Profundiza más

PUCE Virtual

Recurso de Profundización

Clase 3: Tipos de datos, operadores y variables (Parte 1)
Autor: Damián Nicolalde Rodríguez

Objetivo del Documento:
El objetivo de estos recursos de profundización es ampliar, reforzar y clarificar los conceptos fundamentales presentados en la clase, proporcionando ejemplos detallados, comparaciones visuales en tablas que faciliten la comprensión integral y la aplicación práctica de los contenidos. Estos recursos ayudan a los estudiantes a visualizar y estructurar la información de manera clara, promoviendo el aprendizaje autónomo y colaborativo, y sentando las bases necesarias para abordar temas más complejos en el desarrollo de software.

Recurso de Profundización 1: Cuadro Comparativo – Tipos de Datos y Operadores en Python

Tabla 1. Cuadro Comparativo – Tipos de Datos y Operadores en Python

Elemento	Descripción y Aplicación	Ejemplo en Python
int (Enteros)	Representa números enteros sin parte decimal. Se utiliza para conteos, iteraciones y operaciones aritméticas básicas. Es la base para cálculos en algoritmos donde no se requiere precisión decimal.	edad = 25 cantidad = 100
float (Flotantes)	Representa números con parte decimal. Es esencial en cálculos que requieren precisión, como en finanzas o mediciones. Permite operaciones más precisas que los enteros en cálculos complejos.	precio = 19.99 distancia = 3.75
bool (Booleanos)	Almacena valores lógicos: True o False. Se utiliza en la evaluación de condiciones y para controlar el flujo de ejecución de un programa, permitiendo decisiones en estructuras condicionales y bucles.	es_valido = True tiene_acceso = False

Profundiza más

str (Cadenas de Texto)	Maneja secuencias de caracteres. Es fundamental para trabajar con datos alfanuméricos, como nombres, mensajes y descripciones, facilitando la interacción con el usuario y la presentación de resultados en forma textual.	nombre = "Ana" mensaje = "Bienvenido al curso de Programación 1"
Operadores aritméticos	Realizan operaciones matemáticas básicas (suma, resta, multiplicación, división, módulo, exponenciación). Son cruciales para procesar datos numéricos y realizar cálculos que forman parte de la lógica de un algoritmo.	suma = 10 + 5 producto = 10 * 5
Operadores de comparación	Permiten comparar valores y devuelven un resultado booleano. Se usan para establecer relaciones (igualdad, desigualdad, mayor, menor, etc.) y son fundamentales para la construcción de condiciones en estructuras de control.	es_igual = (10 == 10) mayor = (10 > 5)
Operadores lógicos	Combinan expresiones y permiten construir condiciones complejas. Incluyen and, or y not, y son esenciales para evaluar múltiples condiciones simultáneamente y controlar el flujo del programa.	resultado = (10 > 5) and (5 < 3) condicion = not (10 == 5)

Damián Nicolalde (2024)

Este cuadro comparativo permite a los estudiantes visualizar de manera clara las características, aplicaciones y ejemplos prácticos de cada tipo de dato y categoría de operador en Python. Con él, se facilita la comprensión de cómo se combinan estos elementos para resolver problemas mediante cálculos y evaluaciones lógicas en algoritmos.

Recurso de Profundización 2: Tabla de Ejemplos – Variables y Convenciones de Nombres en Python

Tabla2. Ejemplos – Buenas Prácticas en la Declaración de Variables en Python

Variable	Ejemplo Incorrecto	Ejemplo Correcto (snake_case)	Explicación

Nombre de usuario	NombreUsuario	nombre_usuario	Uso de mayúsculas y falta de separador; se recomienda minúsculas y guiones bajos.
Número de clientes	numClientes	num_clientes	Uso de CamelCase en lugar de snake_case, lo que dificulta la legibilidad.
Precio total	PrecioTotal	precio_total	El uso de mayúsculas en "PrecioTotal" reduce la uniformidad en el código.
Estado de la cuenta	EstadoDeLaCuenta	estado_de_la_cuenta	La versión incorrecta mezcla mayúsculas; el formato correcto mejora la claridad.

Damián Nicolalde (2024).

Esta tabla ilustra ejemplos prácticos comparando nombres de variables que siguen las convenciones de snake_case con aquellos que no las siguen. Esta tabla resalta la importancia de utilizar nombres descriptivos, consistentes y en minúsculas para mejorar la legibilidad y el mantenimiento del código, lo cual es fundamental en entornos colaborativos.