## Prácticas de Programación modular (I)

Programación — DAW

Ricardo Pérez López IES Doñana

Curso 2021/2022

1. Consulta la **librería estándar de Python** en la documentación oficial para hacerte una idea general de las posibilidades que ofrecen los módulos predefinidos del lenguaje:

```
https://docs.python.org/3/library/index.html
```

- 2. Importa el módulo math en un programa que necesite la función math. gcd para calcular el máximo común divisor de dos números:
  - a) Usando import math
  - b) Usando from math import gcd
  - c) Usando from math import \*

¿Cuál es la diferencia entre las tres opciones? ¿Cuál es más conveniente? ¿Qué inconvenientes presenta la última opción?

- 3. Usa el módulo random para escribir programas que necesiten mostrar un comportamiento aleatorio:
  - a) La función random. randint (a, b) devuelve un número entero aleatorio entre a y b. Úsala para escribir un programa que juegue a que el usuario tenga que adivinar un número entre 1 y 100.
  - b) La función random.shuffle(x) ordena aleatoriamente la secuencia x. Úsala para escribir un programa que pida al usuario cinco cadenas y que luego las imprima en un orden aleatorio.
- 4. Escribe un módulo llamado fibonacci.py que contenga las siguientes funciones:

- Una función fib que calcule el *n*-ésimo término de la sucesión de Fibonacci de forma recursiva.
- Una función fib\_iter que calcule lo mismo pero de forma iterativa llamando a
  otra función \_fib\_aux (ojo, que empieza por \_), que es la que realmente lleva a
  cabo el proceso iterativo.
- a) Usar el módulo en otro llamado principal. py importándolo mediante:

```
from fibonacci import *
```

¿Cuáles son las funciones que se acaban importando de esta manera? ¿Por qué?

- b) El módulo debe probarse a sí mismo al ejecutarse desde la línea de órdenes del sistema operativo (y sólo entonces):
  - i. Si se ejecuta sin argumentos en la línea de órdenes, deberá comprobar que fib(8) y fib\_iter(8) se calculan correctamente, mostrando un mensaje que indique si el cálculo ha sido correcto o no. Por ejemplo:

```
$ python fibonacci.py
fib(8) vale 21 (correcto)
fib_iter(8) vale 37 (incorrecto)
```

 ii. Si se ejecuta con un argumento en la línea de órdenes, deberá usarse como el argumento de la función fib y mostrar por pantalla el resultado de la función. Por ejemplo:

```
$ python fibonacci.py 7
13
```

- c) ¿Cuál es la interfaz del módulo?
- d) ¿Cuál es la **implementación** del módulo?