Unidad 2

Variables y tipos de datos

1. Variables

En este tema, exploraremos uno de los conceptos más fundamentales en cualquier lenguaje de programación: las variables. Las variables son elementos básicos que permiten almacenar y manipular información en un programa, es como una “caja” donde guardamos un valor que podemos usar y modificar a lo largo de nuestro código. Este valor puede ser un número, una cadena de texto, una lista, entre otros tipos de datos.

Llamamos variable a un nombre asociado a un valor que el programa almacena en la memoria, pudiendo reutilizar estos valores y usarlos en operaciones y cálculos, lo cual facilita escribir programas eficientes, organizados y legibles.

Como vimos en el tema anterior, en Python no necesitamos declarar el tipo de una variable. Basta con darle un nombre y asignarle un valor usando el símbolo =. Por ejemplo:

```
edad = 18         # Una variable que almacena un número entero
print(type(edad))

nombre = "Juan"   # Una variable que almacena una cadena de texto
print(type(nombre))

precio = 19.99    # Una variable que almacena un número decimal
print(type(precio))

compra = ["Leche","Huevos","Azucar"]   # Una variable que almacena una lista
print(type(compra))
```

```
<class 'int'>
<class 'str'>
<class 'float'>
<class 'list'>
```

Las asignaciones se hacen usando el carácter “=”. También es posible hacer varias asignaciones en la misma línea separando los diferentes nombres y los diferentes valores con comas:

```
nombre_padre, nombre_madre = "José", "Maria"

print("Padre: " + nombre_padre)

print("Madre: " + nombre_madre)
```

```
Padre: José
Madre: Maria
```

Podemos diferenciar varios tipos de asignaciones según el valor utilizado:

Literales: el valor viene descrito directamente en el código, se puede saber cuál será el resultado simplemente analizando el código.

```
edad = 18
nombre = "Juan"
```

Expresiones: estos datos se asignan tras la evaluación de otras sentencias, suelen ser dinámicos.

```
nombre completo = nombre + apellido
```

Algunas características de las variables en Python:

Tipado dinámico: Python asigna automáticamente el tipo de dato según el valor que se le da a la variable. Esto permite flexibilidad, ya que podemos cambiar el tipo de dato de una variable simplemente asignándole otro valor.

Convenciones de nombres: Las variables deben tener nombres claros y descriptivos. En Python generalmente se suelen escribir en minúsculas y usando guiones bajos para separar palabras, como edad_usuario. A esta anotación se le conoce como Snake case.

Al nombrar una variable tenemos que tener en cuenta una serie de palabras conocidas como *keywords* o palabras reservadas. Estas palabras reservadas son palabras que tienen un significado especial dentro del lenguaje para realizar tareas específicas (bucles, condiciones, valores booleanos…). Si quieres ver la lista de *keywords*, puedes ejecutar las siguientes líneas:

>>> import keyword

>>> print(keyword.kwlist)

['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

2. Tipos de datos

En este tema, profundizaremos en los **tipos de datos** en Python, un elemento fundamental en cualquier lenguaje de programación. Los tipos de datos definen tanto la naturaleza de la información que una variable puede contener como las operaciones que podemos realizar con ella. En Python, cada dato tiene un tipo específico que determina su comportamiento y las posibles interacciones con otros tipos de datos.

Como hemos visto antes, Python tiene un tipado dinámico, pero hay más características sobre su tipado que son importantes mencionar:

Tipado fuerte: esto significa que no se realizan conversiones automáticas entre tipos incompatibles (a menos que esta conversión esté definica, como de *int* a *float*).

```python
numero = 19
cadena = "Hola"

suma = numero + cadena

print(suma)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[13], line 4
      1 numero = 19
      2 cadena = "Hola"
----> 4 suma = numero + cadena
      6 print(suma)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Unión de tipos: importando *union* del módulo *typing* podremos declarar la entrada de una función como una lista de tipos posibles:

```python
from typing import Union

def procesar_dato(dato: Union[int, str]) -> Union[int, str]:
    if isinstance(dato, int):
        return dato
    elif isinstance(dato, str):
        return dato

# Ejemplos de uso
print(type(procesar_dato(42)))         # Salida: El dato es tipo número
print(type(procesar_dato("Python")))   # Salida: El dato es tipo cadena
```

```
<class 'int'>
<class 'str'>
```

Otra herramienta útil es utilizar *optional* que permite que la salida sea de un tipo, o que no tenga salida (realmente esto es como utilizar *Union[tipo, None]*).

Tipados personalizados: también se pueden definir tipos propios utilizando clases en python dependiendo de lo que busquemos conseguir.

Principales Tipos de Datos en Python

Python ofrece una variedad de tipos de datos que podemos usar para representar información de distintos tipos:

1. **Datos numéricos**: Para representar cantidades y valores numéricos. Los principales son:
    - **Enteros (int)**: Números enteros, como 5, -3 o 100.
    - **Flotantes (float)**: Números decimales, como 3.14 o -2.5.
    - **Números complejos (complex)**: Que incluyen una parte real y una imaginaria, útil en ciertos contextos de cálculo.

2. **Cadenas de texto (str)**: Para almacenar texto. En Python, los textos se representan como cadenas de caracteres y pueden incluir letras, números, símbolos y espacios. No existe un tipo *char* como en otros lenguajes como Java o C.
3. **Booleanos (bool)**: Representa los valores verdadero o falso (True o False). Son esenciales para controlar la lógica del programa, como en condiciones y bucles.
4. **Tipos de colección**: Permiten almacenar múltiples valores en una misma estructura:
    - **Listas (list)**: Una colección ordenada de elementos que puede modificarse, ideal para manejar conjuntos de datos que cambian a lo largo del programa.
    - **Tuplas (tuple)**: Colecciones ordenadas e inmutables, útiles cuando necesitamos listas de datos que no deben modificarse.
    - **Conjuntos (set)**: Colecciones desordenadas de elementos únicos, ideales para eliminar duplicados y realizar operaciones de conjuntos.
    - **Diccionarios (dict)**: Estructuras de pares clave-valor, que permiten organizar y acceder a los datos de forma rápida y eficiente.

Si queremos saber qué tipo de dato es una variable, podemos utilizar la función *type(*nombre_variable*)*.

La Importancia de los Tipos de Datos

Conocer y entender los tipos de datos disponibles en Python es esencial para construir programas eficientes y bien estructurados. Elegir el tipo de dato adecuado permite optimizar el almacenamiento y manipulación de datos, evitar errores y mejorar el rendimiento del código.