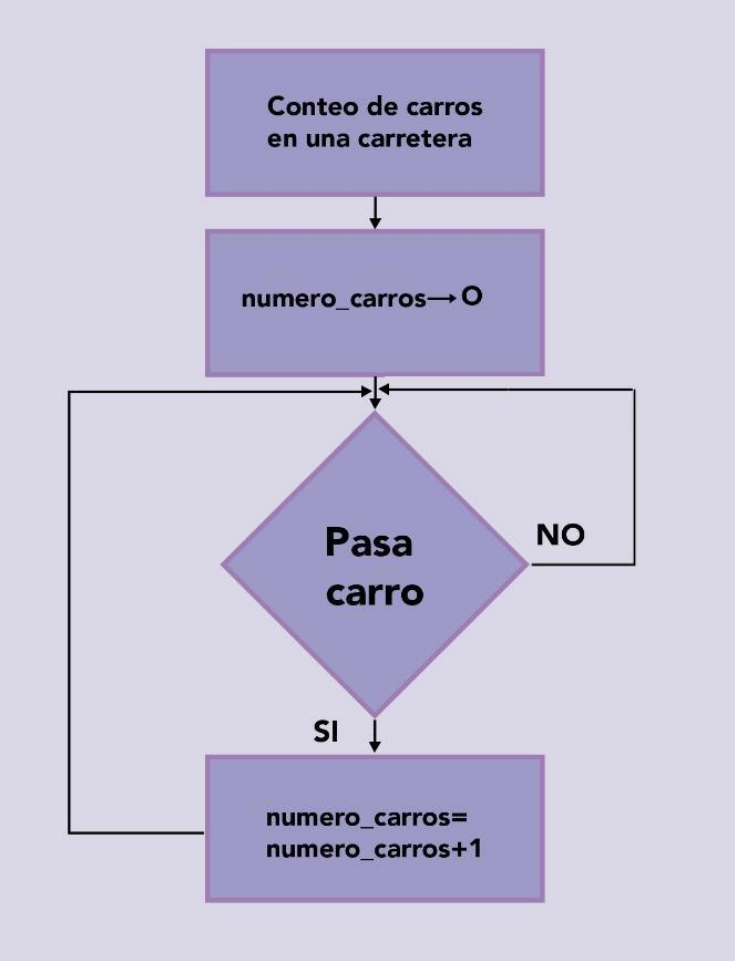
La lavadora solamente **funciona con la tapa cerrada**, si la lavadora está en funcionamiento y se abre la tapa, está se pondrá en **pausa**. Si se cierra la tapa, la lavadora vuelve a ponerse en funcionamiento. Este proceso es mostrado en el diagrama de flujo de la imagen de la derecha.

Variables de control. Acumuladores

El objetivo de este tipo de variables es "**almacenar**" algún tipo de información en una sola variable a medida que se va ejecutando el programa.

Un proceso acumulativo se puede mostrar con el siguiente ejemplo. Si se tiene que **repartir una porción de torta** a ciertas personas, y se quiere obtener al final una **lista de las personas que comieron torta**, se puede hacer con una variable de control tipo **acumulativa**. Este proceso es mostrado en la figura de la derecha.

Variables de control. Contadores



En esta sección aprenderás a hacer uso de los **contadores**, de hecho, están al final de las variables de control, dado que tienen mucha relación con los **acumuladores** de la sección anterior.

Los **contadores** son variables de control que como su nombre lo indica, controlan la **cantidad** de veces que se ejecuta determinada acción, estado, etc. Un ejemplo puede ser un **programa para determinar el flujo de carros en una carretera**, como se muestra en la figura de la derecha.

Revisión de lo aprendido

1. **¿Cuáles variables de control existen?**

    a. Banderas, acumuladores y contadores
    b. Banderas, iteradores y contadores
    c. Banderas, acumuladores e iteradores
    d. acumuladores, iteradores, controladores

Revisión de lo aprendido

2. ¿Cómo se define la variable de control tipo bandera?

a. Por lo general se define como una variable tipo booleana
b. Siempre se define como una variable entera
c. Se suele definir como una variable cadena de caracteres
d. Se define usando variables tipo entero

Revisión de lo aprendido

3. ¿Qué función tiene una variable de control tipo acumulador?

a. Acumula información dentro de una variable
b. Suma la información anterior con la actual
c. Almacena el último valor de toda la información
d. Ninguna de las opciones es correcta

Revisión de lo aprendido

4. ¿Qué función tiene una variable de control tipo contador?

a. Controla la cantidad de veces que se ejecuta una acción
b. Controla el número de variables dentro de un bloque de código
c. Controla el número de veces que el programa es ejecutado
d. Ninguna de las opciones es correcta

Ciclos iterativos

Los **ciclos iterativos** son fragmentos de códigos que permiten condensar la **repetición de tareas**, es decir, si tenemos que resolver una tarea varias veces pero con diferentes condiciones o parámetros, se puede hacer uso de los ciclos iterativos. En los lenguajes de programación existen diferentes maneras de implementar los ciclos iterativos, entre las que se destacan:

- **Los ciclos iterativos controlados por condiciones (bucles while)**
- **Los ciclos iterativos controlados por cantidad (bucles for)**

Bucle while	Bucle for

`valor = 0`
`while valor < 100:`
`    # do something`
`    valor = valor + 1`
	`data = [0,1,2, ..., 99]`
	`for valor in data:`
	`    # do something`

Ciclos iterativos.
Ciclos controlados por condiciones

Ya terminaste de entender las **variables de control**. Estas son la base fundamental de los **ciclos de control**, pues ya notarás que son las encargadas de **determinar el flujo de tu algoritmo**. Ahora te encuentras preparado para empezar por el **ciclo controlado por condiciones**, específicamente el **ciclo while**.

La traducción del nombre de este ciclo al español es "mientras" y verás que su nombre nos indica muy específicamente de su funcionamiento. El **ciclo while** se compone de una condición y su bloque de código; este bloque de código se ejecutará **mientras** que la condición da como resultado **Verdadero** o **True**.

La estructura de un ciclo while es la siguiente:

```
while <condición>:
    <código a ejecutar si se cumple la condición>
    <modificación de la condición>
```

Ciclos iterativos.
Ciclos controlados por condiciones

Un ejemplo de un **ciclo while** es (código optimizado):

```
i = 0
while i < 5:
    print(i)
    i = i + 1
```

Cuyo resultado sería:

```
0
1
2
3
4
```

Ciclos iterativos.
Ciclos controlados por condiciones

Código repetitivo y **cambio de bandera**, valor de `i` (código no optimizado):

```
i = 0
print(i)
i = i + 1
print(i)
i = i + 1
print(i)
i = i + 1
print(i)
i = i + 1
print(i)
```

Ciclos iterativos.
Ciclos controlados por cantidad

De seguro ahora que entendiste los ciclos **while,** te habrás imaginado una serie de situaciones en las que este ciclo tiene una relación directa con un **contador**, y no necesariamente con una condición de verdad, es decir, estarás imaginando diferentes situaciones, como la del **contador de carros en una carretera**, en el que nuestra condición está directamente relacionada con el conteo de elementos y por lo tanto, el uso de contadores.

El ciclo for es una herramienta muy poderosa, cuyos elementos de iteración pueden ser rangos, cadenas de caracteres, estructuras de datos complejas o cualquier otro elemento iterable que tenga una **cantidad** definida, la cual marca la diferencia respecto al **ciclo while**, ya que mientras que el **ciclo while** parte de una **condición de verdad**, el **ciclo for** parte de una **cantidad** definida.

La estructura de un **ciclo for** es la siguiente:

```
for <variable> in <elemento iterable>:
    <código a ejecutar si se cumple la condición>
```

Ciclos iterativos.
Ciclos controlados por condiciones

Un ejemplo de un ciclo for es (código no optimizado):

```
data = [0, 1, 2, 3, 4]
for i in data:
    print(i)
```

Cuyo resultado sería:

```
0
1
2
3
4
```

Ciclos iterativos.
Ciclos controlados por condiciones

Código repetitivo y **cambio de índice**, valor de `i` (código no optimizado):

```
i = 0
print(data[i])
i = 1
print(data[i])
i = 2
print(data[i])
i = 3
print(data[i])
i = 4
print(data[i])
```

Revisión de lo aprendido

1. ¿Qué código permite obtener como resultado `it = 5`?

a.
```
value = 0
it = 0
while value < 10:
    if value > 5:
        value = value + 3
    else:
        value = value + 2
    it = it + 1
```

b.
```
value = 0
it = 0
while value < 10:
    if value > 5:
        value = value*2
    else:
        value = value + 1
    it = it + 1
```

c.
```
value = 0
it = 0
while value < 10:
   if value > 5:
        value = value*2
   else:
        value = value + 2
   it = it + 1
```

d.
```
value = 0
it = 0
while value < 10:
     if value > 5:
      value = value + 3
   else:
        value = value + 1
   it = it + 1
```

Revisión de lo aprendido

2. Complete la siguiente información acerca de bucles tipo for:

```
data = [1, 2, 3, 4, 5]

    _____ idx ____  ______ :
        print(idx)
```

Revisión de lo aprendido

3. El siguiente código duplica cada valor de la lista items

```
items = [3, 4, 5, 6]
for value in items:
    items = 2*items
```

a) Verdadero
b) Falso

Revisión de lo aprendido

4. ¿Qué función cumple el siguiente ciclo for?

```
for value in items:
    if value is not None:
    print(value)
```

a. Filtrar los valores nulos de la lista ítems e imprimir aquellos que no lo son.

b. Filtrar los valores nulos de la lista ítems e imprimir aquellos que son nulos.

c. Almacenar los valores no nulos en la lista ítems.

d. Recorrer los valores no nulos únicamente.

El futuro digital es de todos

MinTIC

Hechos QUE CONECTAN

CICLO 1

EJE TEMÁTICO 3

VARIABLES DE CONTROL Y CICLOS ITERATIVOS

Universidad Industrial de Santander

MisionTIC 2022