Julia

Origen e historia:

El lenguaje Julia comienza a gestarse en 2009 como un proyecto liderado por Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman con el objetivo de crear un lenguaje que tuviera una gran capacidad computacionalmente y fuera veloz. En Febrero de 2012 se lanzó y en 2015 se creó Julia Computing para proveer soporte, entrenamiento y servicios de consultoría a clientes (aquellos que ya utilizaban el lenguaje y potenciales interesados). Desde entonces ha resultado un lenguaje sumamente atractivo para resolver problemáticas de alto contenido matemático y análisis numérico en disciplinas como, sólo para dar los ejemplos más importantes, programación concurrente y distribuida y *data science*. Para enero del 2018, la comunidad de Julia ascendía a 1.800.000 descargas.

Carecterísticas básicas

Manejo de variables y constantes matemáticas:

Julia es un lenguaje que busca cooptar la atención de la comunidad científica y especializarse en la facilidad y el poder de manejo matemático. Es por esto que cuenta con características muy interesantes. Para empezar cualquier variable puede adoptar cualquier valor (si en principio no se declara su tipo, más sobre esto más adelante).

Julia cuenta con una serie de constantes matemáticas predefinidas.

También permite el manejo de números irracionales.

Y números racionales (minimizando la pérdida de dígitos por redondeo).

Paradigmas:

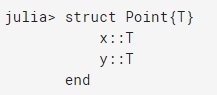
Julia es un lenguaje multiparadigma y como tal permite abordar programación tanto imperativa como funcional y orientada a objetos. Para ello se pueden definir funciones convencionales, funciones lambda, clases e interfaces, solo para mencionar algunas características de estos paradigmas, y combinarlos.

Tipado

En Julia el tipado es dinámico y los tipos son, por defecto, omitidos de manera tal que las variables admitan valores de cualquier tipo. Y esto se da así: son los valores los que tienen tipo mientras que las variables son simples notaciones que hacen referencia a entidades.

De todas formas, el compilador también permite la especificación del tipo esperado para una cierta variable. Esto suele mejorar no solamente la robustez de los programas, sino también su performance.

Una característica interesante de Julia es que la declaración de nuevos tipos puede llevar parámetros y esto introduce una versatilidad importante.



Así, a partir de Point {T} podemos tener una variable que sea de tipo Point {Float64} o Point {Abstract String}, con lo cual tenemos toda una familia de tipos a partir de la definición de una con un parámetro T. Esto es extensible a varios parámetros.

En Julia, todos los tipos son objetos, pero sus métodos no "pertenecen" a la clase, sino que son parte de un listado que el compilador utiliza a la hora de interpretar el código. Ésta es una consideración del diseño del lenguaje que favorece el Multiple Dispatch.

Compilacion e interpretación

Julia posee un compilador denominado JIT (“Just in Time”) que aunqe se utiliza en tiempo de ejecución, el compilador infiere los tipos a través de una primera compilación o “precompilación”. Mediante este mecanismo el lenguaje aparenta ser interpretado en términos de velocidad y así es comparable a lenguajes como C o Fortran, pero con un código muchísimo más simple.

Manejo de errores:

Ante un evento inesperado, Julia tiene un manejo de errores comparable al de la mayoría de los lenguajes. Si la función no puede devolver una salida razonable, entonces levanta una excepción y termina el programa indicando un código de error. En algunos casos, el programador puede especificar hasta el mensaje de error a mostrar.