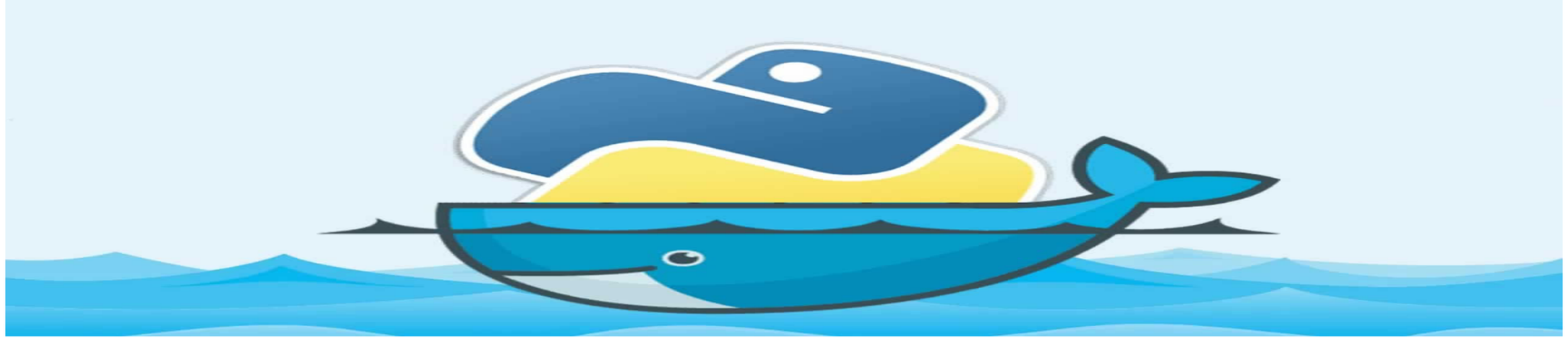

Guión de prácticas 1: Python y entorno de trabajo con docker

Sergio Alonso (zerjioi@ugr.es) y José María Guirao (jmguirao@ugr.es)

Durante las prácticas de la asignatura Desarrollo de Aplicaciones para Internet (DAI) vamos a hacer uso intensivo de diversos lenguajes de programación. Entre otros, utilizaremos Python, un lenguaje de alto nivel sencillo, potente, libre, "fácil de aprender", interpretado y multiplataforma (entre otras características). En esta primera práctica trataremos de resolver algunos problemas genéricos de programación usando dicho lenguaje para familiarizarnos con sus tipos de datos, estructuras de control, etc.

Para aquellos que no tengan conocimientos sobre este lenguaje, existen numerosos manuales en Internet que permiten acercarse a la programación Python de manera sencilla y amena como por ejemplo: A Byte of Python o el manual de Google sobre Python.

Durante todo el curso, usaremos docker y docker-compose para el entorno de desarrollo. Asi conseguimos:

- Aislar nuestro código del resto del ordenador, sin que interfiera lo que pueda instalarse antes o después.
- Conseguir un entorno **idéntico** para todos los participantes del curso, nos ahorramos el problema "pues en mi ordenador funciona" .
- Tener nuestra aplicación lista para subirla a la nube.

Un primer paso será instalar tanto el servidor de Docker como Docker Compose. En este guión no se darán instrucciones concretas para la instalación puesto que depende del sistema operativo anfitrión donde desarrollaremos nuestras prácticas. Consulta los enlaces anteriores para instalarlos en tu sistema operativo. También puedes consultar el siguiente tutorial: What is Docker and How to Use it With Python.

Una vez instalados Docker y Docker compose podemos construir un primer archivo `docker-compose.yml` para ejecutar los ejercicios de este guión. En este archivo especificamos que se monte un directorio `ejercicios` que debe encontrarse en el mismo directorio que `docker-compose.yml` (yml es un formato para marcado ligero y archivos de configuración).

```yaml
# docker-compose.yml
version: '3.7'
services:
  contenedor:
    image: python:3.7-alpine
    volumes:
      - ./ejercicios:/ejercicios
    working_dir: /ejercicios
```

y

```python
# ejercicios/hola.py
print("Hola mundo!")
```

Si todo va bien podemos ejecutar `ejercicios/hola.py` dentro del contenedor`ejercicios` con la siguiente orden:

```
> docker-compose run contenedor python hola.py
```

Ejercicios:

1. Programe un mini-juego de "adivinar" un número (entre 1 y 100) que el ordenador establezca al azar. El usuario puede ir introduciendo números y el ordenador le responderá con mensajes del estilo *"El número buscado el mayor / menor"*. El programa debe finalizar cuando el usuario adivine el número (con su correspondiente mensaje de felicitación) o bien cuando el usuario haya realizado 10 intentos incorrectos de adivinación.

2. Programe un par de funciones de ordenación de matrices (UNIDIMENSIONALES) de números distintas (burbuja, selección, inserción, mezcla, montículos...) (Wikipedia: Algoritmo de ordenamiento). Realice un programa que genere aleatoriamente matrices de números aleatorios y use dicho métodos para comparar el tiempo que tardan en ejecutarse.

3. La Criba de Eratóstenes es un sencillo algoritmo que permite encontrar todos los números primos menores de un número natural dado. Prográmelo.

4. Cree un programa que lea de un fichero de texto un número entero `n` y escriba en otro fichero de texto el `n-ésimo` número de la sucesión de Fibonacci.

5. Cree un programa que:
   - Genere aleatoriamente una cadena de `[` y `]`.
   - Compruebe mediante una función si dicha secuencia está *balanceada*, es decir, que se componga de parejas de corchetes de apertura y cierre correctamente anidados. Por ejemplo:
     - `[]` -> Correcto
     - `[[][[]]]` -> Correcto
     - `[][]` -> Correcto
     - `][` -> Incorrecto
     - `[[][[` -> Incorrecto
     - `[]][[]` -> Incorrecto

6. Utilizando expresiones regulares realice funciones para:
   - Identificar cualquier palabra seguida de un espacio y una única letra mayúscula (por ejemplo: `Apellido N`).
   - Identificar correos electrónicos válidos (empieza por una expresión genérica y ve refinándola todo lo posible).
   - Identificar números de tarjeta de crédito cuyos dígitos estén separados por `-` o espacios en blanco cada paquete de cuatro dígitos: `1234-5678-9012-3456` ó `1234 5678 9012 3456`.