­­­Conceptos y paradigmas de lenguajes de programación – Practica 1

**Ejercicio 1)**

|  |  |
| --- | --- |
| Años | Influencias y nuevas tecnologías |
| 1951 - 1955​: | **Hardware:** Computadoras de tubo de vacío; Memoria de línea de retardo de mercurio.  **Métodos:** lenguajes tipo Assembly; conceptos fundamentales; subprogramas; estructuras de datos.  **Lenguajes: Lenguajes tipo Assembly. Uso experimental de compiladores de expresión.** |
| 1956 - 1960​: | **Hardware:** Almacenamiento de cinta magnética. Memorias centrales. Circuitos de transistores.  **Métodos:** Tecnología de compilación temprana. Optimización de código. Intérpretes. Métodos de almacenamiento dinámico y procesamiento de listas. Gramática BNF.  **Lenguajes: FORTRAN, ALGOL 58, ALGOL 60, LISP.** |
| 1961 - 1965​: | **Hardware:** Familias de arquitecturas compatibles; almacenamiento en disco magnético.  **Métodos:** Multiprogramación de sistemas operativos; compiladores dirigidos por la sintaxis.  **Lenguajes: COBOL, ALGOL 60, SNOBOL, JOVIAL.** |
| 1966 - 1970​: | **Hardware:** Aumentar el tamaño y la velocidad; y disminuir el costo. Circuitos integrados de microprogramación.  **Métodos:** Sistemas de tiempo compartidos. Optimización de compiladores. Traductor de escritura del sistema.  **Lenguajes: APL, FORTRAN 66, BASIC, PL/I, SIMULA 67, ALGOL-W.** |
| 1971 - 1975​: | **Hardware:** Minicomputadoras. Pequeños sistemas de almacenamiento masivo. Memorias semiconductoras.  **Métodos:** Verificación del programa. Programación estructurada. Ingeniería de software.  **Lenguajes: Pascal, C, Scheme, Prolog.** |
| 1976 - 1980​: | **Hardware:** Microcomputadoras. Sistemas de almacenamiento masivo. Computación distribuida.  **Métodos:** Abstracción de datos. semántica formal. Técnicas de programación concurrente, embebidas y en tiempo real.  **Lenguajes: Smalltalk, Ada, FORTRAN 77, ML** |
| 1981 - 1985​: | **Hardware:** Computadoras personales. Estaciones de trabajo. Videojuegos. Redes de área local. ARPANET.  **Métodos:** Programación orientada a objetos. Entornos interactivos. Editores dirigidos por la sintaxis.  **Lenguajes: Smalltalk 80, Turbo Pascal, PostScript** |
| 1986 - 1990​: | **Hardware:** Edad del microordenador. Estación de trabajo de ingeniería. Arquitecturas RISC. Internet.  **Métodos:** Computación cliente/servidor.  **Lenguajes: FORTRAN 90, C++, SML** |
| 1991 - 1995​: | **Hardware:** Muy rápidos estaciones de trabajo de bajo costo y microcomputadoras. Arquitecturas masivamente paralelas. Voz, video, fax, multimedia.  **Métodos:** Sistemas abiertos. Marcos de entorno.  **Lenguajes: TCL, PERL, HTML** |
| 1996 - 2000​: | **Hardware:** Las computadoras como aparatos de bajo costo. Asistentes digitales personales. Red mundial. Redes domésticas basadas en cable. Almacenamiento en disco Gigabyte.  **Métodos:** Comercio electrónico.  **Lenguajes: Java, JavaScript, XML** |

**Ejercicio 2)**

Python fue creado a finales de los ochenta por [Guido van Rossum](https://es.wikipedia.org/wiki/Guido_van_Rossum) en el Centro para las Matemáticas y la Informática (CWI, Centrum Wiskunde & Informatica), en los [Países Bajos](https://es.wikipedia.org/wiki/Pa%C3%ADses_Bajos), como un sucesor del [lenguaje de programación ABC](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_ABC), capaz de [manejar excepciones](https://es.wikipedia.org/wiki/Manejo_de_excepciones) e interactuar con el [sistema operativo Amoeba](https://es.wikipedia.org/wiki/Amoeba_(Inform%C3%A1tica)).

El nombre del lenguaje proviene de la afición de su creador por los humoristas británicos [Monty Python](https://es.wikipedia.org/wiki/Monty_Python).

Van Rossum es el principal autor de Python, y su continuo rol central en decidir la dirección de Python es reconocido, refiriéndose a él como [Benevolente Dictador Vitalicio](https://es.wikipedia.org/wiki/Benevolent_Dictator_for_Life) (en inglés: Benevolent Dictator for Life, BDFL); sin embargo el 12 de julio de 2018 declinó de dicha situación de honor sin dejar un sucesor o sucesora.

**Ejercicio 3)**

• Claridad, sencillez y unidad (legibilidad): La sintaxis del lenguaje afecta la facilidad con la que un programa se puede escribir, por a prueba, y más tarde entender y modificar.

• Ortogonalidad: Capacidad para combinar varias características de un lenguaje en todas las combinaciones posibles, de manera que todas ellas tengan significado.

• Naturalidad para la aplicación: La sintaxis del programa debe permitir que la estructura del programa refleje la estructura lógica subyacente.

• Apoyo para la abstracción: Una parte importante de la tarea del programador es proyectar las abstracciones adecuadas para la solución del problema y luego implementar esas abstracciones empleando las capacidades más primitivas que provee el lenguaje de programación mismo. • Facilidad para verificar programas: La sencillez de la estructura semántica y sintáctica ayuda a simplificar la verificación de programas.

• Entorno de programación: Facilita el trabajo con un lenguaje técnicamente débil en comparación con un lenguaje más fuerte con poco apoyo externo.

• Portabilidad de programas.

• Costo de uso:

– Costo de ejecución del programa.

– Costo de traducción de programas.

– Costo de creación, prueba y uso de programas.

– Costo de mantenimiento de los programas. ,→ Costo total del ciclo de vida.