«Talento Tech»

Iniciación a la Programación con Python

CLASE 3

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 3

Tipos de datos

Temario:

- Conversión entre tipos de datos.
- Operadores algebráicos y expresiones en Python
- Uso de input()
- Programas con entrada, procesamiento y salida de datos

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Conversión entre tipos de datos

La conversión de tipos de datos, también conocida como *casting*, es una operación común en Python y otros lenguajes de programación. Consiste en transformar un valor de un tipo de dato a otro, como de un entero (int) a un flotante (float), o de un número a una cadena de texto (str). Esto es especialmente útil en situaciones donde se requiere manipular los datos de una forma que no es compatible con su tipo original.

Conversión entre números enteros y números decimales

Podés convertir valores entre enteros y decimales para realizar operaciones aritméticas que requieren precisión decimal o simplemente para cambiar la representación de un número. Veamos cómo:

Ejemplo de int a float

```
entero = 10
decimal = float(entero)
print(decimal) # 10.0
```

En este caso **float(entero)** convierte el número entero 10 a un número decimal 10.0.

Ejemplo de float a int

```
decimal = 3.14
entero = int(decimal)
print(entero) # 3
```

En el ejemplo precedente observamos el mismo caso a la inversa: int(decimal) convierte el número decimal 3.14 a un número entero 3, descartando cualquier valor después del punto decimal.

Conversión de números a cadenas de texto y viceversa

La conversión de números a cadenas de texto es útil para concatenar números con texto, mientras que convertir cadenas de texto a números es necesario para realizar operaciones matemáticas con datos que originalmente se ingresaron o se proporcionaron como texto.

Ejemplo de int a str:

```
edad = 25
mensaje = "Tengo " + str(edad) + " años."
print(mensaje) # Tengo 25 años.
```

Como podemos ver, **str(edad)** convierte el número entero 25 a una cadena de texto "25", permitiendo su concatenación con otras cadenas de texto. Si bien pareciera decir lo mismo en ambos casos, recordemos que lo importante es **cómo interpreta Python el valor 25**. Al comenzar edad es un número entero y al finalizar es una cadena de texto.

Ejemplo de str a int:

```
cadena_numerica = "123"
numero = int(cadena_numerica)
print(numero + 7) # 130
```

En este caso **int(cadena_numerica)** convierte la cadena de texto "*123*" a un número entero 123, permitiendo realizar operaciones matemáticas como la adición o suma.

Conversión entre booleanos y otros tipos de datos

Los booleanos también se pueden convertir a otros tipos de datos, y viceversa, dependiendo de las necesidades lógicas del programa. Las expresiones booleanas pueden ser traducidas a **1 (verdadero) y 0 (falso).**

Ejemplo de bool a int:

```
valor_verdadero = True
valor_falso = False
print(int(valor_verdadero)) # 1
print(int(valor_falso))     # 0
```

Aquí, *True* se convierte a **1** y *False* a **0**, lo cual es muy útil en cálculos que dependen de condiciones lógicas.

Ejemplo de int a bool:

```
numero = 0
print(bool(numero))     # False
numero = 5
print(bool(numero))     # True
```

¡Atención! **Cualquier número distinto de 0 se convertirá a *True*** en una conversión a booleano, **mientras que 0 se convierte a *False*.**

La conversión de tipos de datos es una herramienta poderosa en Python que permite manipular la información de manera más flexible. Entender cómo y cuándo usar estas conversiones te ayudará a evitar errores y así lograr perfeccionar la depuración de tu código.

La función type()

Entender cómo verificar el tipo de dato de un objeto en Python es una habilidad fundamental, especialmente útil para depurar programas y asegurar que las operaciones entre datos sean compatibles. **La función type()** es la herramienta que Python proporciona para este propósito.

Uso de la función type()
Esta herramienta de desarrollo retorna el tipo de dato consultado. ¿Cómo funciona? Imaginemos que necesitamos saber cómo está interpretando Python un dato. Para resolverlo le solicitamos que imprima el tipo de dato que es “numero”, es decir que nos devuelve la “clase” (<class>) del **objeto** que le asignamos como *argumento* (entre paréntesis). Se imprimirá **“<class ‘int’>”,** detectando qué tipo de dato es el objeto “numero”: un entero. Esto incluye tipos de datos integrados como enteros, flotantes, cadenas de texto, listas, diccionarios, y más. Veamos algunos otros ejemplos para ilustrar cómo se utiliza **type()** verificando el tipo de datos básicos.

```
numero = 42
print(type(numero))          # <class 'int'>

precio = 19.99
print(type(precio))          # <class 'float'>
```

```python
mensaje = "Hola, Mundo"
print(type(mensaje))         # <class 'str'>

esMayorDeEdad = True
print(type(esMayorDeEdad))   # <class 'bool'>
```

Cada llamada a **type()** devuelve el tipo del valor asignado a la variable: **int** para enteros, **float** para flotantes, **str** para cadenas de texto y **bool** para booleanos.

La función input()

La misma es una herramienta interactiva fundamental que permite a los programas utilizar **datos introducidos por el usuario** a través del teclado. Utilizada comúnmente en aplicaciones de consola, **input()** pausa la ejecución del programa y espera que la persona escriba algo, retornando la entrada como una cadena de texto *(string)* una vez que ésta presione “enter”. Este mecanismo es esencial para crear aplicaciones interactivas que requieren la participación de nuestra persona usuaria. **Input()** tiene los siguientes propósitos:

1. **Interacción:** recoger información de quien utiliza la aplicación, como su nombre, preferencias, comandos, entre otros.
2. **Control de flujo de programas:** basar la lógica del programa en las decisiones de quien interactúa con el mismo.
3. **Entrada de datos para procesamiento:** obtener números, cadenas de texto o cualquier dato que necesite ser procesado posteriormente para su utilización.

Sintaxis básica
La sintaxis básica de **input()** es:

```python
variable = input(prompt)
```

donde **prompt** es un *string*, una cadena de caracteres, que se muestra en la consola indicando al usuario qué debe introducir. Recordamos que este último parámetro es opcional. En el siguiente ejemplo, el prompt dirá: “Introduce tu nombre: “

Solicitar un nombre:

```
nombre = input("Introduce tu nombre: ")
print(f"Hola, {nombre}")
```

Este código pedirá a quien interactúe con él que introduzca su nombre y luego le saludará personalmente. Si alguien introduce el nombre “Mariana”, se imprimirá: “Hola, Mariana”.

Solicitar un número y realizar una operación:
Es importante recordar que **input()** devuelve un *string*, así que si deseás realizar operaciones matemáticas con la entrada, primero tenés que convertirla al tipo de dato adecuado, ya sea *int* o *float*.

```
edad = input("Introduce tu edad: ")
edad = int(edad)
print(f"El próximo año tendrás {edad + 1} años.")
```

Consejos y buenas prácticas

Es fundamental asegurarse de que cualquier entrada proporcionada por el usuario o la usuaria sea validada y verificada antes de ser procesada por el programa. Esto ayuda a evitar errores que podrían interrumpir el funcionamiento del código o generar resultados inesperados. Por ejemplo, si se espera que se introduzca un número, es importante comprobar que el dato ingresado sea válido antes de intentar convertirlo. Esto no sólo mejora la robustez del programa, sino que también proporciona una mejor experiencia a la usuaria o usuario.

Además, es esencial ofrecer instrucciones claras y precisas cuando se solicita información. El texto del *prompt* debe ser lo suficientemente explícito para que quien interactúe con nuestro software entienda exactamente qué tipo de dato se espera que introduzca. Esto reduce la probabilidad de errores y facilita una interacción más fluida con la aplicación.

La función **input()** en Python es una herramienta poderosa que permite crear programas interactivos y adaptables a las necesidades y preferencias de usuarios y usuarias. Al utilizar input(), el programa puede pausar su ejecución y esperar la entrada necesaria para proseguir, lo que abre un amplio rango de posibilidades para desarrollar aplicaciones de consola, scripts personalizados y mucho más. Esta capacidad de interacción convierte a input() en una pieza clave para hacer que los programas sean dinámicos y receptivos a las necesidades del público de nuestras aplicaciones y sitios web.

Los operadores

Se trata de un carácter o conjunto de caracteres que actúa sobre una, dos o más variables y/o literales para llevar a cabo una operación con un resultado determinado.
Algunos ejemplos comunes son los operadores aritméticos, que son los que utilizamos habitualmente para realizar cuentas con números en nuestra vida diaria. Python posee los mismos que encontramos en una calculadora electrónica y agrega algunos tantos más. También podemos encontrar los operadores lógicos y los relacionales, que analizaremos más adelante.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Operadores aritméticos

Los operadores aritméticos son símbolos que representan operaciones matemáticas básicas. Son fundamentales para realizar cálculos en cualquier programa de Python, y permiten sumar, restar, multiplicar, dividir y mucho más. A continuación, analizaremos cada uno de ellos con ejemplos:

Suma (+): Suma dos valores.

```
resultado = 5 + 3  # resultado = 8
```

Resta (-): Resta el segundo valor del primero.

```
resultado = 5 - 3  # resultado = 2
```

Multiplicación (*): Multiplica dos valores.

```
resultado = 5 * 3  # resultado = 15
```

División (/): Divide el primer valor entre el segundo. **¡Atención!** Esta operación siempre retorna un número flotante *(float)*, incluso si el resultado es un entero.

```
resultado = 6 / 3  # resultado = 2.0
```

División entera (//): Divide el primer valor entre el segundo y redondea el resultado hacia abajo para obtener un número entero *(int)*.

```
resultado = 7 // 3  # resultado = 2
```

Módulo (%): Retorna o devuelve el sobrante de la división entre el primer y segundo valor.

```
resultado = 7 % 3  # resultado = 1
```

Exponenciación (**): Eleva el primer valor a la potencia del segundo.

```
resultado = 2 ** 3  # resultado = 8
```

Aplicaciones de los operadores aritméticos

Estos operadores son utilizados frecuentemente para realizar cálculos matemáticos en programas, desde operaciones simples hasta fórmulas matemáticas complejas. Por ejemplo, pueden usarse para calcular el promedio de un conjunto de números, determinar el área de figuras geométricas o ajustar el tamaño de elementos en interfaces gráficas.

Ejemplo práctico
Supongamos que queremos calcular el área de un círculo. Podemos usar el operador de exponenciación para elevar el radio al cuadrado y luego multiplicarlo por Pi (π).

Buenos Aires aprende
Agencia de Habilidades para el Futuro

```python
import math # Importamos el módulo math para usar Pi

radio = 5
area = math.pi * (radio ** 2)    # Calculamos el área

print(f"El área del círculo con radio {radio} es {area:.2f}")
```

Los operadores aritméticos en Python son herramientas básicas pero poderosas que te permiten realizar todo tipo de cálculos matemáticos. Conocer cómo y cuándo utilizarlos es fundamental para cualquier programador, ya que la aritmética es una parte integral de la mayoría de los programas.

Ejercicios prácticos.

Operaciones básicas

Creá un programa que solicite el ingreso de dos números enteros. Realizá las siguientes operaciones: *suma, resta, multiplicación y módulo*. Luego, mostrá el resultado de cada operación en un formato claro. Asegurate de incluir mensajes personalizados que expliquen cada resultado, por ejemplo: "La suma de tus números es: X".

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Resolución

```
numero1 = int(input("Ingresa el primer número entero: "))
numero2 = int(input("Ingresa el segundo número entero: "))

suma = numero1 + numero2
resta = numero1 - numero2
multiplicacion = numero1 * numero2
modulo = numero1 % numero2

print("La suma de tus números es:", suma)
print("La resta de tus números es:", resta)
print("La multiplicación de tus números es:", multiplicacion)
print("El módulo (resto de la división) de tus números es:", modulo)
```

Calculadora de propinas

Escribe un programa en Python que calcule la propina que se debe dejar en un restaurante. El script debe solicitar el monto total de la cuenta y el porcentaje de propina que deseás dejar. Utilizando operadores aritméticos, calculá la cantidad de propina y el total a pagar (incluyendo la propina). Finalmente, mostrá los resultados en la pantalla.

Resolución

```
monto_cuenta = float(input("Ingresa el monto total de la cuenta: "))
porcentaje_propina = float(input("Ingresa el porcentaje de propina que
deseas dejar (por ejemplo, 10 para 10%): "))

propina = (monto_cuenta * porcentaje_propina) / 100
total_pagar = monto_cuenta + propina

print("La propina es:", propina)
print("El total a pagar es:",total_pagar)
```

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad