



Pensamiento Computacional

Ciclos While



Ciclos, bucles, loops

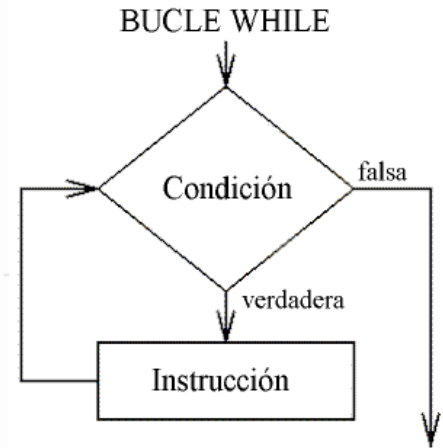
Ya sabemos condicionar un código, ahora
utilizaremos código que se esté repitiendo
mientras se cumpla una condición

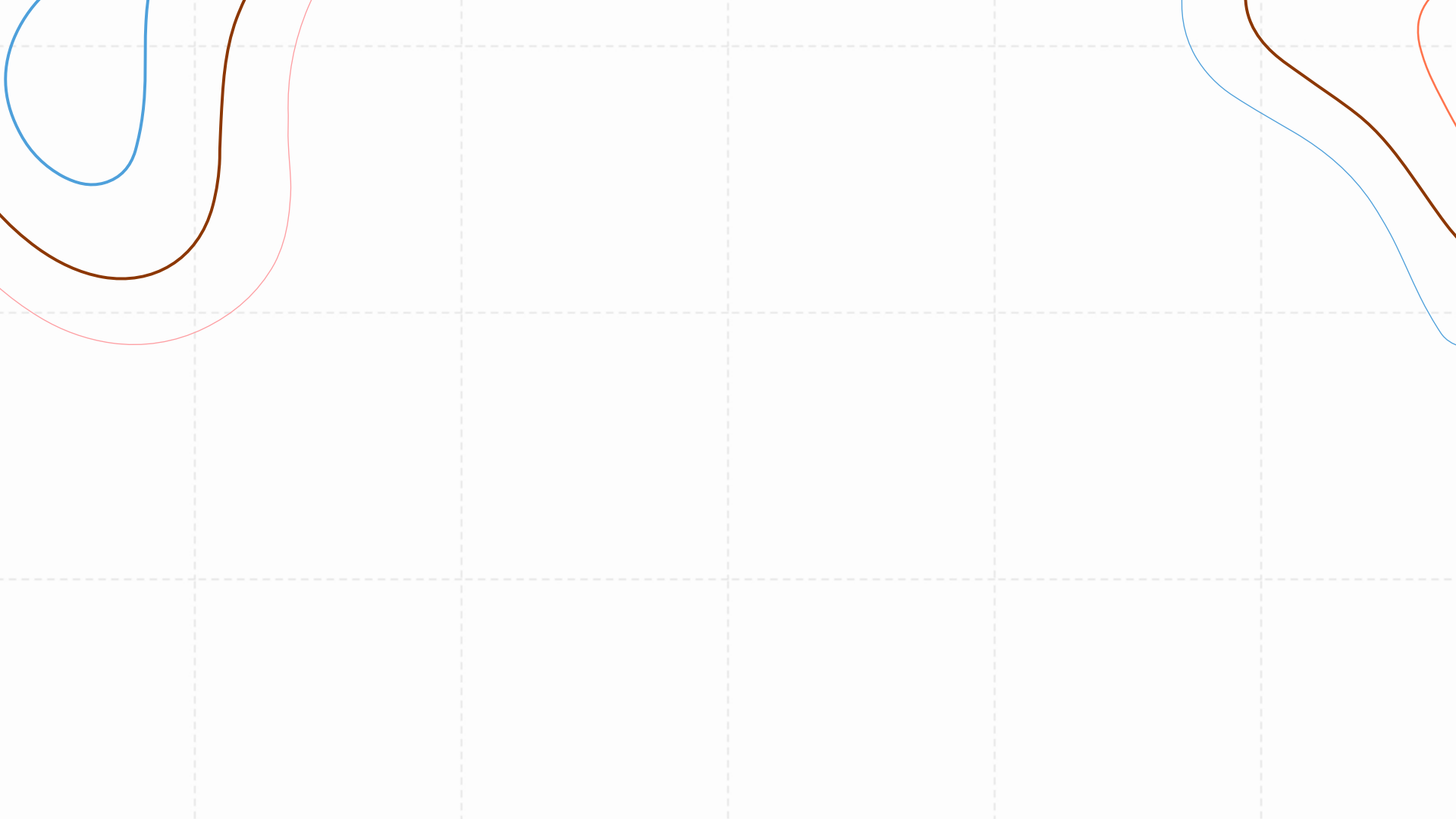
Bucles while

1. Una sentencia if es ejecutada si su condición se evalúa como TRUE
2. Una sentencia `while` es parecida, excepto que se puede ejecutar más de una vez. De hecho las sentencias dentro de una estructura while se repiten siempre y cuando la condición se mantenga.
3. En el momento que la condición se deja de cumplir, el ciclo termina y el programa continúa con la siguiente instrucción debajo del while.
4. *¿Qué te parece hacer un ciclo para volver a ejecutar cualquiera de tus programas?*

Cómo plantear un while

1. Idealmente necesitas un valor inicial para la variable a comparar
2. Utilizas la estructura `while` acompañada de la condición que debe cumplirse para mantenerte en el ciclo
3. Mientras el valor de dicha condición no cambie, el código dentro de la estructura `while` vuelve a ejecutarse
4. Cuando el valor de la condición deje de cumplirse, el programa sale del `while` y continúa con la ejecución del programa fuera de la estructura `while`.





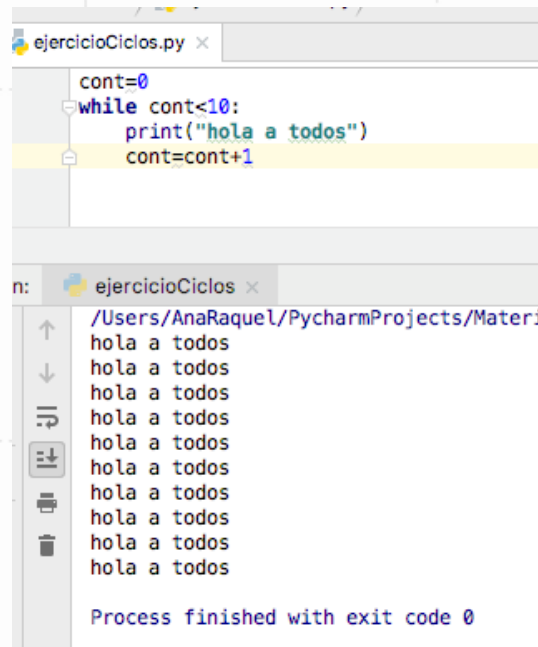
1. Contador simple:

Realiza un programa que repita “Hola a todos” 10 veces

- 1) contador=0
- 2) Mientras contador < 10
 - 2.1 imprimir “hola a todos”
 - 2.2 incrementar el contador
(contador=contador+1)
- 3) terminar.

¿Qué sucede si modificamos la condición
agregando igual?

contador <=10



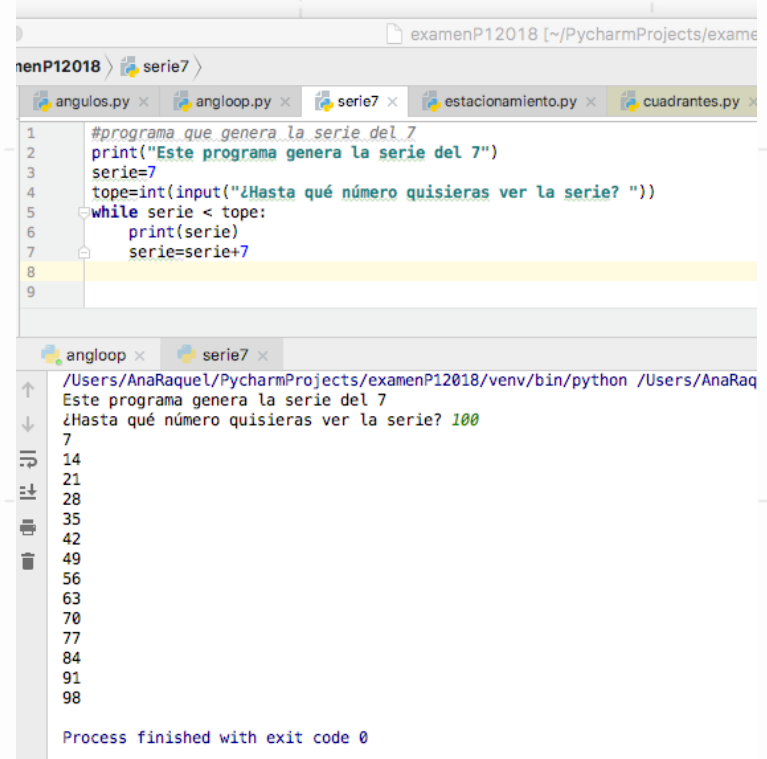
The screenshot shows a Python IDE with a file named 'ejercicioCiclos.py'. The code in the editor is:

```
cont=0
while cont<10:
    print("hola a todos")
    cont=cont+1
```

Below the editor, the output of the program is displayed, showing the text 'hola a todos' printed ten times. At the bottom, it states 'Process finished with exit code 0'.

2. Series (acumulador):

1. Realiza un programa que presente la serie del 7. Pregunta al usuario hasta qué valor desea la serie.
2. Modifica el programa para que el usuario pueda ver la serie del número que él quiera. Cuida las validaciones.
3. Observa como la misma variable que presenta los resultados parciales funciona para detener la ejecución del ciclo



The screenshot shows a PyCharm IDE window titled 'examenP12018 [~/PycharmProjects/examenP12018]'. The editor has several tabs open: 'angulos.py', 'angloop.py', 'serie7', 'estacionamiento.py', and 'cuadrantes.py'. The 'serie7' tab is active, displaying the following Python code:

```
1 #programa que genera la serie del 7
2 print("Este programa genera la serie del 7")
3 serie=7
4 tope=int(input("¿Hasta qué número quisieras ver la serie? "))
5 while serie < tope:
6     print(serie)
7     serie=serie+7
8
9
```

Below the editor, the 'Run' console shows the execution output for the 'serie7' script:

```
angloop x serie7 x
/Users/AnaRaquel/PycharmProjects/examenP12018/venv/bin/python /Users/AnaRaquel/PycharmProjects/examenP12018/venv/bin/python
Este programa genera la serie del 7
¿Hasta qué número quisieras ver la serie? 100
7
14
21
28
35
42
49
56
63
70
77
84
91
98
Process finished with exit code 0
```

Ejercicio:

Escribe un programa que escriba en pantalla los múltiplos de 2 comprendidos entre 1 y 100 y muestre la siguiente salida:

Recuerda los comandos de salida de texto `\t` y `\n`

```
2      4      6      8      10
12     14     16     18     20
22     24     26     28     30
32     34     36     38     40
42     44     46     48     50
52     54     56     58     60
62     64     66     68     70
72     74     76     78     80
82     84     86     88     90
92     94     96     98     100
>>>|
```


3. acumulador con contador

Los ahorros de Juanita...

Leer cantLunes

Leer cantMartes

Leer cantMiercoles

Leer cantJueves

Leer cantViernes

Leer cantSabado

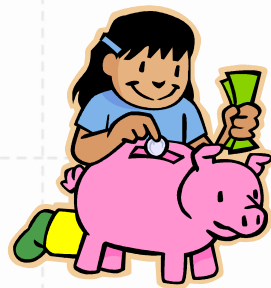
Leer cantDomingo

suma ← cantLunes + cantMartes + cantMiercoles + cantJueves +

cantViernes + cantSabado + cantDomingo

Imprime suma

*Juanita ha estado
ahorrando durante una
semana y quiere que le
digas cuánto ha ahorrado*



¡¡¡Imagínate que Juanita quisiera dictarnos las cantidades de todo el mes o año!!!

Analizando el problema...

- Observa la solución, ¿Qué acciones encuentras que se repiten en tu solución?
- Como te diste cuenta, la lectura de las cantidades es algo que se repite.
- ¿Cuántas veces?
- Y finalmente ¿qué haces con esas cantidades?
- Vamos a simplificar nuestro algoritmo anterior indicando que vamos a repetir instrucciones...

Algoritmo y Pseudocódigo

1. Para poder tener un ACUMULADOR `suma`, tienes que inicializar su valor FUERA del ciclo

`suma=0`

2. Para poder tener un CONTADOR `días`, tienes que inicializar su valor FUERA del ciclo

`días=1`

3. `días` puede inicializarse en 1, pero debes tener cuidado cómo manejas la condición (`<=`)

Mientras `días<=7`

Pedir cantidad

Leer cantidad

`suma ← suma + cantidad`

`días=días + 1`

Entonces, ¿cómo funciona el ciclo?...

- En el ejemplo anterior utilizamos un ciclo, de manera que...
- Las estructuras de repetición o ciclos, nos permiten repetir un conjunto de instrucciones
- La repetición es controlada por una condición.
- Mientras la condición sea verdadera, se repetirán las instrucciones.

Analizando el pseudocódigo...

suma=0

dias=1

suma y días debemos inicializarlas
Al principio de todo, no tenemos acumulado nada,
El día puede empezar en 1

Mientras dias <=7

Leer cantidad

Leemos la cantidad

suma ← suma + cantidad

En suma acumulamos la cantidad
Acumular quiere decir agregarla
a lo que se tiene con anterioridad

dias=dias + 1

Actualizamos el número de días que llevo

*Tenemos la condición que nos dice
cuando terminar de repetir las acciones*

suma y *dias* son variables especiales

- suma es una variable tipo **acumulador**
- dias es una variable de **control de ciclo o contador**
 - Ayudan a que nuestro ciclo deje de repetirse en algún momento y están presentes en la condición del ciclo.

3. acumulador simple con contador:

1. Calcula el promedio de calificaciones de un grupo de 8 alumnos.
2. ¿Qué deberías modificar para que el programa calcule el promedio sin importar el tamaño del grupo de alumnos?

INICIO

Lee calif1

Lee calif2

Lee calif3

Lee calif4

Lee calif5

Lee calif6

Lee calif7

Lee calif8

suma ← (calif1+ calif2+calif3+
calif4+ calif5+ calif6+ calif7+
calif8)

promedio ← suma/8

Imprime promedio

FIN

3. acumulador simple con contador:

- Podrás observar que la captura se repite 8 veces, para las 8 calificaciones.
- Así mismo, la suma **acumula** 8 datos que fueron capturados.
- Vamos a simplificar el código aprovechando estas repeticiones.

INICIO

```
Lee calif1  
Lee calif2  
Lee calif3  
Lee calif4  
Lee calif5  
Lee calif6  
Lee calif7  
Lee calif8
```

```
suma ← (calif1+ calif2+calif3+ calif4+  
calif5+ calif6+ calif7+ calif8)
```

```
promedio ← suma/8
```

```
Imprime promedio
```

FIN

Algoritmo

INICIO

1. Leo la calificación
2. Acumulo la calificación
3. Repito hasta que tenga acumuladas las 8 calificaciones
4. Divido la suma obtenida entre 8
5. Imprimo el resultado

FIN

Pseudocódigo

Suma = 0

Contador = 1

Mientras contador \leq 8

 Lee calif

 suma \leftarrow suma + calif

 contador = contador + 1

promedio = suma/8

Imprime promedio

Puedes utilizar la variable contador para obtener el promedio:
promedio = suma/contador, pero entonces contador debe empezar en 0 y debes modificar la condición del while.

```
cicloPromedio.py x sumaYpromedio.py x
1 suma = 0.0 #Inicializo el Acumulador a 0 para que no tenga nada acumulado.
2 contador = 1 #Inicializo el contador a 1 para empezar a contar alumnos.
3
4 while contador <=8: #Sólo son 8 alumnos
5     calif=float(input(f"Dame la calificación del alumno {contador}: "))
6     #imprimo el número de alumno
7     suma=suma + calif
8     contador = contador +1
9 promedio = suma/8
10 print(f"El promedio del grupo es: {promedio:.2f}")
11
```

¿Recuerdas éste
formato de
impresión?

```
cicloPromedio x
/Users/AnaRaquel/PycharmProjects/cicloPromedio/venv/bin/python /Users/AnaRa
Dame la calificación del alumno 1: 8.9
Dame la calificación del alumno 2: 7.6
Dame la calificación del alumno 3: 6.9
Dame la calificación del alumno 4: 8.0
Dame la calificación del alumno 5: 7.9
Dame la calificación del alumno 6: 7.6
Dame la calificación del alumno 7: 6.8
Dame la calificación del alumno 8: 9.8
El promedio del grupo es: 7.94

Process finished with exit code 0
```

Te permite una
impresión con 2
decimales

¿Qué es una VARIABLE DE CONTROL?

- La variable de control es aquel dato que me permite **DETENER** el ciclo.



Cada vez que se repite el ciclo, es porque se hizo una comparación y la variable de control NO ha llegado o NO ha rebasado el límite que yo marqué.

Pero DENTRO del ciclo, la Variable de Control DEBE cambiar de valor

En el ejemplo anterior, la variable de control debe llegar a **8** porque quiero preguntar **8 calificaciones**

¿Cómo sé cuantas vueltas debe dar el ciclo?

- Normalmente el mismo planteamiento del problema te dará estos datos. Ejemplos:

- *Realiza un programa que pida 8 números y saque el promedio*



Tu variable de control debe llegar a 8, si inicio en cero, o debe llegar a 9 si inició en uno.

- *Realiza un programa que imprima las tabla de multiplicar del número 7.*



Tu defines tu variable de control, tu decides si imprimes las tablas de multiplicar hasta el 10, hasta el 12 o hasta donde el usuario quiera.

4. Validaciones

Validación de división entre 0

1. Un ciclo típico de validación es la división entre 0.
2. Abre el programa que hicimos para tener una calculadora y agrega la opción de division. Modifica la condición y crea un ciclo que no avance hasta que te asegures que el usuario no ingresó un 0 como dividendo.

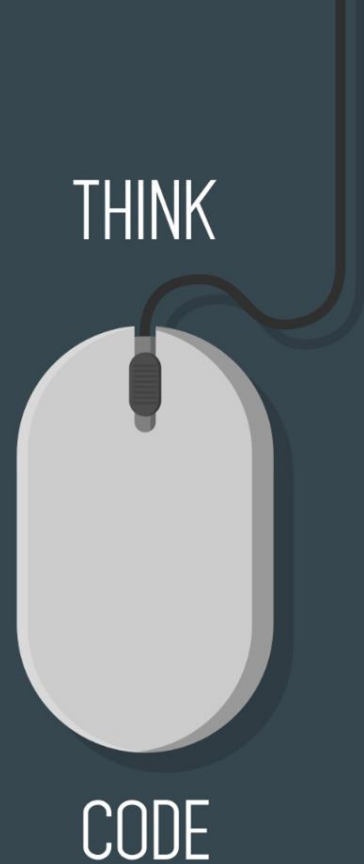
Terminología:

- Un **acumulador** es una variable que va guardando un resultado y que dicho resultado se vuelve a utilizar para guardar un nuevo resultado, pero tomando en cuenta el valor acumulado ANTERIORMENTE
- Ejemplos:
 - $\text{suma} = \text{suma} + N$
 - $\text{multiplica} = \text{multiplica} * N$
- Esta variable también debe **INICIALIZARSE** fuera del ciclo
- Si acumulará sumas, puedes inicializarlo a 0, si acumulará multiplicaciones, deberás inicializarlo a 1.

5. volver a ejecutar

```
volver=1
while volver:
    ang=int(input("Dame el ángulo a acomodar en su cuadrante "))
    cuad=ang % 360          #SACO MODULO PARA QUEDARME CON ÁNGULOS DE 0 A 359 Y ENCONTRAR SU CUADRANTE
    if ang <0:
        print("{0} es un ángulo negativo".format(cuad))
    else:
        print("{0} es un ángulo positivo".format(cuad))
        if cuad<=90:
            print("\t y está en el primer cuadrante")
        elif cuad<=180:
            print("\t y está en el segundo cuadrante")
        elif cuad <=270:
            print("\t y está en el tercer cuadrante")
        elif cuad <= 360:
            print("\t y está en el cuarto cuadrante")
    volver=int(input("\n Deseas volver a ejecutar si=1, no=0 "))
```

A programar



6. Ejercicios con Menu

MENU DE FUNCIONES:

1. Segundos a Días, Horas, Minutos
2. Pies a Centímetros
3. Volumen Esfera
4. Múltiplos Numéricos
5. SALIR

Elige una opción del menú:

- ¿Recuerdas el problema del menú para mostrar mes de verificación y No circula?
- El ciclo while puede ser usado para presentar un menú y lograr que el usuario elija entre mes de verificación o no circula y estar eligiendo la opción que desee hasta que decida salir.
- Puedes practicar haciendo menues de tus programas, utiliza números para plantear las opciones, siempre incluye la opción SALIR para que NO le preguntes al usuario si desea volver a ejecutar.
 - Si elige SALIR, el programa DEBE terminar.

Qué aprendimos...



Ciclos While

Me permite repetir la ejecución de un código **MIENTRAS** se cumpla una condición.

La variable que maneja la condición cambia de valor dentro del ciclo.



contador/acumulador

Dependiendo del problema a resolver, es probable que tengas una variable de tipo **ACUMULADOR** y otra variable de tipo **CONTADOR**.



Inicialización

Para poder utilizar un contador en un ciclo **SIEMPRE** tendrás que **inicializarlo fuera del ciclo**.

Recursos recomendados

- Videos sobre ciclos con funciones desarrollado por Yolanda Martínez.
- <https://youtu.be/4amWkLd4UKA>