UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional de Avellaneda

Carrera: Técnico Superior en Programación

Inglés I

Trabajo Práctico N°1

Aragón, M. Gabriela

Legajo: 106746

División: 1°C

13 programming languages defining the future of coding

Faster, smarter programming, with fewer bugs. Those are the promises coming from the creators of the latest round of languages to capture the attention of programmers. Yes, they're the same buzzwords we've heard before, but the lack of novelty is no reason to dismiss them. The future of coding requires stability and good practices so our innovations will work. In fact, our projects are often so much bigger now, we need the innovation more than ever.

If there's a common theme among the languages I describe below, it's that increasing automation can yield code worthy of the terms "faster, smarter, and bug-free." The newer approaches include more structure and more abstraction, allowing the guts of the languages to do what programmers used to have to do themselves. These automated features give the programmer more leverage to concentrate on the big issues. In many cases, they also produce better performance because the automated mechanisms are better able to find opportunities for efficiency and parallel computation while eliminating some of the simple mistakes that lead to errors.

But beyond this one overarching theme, there's little agreement. One of the languages is built for statistical analysis. Several are meant to modernize classic languages. Some aren't even languages at all—they're merely preprocessors. Still, all of them are changing how we're writing code today and laying the foundation for the future of coding.

Here are 13 languages that are changing how we tell computers what to do. Some of these languages are new, some are already very popular, and some aren't actually languages. If you're looking for an article about new programming languages that have a chance to to become industry mainstays, check out 5 emerging programming languages with a bright future.

1. R

At heart, R is a programming language, but it's more of a standard bearer for the world's current obsession with using statistics to unlock patterns in large blocks of data. R was designed by statisticians and scientists to make their work easier. It comes with most standard functions used in data analysis and many of the most useful statistical algorithms are already implemented as freely distributed libraries. It's got most of what data scientists need to do data-driven science.

Many people end up using R inside an IDE as a high-powered scratchpad for playing with data. R Studio and R Commander are two popular front ends that let you load up your data and play with it. They make it less of a compile-and-run language and more of an interactive world in which to do your work.

Highlights: Clever expressions for selecting a subset of the data and analyzing it
Headaches: Aimed at desktops, not the world of big data where technologies like Hadoop rule.

2. Java 8

Java isn't a new language. It's often everyone's first language, thanks to its role as the lingua franca for AP Computer Science. There are billions of JAR files floating around running the world.

But Java 8 is a bit different. It comes with new features aimed at offering functional techniques that can unlock the parallelism in your code. You don't have to use them. You could stick with all the old Java because it still works. But if you don't use it, you'll be missing the chance to offer the Java virtual machine (JVM) even more structure for optimizing the execution. You'll miss the chance to think functionally and write cleaner, faster, and less buggy code.

Highlights: Lambda expressions and concurrent code
Headaches: A bolted-on feeling makes us want to jump in with both feet and use Scala (see below).

13 lenguajes de programación que definen el futuro de la codificación

Más rápidos, con programación más inteligente y menos errores. Esas son las promesas que vienen de los creadores de la más reciente sucesión de lenguajes para captar la atención de los programadores. Sí, son las mismas palabras de moda que ya hemos escuchado, pero la carencia de novedad no es razón para desecharlas. El futuro de la codificación requiere estabilidad y buenas prácticas de esta manera nuestras innovaciones funcionaran. De hecho, nuestros proyectos usualmente son mucho más grandes ahora, necesitamos la innovación más que nunca.

Si hay un tema en común entre los lenguajes que describo a continuación, es que el aumento de la automatización puede producir un código digno de los términos "más rápido, más inteligente y sin errores". Los más recientes enfoques incluyen más estructura y más abstracción, permitiendo que las entrañas del lenguaje hagan lo que los programadores tenían que hacer por ellos mismos. Estas funciones automatizadas dan al programador más empuje para concentrarse en los grandes problemas. En muchos casos, esto también produce un mejor funcionamiento porque los mecanismos automatizados son mucho más capaces de encontrar oportunidades de eficiencia y cálculo paralelo mientras elimina algunos errores simples que llevan a errores.

Pero más allá de este tema general, hay una pequeña coincidencia. uno de los lenguajes esta construido para el análisis estadístico. Varios están destinados a modernizar los lenguajes clásicos. Algunos no son ni siquiera lenguajes, son simplemente pre-procesadores. Sin embargo, todos ellos estan cambiando el modo en el que hoy que escribimos el código y sentando las bases para el futuro de la codificación.

Aquí están 13 lenguajes que están cambiando cómo le decimos a las computadoras qué hacer. Algunos de estos lenguajes son nuevos, unos ya son muy populares, y otros no son verdaderamente lenguajes. Si usted está buscando un artículo sobre los nuevos lenguajes de programación que tienen una oportunidad de convertirse en pilares de la industria, eche un vistazo a 5 lenguajes de programación emergentes con un futuro brillante.

1.R

En el fondo, R es un lenguaje de programación, pero es más que un portador modelo de la actual obsesión del mundo con el uso de estadísticas para desbloquear patrones en largos bloques de datos. R fue diseñado por estadísticos y científicos para hacer más fácil su trabajo. Viene con la mayoría de las funciones estándar usadas en el análisis de datos y muchos de los algoritmos estadísticos más útiles ya están implementados como bibliotecas distribuidas libremente. Tiene la mayoría de los datos que los científicos necesitan para hacer ciencia basada en datos.

Muchas personas terminan usando R dentro de un IDE como un bloc de notas de alta potencia para trabajar con los datos. R Studio y R Commander son dos interfaces populares que permiten cargar tus datos y trabajar con ellos. Hacen esto no sólo como un lenguaje que compila y ejecuta sino más como un mundo interactivo en el cual hacer su trabajo.

*Aspectos destacados: expresiones inteligentes para seleccionar un subconjunto de datos y analizarlos.

*Dolores de cabeza: dirigido a omputadoras de escirtorio, no al mundo de los grandes datos donde tecnologías como Hadoop dominan.

2.Java 8

Java no es un lenguaje nuevo. A menudo es el primer lenguaje de todo el mundo, gracias a su rol como lenguaje libre para AP Computer Science. Hay miles de milllones de archivos JAR flotando alrededor del mundo de ejecutables.

Pero Java 8 es un poco diferente. Viene con nuevas características destinadas a ofrecer técnicas funcionales que pueden desbloquear el paralelismo en su código. No tiene que usarlos. Usted podría apegarse al viejo Java porque todavía funciona. Pero si no lo usa, estará perdiendo la oportunidad de ofrecer a la máquina virtual Java (JVM) todavía más estructura para optimizar la ejecución. Perderá la oportunidad de pensar funcionalmente y escribir un código más limpio, rápido y con menos errores.

*Aspectos destacados: expresiones Lambda y código coincidente.

*Dolores de cabeza: una sensación rebuscada nos hará querer saltar con ambos pies y usar Scala (ver abajo).

3. Swift

Apple saw an opportunity when programming newbies complained about the endless mess of writing in Objective C. So they introduced Swift and strongly implied that it would replace Objective C for writing for the Mac or the iPhone. They recognized that creating header files and juggling pointers was antiquated. Swift hides this information, making it much more like writing in a modern language like Java or Python. Finally, the language is doing all the scut work, just like the modern code.

The language specification is broad. It's not just a syntactic cleanup of Objective C. There are plenty of new features, so many that they're hard to list. Some coders might even complain that there's too much to learn, and Swift will make life more complicated for teams who need to read each other's code. But let's not focus too much on that. iPhone coders can now spin out code as quickly as others. They can work with a cleaner syntax and let the language do the busy work.

Highlights: Dramatically cleaner syntax and less low-level juggling of pointers
Headaches: The backward compatibility requires thinking about bits and bytes occasionally.

4. Go

When Google set out to build a new language to power its server farms, it decided to build something simple by throwing out many of the more clever ideas often found in other languages. They wanted to keep everything, as one creator said, "simple enough to hold in one programmer's head." There are no complex abstractions or clever metaprogramming in Go—just basic features specified in a straightforward syntax.

This can make things easier for everyone on a team because no one has to fret when someone else digs up a neat idea from the nether reaches of the language specification.

Highlights: Just a clean, simple language for manipulating data.
Headaches: Sometimes a clever feature is needed.

5. CoffeeScript

Somewhere along the line, some JavaScript programmers grew tired of typing all those semicolons and curly brackets. So they created CoffeeScript, a preprocessing tool that turns their syntactic shorthand back into regular JavaScript. It's not as much a language as a way to save time hitting all those semicolons and curly bracket keys.

Jokers may claim that CoffeeScript is little more than a way to rest your right hand's pinkie, but they're missing the point. Cleaner code is easier to read, and we all benefit when we can parse the code quickly in our brain. CoffeeScript makes it easier for everyone to understand the code, and that benefits everyone.

Highlights: Cleaner code
Headaches: Sometimes those brackets make it easier to understand deeply nested code.

6. D

For many programmers, there's nothing like the very clean, simple world of C. The syntax is minimal and the structure maps cleanly to the CPU. Some call it portable Assembly. Even for all these advantages, some C programmers feel like they're missing out on the advantages built into newer languages.

That's why D is being built. It's meant to update all the logical purity of C and C++ while adding in modern conveniences such as memory management, type inference, and bounds checking.

Highlights: Some of the most essential new features in languages.
Headaches: You trade some power away for the safety net.

3.Swift

Apple vio una oportunidad cuando los principiantes de programación se quejaron del interminable embrollo de la escritura en el Objetive C. Así que lanzaron Swift implicando fuertemente que se reemplazaría Objetive C para escribir para Mac o Iphone. Acreditaron que la creación de archivos de encabezado y el malabarismo de punteros era anticuado. Swift oculta esta información, haciéndolo mucho más parecido a cómo se escribe en un lenguaje moderni como Java o Python. Finalmente, el lenguaje está haciendo todo el trabajo de blindar, al igual que el código moderno.

La especificación del lenguaje es amplia. No es sólo una limpieza sintáctica del Objetive C. Hay abundantes características nuevas, tantas que son difíciles de enumerar. Algunos codificadores podrían quejarse de que hay mucho que aprender, y Swift hará la vida más complicada para los equipos que necesitan leer el código de otro. Pero no nos enfoquemos demasiado en eso. Los codificadores de Iphone pueden trabajar con una sintaxis más limpia y dejar que el lenguaje haga el trabajo complicado.

*Aspectos destacados: Sintaxis drasticamente más limpia y menor nivel de malabarismos de punteros.

*Dolores de cabeza: La compatibilidad con versiones anteriores requiere ocasionalmente pensar en bits y bytes.

4.Go

Cuando Google se propuso construir un nuevo lenguaje para potenciar sus torres de servidores, decidió construir algo simple desechando muchas ideas inteligentes que se encuentran con frecuencia en otros lenguajes. Ellos querían continuar con todo, como dijo un creador: "lo suficientemente simple como para guardarse en la cabeza de un programador". No hay abstracciones complejas o metaprogramación inteligente en Go, sólo las características básicas especificadas en una sintaxis directa.

Esto puede hacer las cosas más fáciles para todas las personas de un equipo, porque nadie tiene que preocuparse cuando alguien más descubra una buena de los alcances inferiores de la especificación del lenguaje.

*Aspectos destacados: Sólo un lenguaje limpio y simple para manipular datos.

*Dolores de cabeza: A veces se necesita una función inteligente.

5.CoffeeScript

En algún momento de la línea, algunos programadores de JavaScript se cansaron de escribir todos esos puntos y comas. Así que crearon CoffeScript, una herramienta de preprocesamiento que convierte su taquigrafía sintáctica en JavaScript regular. No es tanto un lenguaje sino una forma de ahorrar tiempo tipeando todos esos puntos y comas y llaves.

Los habladores pueden afirmar que CoffeeScript es nada más que una manera de descansar el dedo meñique de su mano derecha, pero están perdiendo el punto. Un código limpio es fácil de leer, y todos nos beneficiamos cuando podemos analizar el código rápidamente en nuestro cerebro. CoffeeScript facilita que todo el mundo entienda el código, y eso beneficia a todos.

*Aspectos destacados: Código más limpio

*Dolores de cabeza: A veces las llaves facilitan la comprensión de un código profundamente anidado.

6.D

Para muchos programadores, no hay nada como el limpio y simple mundo de C. La sintaxis es mínima y los mapas de estructura se limitan a la CPU. Algunos lo llaman Asamblea portátil. Incluso para todas estas ventajas, algunos programadores C sienten como están perdiendo las ventajas incorporadas en los nuevos lenguajes.

Por eso se está construyendo D. Su propósito es actualizar toda la pureza lógica de C y C++ mientras agrega conveniencias modernas como administración de memoria, inferencia de tipos y comprobación de límites.

*Aspectos destacados: Algunos de las nuevas características más esenciales en lenguajes.

*Dolores de cabeza: Usted intercambia un poco de poder por la seguridad en la red.

7. Less.js

Just like CoffeeScript, Less.js is really just a preprocessor for your files, one that makes it easier to create elaborate CSS files. Anyone who has tried to build a list of layout rules for even the simplest website knows that creating basic CSS requires plenty of repetition; Less.js handles all this repetition with loops, variables, and other basic programming constructs. You can, for instance, create a variable to hold that shade of green used as both a background and a highlight color. If the boss wants to change it, you only need to update one spot.

There are more elaborate constructs such as mixins and nested rules that effectively create blocks of standard layout commands that can be included in any number of CSS classes. If someone decides that the bold typeface needs to go, you only need to fix it at the root and Less.js will push the new rule into all the other definitions.

Highlights: Simpler code
Headaches: A few good constructs leave you asking for more.

8. MATLAB

Once upon a time, MATLAB was a hardcore language for hardcore mathematicians and scientists who needed to juggle complex systems of equations and find solutions. It's still that, and more of today's projects need those complex skills. So MATLAB is finding its way into more applications as developers start pushing deeper into complex mathematical and statistical analysis. The core has been tested over the decades by mathematicians and now it's able to help mere mortals.

Highlights: Fast, stable, and solid algorithms for complex math
Headaches: The math is still complex.

9. Arduino

The Internet of Things is coming. More and more devices have embedded chips just waiting to be told what to do. Arduino isn't so much a new language as a set of C or C++ functions that you string together. The compiler does the rest of the work.

Many of these functions will be a real novelty for programmers, especially programmers used to creating user interfaces for general computers. You can read voltages, check the status of pins on the board, and of course, control just how those LEDs flash to send inscrutable messages to the people staring at the device.

Highlights: The world of devices is your oyster.
Headaches: It's largely C and C++.

10. CUDA

Most people take the power of their video cards for granted. They don't even think about how many triangles the video card is juggling, as long as their world is a complex, first-person shooter game. But if they would only look under the hood, they would find a great deal of power ready to be unlocked by the right programmer. The CUDA language is a way for Nvidia to open up the power of their graphics processing units (GPUs) to work in ways other than killing zombies or robots.

The key challenge to using CUDA is learning to identify the parallel parts of your algorithm. Once you find them, you can set up the CUDA code to blast through these sections using all the inherent parallel power of the video card. Some jobs, like mining Bitcoins, are pretty simple, but other challenges, like sorting and molecular dynamics, may take a bit more thinking. Scientists love using CUDA code for their large, multidimensional simulations.

Highlights: Very fast performance, at least for parallel code.
Headaches: Identifying the easily parallelizable sections of code isn't always easy.

7.Less.js

Al igual que CoffeeScript, Less.js es verdaderamente sólo un preprocesador para sus archivos, uno que hace más fácil crear elaborados archivos CSS. Cualquier persona que ha intentado construir una lista de regla de diseño incluso para el sitio web más sencillo sabe que la creación básica de CSS requiere abundantes repeticiones; Less.js maneja todas estas repeticiones con bucles, variables y otras construcciones básicas de programación. Usted puede, por ejemplo, crear una variable para mantener el mismo tono de verde para utilizarlo como fondo y como color de resaltado. Si el jefe quiere cambiarlo, sólo necesita actualizar un punto especifico.

Hay construcciones más elaboradas como mixins y reglas anidadas que crean efectivamente bloques de comando de disposición estándar que pueden ser incluidos en cualquier numero de clase CSS. Si alguien decide que la fuente en negrita tiene que ir, usted sólo necesita corregirla en su origen y Less.js empujará la nueva regla a todas las demás definiciones.

*Aspectos destacados: Código más simple

*Dolores de cabeza: Algunas buenas construcciones lo dejan pidiendo por más.

8.MATLAB

Había una vez, MATLAB era un lenguaje incondicional para matemáticos y científicos duros que necesitan hacer juegos de malabares con complejos sistemas de ecuaciones y encontrar soluciones. Sigue siendo eso, y más proyectos actualmente necesitan esas habilidades complejas. Entonces MATLAB está encontrando su camino en más aplicaciones a medida que los desarrolladores empiezan a profundizar en complejos análisis matemáticos y estadísticos. El núcleo ha sido probado por décadas por matemáticos y ahora es capaz de ayudar a los simples mortales.

*Aspectos destacados: algoritmos rápidos, estables y sólidos para matemáticas complejas.

*Dolores de cabeza: las matemáticas todavís son complejas.

9.Arduino

La Internet de las cosa está llegando. más y más dispositivos tienen chips incrustados esperando que se les diga que hacer. Arduino no es tanto un nuevo lenguaje sino un conjunto de funciones de C o C++ unidas. El compilador hace el resto del trabajo.

Muchas de estas funciones serán una verdadera novedad para los programadores, especialmente los programadores usados para crear interfaces de usuario para computadoras en general. Puede leer voltajes, comprobar el estado de los pines en la placa, y por supuesto, controlar como esos LEDs parpadean para enviar mensajes enigmáticos a las personas que miran el dispositivo.

*Aspectos destacados: el mundo de los dispositivos es tu ostra.

*Dolores de cabeza: es en gran parte C y C++.

10.CUDA

La mayoría de la gente hace un buen de su placa de vídeo por supuesto. Ni siquiera piensan en cuántos triángulos está malabareando la placa de vídeo, siempre y cuando su mundo sea un complejo juego de disparos en primera persona. Pero si sólo miraran bajo el capó, encontrarían una gran cantidad de potencia lista para ser desbloqueada por el programador correcto. El lenguaje CUDA es una forma de que NVIDIA extienda el poder de sus unidades de procesamiento gráfico (GPU) para trabajar de manera diferente a matar zombies o robots.

El desafío clave para usa CUDA es aprender a identificar las partes paralelas de su algoritmo. Una vez que las encuentre, puede configurar el código CUDA para explotar a través de estas secciones utilizando toda la potencia paralela inherente de la tarjeta de vídeo. Algunos trabajos, como la extracción Bitcoins, son bastante simples, pero otros desafíos, como la clasificación y la dinámica molecular, puede tomar un poco más de razonamiento. A los científicos les encanta usar el código CUDA para sus grandes simulaciones multidimensionales.

*Aspectos destacados: rápido rendimiento, al menos para código paralelo.

*Dolores de cabeza: Identificar las secciones fácilmente paralelizables del código no siempre es sencillo.

11. Scala

Everyone who's taken an advanced course in programming languages knows the academic world loves the idea of functional programming, which insists that each function have well-defined inputs and outputs but no way of messing with other variables. There are dozens of good functional languages, and it would be impossible to add all of them here. Scala is one of the best-known, with one of the larger user bases. It was engineered to run on the JVM, so anything you write in Scala can run anywhere that Java runs—which is almost everywhere.

There are good reasons to believe that functional programming precepts, when followed, can build stronger code that's easier to optimize and often free of some of the most maddening bugs. Scala is one way to dip your toe into these waters.

Highlights: Functional, but flexible enough to play well with others using the JVM
Headaches: Thinking functionally can be difficult for some tasks and applications.

12. Haskell

Scala isn't the only functional language with a serious fan base. One of the most popular functional languages, Haskell, is another good place for programmers to begin. It's already being used for major projects at companies like Facebook. It's delivering real performance on real projects, something that often isn't the case for academic code.

Highlights: Already battle tested
Headaches: Thinking functionally can require fixing some bad habits.

13. Jolt

When XML was the big data format, a functional language called XSLT was one of the better tools for fiddling with large datasets coded in XML. Now that JSON has taken over the world, Jolt is one of the options for massaging your JSON data and transforming it. You can write simple filters that extract attributes and JOLT will find them and morph them as you desire. See also Tempo and using XSLT itself.

Highlights: Very simple for many common JSON problems
Headaches: Some JSON transformations are close to impossible.

No generalizations here

It's hard to generalize much about the new languages, at least beyond the promises that they'll produce code that is faster, smarter, and contains fewer bugs. In fact, it's a bit of a stretch to call them new. The history for some of these languages stretches back years, even decades. They just seem new, now that they're being discovered by the larger world.

https://techbeacon.com/13-programming-languages-defining-future-coding

11.Scala

Todo el mundo que ha tomado un curso avanzado en lenguajes de programación sabe que el mundo académico ama la idea de programación funcional, que insiste en que cada función tiene entradas y salidas bien definidas, pero no hay forma de jugar con otras variables. Hay docenas de buenos lenguajes funcionales, y sería imposible agregar todos ellos aquí. Scala es una de las más conocidas, con una de las bases de usuarios más grandes. Fue diseñado para ejecutarse en JVM, así que cualquier cosa que escriba en Scala puede ejecutarse en cualquier lugar que ejecute Java, lo cual es casi en todas partes.

Hay buenas razones para creer que los preceptos de programación funcional, cuando se siguen, pueden construir un código más fuerte que es más fácil de optimizar ya que algunos de los errores son de lo más enloquecedores. Scala es una forma de sumergir el dedo en estas aguas.

*Aspectos destacados: Funcional, pero lo suficientemente flexible como para trabajar bien con otros usuarios de JVM

*Dolores de cabeza: Pensar funcionalmente puede ser difícil para algunas tareas y aplicaciones.

12.Haskell

Scala no es el único lenguaje funcional con una importante base de aficionados. Uno de los idiomas funcionales más populares, Haskell, es otro buen lugar para que los programadores se inicien. Ya se está utilizando para proyectos importantes en empresas como Facebook. Está dando un rendimiento real en proyectos reales, algo que a menudo no es el caso para el código académico.

*Aspectos destacados: ya probado

*Dolores de cabeza: pensar funcionalmente puede requerir la fijación de algunos malos hábitos.

13.Jolt

Cuando XML era el formato grande de datos, un lenguaje funcional llamado XSLT era una de las mejores herramientas para manipular grandes conjuntos de datos codificados en XML. Ahora que JSON se ha hecho cargo del mundo, Jolt es una de las opciones para manipular sus datos JSON y transformarlos. Puede escribir filtros sencillos que extraigan atributos y JOLT los encontrará y los modificará como desee. Ver también Tempo y usar XSLT.

*Aspectos destacados: muy simple para muchos problemas comunes de JSON

*Dolores de cabeza: algunas transformaciones JSON son casi imposibles.

Aquí o hay que generalizar

Es difícil generalizar mucho sobre los nuevos lenguajes, al menos más allá de las promesas de que producirán código más rápido, más inteligente y contiene menos errores. De hecho, es un poco difícil llamarlos nuevos. La historia de algunos de estos lenguajes se remonta a años, incluso décadas. Simplemente parecen nuevos, ahora que están siendo descubiertos por el extenso mundo.

https://techbeacon.com/13-programming-languages-defining-future-coding

Lista de diez preguntas en base al texto traducido:

1- How many languages are changing how we tell computers what to do?

2- Which are the promises from the creators of the latest round of languages?

3- Were statisticians and scientists designed R?

4- Will you miss the chance to think functionally and write cleaner, faster, and less buggy code?

5- Did Apple see an opportunity?

6- When did Google decide to create Go language?

7- Why programmers created CoffeeScript?

8- Are programmers going to create a new language?

9- What does she do in Arduino usually?

10- Does you write in the programming language called Scala ?