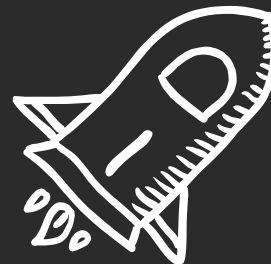




# Semana 2

# ¡Bienvenidos!



# Temas de hoy

---

**1**

Pseudocódigo -  
Estructuras de control:  
Condicionales -  
**Si, SiNo, SiNo Si**

**2**

Pseudocódigo -  
Estructuras de control:  
Bucles -  
**Para, Mientras**

1

# Condicionales - Si, SiNo

Muchas veces necesitamos que el programa siga un curso, un camino u otro, y para esa toma de decisión usamos condicionales.

En la primer semana vimos un pequeño ejemplo con el programa que daba un mensaje si una persona era mayor o menor de edad.

Ahora es momento de profundizar el en tema 

# Condicionales: Si, SiNo, SiNo Si - if, else, elif

## Condicionales

En la programación estas estructuras son fundamentales para tomar decisiones en base a una o varias condiciones que le damos al programa.

En la vida cotidiana constantemente estamos tomando decisiones en base a condiciones que se nos presentan, y, para que un programa se ejecute de una cierta manera, tenemos que darle condiciones para controlar el flujo del programa y que realice diferentes acciones.

Si hablamos de tomar decisiones estamos hablando de que "Si" hacemos una cosa, *sucede algo*, pero "Si no" hacemos esa cosa, *sucede otra cosa*.

## Funcionamiento básico del condicional: Si



se cumple  
esta condición

Entonces ejecuto esto

# Condicionales: Si, SiNo, SiNo Si - if, else, elif

Para el funcionamiento básico del condicional “Si”, incluimos en su estructura una palabra reservada (Si), una *condición*, y una respuesta a esa condición, es decir, el programa tiene que realizar una *acción* si la condición se cumple.

Para ponerlo en práctica vamos a verlo en pseudocódigo usando PSeint, donde vamos a usar su estructura básica:

```
Si expresion_logica Entonces  
    acciones_por_verdadero
```

```
Fin Si
```

Donde “expresión\_logica” se refiere a la *condición* y “acciones\_por\_verdadero” será una *acción* que se ejecutará una vez cumplida la *condición*.

Veamos un ejemplo en pseudocódigo para entenderlo mejor:

```
1  Algoritmo condicionales_ejemplo_1  
2  
3      Definir color_1, color_2 Como Caracter  
4  
5      color_1 = 'Blanco'  
6      color_2 = 'Negro'  
7  
8      Si color_1 = color_2 Entonces  
9          | Imprimir 'Los colores son iguales.'  
10     Fin Si  
11  
12 FinAlgoritmo
```

En este programa estamos verificando si los valores que tienen estas dos variables son iguales, pero como no son iguales (ya que una tiene el valor “Blanco” y la otra “Negro”), el bloque de código que se encuentra dentro de la estructura condicional, no se va a ejecutar.

# Condicionales: Si, SiNo, SiNo Si - if, else, elif

Sin embargo, el ejemplo anterior tiene un inconveniente, que es que, cuando no se cumple la condición, la ejecución del programa sigue.

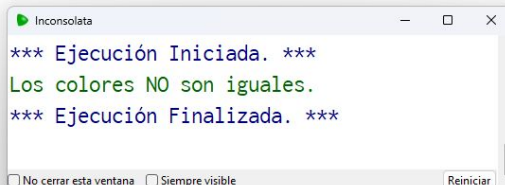
Para solucionar esto, vamos a introducir una palabra reservada más a nuestro pseudocódigo, donde veremos que "Si" se cumple la condición, se ejecutará el bloque de código, pero "SiNo" se cumple la condición, se ejecutará otro bloque de código.

Con esto vamos a tener que las acciones se van a cumplir por Verdadero o por Falso.

```
Si expresion_logica Entonces
    acciones_por_verdadero
SiNo
    acciones_por_falso
Fin Si
```

Veamos el mismo ejemplo en pseudocódigo pero agregando el condicional "SiNo":

```
1  Algoritmo condicionales_ejemplo_2
2
3      Definir color_1, color_2 Como Caracter
4
5      color_1 = 'Blanco'
6      color_2 = 'Negro'
7
8      Si color_1 = color_2 Entonces
9          | Imprimir 'Los colores son iguales.'
10         SiNo
11             | Imprimir 'Los colores NO son iguales.'
12         Fin Si
13
14 FinAlgoritmo
15
```



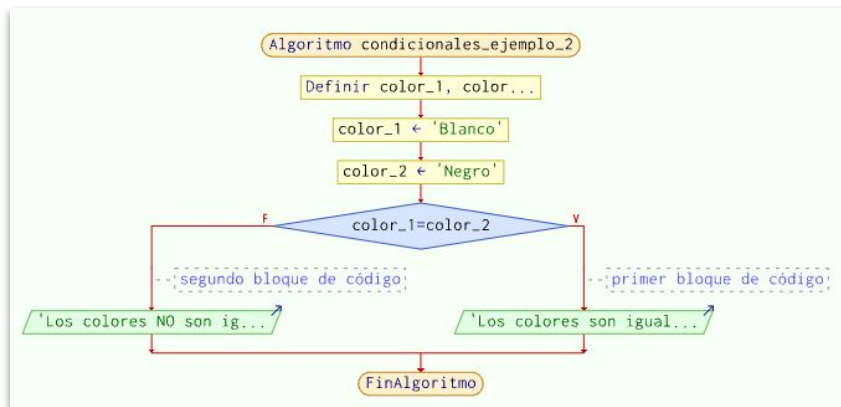
```
*** Ejecución Iniciada. ***
Los colores NO son iguales.
*** Ejecución Finalizada. ***
```

# Condicionales: Si, SiNo, SiNo Si - if, else, elif

De esta forma, podemos controlar mejor el flujo del programa, ya que si la condición se cumple, ejecutaremos un bloque de código, pero si no se cumple, se ejecutará el otro bloque de código.

```
Si color_1 = color_2 Entonces  
    //primer bloque de código  
    Imprimir 'Los colores son iguales.'  
  
SiNo  
    //segundo bloque de código  
    Imprimir 'Los colores NO son iguales.'  
  
Fin Si
```

Podemos ver también este ejemplo en un diagrama de flujo:



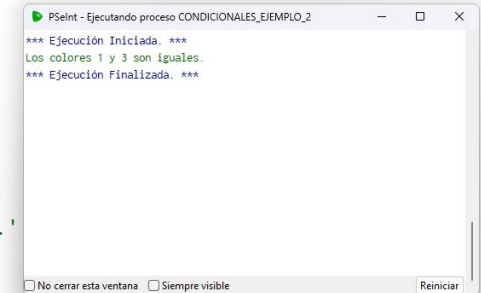
# Condicionales: Si, SiNo, SiNo Si - if, else, elif

Otra cosa que puede ocurrir es que necesitemos que se ejecute una acción, y “SiNo” se ejecute otra acción, y “SiNo”, se ejecute otra acción, y así con la cantidad de acciones que necesitemos.

Para solucionar esto, podemos usar otra cláusula más, o dicho de otra forma, otro comando más.

Donde vemos que si necesitamos agregar otra condición más (ya que tenemos más de 2 variables), usaremos un “SiNo Si” con una nueva condición y un “SiNo” dentro del nuevo bloque de código “SiNo Si”.

```
1 Algoritmo condicionales_ejemplo_2
2
3 Definir color_1, color_2, color_3 Como Caracter
4
5 color_1 = 'Blanco'
6 color_2 = 'Negro'
7 color_3 = 'Blanco'
8
9 Si color_1 = color_2 Entonces
10 //primer bloque de código
11 Imprimir 'Los colores 1 y 2 son iguales.'
12
13 SiNo Si color_1 = color_3 Entonces
14 //segundo bloque de código
15 Imprimir 'Los colores 1 y 3 son iguales.'
16 SiNo
17 //tercer bloque de código
18 Imprimir 'Los colores 2 y 3 son iguales'
19 FinSi
20
21 Fin Si
22
23 FinAlgoritmo
```





# Condicionales: Si, SiNo, SiNo Si - if, else, elif

Si a este pseudocódigo lo llevamos al lenguaje de programación que usaremos

- Python - quedaría de la siguiente manera, donde podemos ver que en vez de “Si” usamos “if”, en vez de “SiNo Si” usamos “elif” y en vez de “SiNo” usamos “else”.

Básicamente son las mismas palabras que en pseudocódigo, pero en inglés, por lo que no es muy difícil traducir del pseudocódigo a Python.

“SiNo si” o “elif” lo podremos usar las veces que necesitemos, por lo que en mentoría podrás ejercitar mucho más con tu mentor.

Además, en el material complementario vas a poder profundizar en estos conceptos.

```
1  color_1 = 'Blanco'
2  color_2 = 'Negro'
3  color_3 = 'Blanco'
4
5  if color_1 == color_2:
6      print('Los colores 1 y 2 son iguales.')
7  elif color_1 == color_3:
8      print('Los colores 1 y 3 son iguales.')
9  else:
10     print('Los colores 2 y 3 son iguales.')
11
```

PROBLEMAS

SALIDA

CONSOLA DE DEPURACIÓN

TERMINAL

```
on.exe c:/Users/Pc/Documents/Info/2.py
Los colores 1 y 3 son iguales.
```

# Ejemplo de condicionales

## Condicionales: Ejemplo 1

En el siguiente ejemplo vamos a plantear la entrada de un auto a un estacionamiento.

El programa debe ser capaz de simular la orden de levantar la barrera para que el auto pase si este paga el monto correcto que sale el ingreso al estacionamiento.

En este caso, el monto será de \$1.000 y una vez simulamos que el usuario ingresa el dinero, la barrera se levantará o no.

```
1 Algoritmo condicionales_ejemplo_1
2
3 Definir monto_ingresado, valor_estacionamiento Como Entero
4
5 valor_estacionamiento = 1000
6
7 Imprimir 'Bienvenido al estacionamiento del Info. El valor del estacionamiento es de: $1.000'
8 Imprimir 'Por favor ingrese el monto que va a pagar (solo números): '
9
10 Leer monto_ingresado
11
12
13 Si monto_ingresado = valor_estacionamiento Entonces
14 | Imprimir 'Muchas gracias. Ahora la barrera se levantará y usted puede ingresar.'
15 SiNo
16 | Imprimir 'Disculpe pero el monto ingresado es incorrecto. La barra no se levantará por lo que usted no puede pasar.'
17 Fin Si
18
19 FinAlgoritmo
20
```

PSeInt - Ejecutando proceso CONDICIONALES\_EJEMPLO\_1

```
*** Ejecución Iniciada. ***
Bienvenido al estacionamiento del Info. El valor del estacionamiento
es de: $1.000
Por favor ingrese el monto que va a pagar (solo números):
> 1000
Muchas gracias. Ahora la barrera se levantará y usted puede ingresar.
*** Ejecución Finalizada. ***
```

☐ No cerrar esta ventana ☐ Siempre visible Reiniciar

PSeInt - Ejecutando proceso CONDICIONALES\_EJEMPLO\_1

```
*** Ejecución Iniciada. ***
Bienvenido al estacionamiento del Info. El valor del estacionamiento es de:
$1.000
Por favor ingrese el monto que va a pagar (solo números):
> 900
Disculpe pero el monto ingresado es incorrecto. La barra no se levantará por
lo que usted no puede pasar.
*** Ejecución Finalizada. ***
```

☐ No cerrar esta ventana ☐ Siempre visible Reiniciar

# Ejemplo de condicionales

## Condicionales: Ejemplo 1 - Explicación

Lo que vimos en el ejemplo anterior son las 2 situaciones que pueden ocurrir para la condición que dimos. Para este caso, le dijimos al programa que “Si” el usuario ingresaba un monto igual a 1000, “Entonces” la barrera se levantaría, pero “SiNo”, es decir, si el monto ingresado no era exactamente de 1000 la barrera no se levantaría.

Esto quiere decir que cuando usamos condicionales

tenemos que contemplar que se va a ejecutar una u otra acción de manera exacta, por lo que el manejo de la condición que le damos al programa tiene que estar muy bien contemplada.

Para solucionar este caso, podemos contemplar una segunda condición usando “SiNo Si”, donde pasamos a verificar que si el usuario ingresa dinero demás, se le dará vuelta y la barrera se levantará. Agregamos una tercer variable para realizar el cálculo del vuelta del usuario.

Vemos el ejemplo en la siguiente página:

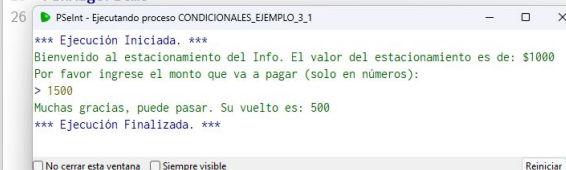
# Ejemplo de condicionales

Ahora estamos controlando mucho mejor el dinero con el que cuenta el usuario, ya sea mayor al monto requerido, igual o menor.

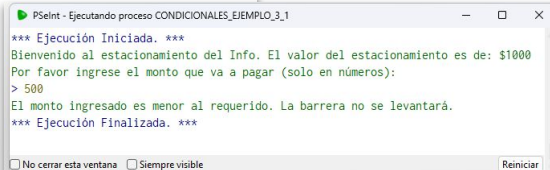
Con este flujo de condiciones y acciones, podemos lograr manejar las distintas situaciones que se puedan presentar.

En mentorías vas a poder seguir practicando con este y otros ejemplos.

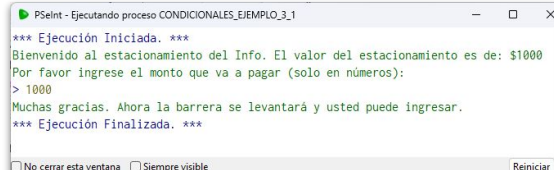
```
1 Algoritmo condicionales_ejemplo_3_1
2
3 Definir monto_ingresado, valor_estacionamiento, vuelto Como Entero
4
5 valor_estacionamiento = 1000
6
7 Imprimir 'Bienvenido al estacionamiento del Info. El valor del estacionamiento es de: $1000'
8 Imprimir 'Por favor ingrese el monto que va a pagar (solo en números): '
9
10 Leer monto_ingresado
11
12 Si monto_ingresado = valor_estacionamiento Entonces
13     //primer bloque de código
14     Imprimir 'Muchas gracias. Ahora la barrera se levantará y usted puede ingresar.'
15 SiNo Si monto_ingresado > valor_estacionamiento Entonces
16     //segundo bloque de código
17     vuelto ← monto_ingresado - valor_estacionamiento //operación para dar vuelto
18     Imprimir 'Muchas gracias, puede pasar. Su vuelto es: ', vuelto
19 SiNo
20     //tercer bloque de código
21     Imprimir 'El monto ingresado es menor al requerido. La barrera no se levantará.'
22 Fin Si
23
24 Fin Si
25 FinAlgoritmo
26
```



```
*** Ejecución Iniciada. ***
Bienvenido al estacionamiento del Info. El valor del estacionamiento es de: $1000
Por favor ingrese el monto que va a pagar (solo en números):
> 1500
Muchas gracias, puede pasar. Su vuelto es: 500
*** Ejecución Finalizada. ***
```



```
*** Ejecución Iniciada. ***
Bienvenido al estacionamiento del Info. El valor del estacionamiento es de: $1000
Por favor ingrese el monto que va a pagar (solo en números):
> 500
El monto ingresado es menor al requerido. La barrera no se levantará.
*** Ejecución Finalizada. ***
```



```
*** Ejecución Iniciada. ***
Bienvenido al estacionamiento del Info. El valor del estacionamiento es de: $1000
Por favor ingrese el monto que va a pagar (solo en números):
> 1000
Muchas gracias. Ahora la barrera se levantará y usted puede ingresar.
*** Ejecución Finalizada. ***
```

2

# Bucles - Mientras, Para

Si bien los condicionales nos sirven para tomar una u otra acción en el programa mediante una condición, pero una vez que se ejecuta la acción, el programa sigue su curso, por lo que no podemos repetir esta acción.

Los bucles vienen para solucionar este problema.

¿Lo vemos? 

# Bucles: Mientras, Para - while, for

## Bucles

Algo que no podemos hacer con condicionales es repetir el bloque de código para que una o varias acciones se ejecuten “mientras” que una condición se cumpla o, en otro caso, que la acción se ejecute una determinada cantidad de veces.

Para solucionar esto, contamos con bucles, los cuales son ciclos de repetición o iteración que se encargarán de que las acciones sean ejecutadas hasta que ya no se necesite más.

“Mientras” realizará iteraciones del bloque de código, justamente, mientras sea necesario y, “Para” realizará iteraciones del bloque de código de forma controlada.

## Funcionamiento básico del bucle: Mientras

**Mientras**

se cumple  
esta condición

Hago esto

# Bucles: Mientras, Para - while, for

## Estructura básica del bucle: Mientras

Si vemos en pseudocódigo la estructura básica de “Mientras”, quedaría de la siguiente manera:

```
Mientras expresion_logica Hacer
    secuencia de acciones
Fin Mientras
```

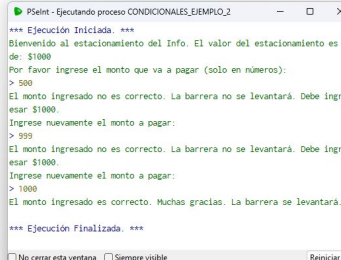
Por lo que, mientras que la *condición* se cumpla, se realizará una *acción* o varias, y esto finalizará cuando la *condición* no se cumpla más.

Para entenderlo mejor, vamos a usar el ejemplo que teníamos en Condicional, pero lo vamos a modificar con “Mientras”.

## Ejemplo del bucle: Mientras

```
1 Algoritmo condicionales_ejemplo_2
2
3 Definir monto_ingresado, valor_estacionamiento, vuelto Como Entero
4
5 valor_estacionamiento = 1000
6
7 Imprimir 'Bienvenido al estacionamiento del Info. El valor del estacionamiento es de: $1000'
8 Imprimir 'Por favor ingrese el monto que va a pagar (solo en números): '
9
10 Leer monto_ingresado
11
12 Mientras monto_ingresado <= valor_estacionamiento Hacer
13     //primer bloque de código
14     Imprimir 'El monto ingresado no es correcto. La barrera no se levantará. Debe ingresar $1000.'
15     Imprimir 'Ingrese nuevamente el monto a pagar: '
16
17     Leer monto_ingresado
18
19 Fin Mientras
20
21 Imprimir 'El monto ingresado es correcto. Muchas gracias. La barrera se levantará.'
```

```
FinAlgoritmo
```



# Bucles: Mientras, Para - while, for

## Explicación del ejemplo anterior

Con unas simples modificaciones, y el uso de “Mientras”, habrás visto que nuestro programa ahora requiere que se ingrese un monto de 1000 exactamente ya que sino, no se va a levantar la barrera.

Nuestra condición cambió para que “Mientras” el monto ingresado sea distinto al valor del estacionamiento, el programa no finalice.

Como para darte una mejor idea, esto nos sería muy útil también para el acceso de una puerta, donde el usuario debe ingresar un código y “Mientras” no ingrese el código correcto, no se abrirá la puerta.

Esto quiere decir que, “Mientras” no tendrá un número de veces específicas para ejecutarse, sino que lo hará *mientras* que la *condición* dada sea *verdadera*, y sólo finalizará cuando la *condición* sea *falsa*.

## Funcionamiento básico del bucle: Para

**Para**

esta cantidad  
de veces

Hago esto



# Bucles: Mientras, Para - while, for

## Estructura básica del bucle: Para

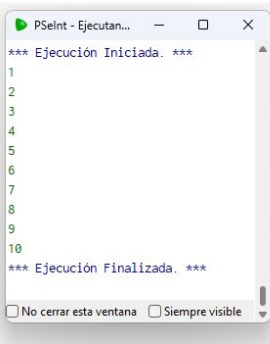
A diferencia de “Mientras”, donde no se tiene un número específico de iteraciones que se ejecutará el bucle, con “Para” sucede lo opuesto, ya que en este bucle sí está controlada la cantidad de veces que se realizarán las iteraciones, es decir, solo una cantidad de veces específicas se ejecutará el bloque de código.

```
Para variable Desde valor_inicial Hasta valor_final Hacer
    secuencia_de_acciones
Fin Para
```

En el caso de “Para” utilizaremos un variable que llevará el control de iteraciones que se realizarán, sin embargo, como la cantidad de iteraciones ya está definida, no se ejecutarán ni más ni menos iteraciones.

## Ejemplo del bucle: Para

```
1  Algoritmo bucles_ejemplo_3
2
3      Definir i Como Entero
4
5      Para i Desde 1 Hasta 10 Hacer
6          Imprimir i
7      Fin Para
8
9      FinAlgoritmo
10
```



En este caso nuestra variable de control “i” aumenta de valor con cada iteración, y como nuestro bloque de código muestra ese valor, en la ejecución se muestran los números del 1 al 10, que es la cantidad exacta que se ejecutará el bucle “Para”.

# Traduciendo a Python: “Mientras” - “while”

## ¿Y cómo se verían en Python?

Así como vimos cómo se vería el condicional en Python, vamos a hacer lo mismo con el bucle “Mientras” y el bucle “Para”.

Primero veremos que para usar “Mientras” escribimos la palabra reservada “while” en Python.

En este ejemplo, el bloque de código mostrará por pantalla los números del 1 al 10, ya que nuestra variable de control aumentará de 1 en 1 por cada iteración.

```
while.py > ...  
1  i = 1  
2  while i <= 10:  
3      print (i)  
4      i += 1  
5  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

# Traduciendo a Python: “Para” - “for”

## ¿Y cómo se verían en Python?

A diferencia de “Mientras” o “while” en Python, donde por cada iteración aumentamos el valor de nuestra variable de control, en “Para” o “for” en Python, la cantidad de iteraciones está definida, por lo que no es necesario realizar ninguna operación de aumentar o incrementar el valor de nuestra variable.

Sin embargo, hemos usado una cláusula más donde pusimos las palabras reservadas “in” y la palabra reservada “range”, pero esto lo veremos a profundidad en las próximas clases.

```

for.py > ...
1  i = 1
2  for i in range(1, 11):
3      print(i)
4

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10



# Lo que vimos hoy 🤔

Hasta acá pudimos ver el uso de condicionales “Si” y sus respectivas cláusulas “SiNo” y “SiNo Si” y el uso de bucles “Mientras” y “Para”. Además, ya pudimos ver cómo sería su traducción al lenguaje de programación Python.

Recordá que vas a poder profundizar sobre el uso de estas estructuras de control tanto en las clases de mentoría como en el material complementario que ya se encuentra a tu disposición.



**¡Nos vemos  
En la próxima  
clase!**

