P1

Programación 1

UNIDAD 03

CLASE 03

ESTRUCTURAS REPETITIVAS & CONTADORES

- Estructuras repetitivas
- Estructura While
- Contadores
- Acumuladores

Al final de esta clase ya podrás:

- Comprender situaciones problemáticas que para su resolución impliquen el uso de ciclos repetitivos con diferentes características.
- Valorar la conveniencia de aplicar contadores y acumuladores para facilitar el diseño de algoritmos eficientes.

ISSD

-Des-
Desarrollo de Software

MÓDULO DIDÁCTICO

2023

Estructuras repetitivas

¡Bienvenidos a la tercera clase de Programación 1!

En esta oportunidad nos introduciremos a las **Estructuras Repetitivas** y trabajaremos con situaciones problemáticas ligeramente más complejas que nos permitan aplicar y afianzar nuestro conocimiento.

Específicamente, hoy estudiaremos dos temas:
- El primero de ellos aborda las **Estructuras Repetitivas y el uso de la función "While"**.
- El segundo nos introducirá el uso de un **"Contador"** para almacenar datos.

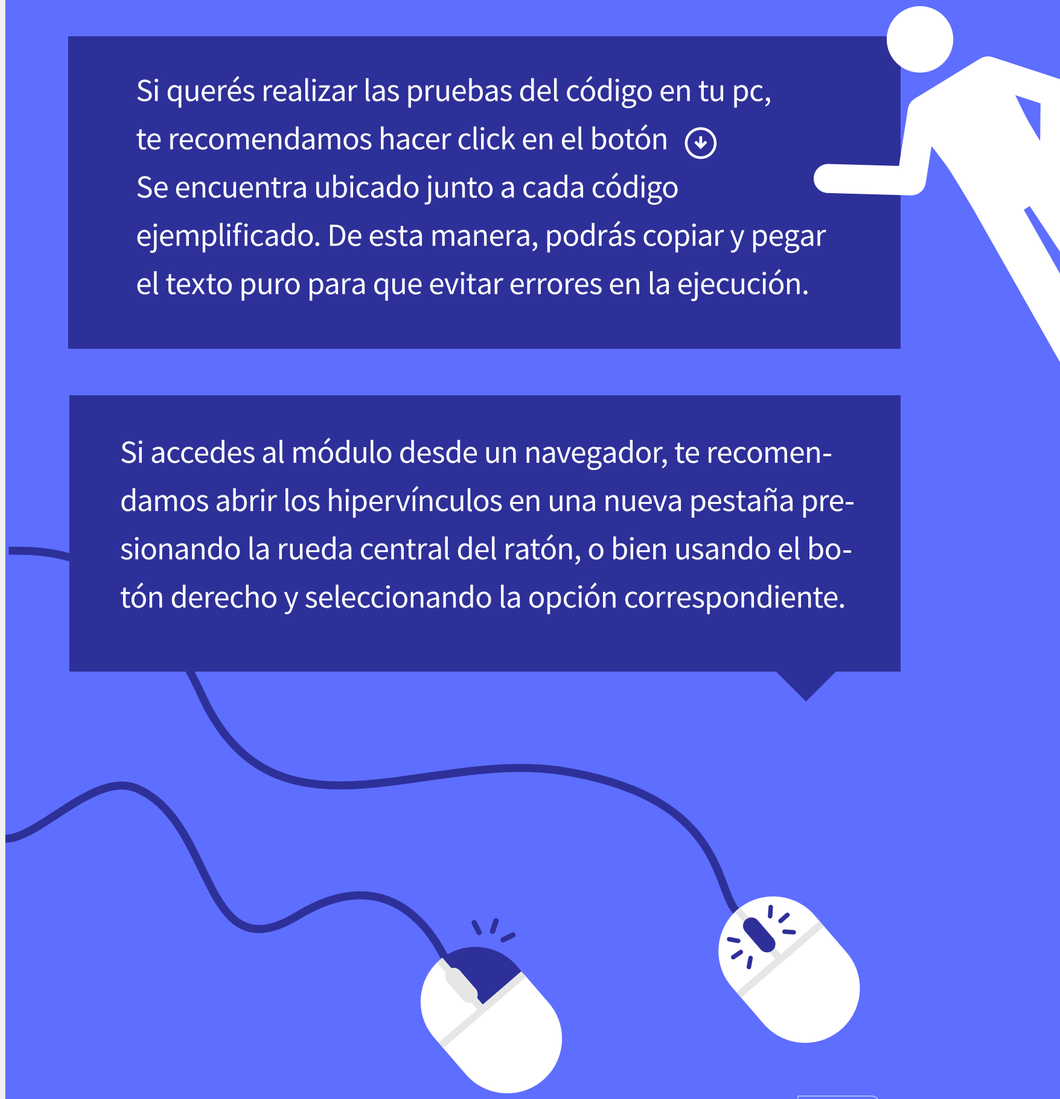
A lo largo de la clase te encontrarás con ejercicios para que realices vos y también con otros ejemplos que iremos comentando y explicando para que practiquemos juntos.

Durante la última clase, aprendimos a diseñar programas que pudieran tomar decisiones mediante las estructuras condicionales. Pero hay veces que estos problemas requieren soluciones más creativas como crear un ciclo que se repita "n" cantidad de veces.

¡Nos vemos!

Si querés realizar las pruebas del código en tu pc, te recomendamos hacer click en el botón ⓘ
Se encuentra ubicado junto a cada código ejemplificado. De esta manera, podrás copiar y pegar el texto puro para que evitar errores en la ejecución.

Si accedes al módulo desde un navegador, te recomendamos abrir los hipervínculos en una nueva pestaña presionando la rueda central del ratón, o bien usando el botón derecho y seleccionando la opción correspondiente.



Antes de comenzar con la clase, leé el siguiente artículo.

Lavarropas y Wi-Fi, ¿para qué?

Internet de las Cosas. La empresa cordobesa Alladio presentó su lavarropas Drean Next, que incorpora conectividad Wi-Fi. Describimos su modo de funcionamiento, su integración con una aplicación móvil y las posibilidades que esta tecnología abre a futuro para la marca.

Los electrodomésticos conectados dejaron de ser una promesa del futuro: no sólo han llegado a nuestro país, sino que ya se producen localmente, como lo confirma el lanzamiento del lavarropas Drean Next, un producto íntegramente desarrollado en nuestra provincia por los ingenieros de Alladio con la colaboración de empresas tecnológicas locales.

Wi-Fi, ¿para qué?

Un sector fundamental de la Internet de las Cosas (IoT, por sus siglas en inglés) es el de los electrodomésticos conectados, que abren un nuevo abanico de posibilidades en cuanto a control remoto e integración en el hogar inteligente.

El lavarropas Drean Next se constituye en el primer electrodoméstico desarrollado en el país que integra conectividad. Se complementa con una aplicación móvil que permite su operación desde un teléfono, tablet o cualquier dispositivo conectado a Internet. Además, comunica al usuario el estado del proceso de lavado y centrifugado, brinda avisos para el mantenimiento preventivo, permite acceder a tutoriales, actualizar el software y personalizar programas, entre otras funciones.

A la hora de pensar en utilidades concretas, desde la empresa plantean diversos escenarios frecuentes en la vida actual. Por caso, los usuarios de estos equipos podrán dejar la carga lista e iniciar el proceso de lavado remotamente, para que al volver a casa la ropa esté recién lavada y lista para secar. O, frente a una demora imprevista, activar desde la app móvil un nuevo proceso de enjuague, para que las prendas no tomen mal olor. También resulta muy útil recibir en el smartphone las notificaciones de finalización del lavado.

La inteligencia de este lavarropas no sólo radica en sus posibilidades de conectividad: además, reconoce el programa más utilizado y permite darle inicio con sólo presionar un botón. Del mismo modo, es posible modificar

los programas de lavado predefinidos de fábrica para crear nuevas versiones personalizadas de ellos.

Recopilación de datos

Uno de los temas sobre los que más se discute a la hora de hablar de la Internet de las Cosas y de electrodomésticos conectados es el de la privacidad de los datos recopilados por los fabricantes. En esta línea, desde Alladio destacan que recolectarán información de uso de sus productos con el único propósito de contribuir a su proceso de mejora continua. Así, Agustín Roberi, director y gerente de Marketing de la firma, sostiene que los datos recopilados *"nos permitirán contar con información fidedigna sobre patrones de uso que volcaremos en nuevos desarrollos, aprendiendo mucho más sobre cómo usa la gente nuestros lavarropas: por ejemplo, sus horarios, con qué carga, en qué programas".*

El ejecutivo arriesga que *"en el futuro, según la zona del país en la cual se esté utilizando, podremos indicarle al lavarropas que emplee más o menos cantidad de jabón al lavar".*

Tecnología cordobesa

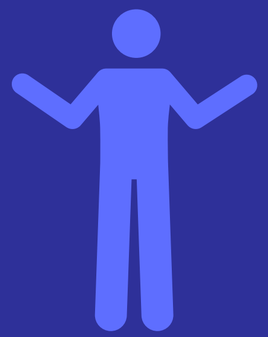
Tanto la aplicación móvil para el control remoto como la electrónica propia del lavarropas fueron desarrolladas por tecnológicas locales. La app pertenece a Geminus-Qhom, mientras que el módulo de Wi-Fi fue desarrollado por Linetec, otro emprendimiento local que hace tiempo viene fabricando las plaquetas utilizadas en los lavarropas cordobeses.

"Es todo un desafío, porque estamos entre las primeras empresas a nivel internacional en introducir esto para lo que, además, las partes vitales del producto también son de empresas cordobesas. O sea, somos pioneros con un desarrollo bien autóctono, que se ubica a la vanguardia del lavado de ropa en el mundo", indica Marcos Alladio. Esto abre perspectivas de generar un ecosistema de electrodomésticos conectados e interactuando en el hogar, tomando en cuenta que la firma produce bajo otras denominaciones comerciales tanto heladeras como hornos de microondas, extractores, anafes y hornos eléctricos.

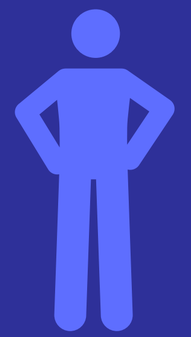
La Voz del Interior – 11/7/2015

La programación está invadiendo hasta las áreas más impensadas: ¿lavarropas controlados por WiFi?

Pero lo más importante, es que no se trata de una exposición futurista que se desarrolla en los países centrales, sino de productos que se fabrican en nuestra provincia y con desarrollos de hardware y software construidos por profesionales y empresas cordobesas.



La Internet de las cosas ya está entre nosotros, y vos te estás preparando para ser protagonista de este futuro inmediato ¡A programar!



Estructura Repetitiva “While”

Hasta ahora hemos empleado estructuras SECUENCIALES y CONDICIONALES. Existe otro tipo de estructuras tan importantes como las anteriores que son las estructuras **REPETITIVAS**.

Una estructura repetitiva permite ejecutar una instrucción o un conjunto de instrucciones varias veces. Una ejecución repetitiva de sentencias se caracteriza por:

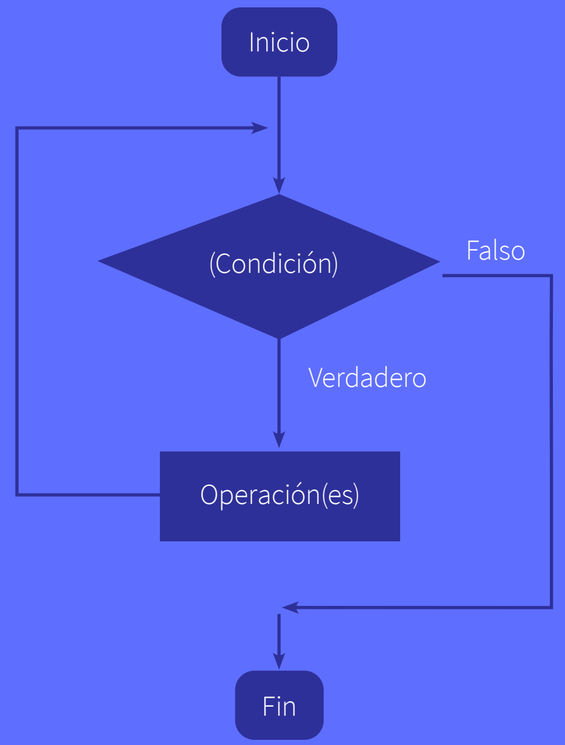
| La sentencia o las sentencias que se repiten.

| El test o prueba de condición antes de cada repetición, que motivará que se repitan o no las instrucciones.

El bloque se repite MIENTRAS la condición sea Verdadera.

Veamos un ejemplo de su aplicación |1|.

|1|



Si analizamos el gráfico |1| con cuidado, notaremos que la representación es similar a la estructura condicional “If”, pero no debemos confundirlo.

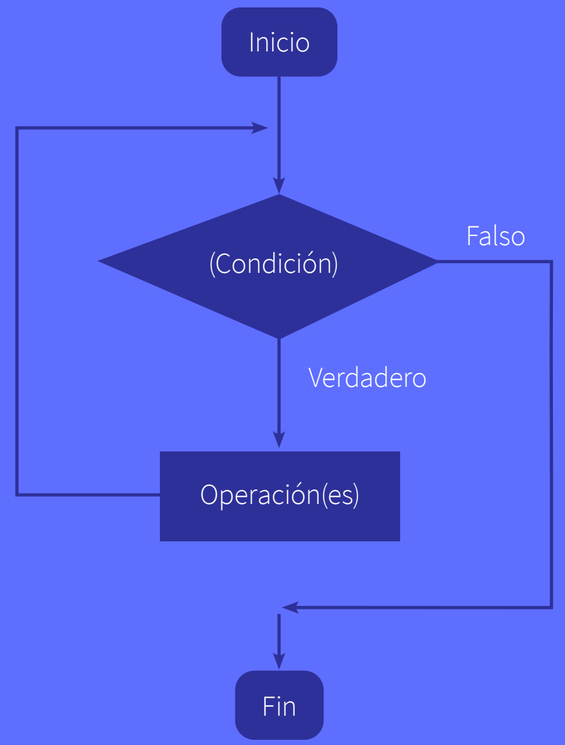
Su funcionamiento es el siguiente:
| En primer lugar se verifica la condición, si la misma resulta **verdadera** se ejecutan las operaciones que indicamos por la ***rama del Verdadero.***
| A la rama del verdadero la graficamos en la parte inferior de la condición.
| Una línea al final del bloque de repetición conecta la operación con la parte superior de la estructura repetitiva y ***vuelve a ejecutar la condición.***
| En caso que la condición sea **Falsa** continúa por la ***rama del Falso*** y **sale** de la estructura repetitiva para continuar con la ejecución del algoritmo.

El bloque se repite MIENTRAS la condición sea Verdadera.

¡Importante!

Si la condición **siempre retorna verdadero** estaremos en presencia de un ciclo repetitivo infinito. Dicha situación es un error de programación lógico, nunca finalizará el programa.

|1|



Ejemplo 11

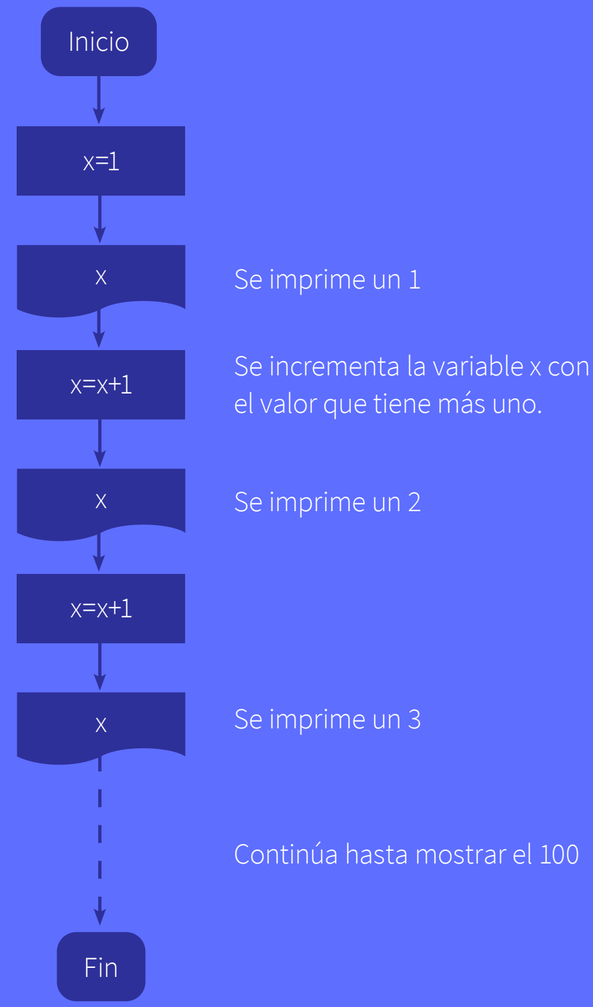
Enseñándole a contar al programa

Realizá un programa que imprima en pantalla los números del 1 al 100.

| Pensemos un poco: Sin conocer las estructuras repetitivas podemos resolver el problema empleando una estructura secuencial. Iniciamos una variable con el valor 1, luego imprimimos la variable, incrementamos nuevamente la variable y así sucesivamente.

Si continuamos con el diagrama |2| veríamos que es **casi interminable**.
Emplear una estructura secuencial para resolver este problema produce un diagrama de flujo y un programa en Python muy largo.

|2|



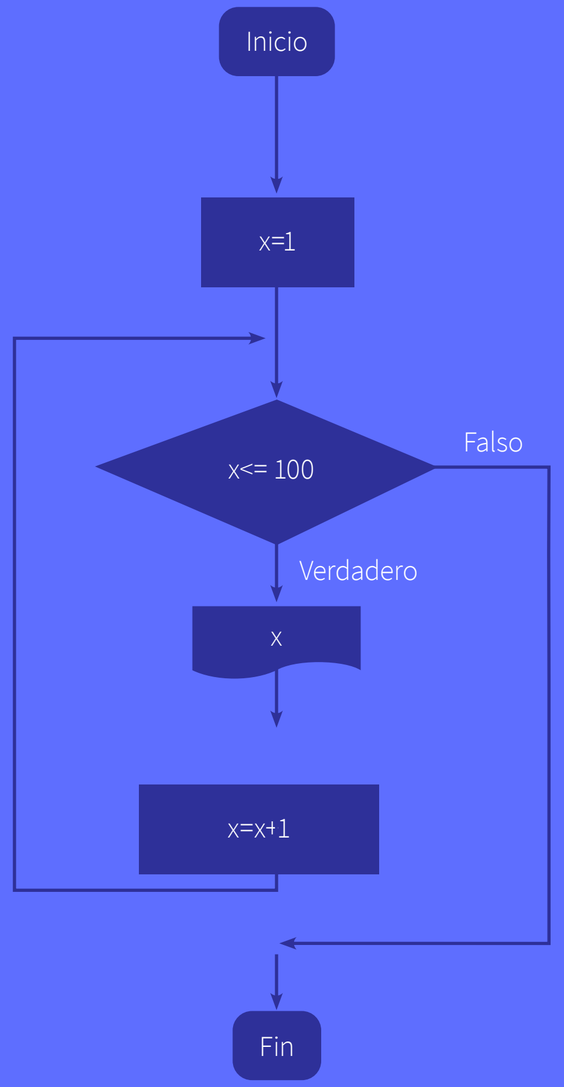
| Ahora veamos la solución empleando una estructura repetitiva while:

Es muy importante analizar este diagrama: |3|
La primera operación inicializa la variable **x en 1,** seguidamente comienza la estructura repetitiva while y disponemos la siguiente condición (x <= 100), ***se lee MIENTRAS la variable x sea menor o igual a 100.***

Al ejecutarse la condición retorna VERDADERO porque el contenido de x (1) es menor o igual a 100.
Al ser la condición verdadera se ejecuta el bloque de instrucciones que contiene la estructura while.
El bloque de instrucciones contiene una salida y una operación.
Se imprime el contenido de x, y seguidamente se incrementa la variable x en uno.

La operación ***x=x+1*** se lee como "**en la variable x se guarda el contenido de x más 1**".
Es decir, ***si x contiene 1,*** luego de ejecutarse esta operación se almacenará en x un **2**.

|3|

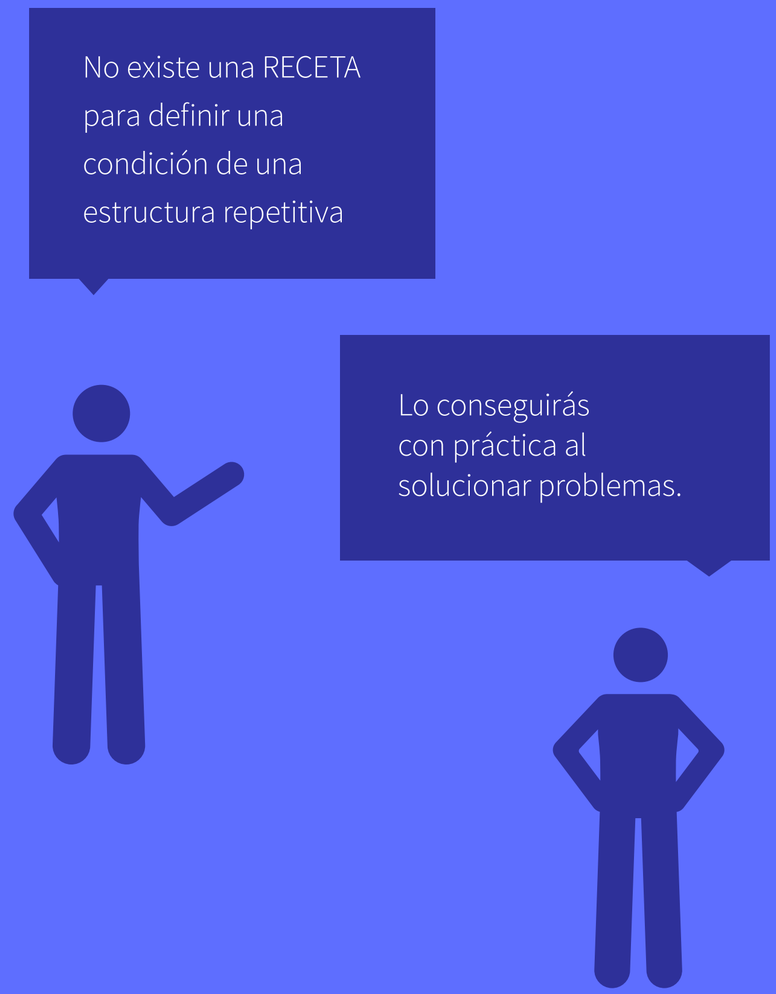


Al finalizar el bloque de instrucciones que contiene la estructura repetitiva, se verifica nuevamente la condición de la estructura repetitiva y se repite el proceso explicado anteriormente.

Lo más complejo, es definir correctamente la condición de la estructura while y definir las instrucciones que se repetirán.

Observá que si, por ejemplo, disponemos la condición x >=100 ***(si x es mayor o igual a 100)*** no provoca ningún error sintáctico pero estamos en presencia de un **error lógico** porque al evaluarse por primera vez la condición retorna falso y no se ejecuta el bloque de instrucciones que queríamos repetir 100 veces.

Una vez planteado el diagrama debemos verificar si el mismo **es una solución válida al problema** (en este caso se debe imprimir los números del 1 al 100 en pantalla), para ello podemos hacer un seguimiento del flujo del diagrama y los valores que toman las variables a lo largo de la ejecución:



A continuación te compartiremos la visualización del diagrama anterior:

x
1
2
3
4
.
.
100
101 Cuando x vale 101 la condición de la estructura repetitiva retorna falso,
en este caso finaliza el diagrama.

¡Importante!

Podemos observar que el bloque repetitivo puede no ejecutarse ninguna vez si la condición retorna falso la primera vez. La variable x debe estar inicializada con algún valor antes que se ejecute la operación x=x+1 en caso de no estar inicializada aparece un error de compilación.

No te olvides de inicializar la "x" al comenzar la operación

Programa:

```
x=1
while x<=100:
    print(x)
    x=x+1
```

¡Importante!

Recordá nuestro problema no estará 100% solucionado si no codificamos la devolución de los resultados buscados en pantalla o el medio requerido.

Es importante notar que seguido de la palabra clave **while** disponemos la condición y finalmente los dos puntos. Todo el código contenido en la estructura repetitiva debe estar indentado (normalmente a cuatro espacios)

Desempeño 17

Soluciones

| Probemos algunas modificaciones de este programa y veamos que cambios se deberían hacer para los siguientes enunciados:

a| Imprimir los números del 1 al 500.
b| Imprimir los números del 50 al 100.
c| Imprimir los números del -50 al 0.
d| Imprimir los números del 2 al 100 pero de 2 en 2 (2,4,6,8 ....100).

Cuando te sientas listo, hacé click en el botón "Soluciones" para verificar tus respuestas

Ejemplo 12

Enseñandole a contar hasta el número solicitado

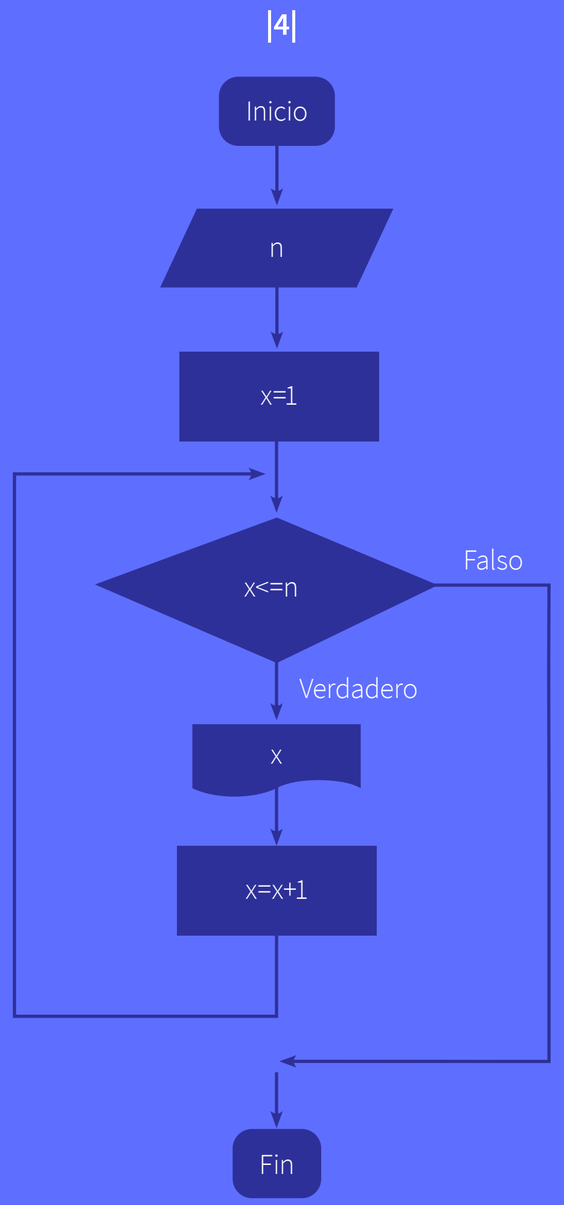
Realizá un programa que solicite la carga de un número positivo y nos muestre desde 1 hasta el valor ingresado de uno en uno de uno en uno.

| Por ejemplo: Si ingresamos 30 se debe mostrar en pantalla los números del 1 al 30.

Es de **FUNDAMENTAL** importancia analizar los diagramas de flujo y la posterior codificación en Python de los siguientes problemas, ya que en varios problemas posteriores se presentarán situaciones no vistas en el ejercicio anterior.

| Mirando el diagrama de este ejercicio **|4|** con atención podremos observar que se ingresa por teclado la variable n, es decir que el operador puede cargar cualquier valor.

Si el usuario carga "**10**", el bloque repetitivo se ejecutará **10 veces**, ya que la condición es “Mientras x<=n ”, es decir “mientras x sea menor o igual a 10”. Tené en cuenta que x comienza en uno y se incrementará en uno cada vez que se ejecuta el bloque repetitivo.



A la prueba del diagrama |4| la podemos realizar dándole valores a las variables; por ejemplo, si ingresamos 5 visualizamos en pantalla el siguiente resultado:

n	x
5	1 (Se imprime el contenido de x)
2	" "
3	" "
4	" "
5	" "
6	(Sale del while porque 6 no es menor o igual a 5)

Programa:

```
n=int(input("Ingrese el valor final:"))
x=1
while x<=n:
    print(x)
    x=x+1
```

¡Importante!

Los nombres de las variables **n** y **x** pueden ser palabras o letras (como en este caso)
La variable x recibe el nombre de **CONTADOR**.
Un contador es un tipo especial de variable que se incrementa o disminuye con valores constantes durante la ejecución del programa.

En esta situación, el contador x nos indica en cada momento la cantidad de valores impresos en pantalla.

Ejemplo 13

Enseñandole a sumar a nuestro programa

Realizá un programa que permita la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y también su promedio.

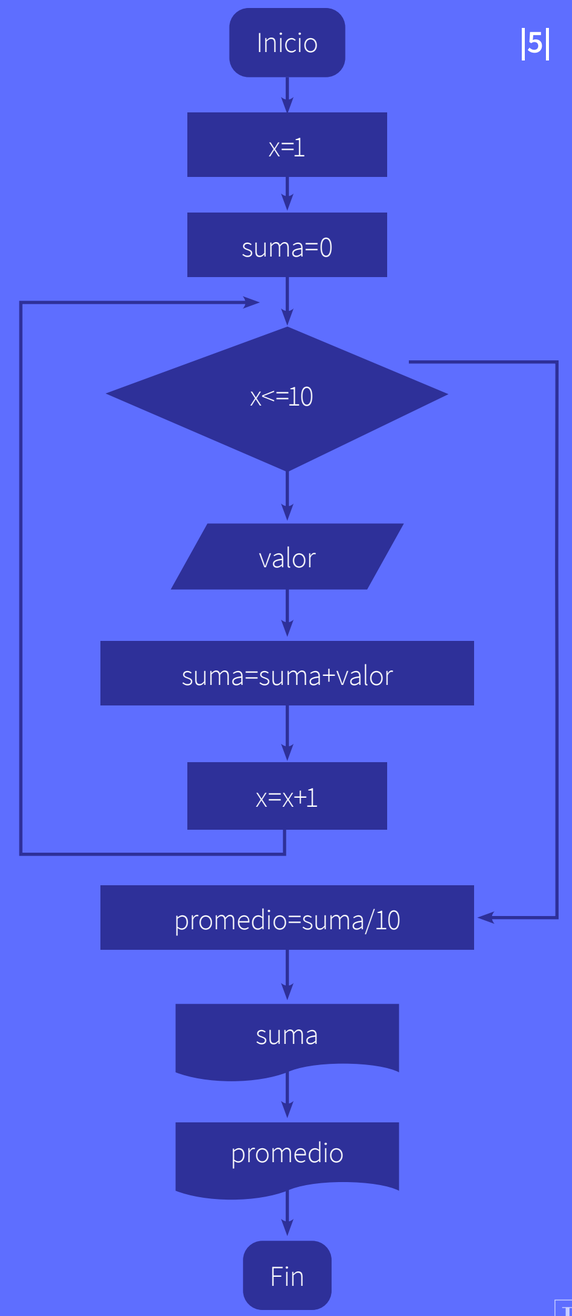
| Mirando el diagrama de este ejercicio |5| observaremos que es semejante a los anteriores, aquí tenemos un **CONTADOR** llamado x que nos sirve para contar las vueltas que debe repetir el while.

| También aparece el concepto de **ACUMULADOR** (un acumulador es un tipo especial de variable que se incrementa o disminuye con valores variables durante la ejecución del programa)

| Hemos dado el nombre de **suma** a nuestro acumulador.

Cada ciclo que se repita nuestra estructura repetitiva, la variable suma se incrementará con el contenido ingresado en la variable valor.

|5|



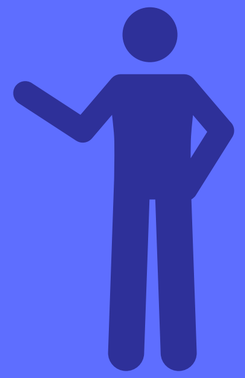
Realizaremos la prueba a este diagrama dándole valores a las variables y observando su resultado.

valor	suma	x	promedio
	0	0	
5	5	1	
16	21	2	
7	28	3	
10	38	4	
2	40	5	
20	60	6	
5	65	7	
5	70	8	
10	80	9	
2	82	10	
8	90	11	
			9

¡Importante!

Cuando en la variable valor se carga el primer valor (en este ejemplo **5**) **al cargarse el segundo valor (16) el valor anterior 5 se pierde**, por ello la necesidad de ir almacenando en la variable suma los valores ingresados.

La primera linea muestra los valores antes de entrar a la estructura repetitiva.



Los números que toma la variable valor dependerá de qué cifras cargue el operador durante la ejecución del programa.

El promedio se calcula al salir de la estructura repetitiva (es decir primero sumamos los 10 valores ingresados y luego los dividimos por 10)

Programa:

```
x=1
suma=0
while x<=10:
    valor=int(input("Ingrese un valor:"))
    suma=suma+valor
    x=x+1
promedio=suma/10
print("La suma de los 10 valores es")
print(suma)
print("El promedio es")
print(promedio)
```

El resultado del promedio es un valor real es decir con coma. Si queremos que el resultado de la división solo retorne la parte entera del promedio debemos utilizar el operador // :

```
promedio=suma//10
```

El interprete de Python sabe que el promedio se calcula al finalizar el while ya que se encuentra codificado en la columna 1. Las tres instrucciones contenidas en el while están indentadas.

Ejemplo 14

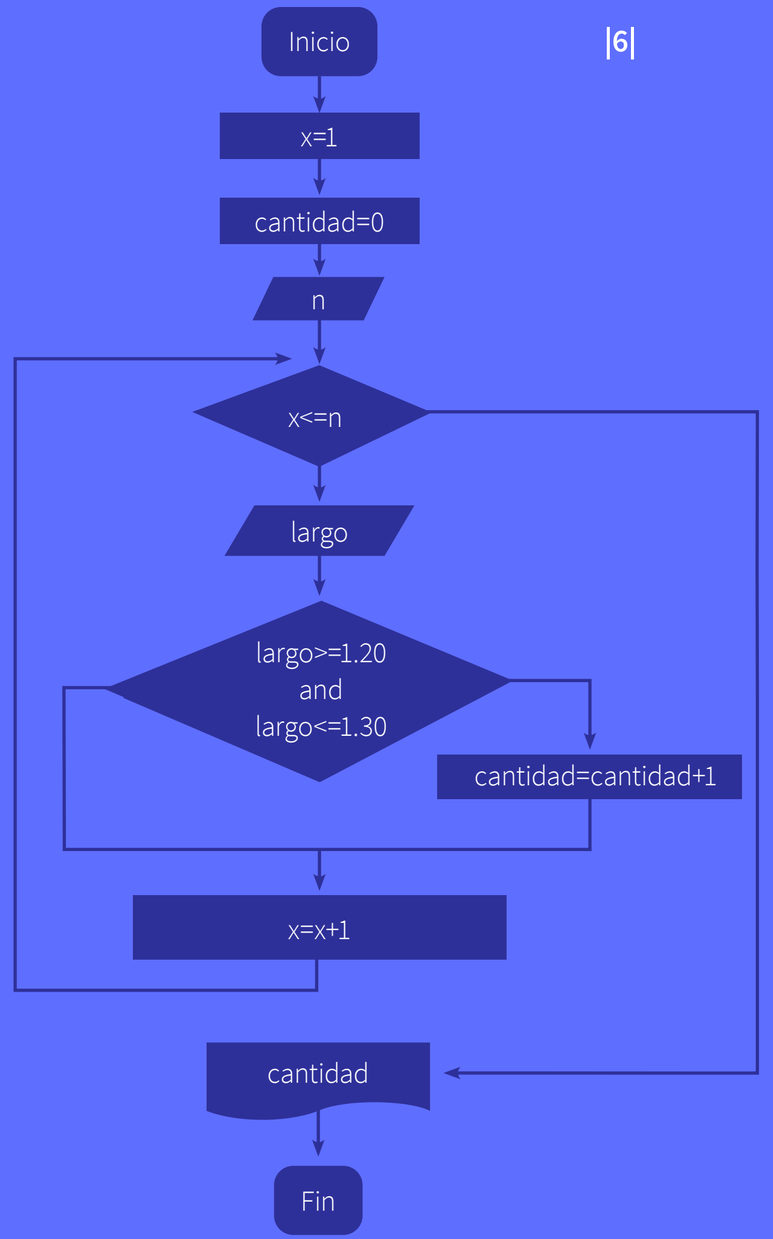
Contador de piezas aptas para stock

Una planta que fabrica perfiles de hierro posee un lote de n piezas.
Confeccionar un programa que pida ingresar por teclado la cantidad de piezas a procesar y luego ingrese la longitud de cada perfil; sabiendo que la pieza cuya longitud esté comprendida en el rango de 1.20 y 1.30 son aptas. Imprimir por pantalla la cantidad de piezas aptas que hay en el lote.

| Mirando el diagrama de este ejercicio |6| podemos observar que dentro de una estructura repetitiva puede haber estructuras condicionales (inclusive puede haber otras estructuras repetitivas que veremos más adelante)

En este problema hay que cargar inicialmente la cantidad de piezas a ingresar (n), seguidamente se cargan n valores de largos de piezas.

Cada vez que ingresamos un largo de pieza **(largo)** verificamos si es una medida correcta (debe estar entre 1.20 y 1.30 el largo para que sea correcta), en caso de ser correcta la CONTAMOS (incrementamos la variable **cantidad** en 1)



Al contador **cantidad** lo inicializamos en cero porque inicialmente no se ha cargado ningún largo de pieza. Cuando salimos de la estructura repetitiva porque se han cargado n largos de piezas mostramos por pantalla el contador **cantidad** (que representa la cantidad de piezas aptas)

En este problema tenemos **2 CONTADORES**:

- x (Cuenta la cantidad de piezas cargadas hasta el momento)
- cantidad (Cuenta los perfiles de hierro aptos)

Programa:

```
cantidad=0
x=1
n=int(input("Cuantas piezas cargara:"))
while x<=n:
    largo=float(input("Ingrese la medida de la pieza:"))
    if largo>=1.2 and largo<=1.3:
        cantidad=cantidad+1
    x=x+1
print("La cantidad de piezas aptas son")
print(cantidad)
```

Cuando queremos cargar por teclado un valor con decimales debemos utilizar la función float en lugar de int

Ha llegado la parte fundamental, que es el momento donde desarrollarás individualmente tus algoritmos para la resolución de los problemas.

La experiencia dice que debemos dedicar el 80% del tiempo a la resolución individual de problemas y el otro 20% al análisis y codificación de problemas ya resueltos por otras personas.

Es de vital importancia para llegar a ser un buen PROGRAMADOR poder resolver problemas en forma individual.

80%

Es de vital importancia para llegar a ser un buen PROGRAMADOR poder resolver problemas en forma individual.
La experiencia dice que debemos dedicar el 80% del tiempo a la resolución individual de problemas y el otro 20% al análisis y codificación de problemas ya resueltos por otras personas.

El tiempo a dedicar a esta sección EJERCICIOS PROPUESTOS debe ser mucho mayor que el empleado a la sección de EJERCICIOS RESUELTOS.

Desempeño 18

Escribir un programa que solicite ingresar 10 notas de alumnos y nos informe cuántos tienen notas mayores o iguales a 7 y cuántos menores.

Desempeño 19

Se ingresan un conjunto de n alturas de personas por teclado. Mostrar la altura promedio de las personas.

Desempeño 20

| En una empresa trabajan n empleados cuyos sueldos oscilan entre $100 y $500, realizar un programa que lea los sueldos que cobra cada empleado e informe cuántos empleados cobran entre $100 y $300 y cuántos cobran más de $300.
| Además el programa deberá informar el importe que gasta la empresa en sueldos al personal

Desempeño 21

| Realizar un programa que imprima 25 términos de la serie 11 - 22 - 33 - 44, etc. (No se ingresan valores por teclado)

Desempeño 22

Mostrar los múltiplos de 8 hasta el valor 500. Debe aparecer en pantalla 8 - 16 - 24, etc.

Desempeño 23

Realizar un programa que permita cargar dos listas de 15 valores cada una. Informar con un mensaje cual de las dos listas tiene un valor acumulado mayor (mensajes "Lista 1 mayor", "Lista 2 mayor", "Listas iguales")

Tené en cuenta que puede haber dos o más estructuras repetitivas en un algoritmo.

Desempeño 24

Soluciones

Desarrollar un programa que permita cargar n números enteros y luego nos informe cuántos valores fueron pares y cuántos impares.

Emplear el operador “%” en la condición de la estructura condicional (este operador retorna el resto de la división de dos valores, por ejemplo 11%2 retorna un 1):

```
if valor%2==0:
```

Cuando te sientas listo, hacé click en el botón "Soluciones" para verificar tus respuestas

Esta semana avanzamos mucho: ya podemos diseñar programas que toman decisiones bastante complejas. La semana próxima nos meteremos en un nuevo tema. No olviden que deben realizar y probar en sus propias computadoras todos los ejemplos y ejercicios propuestos.

¡Hasta la semana que viene!

Soluciones a los problemas

Te recomendamos utilizar esta sección luego de haber intentado por un largo tiempo la resolución y también para verificar tus soluciones.

Volver a la clase

Solución al desempeño 17

a| Debemos cambiar la condición del while con x<=500.

b| Debemos inicializar x con el valor 50.

c| Inicializar x con el valor -50 y fijar la condición x<=0.

d| Inicializar a x con el valor 2 y dentro del bloque repetitivo incrementar a x en 2 **(x = x + 2)**

Soluciones a los problemas

Te recomendamos utilizar esta sección luego de haber intentado por un largo tiempo la resolución y también para verificar tus soluciones.

Solución al desempeño 18

```
x=1
conta1=0
conta2=0
while x<=10:
    nota=int(input("Ingrese nota:"))
    if nota>=7:
        conta1=conta1+1
    else:
        conta2=conta2+1
    x=x+1
print("Cantidad de alumnos con notas mayores o iguales a 7")
print(conta1)
print("Cantidad de alumnos con notas menores a 7")
print(conta2)
```

Soluciones a los problemas

Te recomendamos utilizar esta sección luego de haber intentado por un largo tiempo la resolución y también para verificar tus soluciones.

Solución al desempeño 19

```
n=int(input("Cuantas personas hay:"))
x=1
suma=0
while x<=n:
    altura=float(input("Ingrese la altura:"))
    suma=suma+altura
    x=x+1
promedio=suma/n
print("Altura promedio")
print(promedio)
```

Solución al desempeño 20

```
n=int(input("Cuantos empleados tiene la empresa:"))
x=1
conta1=0
conta2=0
gastos=0
while x<=n:
    sueldo=float(input("Ingrese el sueldo del empleado:"))
    if sueldo<=300:
        conta1=conta1+1
    else:
        conta2=conta2+1
    gastos=gastos+sueldo
    x=x+1
print("Cantidad de empleados con sueldos entre 100 y 300")
print(conta1)
print("Cantidad de empleados con sueldos mayor a 300")
print(conta2)
print("Gastos total de la empresa en sueldos")
print(gastos)
```

Solución al desempeño 21

```
x=1
termino=11
while x<=25:
    print(termino)
    x=x+1
    termino=termino+11
```

Solución al desempeño 22

```
mult8=8
while mult8<=500:
    print(mult8)
     mult8=mult8+8
```

Solución al desempeño 23

```
x=1
suma1=0
suma2=0
print("primer lista")
while x<=15:
    valor=int(input("Ingrese valor:"))
    suma1=suma1+valor
    x=x+1
print("Segunda lista")
x = 1
while x<=15:
    valor=int(input("Ingrese valor:"))
    suma2=suma2+valor
    x=x+1
if suma1>suma2:
    print("Lista 1 mayor.")
else:
    if suma2>suma1:
        print("Lista2 mayor.")
    else:
        print("Listas iguales.")
```

Solución al desempeño 24

Volver a la clase

```
x=1
pares=0
impares=0
n=int(input("Cuantos números ingresará:"))
while x<=n:
    valor=int(input("Ingrese el valor:"))
    if valor%2==0:
        pares=pares+1
    else:
        impares=impares+1
    x=x+1
print("Cantidad de pares")
print(pares)
print("Cantidad de impares")
print(impares)
```

Créditos

Imágenes

Encabezado: Image by Steve Buissinne from Pixabay
https://pixabay.com/photos/pawn-chess-pieces-strategy-chess-2430046/

Tipografía

Para este diseño se utilizó la tipografía ***Source Sans Pro*** diseñada por Paul D. Hunt.
Extraída de Google Fonts.

Si detectás un error del tipo que fuere (falta un punto, un acento, una palabra mal escrita, un error en código, etc.), por favor comunicate con nosotros a *correcciones@issd.edu.ar* e indicanos por cada error que detectes la página y el párrafo. Muchas gracias por tu aporte.