1. Toda la info que pude encontrar:

FORTRAN (1955)

Primer lenguaje comercial

Lenguaje imperativo: estado, estructuras de control, contador de programa, celdas de memoria

Lenguaje compilado

LISP (1958)

Lenguaje funcional: funciones, recursión, listas, símbolos

Lenguaje interpretado

COBOL (1959) lenguaje de programación universal (portabilidad), con sintaxis más legible

Estructuras de bloques anidadas (código y declaraciones asociadas en bloques)

Bloques con variables, procedimientos y funciones propias (ocultamiento de la información)

Sintaxis con BNF

BASIC sencillo de aprender y interprete ocupaba poca memoria

Logo programación de una forma gráfica

Algol 68

la sintaxis y la semántica se hizo aún más ortogonal, con rutinas anónimas, un sistema de tipificación recursiva con funciones de orden superior, etc.

Tanto la sintaxis como la semántica del lenguaje completo fueron definidos formalmente, en términos de una gramática de Van Wijngaarden

bloques simultáneos y paralelos complejo sistema de atajos sintácticos y coerciones automáticas de tipo

Simula

el primer lenguaje que define conceptos como clase u objeto. Es el origen de la programación orientada a objetos estática y fuertemente tipeada

Smalltalk (mediados de los 70) proporcionaron un completo diseño de un lenguaje orientado a objetos.

C, en principio un lenguaje de programación de sistemas

Prolog, fue el primer lenguaje de programación lógica.

\_\_

1959

LENGUAJES DE ALTO NIVEL

1967

La Programación Orientación a Objetos (P.O.O.)

1970

tecnologías de inteligencia artificial tanto en el plano del hardware como del software

1980

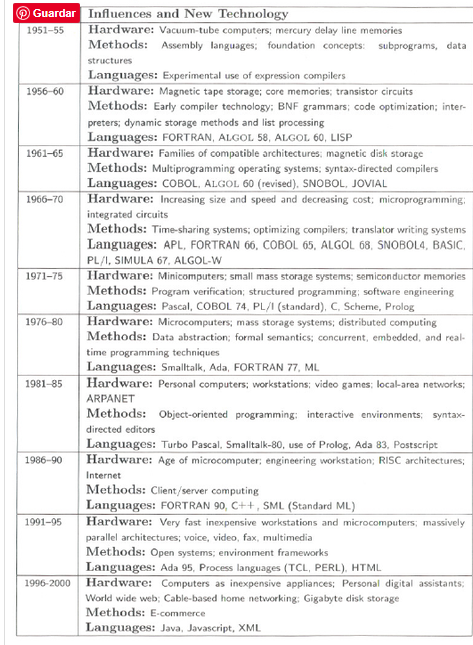
consolidación relativa en los lenguajes imperativos

C++ combinaba la programación orientada a objetos y la programación de sistemas

Lenguajes que incorporaban construcciones de la programación lógica

1990 a 2000

JavaScript se hizo popular debido a su pronta integración con el navegador web Netscape Navigator, y varios lenguajes de scripting alcanzaron un amplio uso en el desarrollo de aplicaciones personalizadas para servidores web

IDE, recolector de basura

Más info acá:

https://www.negocioscontralaobsolescencia.com/tecnologia/el-arbol-genealogico-de-los-lenguajes-de-programacion-de-alto-nivel-los-hijos-de-fortran

1. Historia de Javascript:

A principios de los años 90, la mayoría de usuarios que se conectaban a Internet lo hacían con módems a una velocidad máxima de 28.8 kbps. En esa época, empezaban a desarrollarse las primeras aplicaciones web y por tanto, las páginas web comenzaban a incluir formularios complejos.

Con unas aplicaciones web cada vez más complejas y una velocidad de navegación tan lenta, surgió la necesidad de un lenguaje de programación que se ejecutara en el navegador del usuario, para que las velocidades de respuesta sean mejores.

Brendan Eich, un programador que trabajaba en Netscape, pensó que podría solucionar este problema adaptando otras tecnologías existentes (como ScriptEase) al navegador Netscape Navigator 2.0, que iba a lanzarse en 1995.

Netscape firmó una alianza con Sun Microsystems para el desarrollo del nuevo lenguaje de programación, JavaScript. En un principio iba a llamarse Mocha, y luego LiveScript, pero la palabra Java era más popular, además que en el momento que se decidió cambiarlo a JavaScript, Netscape agregó compatibilidad con la tecnología Java.

La primera versión de JavaScript fue un completo éxito y Netscape Navigator 3.0 ya incorporaba la siguiente versión del lenguaje, la versión 1.1. Al mismo tiempo, Microsoft lanzó JScript con su navegador Internet Explorer 3.

Para evitar una guerra de tecnologías, Netscape decidió que lo mejor sería estandarizar el lenguaje JavaScript. De esta forma, en 1997 se envió la especificación JavaScript 1.1 al organismo ECMA European Computer Manufacturers Association).

ECMA creó el comité TC39 con el objetivo de "estandarizar de un lenguaje de script multiplataforma e independiente de cualquier empresa". El primer estándar que creó el comité TC39 se denominó ECMA-262, en el que se definió por primera vez el lenguaje ECMAScript.

JavaScript no es más que la implementación que realizó la empresa Netscape del estándar ECMAScript.

La organización internacional para la estandarización (ISO) adoptó el estándar ECMA-262 a través de su comisión IEC, dando lugar al estándar ISO/IEC-16262.

Con el paso del tiempo y la popularización de JavaScript (el cuál no fue tan aceptado en sus comienzos), hubo una proliferación de un conjunto de frameworks y librerías de ámbito general, mejorando las prácticas de programación con JavaScript, y aumentado el uso de JavaScript fuera de los navegadores web, como se ha visto con la proliferación de entornos JavaScript del lado del servidor.

1. Los atibutos que debería tener un buen lenguaje de programación son:

* Simplicidad y legibilidad (como Pascal)
* Claridad en los bindings (como Java)
* Confiabilidad (como Java)
* Soporte (como Python)
* Abstracción (como Python)
* Ortogonalidad (como Ada o Python)
* Eficiencia (como Python supongo)

4)

Los lenguajes son Java y Python

1. Tipo de expresiones de Java:

int valor = 1; (inicialización)

if (valor1 > valor2) { ... } (condicional)

for (i=1;i<10;i++) { … } (repetición)

Circulo miCirculo = new Circulo(2,3); (creación de un objeto)

Tipo de expresiones de Python:

Variable = 1 (inicialización)

If (valor1 < valor2):

…

(condicional)

for x in range(25):

…

(repetición)

Etc

b) Python nos permite hacer directorios con un archivo llamado init que nos permite setear ciertas variables incluso, la palabra function para hacer funciones, y ciertas reglas de escritura para utilizar objetos. Más allá de eso no posee alguna organización.

Java nos permite estructurar todo el código que escribimos en diferentes archivos y utilizarlos con un main posteriormente. Tiene varias reglas para programar y utilizar los objetos.

c)

Para Java:

* Simplicidad y legibilidad: No, para ejecutar un programa en Java se deben seguir un conjunto de reglas que son difíciles de entender en un principio, el ser tan estructurado es difícil de aprender/enseñar.
* Claridad en los bindings: Si.
* Confiabilidad: Si, chequeo de tipos muy desarrollado y posee manejo de excepciones
* Soporte: Si, es público y posee gran cantidad de información acerca de él.
* Abstracción: Si, posee elementos como el garbage collector, también librerías varias que permiten abstraernos de realizar varias tareas de bajo nivel.
* Ortogonalidad: Si, considero que es ortogonal porque se pueden combinar varios elementos tanto para la producción de objetos, como para la recursividad y el pasaje de parámetros.
* Eficiencia: Si.

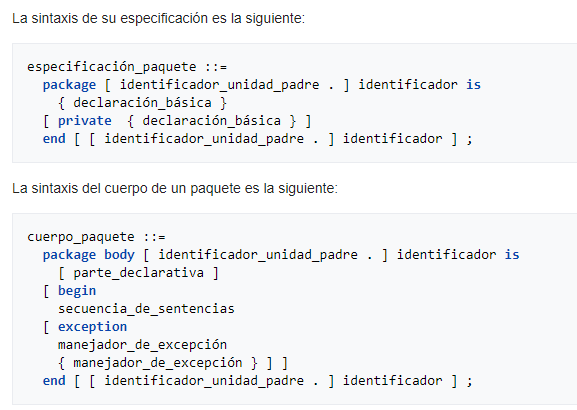
Para Python:

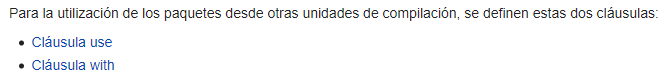
* Simplicidad y legibilidad: Si, es un lenguaje que no requiere ningún tipo de estructura compleja para ser interpretado, muy fácil a la hora de aprender. También posee un buen uso de la indentación, lo que lo hace bastante legible. Sin embargo hay funciones de Python que pueden cargar un poco cargada de operadores e ir encontra de este principio, pero en general cumple con las condiciones.
* Claridad en los bindings: Si.
* Confiabilidad: Relativa, posee un buen manejo de excepciones pero al tener un tipado dinámico puede perder un poco de confiabilidad en ese sentido.
* Soporte: Si, fácil de instalar, de dominio público y con toneladas de documentación tanto oficial como de la comunidad.
* Abstracción: Si, posee varias librerías que nos permiten abstraernos con funciones muy útiles, por ejemplo: lambda, map, list, sort, etc.
* Ortogonalidad: Si.
* Eficiencia: Si, es frecuentemente utilizado en ámbitos científicos.

5) Ada:

Tipos de datos:

* Integer
* Natural
* Positive (naturales sin 0)
* Float
* Character
* String

Tipos abstractos de datos – paquetes:

Los paquetes se pueden declarar en cualquier parte declarativa, es decir, en un bloque, subprograma o dentro de otro paquete. 

Estructuras de datos (tipos de datos estructurados):

* Arrays (estructuras homogéneas). Hay dos clases: restringidos y no restringidos. Los restringidos son arrays para los que se especifica el tamaño en su definición (tamaño fijo). Los no restringidos son aquellos en los que el rango de los índices no se establece al definir el tipo, sino que se concreta posteriormente.
* Rodajas (slices). Se puede hacer referencia a un trozo de un array monodimensional, simplemente especificando sus límites. Este tipo de "corte" se conoce como rodaja o slice.
* Records (estructuras heterogéneas).

Manejo de excepciones:

Las excepciones de ada son:

* Constraint\_error.- ocurre cuando se intenta asignar a una variable un valor no válido, o cuando se intenta acceder a una posición de un array fuera del rango permitido.
* Program\_error.- ocurre en situaciones extrañas cuando parte de un programa no es accesible, o cuando se alcanza el "end" de una función sin encontrar un "return".
* Storage\_error.- ocurre cuando se agota la memoria disponible.
* Tasking\_error.- está relacionado con errores en programas que utilicen programación concurrente.

Y relacionadas a la E/S:

* Status\_Error.- ocurre cuando se intenta leer o escribir en un fichero que no está abierto, o abrir un fichero que está abierto.
* Mode\_Error.- ocurre cuando se intenta leer de un fichero que está abierto para escritura o escribir en un fichero que está abierto para lectura.
* Name\_Error.- ocurre cuando se intenta abrir un fichero y el nombre externo es incorrecto.
* Use\_Error.- ocurre cuando se intenta abrir un fichero para un uso ilegal (p.e. si se intenta crear un fichero con un nombre externo que ya existe).
* Device\_Error.- ocurre cuando se produce un fallo técnico en un dispositivo de entrada/salida.
* End\_Error.- ocurre cuando se intenta leer de un fichero en el que se ha alcanzado la marca de fin de fichero.
* Data\_Error.- ocurre cuando se intenta leer un valor entero, real o enumerado y los datos de entrada tienen un formato incorrecto.

Podemos capturar las excepciones y controlarlas o reenviarlas a otra parte del programa. También se nos permite ignorarlas.

Manejo de concurrencia:

En Ada, las actividades concurrentes reciben el nombre de tareas

Las tareas se pueden comunicar y sincronizar mediante diversos mecanismos:

* Variables compartidas
* Objetos protegidos
* Citas

Las declaraciones de tareas tienen dos partes:

* Especificación: contiene la interfaz visible de la tarea (nombre, parámetros, elementos de comunicación y otras cosas)
* Cuerpo: contiene las instrucciones que ejecuta la tarea.

La ejecución de una tarea pasa por tres fases:

* Activación: se elabora la zona declarativa del cuerpo de la tarea (se crean y se inician los objetos locales)
* Ejecución: se ejecuta la secuencia de instrucciones del cuerpo de la tarea
* Finalización: se ejecutan operaciones finales en los objetos locales

6) Java se creó como una herramienta de programación para ser usada en un proyecto de set-top-box en una pequeña operación denominada the Green Project en Sun Microsystems en el año 1991.

El objetivo de java era crear un lenguaje de programación parecido a C++ en estructura y sintaxis, fuertemente orientado a objetos, pero con una máquina virtual propia. Esto se hizo bajo el principio, de poder ser usado bajo cualquier arquitectura "Write Once, Run Anywhere (escríbelo una vez, ejecútalo en cualquier sitio)".

7) Está basado en C++ principalmente, también Pascal y Objetive-C

8)

Applet:

Son programas escritos en Java, por lo general de poco tamaño, que se incrustan en un documento HTML, y el navegador lo ejecuta en las páginas webs para proporcionar alguna funcionalidad.

La idea de los applets es que sean lo suficientemente pequeños como para proporcionar una funcionalidad específica y claramente definida.

Servlets:

Es una clase de Java utilizada para ampliar las capacidades de un servidor. Son utilizados comúnmente para extender las aplicaciones alojadas por servidores web (serían como applets pero para servidores).

9)

Si existe el anidamiento de funciones, la estructura del programa se basa en una función principal (llamada main) la cual se ejecuta y en base al código de esa función se va desarrollando el programa, pudiendo llamar a otras funciones.

10)

Una expresión es un conjunto de variables y constantes, y también de otras expresiones más sencillas, relacionadas mediante distintos operadores, cuya evaluación debe dar un resultado del mismo tipo de la expresión. Las expresiones pueden contener paréntesis (...) que agrupan a algunos de sus términos. Puede haber paréntesis contenidos dentro de otros paréntesis. El significado de los paréntesis coincide con el habitual en las expresiones matemáticas.

11)

Python se usa para:

* Desarrollo web
* Ciencia de datos, incluido el aprendizaje automático, el análisis de datos y la visualización de datos
* Scripting

Pertenece al paradigma: es multiparadigma, nos permite utilizar el paradigma que queramos

Ruby se usa para:

Propósito general, es decir, con Ruby se pueden desarrollar todo tipo de aplicaciones diferentes: aplicaciones de servicio web, clientes de correo electrónico, procesamiento de datos en Backend, aplicaciones de red, etc.

Pertenece al paradigma: es multiparadigma, aunque tiene una tendencia fuerte a orientado a objetos.

PHP se usa para:

Scripts del lado del servidor (principalmente)

Scripts desde la línea de comandos

Escribir aplicaciones de escritorio

Pertenece al paradigma: es multiparadigma, nos permite utilizar el paradigma que queramos

12)

Python es interpretado, fuertemente tipado, dinámico, multiplataforma

Ruby es interpretado, reflexivo, es portable, tiene manejo de excepciones, manejo de hilos, garbage collector, tiene cuatro niveles de ámbito de variable: global, clase, instancia y local. ‎Fuertemente tipado‎, ‎dinámico. Soporta polimorfismo de tipos.

PHP tiene manejo de excepciones, es interpretado, tipado dinámico y débil. Tiene gran compatibilidad con muchas bases de datos.

Grobstone está diseñado para la enseñanza de ideas básicas de programación. Posee bloques de código encerrados por llaves, definiciones de procedimientos y funciones, parámetros por valor, variables exclusivamente locales. Tiene tipos primitivos que intentan ser intuitivos, pertenecientes a un universo de discurso diseñado especialmente para este lenguaje.

Presenta un tablero que posee un elemento activo, denominado "cabezal", que apunta en todo momento a una celda del mismo. El cabezal puede realizar diversas acciones, tales como poner y sacar bolitas de diferentes colores, moverse a otras celdas del tablero y otras. Si el cabezal se mueve a una celda no existente o intenta sacar bolitas de una celda vacía, el programa finaliza de forma anormal. Los comandos describen acciones que realiza el cabezal sobre el tablero, y por lo tanto generan efectos. Las expresiones se limitan solo a denotar valores (no poseen efectos laterales). El usuario puede definir procedimientos (que dan nombre a comandos nuevos) y funciones puras. El lenguaje no posee entrada y salida de datos. Al final de la ejecución de un programa Gobstones se visualiza un tablero final que resulta de los efectos descritos en el programa.

Processing sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital.

Processing incluye una ventana visual como complemento al contorno del entorno de desarrollo integrado (IDE) para organizarlas en los proyectos.

Cada esquema de Processing es en realidad una subclase de PApplet, un tipo Java que pone en funcionamiento la mayor parte de las características del lenguaje del Processing.

Al programar en Processing, todas las clases adicionales definidas serán tratadas como clases internas cuando el código se traduce en puro Java antes de compilar. Esto significa que el uso de variables estáticas y métodos de las clases está prohibido a menos que se indique específicamente a Processing qué quiere el código en modo puro Java.

Processing también permite a los usuarios crear sus propias clases de PApplet en la ventana. Esto permite que los tipos de datos complejos puedan incluir cualquier número de argumentos y evita las limitaciones al uso de tipos de datos estándar como int (entero), char (caracteres), float (número real) o color (RGB, hexadecimal ARGB).

13)

Es un lenguaje de programación interpretado, orientado a objetos.

14)

Es basado en prototipos, imperativo, débilmente tipado y dinámico, tiene variables numéricas, strings, arrays y booleanos. La estructura no está definida, pero suele ser: comentario informativo del script, declaración de variables, declaración de funciones, instrucciones,