1. Introducción.

Caso práctico

Pixabay (Dominio público)

Julián es un joven programador que se ha incorporado a una empresa de desarrollo de software. Conoce varios lenguajes de programación, pero se siento más cómodo con Python, si bien ha tenido experiencia programando algunas aplicaciones con C y C++.

Sabe que muchas veces los requisitos del cliente respecto al software condicionarán el lenguaje elegido, ya sea porque se necesita tener el software cuanto antes, necesite tener un alto rendimiento o tenga que comunicarse con alguna otra pieza de software o hardware y por tanto deba de ir en un lenguaje concreto. Por lo que Julián sabe que es posible que por requisitos técnicos tenga que abandonar su lenguaje favorito y adaptarse a las necesidades del proyecto.

Julián, en uno de sus primeros proyectos recibe el encargo de desarrollar una aplicación bastante sencilla donde el rendimiento no es relevante y tiene dudas

sobre qué lenguaje de programación usar, ya que según el tipo que sea tiene unas ventajas e inconvenientes. Valorando los tipos de lenguaje se encuentra con lo siguiente:

- **Rendimiento**: un lenguaje de programación compilado tiene mayor rendimiento que un lenguaje de programación interpretado pero sólo funciona para la plataforma para la que se ha compilado
- **Facilidad:** los lenguajes de programación interpretado, por lo general, son más sencillos de programar en ellos debido a un mayor nivel de abstracción

Decide hacer finalmente el programa en Python, que es un lenguaje de programación interpretado. Sabe que programar en Python será más sencillo y por tanto le requerirá menos tiempo de programación pero a cambio necesitará que la máquina donde se ejecute tenga el intérprete de Python instalado. Como la aplicación es bastante sencilla y el rendimiento no es prioritario a Julián no le importa las desventajas en cuanto a rendimiento de un lenguaje interpretado.

En un mundo que ha sufrido un importante proceso de transformación digital cada vez hay mas aplicativos desarrollados con el fin de interactuar con los seres humanos, si estos aplicativos no están correctamente programados un atacante puede hacer uso mal intencionado del software y controlar el flujo de ejecución, pudiendo alterar la información y/o robarla.

Antes de nada haremos una distinción fundamental entre hardware y software

- **Hardware**: todos los componentes físicos de un ordenador
- **Software**: programas e instrucciones para ejecutar tareas en un ordenador

Una vez tenemos claro la diferencia, en este curso nos centraremos en el software, y en una de sus unidades fundamentales: los programas

Definiríamos **programa** como:

1.- Secuencia de instrucciones
2.- Entendible por el ordenador
3.- Tienen un objetivo o tarea concreta

En esta unidad el alumno aprenderá:

- Fundamentos básicos de la programación.
- La diferencia entre lenguajes de programación interpretados y compilados.
- Código fuente y entornos de desarrollo.
- Modos de ejecución de software.
- Los principales elementos dentro de un programa.
- Pruebas de ejecución de software.
- Seguridad en los lenguajes de programación y sus entornos de ejecución

(“sandboxes”).

Pixabay (Dominio público)

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Ministerio de Educación y Formación Profesional (Dominio público)

Materiales formativos de FP Online propiedad del Ministerio de Educación y Formación Profesional.
Aviso Legal

1.1.- Fundamentos de Programación.

Los ordenadores son máquinas eléctricas que sólo entienden de 0 y 1, siendo:

- 0 no hay corriente
- 1 hay corriente

Este es el único lenguaje que entiende el ordenador, llamado lenguaje binario. Como crear programas, de aquí en adelante programar, es difícil en este lenguaje aparecieron los lenguajes de programación, que podemos dividir en dos grupos:

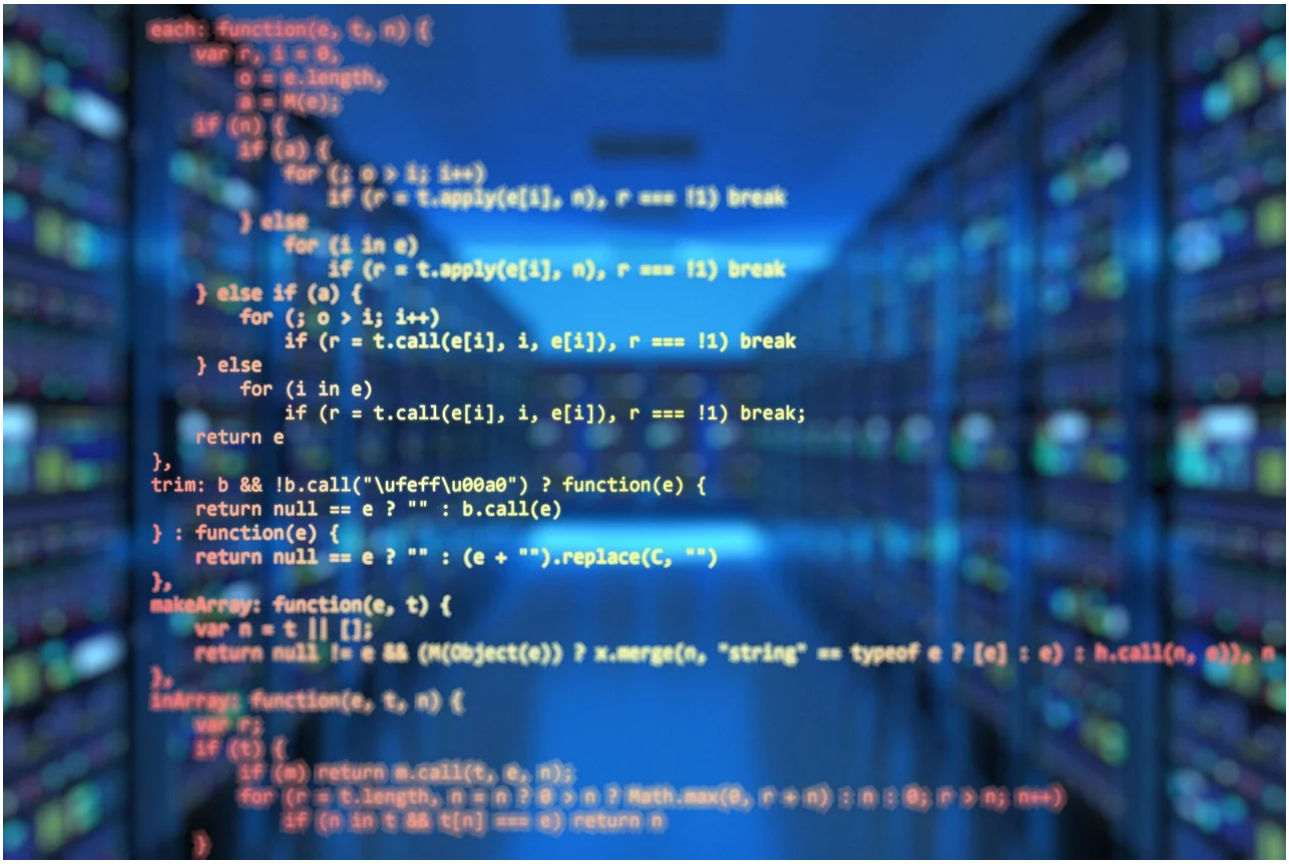
- **Lenguajes de bajo nivel**: son los mas cercanos al binario, pero son difíciles de programar
- **Lenguajes de alto nivel**: necesitan ser traducidos antes de llegar al ordenador pero son fáciles de programar ya que son mas cercanos al lenguaje natural de las personas

Algunos ejemplos:

- Lenguajes bajo nivel: lenguaje máquina
- Lenguajes alto nivel: C, C++, Python, Go

En el desarrollo de software hay muchas condicionantes que marcan que elección de lenguaje se usará en el proyecto:

- Requisitos técnicos marcados por otras piezas de software/hardware
- Tiempo de entrega del software
- Necesidad de rendimiento



Pixabay (Dominio público)

1.2.- Modelos de Ejecución Software.

Podemos clasificar los lenguajes de programación según su modo de ejecución en 3 tipos distintos

1.- Compilado
2.- Interpretado
3.- Intermedio

Pixabay (Dominio público)

Lenguaje compilado:

- Nuestro código fuente se transforma en un binario mediante compilación
- El ordenador ejecuta el binario
- No necesitamos por tanto ningún programa adicional

Lenguaje interpretado:

- Nuestro código fuente es leído en tiempo real por un programa (intérprete) que lo traduce a código máquina(objeto binario)
- El ordenador ejecuta ese binario
- Necesita por tanto un programa adicional, llamado intérprete, que hace de traductor entre las instrucciones y el código máquina

Lenguaje intermedio:

- El código fuente se compila a un lenguaje intermedio
- Este lenguaje intermedio se ejecuta en una máquina virtual

Para saber más

Python se ha convertido en uno de los lenguajes mas usados a nivel mundial para el desarrollo de software. Si bien no tiene el rendimiento que podría tener lenguajes como C o C++ es sencillo de programar, tiene una sintaxis clara, tiene numerosas librerías disponibles y resulta muy divertido programar en Python!

Si te interesa existe un "zen" o mejores prácticas de Python que seguro te resultará interesante. Aquí el link.

Debes conocer

Un compilador no es mas que un software que se encarga de transformar código fuente en lenguaje binario que es mas fácil de entender por los ordenadores. Cuando compilamos un programa elegimos el tipo de plataforma y arquitectura en la que queremos que el binario funcione según el compilador que estemos usando.

Hay lenguajes como Golang que permiten hacer un compilado multiplataforma especificando sobre que plataforma y arquitectura queremos que nuestro binario funcione cuando lo estamos compilando.

Autoevaluación

Identifica si las siguientes frases son verdaderas o falsas

Python es un lenguaje compilado de alto rendimiento

Verdadero Falso

Falso

Python es un lenguaje que necesita un intérprete, que hace un trabajo de compilado en tiempo real, es decir es un lenguaje NO compilado y por tanto su rendimiento será menor.

Ruby es un lenguaje interpretado, es decir necesita un intérprete y por tanto NO es compilado.

Verdadero Falso

Verdadero

Los lenguajes interpretados tienen, valga la redundancia, un intérprete que los compila y ejecuta en tiempo real. Es decir no ha sufrido un proceso de compilación previo.

Golang es un lenguaje compilado y por tanto si rendimiento será mayor.

Verdadero Falso

Verdadero

Golang o go es un lenguaje que esta adquiriendo gran popularidad porque es sencillo de programar como Python pero tiene el rendimiento de un lenguaje compilado.

Scala es un lenguaje intermedio puesto que el código se compila a un lenguaje intermedio para mas tarde ser ejecutado en un entorno controlado o maquina virtual.

Verdadero Falso

Verdadero

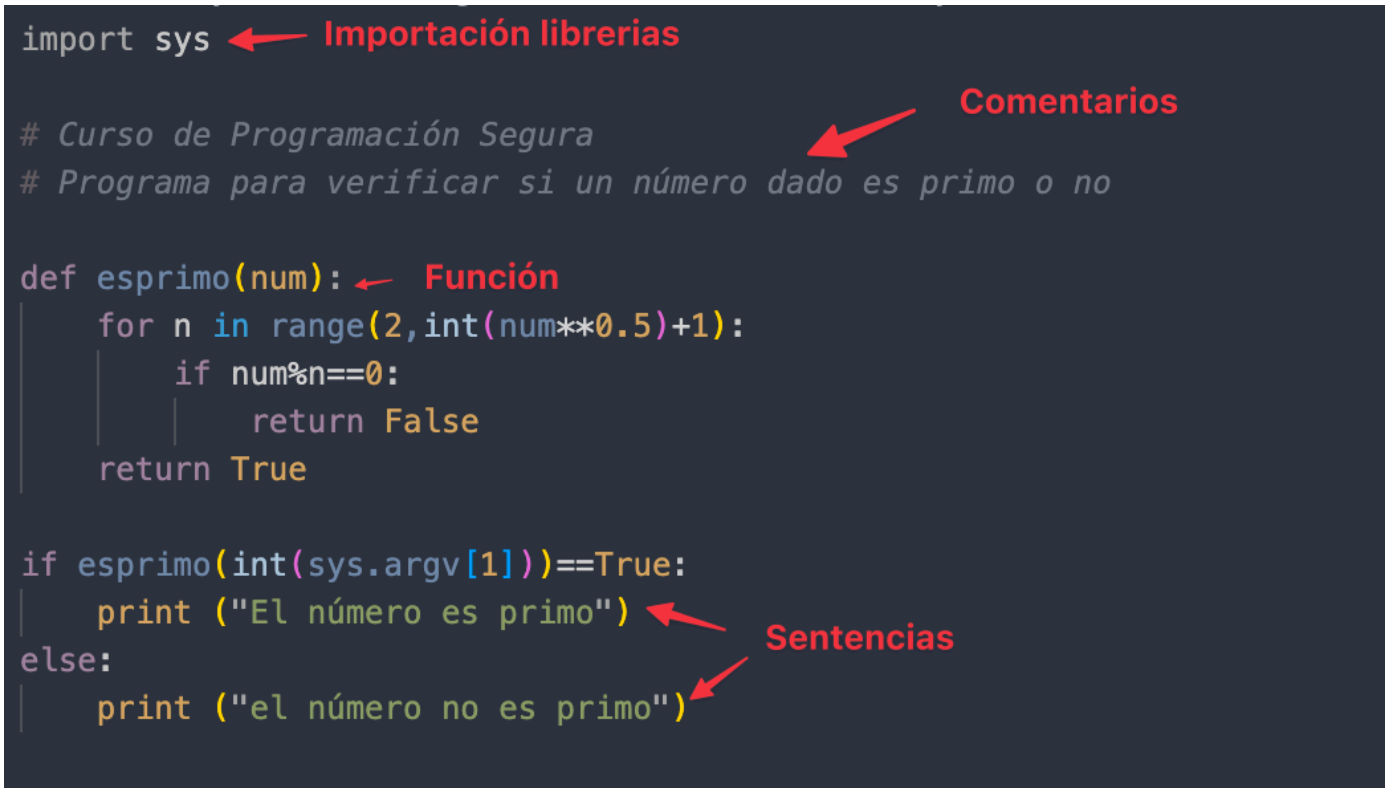
Los lenguajes intermedios sufren de un proceso de compilación y ejecución en un entorno virtual o controlado.

1.3.- Elementos básicos.

Dentro de nuestro programa o código fuente podemos distinguir varios componentes:

- Comandos de pre-procesamiento: importación de módulos/librerías.
- Funciones.
- Declaración de variables.
- Sentencias & Expresiones.
- Comentarios.

En la imagen dada a continuación se puede ver un pequeño programa desarrollado en Python en el que se muestra la estructura básica que sigue el código fuente de un programa.



David Conde (Dominio público)

Citas Para Pensar

"Si la depuración es el proceso de eliminar errores, entonces la

programación debe ser el proceso de introducirlos". Edsger W. Dijkstra

1.4.- Pruebas de software.

Las pruebas de software son un elemento crítico para garantizar la calidad de un producto de software. La necesidad de las compañías de desarrollar software de calidad, ha motivado el diseño de diversas metodologías para el desarrollo y ejecución de pruebas de software. Diversas compañías buscan implementar satisfactoriamente estas metodologías, pero definir los pasos para mejorar y controlar las fases del proceso de pruebas de software y el orden en que estas se implementan es, en general, una tarea difícil.

De forma general las pruebas de software las podemos clasificar en base a cómo las ejecutamos:

1.- Pruebas manuales
2.- Pruebas automáticas

Pero también podemos clasificarlas, en base a lo que estamos midiendo y probando, teniendo muchos tipos distintos. las más comunes son:

- **Pruebas de aceptación**: Verificar si todo el conjunto de software se comporta según lo esperado
- **Pruebas de integración**: Verifica que distintos módulos que funcionan por separado funcionen de manera conjunta
- **Examen de la unidad**: Valida que cada unidad de software funcione como se esperaba. Una unidad es el componente comprobable más pequeño de una aplicación.
- **Pruebas funcionales**: Verifica las funciones emulando escenarios comerciales, basados en requisitos funcionales. Las pruebas de caja negra son una forma común de verificar funciones.
- **Pruebas de rendimiento**: Prueba el rendimiento del software en diferentes cargas de trabajo. Las pruebas de carga, por ejemplo, se utilizan para evaluar el rendimiento en condiciones de carga reales.
- **Pruebas de regresión**: Verifica si las el código introducido rompe o degrada la funcionalidad del software original. Suelen realizarse cuando hacemos cambios en un programa.
- **Pruebas de estrés**: Prueba cuánta tensión puede soportar el sistema antes de que falle. Está considerada como un tipo de prueba no funcional.
- **Pruebas de usabilidad**: Valida qué tan bien un cliente puede usar un sistema o una aplicación web para completar una tarea.

1.5.- Evaluación de lenguajes de programación.

Evaluar los lenguajes de programación en cuanto a seguridad no es tarea sencilla, ya que la mayoría de veces no depende del lenguaje en si, sino del programador, de la tecnología que implementa ese lenguaje y sobre todo del volumen de código que hay publicado de ese lenguaje de programación.

Si tuviéramos que clasificar el lenguaje de programación con mas vulnerabilidades sería sin duda el lenguaje C, casi la mitad de las vulnerabilidades de software reportadas los últimos años están relacionadas con el lenguaje C. Esto nos haría pensar que C es un lenguaje menos seguro que los otros lenguajes pero la explicación de la gran cantidad de vulnerabilidades que hay en C es sencilla:

- El lenguaje C es uno de los lenguajes más antiguos y por tanto se ha usado durante más tiempo, por lo tanto a más tiempo expuesto mas vulnerabilidades encontradas
- Es el lenguaje con mayor volumen de código abierto publicado en internet
- Muchos sistemas operativos como UNIX/Linux, incluso partes de Windows usan C

Realmente si tenemos que evaluar los lenguajes de programación en cuanto a su seguridad deberíamos de valorar mas que el propio lenguaje el ciclo de desarrollo del mismo dentro de un proyecto

Debes conocer

Cuando estamos desarrollando y probando software necesitamos disponer de un entorno de desarrollo aislado donde poder hacer nuestras pruebas y verificar si el software concreto que hemos desarrollado se comporta de la manera esperada ante las distintas variables o datos procesados.

La mejor manera de hacerlo es tener un entorno controlado, no influenciado por otras piezas de software o hardware y donde todos los programadores pueden tener las mismas condiciones y por tanto poder medir y probar el software de manera eficaz y eficiente.

Estos entornos se conocen como s*andbox,* es decir es un entorno cerrado en el que probar los cambios de código de forma aislada, sin comprometer la producción ni la edición del programa en desarrollo. Así, el sandbox protege, controla de forma preventiva y en tiempo real la ejecución del código, evitando cambios que podrían comprometer la estabilidad del programa.

Para saber más

Si bien el término *sandbox* se refiere a un entorno aislado y controlado, dentro de la informática tiene dos usos. A nivel de desarrollo de software, nuestro caso, se podría definir como el entorno donde creamos y probamos nuestro software de manera controlada, mientras que a nivel de ciberseguridad se entiende por *sandbox* el entorno donde se ejecuta de manera aislada el software malicioso o malware para ver como se comporta sin suponer un riesgo.

Autoevaluación

Marca las siguientes sentencias como verdaderas o falsas según consideres.

El lenguaje C es el mas vulnerable puesto que tiene fallos graves de diseño.

○ Verdadero ○ Falso

Falso

Las vulnerabilidades del lenguaje C están más relacionadas con la cantidad de código disponible y con el tiempo que lleva ese código publicado.

Python es un lenguaje muy seguro porque tiene librerías para depurar y hacer correcciones de código.

○ Verdadero ○ Falso

Falso

La mayoría de lenguajes modernos incorporan módulos o librerías que permiten revisar y detectar errores de manera automática o semi automática. No podemos marcar lenguajes como seguros o inseguros de manera categórica, sino su diseño, implantación y validación dentro de un proyecto.

No tiene sentido analizar los posibles datos de entrada de una función mientras estamos programando porque ya haremos la revisión más adelante de manera automática.

○ Verdadero ○ Falso

Falso

Cuando definimos una función deberemos de entender que datos estamos permitiendo de entrada y realmente revisar si lo podemos limitar lo máximo posible para prevenir situaciones no esperadas.

El lenguaje sistema operativo Linux no es muy porque está escrito mayoritariamente en C.

○ Verdadero ○ Falso

Falso

Que un sistema operativo sea robusto o no no depende tanto del lenguaje de programación, como el hecho de si es un sistema operativo abierto y su código está publico, así como el numero de librerías o módulos de terceros que se incluyen sin revisar.