TEL101 - Iniciación a la Programación

Clase 3: Ciclos

Prof. Nicolás Torres
nicolas.torresr@usm.cl
Ingeniería Civil Telemática
Departamento de Electrónica
Universidad Técnica Federico Santa María

Ciclo `while`

El ciclo `while` ("mientras") ejecuta las instrucciones mientras se cumple una condición. Cuando la condición se convierte en falsa, entonces las instrucciones dejan de realizarse.

Sintaxis:

```
while condición:
    instrucciones
```

- La **Indentación** después de la instrucción `while` indica que acciones se realizarán reiterativamente mientras se cumpla la condición.
- Cada vuelta o repetición en un ciclo es llamada **Iteración**.

In [1]:
```
i = 1
while i < 10:
    print(i)
    i += 1
print("Cuando i es igual a", i,"la condición es falsa y no se itera más.")
```

```
1
2
3
4
5
6
7
8
9
Cuando i es igual a 10 la condición es falsa y no se itera más.
```

Condición de término no conocida (uso de "flag")

Se conoce como "flag" a una variable de tipo Booleana que controla el inicio y término de un ciclo. Cuando la condición de término no es conocida, se suelen utilizar este tipo de variables de control porque su estado, que controla el ciclo, puede cambiar fácilmente en el momento deseado y detener las iteraciones.

```
flag = True
while flag:
    if <condición>:
```

```
        flag = False
    else:
        sentencias mientras la <condición> sea verdadera
```

In [2]:
```
flag = True
while flag:
    n = int(input("Ingrese un número par: "))
    if n % 2 == 0:
        print("Correcto!")
        flag = False
    else:
        print("El número ingresado no es par.")
```

```
Ingrese un número par: 13
El número ingresado no es par.
Ingrese un número par: 7
El número ingresado no es par.
Ingrese un número par: 16
Correcto!
```

Ciclos anidados

Al igual que las condicionales anidadas, es posible ubicar un ciclo dentro de otro. El "ciclo interno" se ejecutará una vez por cada iteración del "ciclo externo".

In [3]:
```
x = 1
while x <= 4:
    y = 1
    while y <= 4:
        print(x,y)
        y += 1
    x += 1
```

```
1 1
1 2
1 3
1 4
2 1
2 2
2 3
2 4
3 1
3 2
3 3
3 4
4 1
4 2
4 3
4 4
```

Patrones comunes

Los "Patrones" en diseño de software son una solución general, reutilizable y aplicable para resolver problemas comunes. Es conveniente conocer algoritmos básicos de programación y entender su funcionamiento para poder aplicarlos en distintos contextos.

Par/Impar

En matemáticas, un número "par" es un número entero que es divisible por 2. Por lo tanto, un número par tendrá resto cero al ser dividido por dos. El operador módulo permite obtener el resto de la división entre dos números, y se utiliza para determinar si un número es par o impar.

In [4]:
```
numero = int(input("Ingrese un numero:"))
if numero % 2 == 0:
    print("Par")
else:
    print("Impar")
```

```
Ingrese un numero:7
Impar
```

Conteo

Un "contador" es el nombre que se le da a una variable de tipo entera que se utiliza para llevar un conteo de ciertos eventos dentro del programa. Generalmente, esta variable comienza con valor inicial `0` y su valor se incrementa en uno cuando se cumple alguna condición.

In [5]:
```
contador = 0
i = 1
while i <= 5:
    numero = int(input("Ingrese un número entero: "))
    if numero % 2 == 0:
        contador += 1
    i += 1
print("Se ingresaron",contador,"números pares")
```

```
Ingrese un número entero: 2
Ingrese un número entero: 3
Ingrese un número entero: 4
Ingrese un número entero: 5
Ingrese un número entero: 6
Se ingresaron 3 números pares
```

Acumulación

Un "acumulador", funciona de manera similar a un contador, pero generalmente el incremento no es contante.

In [6]:
```
suma = 0
i = 1
while i <= 5:
    numero = int(input("Ingrese un número entero: "))
    if numero % 2 == 0:
        suma += numero
    i += 1
print("La suma de los números pares ingresados es",suma)
```

```
Ingrese un número entero: 2
Ingrese un número entero: 3
Ingrese un número entero: 4
Ingrese un número entero: 5
Ingrese un número entero: 6
La suma de los números pares ingresados es 12
```

Mayor

El "patrón del mayor" permite encontrar el valor más grande entre una serie de números ingresados por el usuario. La idea es usar una variable que vaya "recordando" cuál es el mayor valor visto hasta el momento y si aparece un número más grande, actualizar su valor asignándole ese número. La variable debe ser inicializada con un valor que sea menor a todos los números ($-\infty$).

In [7]:
```
mayor = -float("inf")
i = 1
while i <= 5:
    numero = float(input('Ingrese un número:'))
    if numero > mayor:
        mayor = numero
    i += 1
print('El mayor número ingresado es',mayor)
```

```
Ingrese un número:3
Ingrese un número:8
Ingrese un número:1
Ingrese un número:4
Ingrese un número:7
El mayor número ingresado es 8.0
```

Menor

El patrón para encontrar el menor es análogo al mayor, con la diferencia que se invierte la condición y, la variable que "recuerda" comienza con valor infinito positivo.

In [8]:
```
menor = float("inf")
i = 1
while i <= 5:
    numero = float(input('Ingrese un numero:'))
    if numero < menor:
        menor = numero
    i += 1
print('El menor número ingresado es',menor)
```

```
Ingrese un numero:3
Ingrese un numero:8
Ingrese un numero:1
Ingrese un numero:4
Ingrese un numero:7
El menor número ingresado es 1.0
```