# Introducción a la programación

## Práctica 7: Introducción al Lenguaje Imperativo

# Empezando con Python

**Primeros pasos:**

- Por ahora vamos a trabajar similar que con Haskell
  1. Hacemos nuestro archivo con las funciones (`test.py`)
  2. Abrimos el compilador interactivo (`python` o `python3`, ver version)
  3. Importamos el archivo con nuestras funciones (`import test`)
  4. Las usamos interactivamente (`test.<nombre_funcion>`)
  5. Para recargar el archivo hay que cerrar el intérprete (`exit()`) y volver a empezar

# Empezando con Python

**Primeros pasos:**

- Por ahora vamos a trabajar similar que con Haskell
  1. Hacemos nuestro archivo con las funciones (`test.py`)
  2. Abrimos el compilador interactivo (`python` o `python3`, ver version)
  3. Importamos el archivo con nuestras funciones (`import test`)
  4. Las usamos interactivamente (`test.<nombre_funcion>`)
  5. Para recargar el archivo hay que cerrar el intérprete (`exit()`) y volver a empezar
- También podemos ejecutar todo el script desde consola (`python3 test.py`)
  - En este caso además de las definiciones de las funciones tiene que haber código "libre" que las ejecute

# Empezando con Python

**Primeros pasos:**

- Por ahora vamos a trabajar similar que con Haskell
  1. Hacemos nuestro archivo con las funciones (`test.py`)
  2. Abrimos el compilador interactivo (`python` o `python3`, ver version)
  3. Importamos el archivo con nuestras funciones (`import test`)
  4. Las usamos interactivamente (`test.<nombre_funcion>`)
  5. Para recargar el archivo hay que cerrar el intérprete (`exit()`) y volver a empezar
- También podemos ejecutar todo el script desde consola (`python3 test.py`)
  - En este caso además de las definiciones de las funciones tiene que haber código "libre" que las ejecute
- O usar las herramientas de *debugging* que vamos a ver más adelante

# Ej 1.2

**Ejercicio 1.** Definir las siguientes funciones y procedimientos

2. `problema imprimir_hola ()` {
   `requiere:` { True }
   `asegura:` { imprime 'hola'por consola}
   }

# Ej 2.4

**Ejercicio 2.** Definir las siguientes funciones y procedimientos con parámetros

4. `problema es_multiplo_de` (in n: $\mathbb{Z}$, in m:$\mathbb{Z}$) {
   `requiere:` { m $\neq$ 0 }
   `asegura:` $\{res = True \leftrightarrow (\exists k : \mathbb{Z})(n = m * k)\}$
   }

# Ej 3.3

**Ejercicio 3.** Resuelva los siguientes ejercicios utilizando los operadores lógicos `and`, `or`, `not`. Resolverlos sin utilizar alternativa condicional (if).

3. `problema es_nombre_largo` (in nombre: String) : Bool {
   `requiere:` { True }
   `asegura:` $\{res = True \leftrightarrow 3 \leq |nombre| \leq 8\}$
   }

# Ej 5.1

**Ejercicio 5.** Implementar los siguientes problemas de alternativa condicional (if).

1. `devolver_el_doble_si_es_par(un_numero)` . Debe devolver el mismo número en caso de no ser par.

# Ej 6.2

**Ejercicio 6.** Implementar las siguientes funciones usando repetición condicional `while`.

2. Escribir una función que imprima los números pares entre el 10 y el 40.

# Ej 6.2

**Ejercicio 6.** Implementar las siguientes funciones usando repetición condicional `while`.

2. Escribir una función que imprima los números pares entre el 10 y el 40.

**Ejercicio 7.** Implementar las funciones del ejercicio 6 utilizando `for num in range(i,f,p):`.