Programación para Física y Astronomía

Departamento de Física.

Coordinadora: C Loyola, Profesores/as C Femenías / Y Navarrete

Primer Semestre / 2025

Universidad Andrés Bello

Universidad Andrés Bello®

Resumen - Parte 2 (Semana 03)

- Estructuras de control
  - Declaración IF
  - La declaración WHILE
- La secuencia de Fibonacci
- Funciones definidas por el usuario
- Recursividad
- Actividades

Estructuras de control

Estructuras de control

Hasta ahora, hemos visto programas **lineales**: se ejecutan de arriba abajo de forma secuencial. Para hacerlos más flexibles y tomar decisiones (o repetir bloques de código), Python ofrece *estructuras de control*:

- **if, else, elif** (condicionales)
- **while** (bucles)
- Más adelante: `for`, `try-except`, etc.

La declaración IF

```
1  x = int(input("Ingrese un entero no mayor a diez:"))
2  if x > 10:
3    print("Usted ingreso un numero mayor que diez.")
4    print("Permítame corregirle.")
5    x = 10
6  print("Su número es", x)
```

Nota: Observa la indentación. Todo lo que esté “sangrado” debajo de `if` será el bloque que se ejecuta si la condición es verdadera.

Comparaciones

- `x == 1` compara si *x* es igual a 1.
- `x != 1` compara si *x* es distinto de 1.
- `x > 1`, `x >= 1`, `x < 1`, `x <= 1`.
- Se pueden combinar condiciones con **`and`** / **`or`**.

```
1  if x > 10 or x < 1:
2    # hacer algo
3  if x >= 1 and x <= 10:
4    # hacer otra cosa
```

else y elif

```
1  if x>10:
2    print("Su numero es mayor que diez.")
3  else:
4    print("Su numero esta correcto. Nada por hacer.")
```

```
1  if x>10:
2    print("Su numero es mayor que diez.")
3  elif x>9:
4    print("Su numero esta bien, pero cercano a 10.")
5  else:
6    print("Su numero esta correcto. Nada por hacer.")
```

La declaración WHILE

Útil para repetir un bloque mientras se cumpla alguna condición:

```
x = int(input("Ingrese numero <= 10: "))
while x > 10:
  print("Es mayor que diez. Reintente.")
  x = int(input("Ingrese numero <= 10: "))
print("Su numero es", x)
```

Combinando:

```
while x>10 or x<1:
  # ...
```

La secuencia de Fibonacci

La secuencia de Fibonacci

Los números de Fibonacci se definen como:

$$F_n = F_{n-1} + F_{n-2}, \quad F_1 = 1, F_2 = 1$$

Los primeros términos: 1, 1, 2, 3, 5, 8, 13, 21, ...

Ejemplo: Fibonacci hasta 1000

```
1  f1 = 1
2  f2 = 1
3  next_ = f1 + f2
4
5  while f1 <= 1000:
6    print(f1)
7    f1 = f2
8    f2 = next_
9    next_ = f1 + f2
```

Funciones definidas por el usuario

Funciones definidas por el usuario

A menudo necesitamos funciones específicas. Ej: calcular factorial $n!$:

```
1  def factorial(n):
2    f = 1
3    k = 1
4    while (k <= n):
5      f *= k
6      k += 1
7    return f
8
9  # Programa principal
10 val = int(input("Ingrese un entero positivo: "))
11 print(val, "! =", factorial(val))
```

Funciones definidas por el usuario

- `def nombre_funcion(argumento):` abre la definición de la función.
- Bloque indentado = cuerpo de la función.
- `return` finaliza la función y devuelve un valor.

Recursividad

Recursividad

La recursividad consiste en que la función se llama a sí misma (para subproblemas más pequeños):

```
1  def factorial(n):
2    if n == 1:
3      return 1
4    else:
5      return n * factorial(n-1)
```

Actividades

Actividades para Hoy

- Implemente un programa que pida un número y verifique si es par o impar. Use un condicional `if` o `while` hasta que sea válido.
- Extienda el ejemplo de Fibonacci para que el usuario indique hasta qué número máximo desea imprimir.
- Defina una función que calcule la potencia a^b mediante multiplicaciones sucesivas (sin usar el operador `**`). Compare con la función incorporada.

Fin de la Parte 2