07/04/2025

Thiago Ruiz y Gonzalo Ruiz

¿Qué es Python y en qué se diferencia de otros lenguajes de programación de alto nivel?

* 1. **Python es un lenguaje de programación que nos sirve para el aprendizaje y desarrollo ya sea para páginas web, aplicaciones, videojuegos, etc..**

**Se diferencia de los demás en que Python es mas intuitivo y mas accesible a la hora de aprender a programar.**

1. Menciona tres campos de aplicación donde el lenguaje Python tenga especial relevancia actualmente.

**2. En la actualidad el lenguaje Python es esencial a la hora de programar, ya sea una pagina web,un videojuego, o lo que se desee hacer con dicha programación.**

1.2 Características del Lenguaje    
3. Enumera al menos cinco características fundamentales del lenguaje Python.    
4. Explica brevemente la diferencia entre tipado dinámico e implícito en Python.

**3. El mismo Tipado dinámico, el “Interpretado”, la extensa biblioteca que tiene disponible, que es Multiplataforma y que Python tiene una sintaxis clara y fácil de entender**

**4. En Python, el \*\*tipado dinámico\*\* asigna el tipo de variable en tiempo de ejecución, y el \*\*implícito\*\* simplemente lo hace automáticamente sin declarar el tipo.**

**Básicamente el tipado dinamico asigna el tipo de variable, mientras el implícito lo hace automáticamente**

1.3 Instalación de Python    
5. ¿Qué consideraciones deben tenerse en cuenta al instalar Python en un sistema operativo Windows?    
6. ¿Cuál es la utilidad del gestor de paquetes `pip` y cómo se verifica su instalación?

**5. Lo primero sería asegurándonos de que nuestro sistema operativo sea compatible con la versión de Python que deseamos instalar, por ejemplo, Python está disponible para las versiones de Windows 7 en adelante.**

**Lo segundo sería descargar Python desde la página oficial, así evitamos que al descargar por otro medio terminemos descargando otro tipo de archivo.**

**Luego sería escoger la versión, ya sea la de 32bits o 64bits, dependiendo cuanto tiene nuestra computadora.**

**Por último, podríamos verificar si la instalación se hizo correctamente y ver si ya está disponible para el uso.**

**6. El gestor de paquetes PiP nos va a servir para después de la instalación Python, para verificar la instalación hay que ejecutarlo de la siguiente manera:**

**El primer paso sería abrir la terminal (en Windows, puedes usar CMD o PowerShell; en macOS o Linux, abre la terminal).**

**Como segundo paso sería ingresar el siguiente código: pip –version o pip3 –version.**

**Si está instalado te aparecerá pip 23.0.1 from /usr/lib/python3/dist-packages/pip (python 3.10), si no lo está te aparecerá un mensaje que dirá “'pip' no se reconoce como un comando interno o externo...”**

1.4 ¿Por qué Python?    
7. Desde un punto de vista técnico y de productividad, ¿por qué elegir Python sobre otros lenguajes como Java o C++ en proyectos de prototipado rápido?

**Desde un punto de vista técnico y de productividad, elegir Python para proyectos de prototipado rápido es una decisión acertada por su sintaxis sencilla, su flexibilidad y la rapidez con la que permite desarrollar y probar ideas**.

8. ¿Qué es Visual Studio Code y qué ventajas ofrece como entorno de desarrollo para Python?

**8. Visual Studio Code es un editor de código liviano y gratuito que, con la extensión oficial de Python, se convierte en un entorno de desarrollo muy completo. Es rápido, personalizable y perfecto para proyectos en Python, ideal tanto para principiantes como para desarrolladores avanzados.**

9. Menciona al menos tres extensiones recomendadas para trabajar con Python en VS Code.

**9. Python (de Microsoft)**

**Imprescindible: ofrece autocompletado, linting, debugging, ejecución de scripts y más.**

**Pylance: Mejora el rendimiento de IntelliSense (autocompletado) y proporciona análisis de código más rápido y preciso.**

**Jupyter:Permite abrir, editar y ejecutar notebooks .ipynb directamente desde VS Code, ideal para ciencia de datos y pruebas rápidas.**

1.6 Integración de Python con VS Code

10. Describe el proceso de configuración del intérprete de Python en Visual Studio Code.    
11. ¿Cómo se activa el entorno virtual desde VS Code?

**10.**

**Instala la extensión de Python desde la pestaña de extensiones (Ctrl+Shift+X).**

**Abre tu proyecto o archivo .py.**

**Abre la paleta de comandos (Ctrl+Shift+P) y selecciona:**

**👉 Python: Select Interpreter.**

**Elige el intérprete que prefieras (global o entorno virtual).**

**Verifica en la esquina inferior izquierda o con python --version en la terminal.**

Si abrís la carpeta del proyecto y luego seleccionás el entorno con  
Python: Select Interpreter, VS Code puede **activar automáticamente** el entorno virtual cada vez que abras la terminal.

**11.**

**Si abrís la carpeta del proyecto y luego seleccionás el entorno con Python: Select Interpreter, VS Code puede activar automáticamente el entorno virtual cada vez que abras la terminal.**

**Cuando esté activado, verás el nombre de la siguiente manera:**

**(nombre\_entorno) C:\ruta\del\proyecto>**

1.7 Ejecutar código Python    
12. Explica dos formas distintas de ejecutar un script de Python en un sistema operativo.

12.

**12. \*\*Desde la terminal\*\*: Abre la terminal, navega al script con `cd`, y ejecuta:**

**- Windows: python nombre\_script.py**

**- macOS/Linux: python3 nombre\_script.py**

**Desde un editor (VS Code). Abre el script y presiona `F5` o haz clic en "Run" para ejecutarlo directamente desde el editor.**

1.8 Ejecutar código Python en VS Code    
13. ¿Cuál es la diferencia entre ejecutar código Python desde la terminal integrada de VS Code y utilizar el botón "Run" de la interfaz?    
14. ¿Qué ventajas ofrece el uso del "Debug Mode" en VS Code para el desarrollo en Python?

**13. Terminal integrada: Control total, ideal para comandos personalizados y entornos específicos.**

**Botón "Run": Ejecución rápida, cómoda y configurada automáticamente para depuración.**

**14. El "Debug Mode" en VS Code permite depurar el código paso a paso, inspeccionar variables en tiempo real, controlar el flujo de ejecución y detectar errores rápidamente. Esto mejora la productividad y facilita la corrección de fallos en el código de manera eficiente.**

1.9 ¿Por qué VS Code?    
15. Desde tu experiencia o perspectiva técnica, ¿qué beneficios específicos tiene VS Code sobre otros IDEs como PyCharm o Jupyter para desarrollos educativos o introductorios?

**15. Desde una perspectiva técnica y educativa, VS Code ofrece varios beneficios específicos que lo hacen una excelente opción para desarrollos educativos o introductorios en Python, en comparación con otros IDEs como PyCharm o Jupyter. Como por ejemplo Ligereza, velocidad, Facilidad de configuración y personalización**

2.1 ¿Qué es un algoritmo?

**2.1. un algoritmo es un conjunto de pasos o instrucciones ordenadas y finitas que se siguen para resolver un problema o realizar una tarea específica.**

16. **Define qué es un algoritmo y describe sus principales propiedades.**

**16. Un algoritmo es un conjunto de instrucciones ordenadas y finitas que se siguen para resolver un problema o realizar una tarea específica. Sus principales propiedades son: claridad (cada paso debe ser preciso y comprensible), finitud (debe terminar después de un número determinado de pasos), determinismo (produce siempre el mismo resultado para las mismas entradas), entrada (requiere datos de entrada para ejecutarse), salida (genera un resultado al final) y eficiencia (debe resolver el problema de manera óptima en términos de tiempo y recursos**).

17. ¿Por qué es importante que un algoritmo tenga una lógica secuencial clara y no ambigua?

**17. una lógica secuencial clara y no ambigua en un algoritmo es crucial para asegurar su correcta ejecución, predictibilidad de resultados, facilidad de comprensión y mantenimiento, optimización del rendimiento y para evitar errores.**

18. Explica la utilidad del pseudocódigo en la planificación de algoritmos.

**18. El pseudocódigo es una herramienta crucial en la planificación de algoritmos, ya que permite estructurar y comunicar la lógica del algoritmo de manera clara y comprensible antes de implementarlo en un lenguaje de programación**

19. ¿Cuál es la ventaja de utilizar diagramas de flujo frente a pseudocódigo en la enseñanza inicial de la lógica algorítmica?

**19. La principal ventaja de los diagramas de flujo frente al pseudocódigo en la enseñanza inicial de la lógica algorítmica es su capacidad para visualizar claramente el flujo de control, lo que facilita la comprensión de conceptos como decisiones, bucles y secuencias. Son más accesibles, fáciles de entender y permiten a los estudiantes novatos pensar de manera más lógica y estructurada**

20. ¿Qué diferencias hay entre datos numéricos y alfanuméricos en Python?

**20. Los datos numéricos en Python son utilizados para realizar operaciones matemáticas y se representan con tipos como int y float. Por otro lado, los datos alfanuméricos se usan para manejar texto o cadenas de caracteres y se representan con el tipo str. La diferencia principal es que los datos numéricos permiten cálculos, mientras que los alfanuméricos se manipulan como texto**.

21. Proporciona un ejemplo de dato lógico en Python y explica su utilidad en una estructura condicional.

**21.** En **Python**, los **datos lógicos** (también llamados **booleanos**) representan dos posibles valores: **True** (verdadero) y **False** (falso). Estos valores son fundamentales para el control de flujo en los programas, especialmente dentro de **estructuras condicionales** como **if**, **elif** y **else**.

### ****Ejemplo de dato lógico en Python****:

# Ejemplo de una variable booleana

es\_estudiante = True

# Uso del dato lógico en una estructura condicional

if es\_estudiante:

print("Tienes descuento en la matrícula.")

else:

print("No tienes descuento en la matrícula.")

**Explicación:** Un **dato lógico** en Python (como True o False) se usa **comúnmente en estructuras condicionales para decidir qué bloque de código ejecutar, lo cual es esencial para el control de flujo en los programas**.

22. ¿Qué es un identificador válido en Python? Proporciona tres ejemplos y tres no válidos.

**22.** **En \*\*Python\*\*, un \*\*identificador\*\* es el nombre que se le da a las variables, funciones, clases, módulos, etc. Un identificador válido sigue ciertas reglas que permiten al intérprete de Python reconocerlo correctamente.**

**### \*\*Reglas para un identificador válido en Python\*\*:**

**1. Debe comenzar con una \*\*letra (a-z, A-Z)\*\* o un \*\*guion bajo (\_)\*\***

**2. Los siguientes caracteres pueden ser \*\*letras\*\*, \*\*números (0-9)\*\* o \*\*guiones bajos\*\*.**

**3. No puede ser una \*\*palabra reservada\*\* de Python (como `if`, `else`, `class`, etc.).**

**4. Los identificadores son \*\*sensibles a mayúsculas y minúsculas\*\* (`variable` y `Variable` son diferentes).**

**### \*\*Ejemplos de identificadores válidos\*\*:**

**1. `nombre`**

**2. `\_edad`**

**3. `numero\_1`**

**### \*\*Ejemplos de identificadores no válidos\*\*:**

**1. `2nombre`**

**- \*\*No válido\*\* porque comienza con un número.**

**2. `mi variable`**

**- \*\*No válido\*\* porque contiene un espacio, lo cual no es permitido.**

**3. `if`**

**- \*\*No válido\*\* porque \*\*`if`\*\* es una palabra reservada de Python.**

**### \*\*Resumen\*\*:**

**Un identificador válido en Python debe comenzar con una letra o un guion bajo, seguido de letras, números o guiones bajos. Además, no puede ser una palabra reservada. Los identificadores no válidos incluyen aquellos que empiezan con un número, contienen espacios o son palabras reservadas.**

**2**3. ¿Cuál es la diferencia entre una variable y una constante en el contexto de Python, considerando que el lenguaje no tiene constantes formales?

**23. e**n **Python**, la principal diferencia entre una **variable** y una **constante** radica en el **uso y el concepto** que tenemos de cada uno, ya que el lenguaje **no tiene constantes formales** como otros lenguajes de programación (por ejemplo, const en C o final en Java).

24. Escribe un ejemplo de cada una de las siguientes operaciones en Python:    
   a) Aritmética    
   b) Relacional    
   c) Lógica

* **24. Aritmética**: Operaciones como +, -, \*, /, %, \*\* para realizar cálculos.
* **Relacional**: Operaciones como ==, !=, >, <, >=, <= para comparar valores.
* **Lógica**: Operaciones como and, or, not para combinar valores booleanos.

25. ¿Qué resultado se obtiene de la siguiente expresión en Python?:

python

(10 > 5) and (3 == 3) or not (4 < 2)

25. La expresión completa **(10 > 5) and (3 == 3) or not (4 < 2)** evalúa a **True**.

**Resumen:**

El resultado de la expresión es **True**.