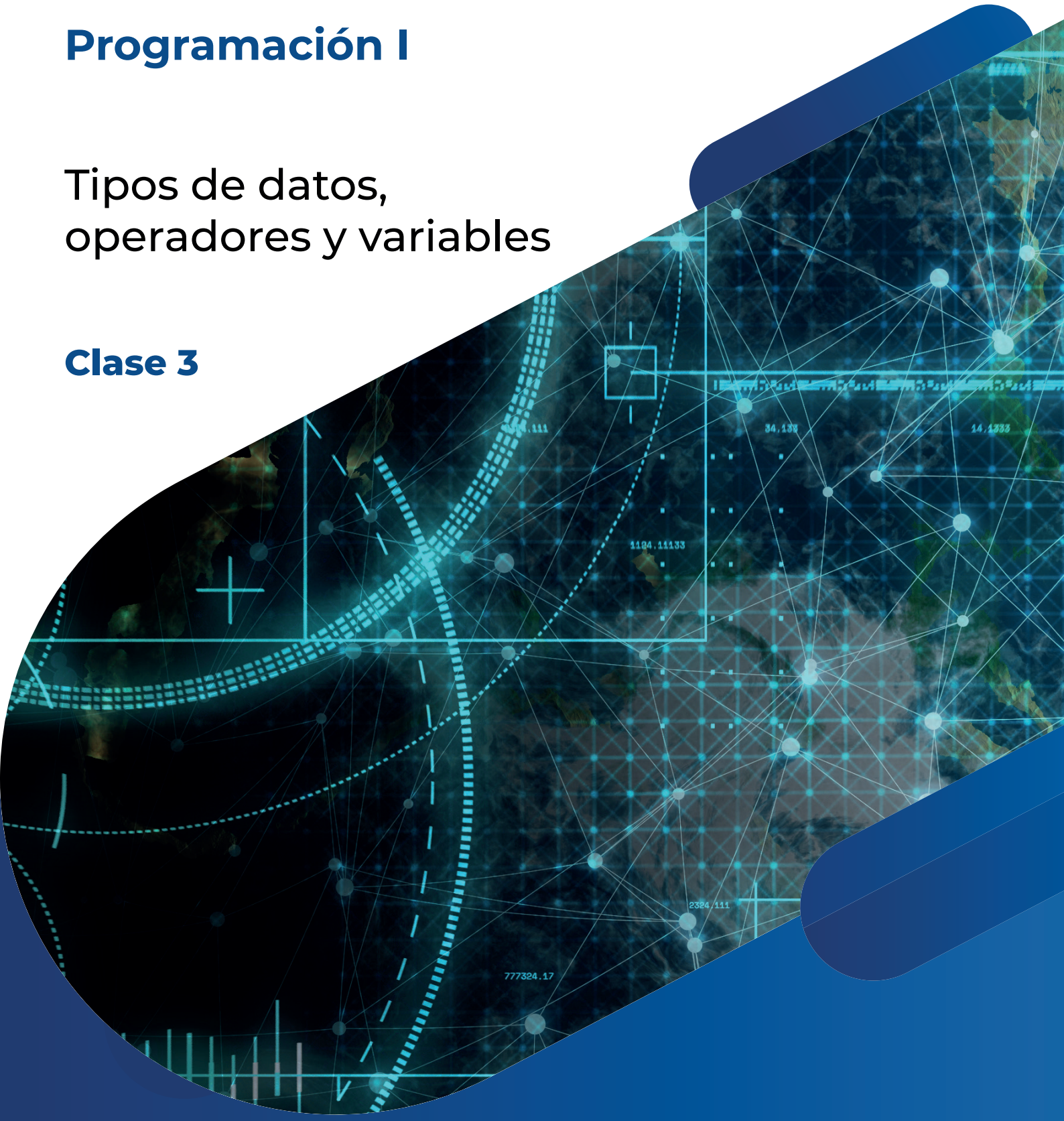
Programación I

Tipos de datos, operadores y variables

Clase 3



Ingeniería en ciberseguridad

La excelencia no se improvisa

1. INTRODUCCIÓN DE LA CLASE

Hoy vamos a adentrarnos en los fundamentos de Python, un lenguaje de programación que se caracteriza por su sintaxis clara y sencilla. Imagina que, al aprender a programar, lo primero es conocer los "tipos" básicos que nos permiten trabajar con datos: números, textos, y valores lógicos. Esto es fundamental, ya que cada dato se comporta de manera distinta y Python nos ofrece herramientas muy potentes para manipularlos. Así es, y no solo veremos cómo se definen estos datos, sino también cómo declarar variables siguiendo convenciones que facilitan la legibilidad y el mantenimiento del código. Además, exploraremos los operadores que nos permiten realizar operaciones aritméticas, lógicas y de comparación, esenciales para el control del flujo de un programa. La claridad en la definición de estos elementos es el primer paso para desarrollar programas robustos y eficientes.

El profesor enfatiza la importancia de dominar estos conceptos, explicando que Python será el lenguaje base para todos los ejemplos y ejercicios del curso. La familiaridad con los tipos de datos, operadores y la correcta declaración de variables facilitará la comprensión de temas más complejos en el futuro y permitirá a los estudiantes aplicar buenas prácticas en el desarrollo de software. Dominar estos fundamentos es esencial para cualquier programador.

Clase 3: Tipos de datos, operadores y variables (Parte 1)

Resultado o resultados de aprendizaje que será abordado con el contenido de la clase.

Asociar los conceptos fundamentales de la programación: sintaxis, manejo de tipos de datos, variables, entrada/salida de datos y estructuras de control básicas, en un lenguaje estructurado.

Reto # 1

Contenido de la Clase:

3. **Tipos de datos, operadores y variables (Parte 1)**

En esta sección profundizaremos de manera detallada en aspectos fundamentales para el dominio inicial de Python, un lenguaje ampliamente utilizado en múltiples ámbitos debido a su flexibilidad y claridad sintáctica. Nos centraremos en comprender a fondo los tipos de datos primitivos, las convenciones para nombrar variables, y los operadores básicos que Python utiliza para manipular información, conceptos esenciales para desarrollar algoritmos eficientes y claros.

Iniciaremos explorando los tipos de datos más básicos, conocidos como datos primitivos, los cuales constituyen la piedra angular para manejar información en cualquier lenguaje de programación. Entre estos destacan

los enteros (int), los números con decimales (float), los valores booleanos (bool) y las cadenas de texto (str). Cada uno de estos tipos de datos posee características particulares y es esencial comprender sus propiedades y limitaciones para utilizarlos de forma efectiva en programas estructurados. Adicionalmente, se detallarán las reglas prácticas para declarar variables en Python, así como las convenciones recomendadas para nombrarlas adecuadamente, lo que promueve una mejor legibilidad y mantenimiento del código especialmente cuando se trabaja en equipos colaborativos.

Adicionalmente, abordaremos el uso y funcionalidad de los operadores básicos en Python, herramientas fundamentales que permiten la manipulación eficiente y precisa de datos en diversas situaciones. Los operadores pueden clasificarse principalmente en tres grandes grupos: operadores aritméticos, operadores relacionales y operadores lógicos. Cada grupo de operadores cumple funciones específicas dentro del desarrollo de programas, facilitando desde cálculos matemáticos simples hasta la evaluación de condiciones más complejas para controlar el flujo lógico de la ejecución.

Por ejemplo, mediante los operadores aritméticos (+, -, *, /, %), podemos realizar desde cálculos básicos, como sumar dos valores numéricos, hasta complejas operaciones que impliquen la combinación y manipulación de múltiples datos numéricos. Por su parte, los operadores relacionales como mayor que (>), menor que (<), igual a (==), entre otros, son herramientas esenciales para determinar relaciones entre valores y tomar decisiones de forma precisa dentro de los programas. Finalmente, los operadores lógicos and, or y not permiten construir expresiones compuestas capaces de evaluar múltiples condiciones, otorgando una enorme flexibilidad al controlar el flujo del programa.

3.1. Tipos de datos primitivos: int, float, bool, str.

3.1.1. Introducción a Python como Lenguaje de Programación

Python es un lenguaje interpretado, considerado de alto nivel debido a su proximidad al lenguaje humano. Se caracteriza por permitir diversos enfoques de programación, conocidos como paradigmas. Este lenguaje ha ganado popularidad notable en la última década, principalmente debido a su facilidad de aprendizaje, su código limpio y legible, y su gran versatilidad en proyectos complejos. La simplicidad en la sintaxis es uno de sus rasgos distintivos, permitiendo a los desarrolladores escribir soluciones de manera ágil y eficiente, lo que facilita enormemente la curva de aprendizaje en comparación con otros lenguajes más rígidos y sintácticamente complejos.

Creado inicialmente por Guido van Rossum hacia finales de la década de 1980 como un proyecto personal para mejorar la claridad del código en comparación con otros lenguajes populares en aquel entonces, Python fue lanzado oficialmente en el año 1991. Desde su publicación, ha experimentado múltiples actualizaciones que han incorporado mejoras significativas, convirtiéndose hoy en día en una herramienta indispensable para distintos campos profesionales. Actualmente, Python no solo destaca en áreas tradicionales del desarrollo como la creación de sitios web dinámicos, aplicaciones de escritorio o scripts de automatización, sino que se ha convertido en el lenguaje preferido para la ciencia de datos, el aprendizaje automático, la inteligencia artificial, y múltiples áreas tecnológicas emergentes debido a su flexibilidad y potencia.

Entre las características fundamentales que distinguen a Python están:

- Sintaxis Sencilla y Clara: La estructura del lenguaje está diseñada para ser intuitiva y cercana al idioma inglés, lo cual facilita que personas con poca experiencia previa en programación puedan entender rápidamente el código. Esta claridad disminuye notablemente la probabilidad de errores, facilitando además las tareas de mantenimiento y modificación del código a largo plazo.
- Tipado Dinámico: A diferencia de lenguajes con tipado estático, en Python no es obligatorio definir explícitamente el tipo de una variable antes de su uso. El intérprete reconoce automáticamente el tipo en el momento de la ejecución del programa. Esta característica reduce el tiempo necesario para

escribir código, otorgando flexibilidad al desarrollador, especialmente durante las primeras fases del desarrollo de una aplicación.

- Amplio Ecosistema y Bibliotecas Robustas: Una de las ventajas más atractivas de Python es su biblioteca estándar extensa, acompañada por miles de paquetes externos disponibles de forma gratuita y mantenidos por la comunidad. Herramientas populares como NumPy y Pandas para análisis y manipulación de datos, Django y Flask para desarrollo web, o TensorFlow y PyTorch para inteligencia artificial y aprendizaje automático, demuestran cómo Python cubre con solvencia prácticamente cualquier necesidad tecnológica actual.
- Comunidad Activa y Colaborativa: Python cuenta con una de las comunidades más grandes y colaborativas dentro del mundo de la programación. Esto implica una gran cantidad de recursos disponibles para aprendizaje, documentación bien estructurada, cursos gratuitos en múltiples plataformas educativas, y foros activos donde los desarrolladores intercambian soluciones a problemas frecuentes y se ayudan mutuamente. Esta comunidad vigorosa es esencial para que el lenguaje se mantenga en constante evolución, integrando mejoras continuas y adaptándose a las necesidades cambiantes de sus usuarios.

Además, para maximizar la productividad al utilizar Python, es recomendable trabajar en un entorno de desarrollo integrado (IDE). Estos entornos proporcionan herramientas que facilitan enormemente la programación, tales como el autocompletado de código, funciones de depuración avanzadas que permiten identificar y resolver errores con facilidad, gestión organizada de proyectos grandes con múltiples archivos, y una interfaz visual cómoda para la escritura del código. Entre los IDE más populares se encuentran Visual Studio Code y PyCharm. Ambos son ampliamente usados en la industria y por desarrolladores profesionales debido a su robustez y a las facilidades que ofrecen para mantener el orden y la estructura del código, lo que repercute directamente en una mejora considerable en la eficiencia y calidad del software desarrollado.

Adicionalmente, Python es un lenguaje multiplataforma. Esto quiere decir que un mismo programa puede ejecutarse sin mayores modificaciones en diversos sistemas operativos como Windows, macOS y distribuciones de Linux, haciendo que las aplicaciones desarrolladas con Python sean altamente portables y accesibles a un público amplio. Esta característica fortalece la adopción del lenguaje en empresas que requieren soluciones tecnológicas flexibles y compatibles con diversas plataformas de trabajo.

Un entorno de desarrollo adecuado potencia la eficiencia y la calidad del código.

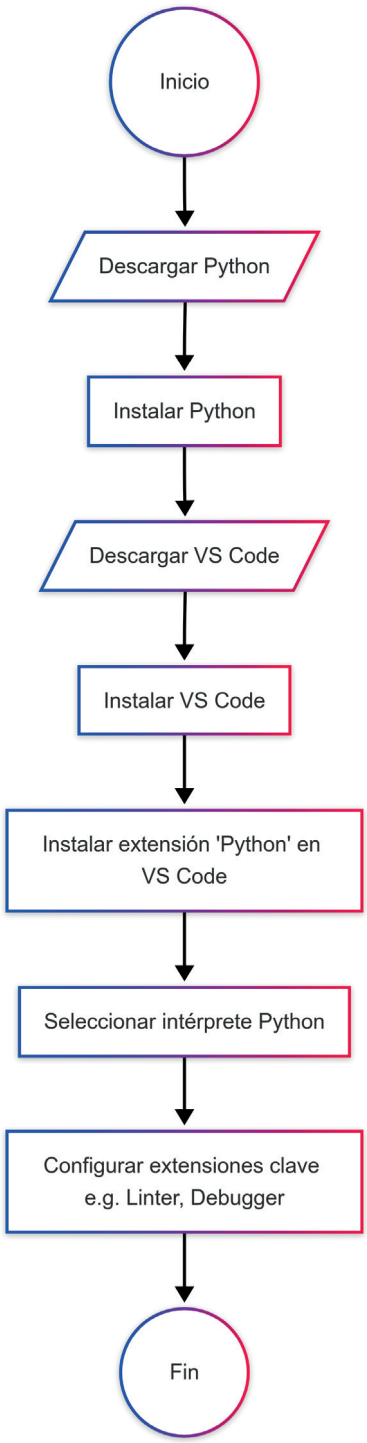


Imagen 1: Diagrama que ilustra la Instalación y Configuración de Python
Damián Nicolalde Rodríguez. (2024). Instalación y Configuración de Python en VS Code

Para complementar y profundizar, se sugieren acceder a este recurso:

- **Título:** Instalación y Configuración de Python en Visual Studio Code
- **Descripción:** Este tutorial detalla el proceso de instalación y configuración de Python en Visual Studio Code (VS Code), incluyendo la selección del intérprete, la integración de extensiones clave (como Pylint o Python Extension Pack) y la personalización del entorno para desarrollo eficiente. Se explican pasos esenciales como la creación de entornos virtuales, la depuración de código y la optimización de

herramientas de autocompletado, dirigido a desarrolladores que buscan establecer un flujo de trabajo robusto en VS Code.

- **URL:** 1.1 Instalación y Configuración de Python en Visual Studio Code | Programar desde cero en Python

3.2. Tipos de datos primitivos: int, float, bool, str.

Los tipos de datos primitivos son la base para la manipulación de la información y el control de los procesos en un programa. Estos tipos se utilizan para representar valores y realizar operaciones matemáticas y lógicas.

En Python, los datos primitivos incluyen principalmente:

- **int:** Números enteros.
- **float:** Números con parte decimal.
- **bool:** Valores lógicos (True o False).
- **str:** Cadenas de texto.

3.2.1. int (entero)

El tipo int se utiliza para representar números enteros, es decir, aquellos sin parte decimal. Estos números se emplean en operaciones aritméticas básicas, conteos y estructuras de iteración. Por ejemplo, al definir la variable edad = 25, se está asignando un valor entero.

Ejemplo en Python:

```
edad = 25
cantidad_de_items = 100
```

Aquí, edad y cantidad_de_items son variables de tipo entero que pueden usarse en cálculos, comparaciones y ciclos. El manejo de números enteros es esencial para cálculos precisos y control de flujos mediante bucles y condicionales.

3.2.2. float (Flotantes)

El tipo float representa números con parte decimal, lo que es fundamental en aplicaciones que requieren precisión en cálculos, como mediciones o finanzas. Los números flotantes permiten trabajar con decimales y son imprescindibles para operaciones matemáticas complejas.

Ejemplo en Python:

```
precio = 19.99
distancia = 3.75
```

Estos valores se usan cuando se requiere un mayor grado de precisión en cálculos matemáticos.

3.2.3. **bool (Booleanos)**

El tipo bool es utilizado para almacenar valores lógicos, que pueden ser True o False. Son fundamentales en la toma de decisiones dentro del flujo de un programa, ya que permiten evaluar condiciones y determinar la ejecución de bloques de código.

Ejemplo en Python:

```
es_aprobado = True
tiene_permiso = False
```

El uso de valores booleanos es crucial para la toma de decisiones en el flujo del programa.

3.2.4. **str (Cadenas de Texto)**

El tipo str se utiliza para representar secuencias de caracteres. Es la base para trabajar con textos, nombres, mensajes y cualquier dato alfanumérico. Las cadenas permiten almacenar y manipular información textual de manera eficiente.

Ejemplo en Python:

```
nombre = “Juan”
```

```
mensaje = “Bienvenido al curso de Programación 1”
```

La manipulación de cadenas es vital en aplicaciones que requieren la interacción con el usuario o el procesamiento de texto.

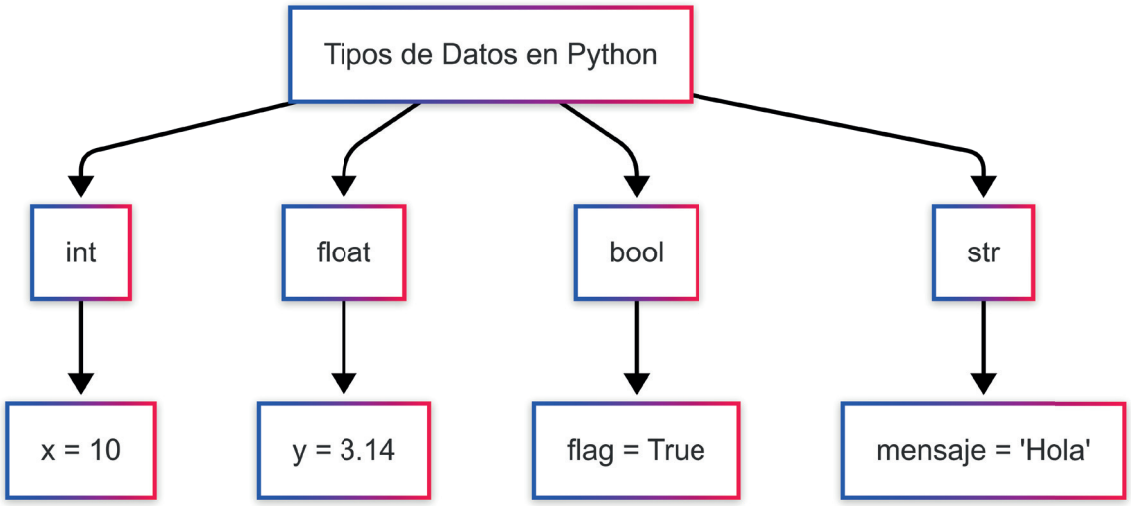


Imagen 2: Ejemplo Visual de Tipos de Datos en Python

Damián Nicolalde Rodríguez. (2024). Ejemplo Visual de Tipos de Datos en Python.

Para complementar y profundizar, se sugieren acceder a este recurso:

- **Título:** APRENDE TIPOS, VARIABLES y OPERADORES en PYTHON como NUNCA TE LO HAN EXPLICADO
- **Descripción:** Este video ofrece una introducción a Python, abarcando la declaración de variables, tipos de datos y uso de operadores.
- **URL:** APRENDE TIPOS, VARIABLES y OPERADORES en PYTHON como NUNCA TE LO HAN EXPLICADO

3.2.5. **Operadores**

Los operadores son símbolos que realizan operaciones sobre los datos. Python los clasifica en varias categorías que permiten llevar a cabo cálculos, comparaciones y evaluaciones lógicas.

3.2.5.1. **Operadores Aritméticos**

Estos operadores realizan operaciones matemáticas básicas, tales como suma, resta, multiplicación, división, módulo y exponenciación. Son esenciales para realizar cálculos y transformar datos numéricos.

Ejemplos:

- Suma: +
- Resta: -
- Multiplicación: *
- División: /
- Módulo (resto): %
- Exponenciación: **

Ejemplo en Python:

```
suma = 10 + 5 # 15

producto = 10 * 5 # 50

resta = 10 - 5 # 5

division = 10 / 5 # 2.0

modulo = 10 % 3 # 1


potencia = 2 ** 3 # 8
```

Cada uno de estos operadores permite manipular números de manera efectiva y es fundamental en el desarrollo de algoritmos.

3.2.5.2. **Operadores de Comparación**

Los operadores de comparación se utilizan para comparar dos valores y devuelven un resultado booleano. Permiten establecer relaciones entre variables, lo que es clave para la toma de decisiones en el código.

Ejemplos:

- Igual a: ==
- No igual a: !=
- Mayor que: >
- Menor que: <
- Mayor o igual que: >=
- Menor o igual que: <=

Ejemplo en Python:

```
es_igual = (10 == 10) # True
no_es_igual = (10 != 5) # True
mayor = (10 > 5) # True
menor = (5 < 10) # True
```

Estos operadores son esenciales para construir condiciones en estructuras de control, permitiendo decisiones precisas en la lógica del programa.

Tabla 1: Resumen Operadores en Python

Categoría	**Ejemplos de Operadores**	**Uso Principal**
Aritméticos	+, -, *, /, %, **	Realizar operaciones matemáticas.
Comparación	==, !=, >, <, >=, <=	Comparar valores y devolver un valor booleano.
Lógicos	and, or, not	Combinar condiciones y formar expresiones lógicas.

Damián Nicolalde Rodríguez. (2024)

3.2.5.3. **Operadores Lógicos**

Los operadores lógicos combinan expresiones y permiten la formación de condiciones complejas. Los principales son:

- and: Devuelve True si ambas condiciones son verdaderas.
- or: Devuelve True si al menos una condición es verdadera.
- not: Invierte el valor lógico.

Ejemplo en Python:

```
resultado = (10 > 5) and (5 < 3)  # Resultado: False
condicion = not (10 == 5) # Resultado: True
```

El uso de operadores lógicos es crucial para la evaluación de condiciones complejas y el control del flujo de ejecución.

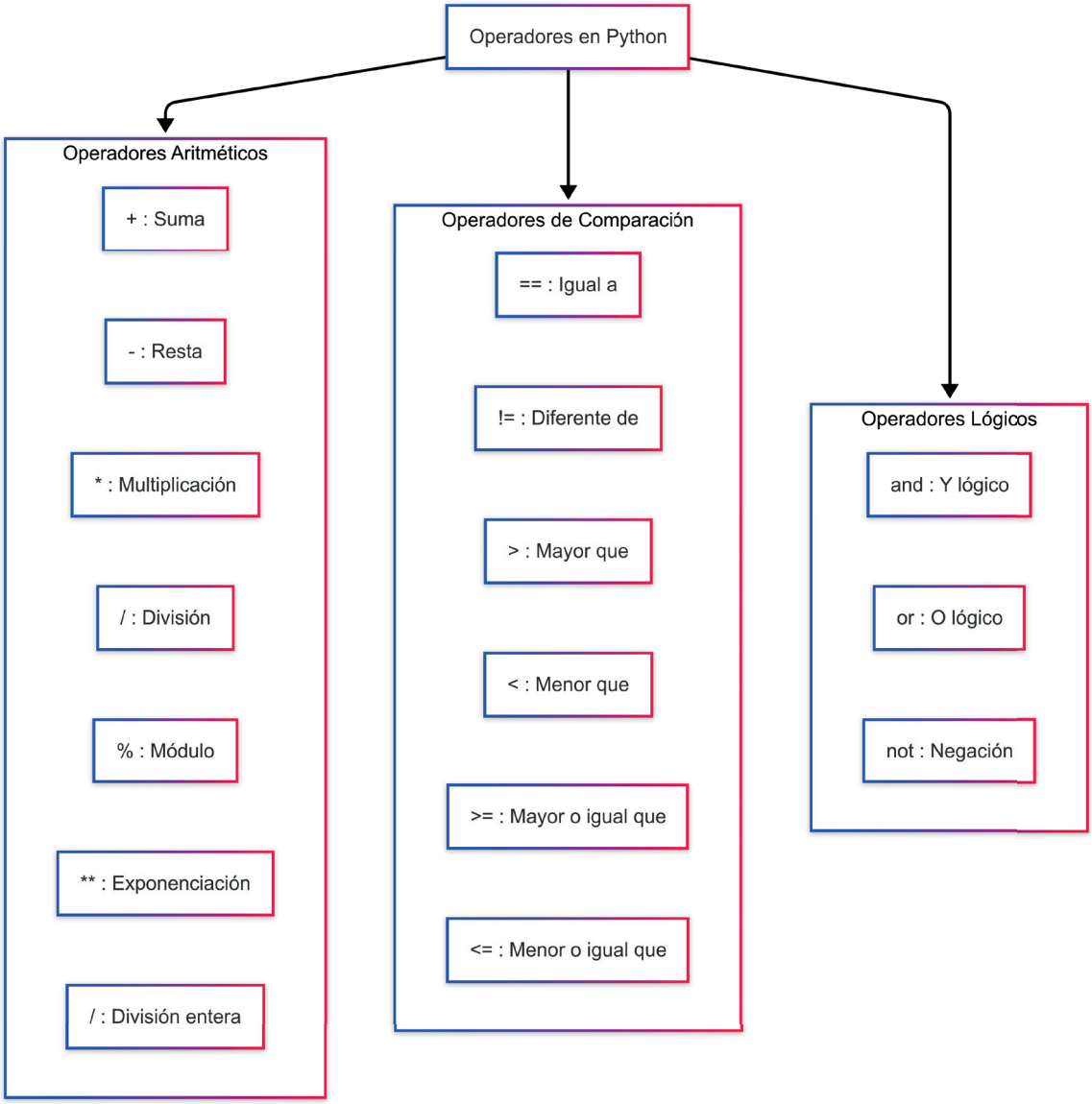


Imagen 3: Diagrama de Operadores en Python

Descripción: clasificación y ejemplificación de los operadores aritméticos, de comparación y lógicos en Python.

Damián Nicolalde Rodríguez. (2024). clasificación y ejemplificación de los operadores.

3.3. **Declaración de variables y convenciones de nombres.**

a declaración de variables en Python es sencilla debido al tipado dinámico, lo que significa que el tipo de una variable se determina automáticamente en el momento de la asignación. Sin embargo, seguir convenciones de nombres es esencial para mantener la claridad y la legibilidad del código.

3.3.1. **Declaración de Variables**

Para crear una variable, se asigna un valor directamente. La simplicidad de Python permite definir variables en una sola línea, lo que reduce la complejidad del código y agiliza la programación.

Ejemplo en Python:

```
nombre = “Carlos”
edad = 30
precio = 15.75
activo = True
```

Cada variable se crea de manera simple y sin la necesidad de declarar su tipo explícitamente.

3.3.2. **Convenciones de Nombres**

Las convenciones de nombres son pautas que aseguran que los identificadores sean claros y consistentes. En Python, se recomienda usar el estilo snake_case, en el que se escriben las palabras en minúsculas y se separan con guiones bajos.

Ejemplos:

- Correcto: numero_de_clientes
- Incorrecto: numClientes o NumeroDeClientes

Estas convenciones facilitan la lectura del código y su mantenimiento en equipos de trabajo, ya que se genera un estándar que todos los miembros pueden seguir.

Tabla 2: Ejemplos de Declaración de Variables y Convenciones de Nombres

Variable	**Valor**	**Convención Correcta**
nombre_usuario	“María”	Uso de minúsculas y guiones bajos (snake_case).
edad_usuario	28	Nombre descriptivo en minúsculas.
precio_producto	12.50	Uso adecuado de separación de palabras.
es_activo	True	Nombre claro que indica un valor booleano.

Damián Nicolalde Rodríguez. (2024).

3.3.3. Integración de Conceptos en Ejemplos Prácticos

Para consolidar el conocimiento, es fundamental aplicar los conceptos en ejemplos prácticos que integren la declaración de variables, el uso de operadores y estructuras de control. Un ejercicio práctico refuerza la teoría y muestra cómo los fundamentos se traducen en código funcional.

Ejercicio Propuesto

Elabore un pequeño programa en Python que realice lo siguiente:

- Solicite al usuario su nombre, edad y salario.
- Determine si el usuario es mayor de edad y si su salario supera un valor umbral (por ejemplo, 1000).
- Imprima un mensaje que combine ambos resultados utilizando operadores aritméticos, de comparación y lógicos.

Ejemplo de Código:

```
# Solicitar datos al usuario
nombre_usuario = input("Ingrese su nombre: ")
edad_usuario = int(input("Ingrese su edad: "))
salario_usuario = float(input("Ingrese su salario: "))

# Evaluar condiciones
es_mayor = edad_usuario >= 18
salario_alto = salario_usuario > 1000

# Imprimir resultados
if es_mayor and salario_alto:
    print(f"{nombre_usuario}, usted es mayor de edad y tiene un salario alto.")
elif es_mayor and not salario_alto:
    print(f"{nombre_usuario}, usted es mayor de edad pero su salario es menor o igual a 1000.")
else:
    print(f"{nombre_usuario}, usted no es mayor de edad.")
```

```
# Fin del programa
```

Este ejercicio combina la declaración de variables, el uso de diferentes tipos de operadores y la aplicación de estructuras condicionales, consolidando el aprendizaje de conceptos fundamentales en Python.

```python
# Ejemplo de uso de variables, operadores y estructuras de control

# Declaración de variables
numero = 10           # int
pi = 3.1416           # float
saludo = "Hola Mundo" # str
activo = True         # bool

# Operadores aritméticos
suma = numero + 5
producto = numero * 2

# Estructura de control: condicional if-else
if numero > 5:
    print("El número es mayor que 5")
else:
    print("El número es menor o igual a 5")

# Estructura de control: bucle for
for i in range(3):
    print("Iteración", i)
```

Imagen 4: Captura de Pantalla de un Código en Python

Damián Nicolalde Rodríguez. (2024). Captura de Pantalla de un Código en Python.

El dominio de los tipos de datos primitivos, operadores y la correcta declaración de variables en Python es crucial para desarrollar programas robustos y eficientes. Estos elementos básicos constituyen el andamiaje sobre el cual se construyen algoritmos complejos y se implementan soluciones de software.

Python, con su sintaxis sencilla y tipado dinámico, permite a los programadores concentrarse en la lógica del problema sin verse agobiados por detalles sintácticos, lo que acelera el proceso de desarrollo y minimiza errores. La declaración de variables y las convenciones de nombres no solo mejoran la legibilidad del código, sino que también promueven la consistencia en equipos de trabajo, facilitando la colaboración y el mantenimiento del software.

Las herramientas de planificación, como el pseudocódigo y los diagramas de flujo, son complementarias y esenciales para estructurar la lógica antes de codificar. El pseudocódigo ayuda a detallar cada paso de un algoritmo de forma textual, mientras que los diagramas de flujo proporcionan una visión gráfica que facilita la identificación de errores y la comunicación entre los miembros del equipo. Esta combinación de metodologías

fortalece la capacidad de los desarrolladores para diseñar soluciones efectivas y colaborativas.
La planificación y la estructura clara del código son la base de un software robusto y mantenible.

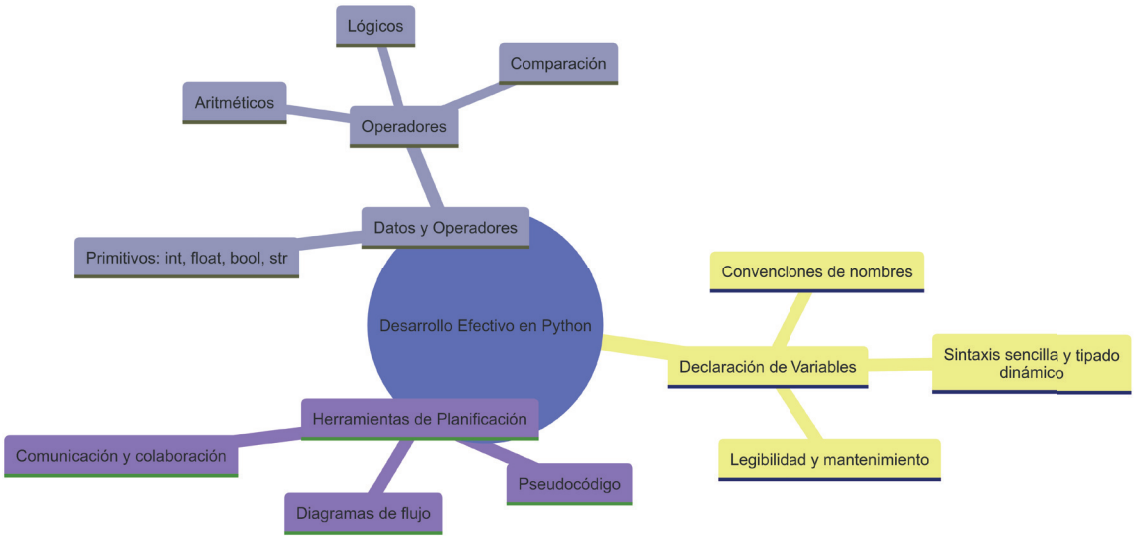


Imagen 5: Infografía: Desarrollo Efectivo en Python

Damián Nicolalde Rodríguez. (2024). Infografía: Desarrollo Efectivo en Python.

Referencias citadas en la Clase 3.

- https://elibro.puce.elogim.com/es/ereader/puce/230298
- https://puce.odilo.us/info/facil-aprendizaje-estructuras-de-datos-algoritmos-c-aprenda-facilmente-estructuras-de-datos-graficamente-03127232
- https://puce.odilo.us/info/aprende-c-en-un-fin-de-semana-03105596

Definición de los términos citados en la Clase 3.

Variables: Son identificadores que actúan como contenedores para almacenar datos en la memoria del sistema. En Python, las variables se asignan mediante el operador "=", y su tipo se determina dinámicamente en tiempo de ejecución. Permiten guardar, modificar y manipular información durante la ejecución del programa.

Convenciones de nombres: Son un conjunto de reglas y pautas que determinan cómo deben nombrarse las variables, funciones, clases y otros elementos en el código. En Python, se recomienda utilizar el estilo *snake_case*—es decir, utilizar minúsculas y separar las palabras con guiones bajos—para mejorar la legibilidad y mantener la coherencia del código en entornos colaborativos.

Profundización Clase 3.

Recurso_profundizacion_clase3.docx

PUCE Virtual

La excelencia no se improvisa

síguenos