**Curso: Fundamentos de Python.**

**Las computadoras no son inteligentes**, necesitan que el usuario ingrese los datos para que la computadora almacene la información y la procese, pero el usuario es el que decide qué operación realizar, de hecho, la computadora no sabe para que se utilizan dichos datos, solo los procesa.

**El lenguaje maquina son desarrollados por humanos.**

**Tanto el lenguaje maquina como el lenguaje natural cuentan con 4 características:**

**1.- Un Alfabeto:** Es un conjunto de símbolos utilizados para formar una palabra de un determinado lenguaje.

**2.- Un Léxico:** (También conocido como diccionario)Un conjunto de palabras que el lenguaje ofrece a sus usuarios.

**3.- Una Sintaxis:** Un conjunto de cadenas que determinan una cadena de palabras forman una oración valida.

**4.- Una Semántica:** Un conjunto de reglas que determinan si una frase tiene sentido.

**IL=Instrucción list (lista de instrucciones)**

**El lenguaje maquina es muy difícil de comprender para los humanos y los lenguajes naturales son muy difíciles de comprender para la máquina, por lo que se necesitan de puentes o conexiones entre dichos lenguajes y a estos puentes se les denomina lenguaje de alto nivel, que son muy similares a el lenguaje natural, solo que contiene símbolos, y características diferentes, también es más fácil de comprender para el lenguaje máquina.**

Un programa escrito en un lenguaje de alto nivel se le denomina código fuente y el archivo que contiene el código fuente se le denomina archivo fuente

La composición de un programa informático:

* **Alfabéticamente:** Un programa debe estar escrito en un alfabeto reconocible.
* **Léxicamente:** Cada lenguaje tiene su diccionario y hay que comprenderlo, afortunadamente es más reducido que un lenguaje natural cualquiera.
* **Sintácticamente:** Cada idioma tiene sus reglas y hay que obedecerlas.
* **Semánticamente:** El programa debe tener sentido.

**Hay 2 formas diferentes de trasformar un programa de lenguaje de alto nivel a lenguaje máquina.**

1. **Compilación:** Un programa se compila una vez, sin embargo, si el código se modifica se tiene que volver a compilar (a los programas que hace esto se les denomina compiladores o traductores).
2. **Interpretación:** Cualquier usuario del código puede traducir el programa fuente cada vez que se deba ejecutar, el programa que realiza este tipo de transformación se le denomina interprete ya que interpreta el código cada vez que se quiere ejecutar. También significa que no puedes distribuir el código fuente tal cual, porque el usuario final también necesita el intérprete para ejecutarlo.

**Un lenguaje de programación se realiza con la pregunta ¿Sera compilado o será interpretado?**

Tiene que ser un texto puro, sin imágenes, fuentes o colores, de lo contrario el intérprete no podrá leer o detectar el texto correctamente.

|  |  |  |
| --- | --- | --- |
|  | Compilar | Interpretar |
| Ventajas | * La ejecución del código traducido suele ser más rápida. * Solo el usuario debe tener el compilador; el usuario final puede usar el código sin él; * El código traducido se almacena usando lenguaje maquina; como es muy difícil de entender, es probable que tus propios trucos e inventos sigan siendo tu secreto | * Puedes ejecutar el código tan rápido como lo completes; no hay frases adicionales de traducción. * El código se almacena usando un lenguaje de programación, no un lenguaje maquina; esto significa que se pueden ejecutar en computadoras con diferente lenguaje maquina; no compila tu código por separado para cada arquitectura. |
| Desventajas | * La compilación en si puede ser un proceso que consuma mucho tiempo; es posible que no puedas ejecutar tu código inmediatamente después de realizar una modificación; * Debes tener tantos compiladores como plataformas de hardware donde desees que se ejecute tu código. | * No esperes que la interpretación acelere tu código a alta velocidad: tu código compartirá el poder de la computadora con el intérprete, por lo que no puede ser realmente rápido; * Tanto tu como el usuario final deben tener el interprete para ejecutar tu código. |

Los lenguajes diseñados para ser interpretados son denominados lenguajes de scripting, mientras que a los programas interpretados en ellos se les denomina scripts.

Python tiene 2 competidores directos, Perl y Ruby.

**Cython** es una de las posibles soluciones al rasgo de Python más doloroso - la falta de eficiencia. Los cálculos matemáticos grandes y complejos pueden ser fácilmente codificados en Python (mucho más fácil que en "C" o en cualquier otro lenguaje tradicional), pero la ejecución del código resultante puede requerir mucho tiempo.

¿Cómo se reconcilian estas dos contradicciones? Una solución es escribir tus ideas matemáticas usando Python, y cuando estés absolutamente seguro de que tu código es correcto y produce resultados válidos, puedes traducirlo a "C". Ciertamente, "C" se ejecutará mucho más rápido que Python puro.

Esto es lo que pretende hacer Cython: traducir automáticamente el código de Python (limpio y claro, pero no demasiado rápido) al código "C" (complicado y hablador, pero ágil).

Otra versión de Python se llama **Jython.**

"J" es de "Java". Imagina un Python escrito en Java en lugar de C. Esto es útil, por ejemplo, si desarrollas sistemas grandes y complejos escritos completamente en Java y deseas agregarles cierta flexibilidad de Python. El tradicional CPython puede ser difícil de integrar en un entorno de este tipo, ya que C y Java viven en mundos completamente diferentes y no comparten muchas ideas comunes.

Jython puede comunicarse con la infraestructura Java existente de manera más efectiva. Es por esto que algunos proyectos lo encuentran útil y necesario.

Nota: la implementación actual de Jython sigue los estándares de Python 2. Hasta ahora, no hay Jython conforme a Python 3.

Echa un vistazo al logo de abajo. ¿Puedes resolverlo?

Es el logotipo de **PyPy** - un Python dentro de un Python. En otras palabras, representa un entorno de Python escrito en un lenguaje similar a Python llamado RPython (Restricted Python). En realidad es un subconjunto de Python.

El código fuente de PyPy no se ejecuta de manera interpretativa, sino que se traduce al lenguaje de programación C y luego se ejecuta por separado.

Esto es útil porque si deseas probar cualquier característica nueva que pueda ser o no introducida en la implementación de Python, es más fácil verificarla con PyPy que con CPython. Esta es la razón por la que PyPy es más una herramienta para las personas que desarrollan Python que para el resto de los usuarios.

Esto no hace que PyPy sea menos importante o menos serio que CPython.

Además, PyPy es compatible con el lenguaje Python 3.

Hay muchos más Pythons diferentes en el mundo. Los encontrarás sí los buscas, pero este curso se centrará en CPython.

**MicroPython** es una implementación eficiente de software de código abierto de Python 3 que está optimizada para ejecutarse en microcontroladores. Incluye un pequeño subconjunto de la biblioteca estándar de Python, pero está repleto de una gran cantidad de funciones, como mensajes interactivos o números enteros de precisión arbitraria, así como módulos que dan acceso al programador a hardware de bajo nivel.

Creado originalmente por Damien George, un programador australiano, que en el año 2013 realizó una exitosa campaña en Kickstarter y lanzó la primera versión de MicroPython con una placa de desarrollo con tecnología STM32F4 llamada pyboard.

En 2017, MicroPython se utilizó para crear CircuitPython, otro lenguaje de programación de código abierto que se ejecuta en el hardware del microcontrolador, que es un derivado del lenguaje MicroPython.

Evaluacion Modulo 1 (9/10)

1. **¿Qué es el código máquina?**

Un lenguaje de programación de bajo nivel que consiste en dígitos/bits binarios que la computadora lee y entiende.

1. **¿Cuáles son los cuatro elementos fundamentales que componen un lenguaje?**

Un alfabeto, un léxico, una sintaxis y una semántica.

1. **¿Cómo se llama a un archivo que contiene un programa escrito en un lenguaje de programación de alto nivel?**

Un archivo fuente.

1. **¿Qué es cierto sobre la compilación?**

El código se convierte directamente en código máquina ejecutable por el procesador.

Tiende a ser más rápida que la interpretación.

1. **¿Cuál es la mejor definición de un script?**

Es un archivo de texto que contiene instrucciones que componen un programa de Python

1. **Selecciona las afirmaciones que sean verdaderas.**

Python es una buena opción para crear y ejecutar pruebas para aplicaciones.

Python es gratuito, de código abierto y multiplataforma.

1. **¿Cómo se llama un intérprete de línea de comandos que te permite interactuar con tu sistema operativo y ejecutar comandos y scripts de Python?**

Una consola.

1. **¿Cuál es el comportamiento esperado del siguiente programa**?

Print(“¡ Hola ¡”)

El programa mostrará ¡Hola! en la pantalla.

1. **¿Cuál es el comportamiento esperado del siguiente programa?**

**Prin(“¡ Adiós ¡”)**

El programa generará un mensaje de error en la pantalla.

Funciones:

Print(): se utiliza para mostar un mensaje en pantalla, ya sean Cadenas, números, caracteres, valores lógicos, objetos.

No se puede iniciar un print y terminarlo en otra línea.

\ : es un escape breve de la línea

\n: significa newline (lo que provoca que se genere una nueva línea en la consola)

al colocar una \ dentro de una cadena no significa nada, para que signifique algo tendría que ser \\