

Tipos de datos

Edison Achalma

Escuela Profesional de Economía, Universidad Nacional de San Cristóbal de Huamanga

Resumen

Este abstract será actualizado una vez que se complete el contenido final del artículo.

Tabla de contenidos

Introduction	2
0.1 Motivación	2
0.2 Intuición	2
0.3 Modelo mental	2
0.4 Desarrollo	2
0.4.1 1. Tipos de datos principales	2
0.4.2 2. Verificar el tipo con <code>type()</code>	3
0.4.3 3. Asignación automática de tipo	3
0.4.4 4. Casting (conversión explícita)	3
0.4.5 5. Verificar el tipo en código	4
0.5 Ejemplos	4
0.6 Errores comunes	5
0.7 Buenas prácticas	5
0.8 Ejercicio	6
0.9 Síntesis	7
0.10 Qué sigue	7
1 Publicaciones Similares	8

Edison Achalma  <https://orcid.org/0000-0001-6996-3364>

El autor no tiene conflictos de interés que revelar. Los roles de autor se clasificaron utilizando la taxonomía de roles de colaborador (CRediT; <https://credit.niso.org/>) de la siguiente manera: Edison Achalma: conceptualización, metodología, análisis formal, investigación, recursos, curación de datos, redacción, visualización, supervisión, administración del proyecto

La correspondencia relativa a este artículo debe dirigirse a Edison Achalma, Escuela Profesional de Economía, Universidad Nacional de San Cristóbal de Huamanga, Portal Independencia N° 57, Ayacucho, AYA 5001, Perú, Email: elmer.achalma.09@unsch.edu.pe

Tipos de datos

0.1 Motivación

Los tipos de datos son la base de la programación. Determinan qué tipo de información puede almacenar una variable, qué operaciones puedes realizar con ella y cómo se comporta en diferentes situaciones. Entender los tipos de datos te permite escribir código más eficiente, evitar errores inesperados y aprovechar al máximo las capacidades de Python.

0.2 Intuición

Imagina que tienes diferentes tipos de contenedores:

- **Números** son como una calculadora: puedes sumarlos, restarlos, multiplicarlos.
- **Texto (strings)** son como palabras: puedes unirlos, dividirlos, buscar letras.
- **Listas** son como una bandeja con compartimentos: puedes guardar varios elementos ordenados.
- **Booleanos** son como un interruptor: solo pueden estar encendidos (True) o apagados (False).

Cada contenedor tiene sus propias reglas y operaciones, y Python necesita saber con qué tipo está trabajando para actuar correctamente.

0.3 Modelo mental

En Python:

1. **Tipado dinámico:** no declaras el tipo; Python lo deduce al asignar.
2. **Todo es un objeto:** incluso los números y el texto son objetos con métodos.
3. **Los tipos determinan el comportamiento:** lo que puedes hacer con una variable depende de su tipo.
4. **type():** tu herramienta para verificar el tipo de cualquier variable.

0.4 Desarrollo

0.4.1 1. Tipos de datos principales

Categoría	Tipos	Ejemplo	Cuándo usar
Texto	str	"Hola"	Cualquier cadena de caracteres
Números	int, float, complex	5, 3.14, 1j	Operaciones matemáticas
Booleanos	bool	True, False	Condiciones lógicas
Secuencias	list, tuple, range	[1,2,3], (1,2), range(5)	Colecciones de elementos ordenados
Mapeo	dict	{"clave": "valor"}	Pares clave-valor
Conjuntos	set, frozenset	{1,2,3}	Colecciones sin duplicados
Binarios	bytes, bytearray, memoryview	b"Hello"	Datos binarios

Categoría	Tipos	Ejemplo	Cuándo usar
Nulo	NoneType	None	Valor “vacío” o “sin valor”

0.4.2 2. Verificar el tipo con `type()`

```
x = 5
print(type(x)) # <class 'int'>

y = "Hola"
print(type(y)) # <class 'str'>

z = 3.14
print(type(z)) # <class 'float'>

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

0.4.3 3. Asignación automática de tipo

Python asigna el tipo cuando asignas un valor:

```
# Asignación automática
nombre = "Ana"      # str
edad = 25           # int
precio = 99.99      # float
es_valido = True    # bool
lista = [1, 2, 3]   # list
```

0.4.4 4. Casting (conversión explícita)

Puedes convertir entre tipos usando funciones constructoras:

```
# Convertir a string
str(25)      # "25"
str(3.14)    # "3.14"

# Convertir a entero
int("25")    # 25
int(3.14)    # 3 (trunca decimales)

# Convertir a float
float("3.14") # 3.14
float(5)     # 5.0

# Convertir a booleano
bool(0)      # False (cero es False)
```

```
bool(1)          # True
bool("")         # False (string vacío es False)
bool("Hola")     # True
```

0.4.5 5. Verificar el tipo en código

A veces necesitas saber el tipo para tomar decisiones:

```
dato = 25

if type(dato) == int:
    print("Es un entero")
elif type(dato) == str:
    print("Es texto")

# Alternativa más idiomática
if isinstance(dato, int):
    print("Es un entero")
```

0.5 Ejemplos

Mínimo — Tipos básicos:

```
texto = "Python"          # str
entero = 42                # int
decimal = 3.14            # float
booleano = True           # bool
none = None               # NoneType

print(type(texto))        # <class 'str'>
print(type(entero))       # <class 'int'>
```

Progresivo — Conversión entre tipos:

```
# Conversión de tipos
edad = "25"
print(type(edad))        # str
print(type(int(edad)))   # int

# Problemas comunes
# print("Edad: " + 25)    # Error
print("Edad: " + str(25)) # Convierte a string

# Operaciones con números
x = 5
y = 2
print(x / y)             # 2.5 (float)
print(x // y)            # 2 (int, división entera)
print(x % y)             # 1 (resto)
```

Real — Sistema de usuario con validación:

```
# Simulación de entrada de usuario
nombre = "Carlos"      # str
edad = 28              # int
altura = 1.75          # float
es_estudiante = True   # bool

# Validaciones
print(f"Tipo de nombre: {type(nombre)}")
print(f"Tipo de edad: {type(edad)}")
print(f"Tipo de altura: {type(altura)}")
print(f"Tipo de estudiante: {type(es_estudiante)}")

# Conversiones necesarias
edad_texto = str(edad)
altura_texto = str(altura)

print(f"Edad: {edad_texto} años")
print(f"Altura: {altura_texto} metros")
```

0.6 Errores comunes

Error	Síntoma	Solución
Sumar string + número	TypeError	Convierte: str(edad) o usa coma en print
Dividir por cero	ZeroDivisionError	Verifica que el divisor no sea 0
Índice fuera de rango	IndexError	Verifica la longitud de la lista
Comparar tipos incompatibles	TypeError	Convierte al mismo tipo antes de comparar
Usar == en lugar de is con None	Comparación válida pero no idiomática	Usa is None para verificar None

0.7 Buenas prácticas

1. Usa isinstance() en lugar de type() para verificar tipos:

```
# Más flexible
if isinstance(valor, int):
    print("Es entero")

# Menos flexible (no maneja herencia)
if type(valor) == int:
    print("Es entero")
```

2. Convierte tipos explícitamente cuando sea necesario:

```
edad = input("Edad: ") # input siempre devuelve str
edad_int = int(edad)    # Convierte a int
```

3. Sé consciente de los tipos en operaciones mixtas:

```
# La división siempre da float
print(5 / 2)    # 2.5 (float)
print(5 // 2)   # 2 (int)

# Suma de int + float da float
print(5 + 2.0) # 7.0 (float)
```

4. Usa nombres de variables que indiquen el tipo (implícitamente):

```
# Convención
nombres_lista = ["Ana", "Carlos"]
contador_int = 5
es_valido_bool = True
```

5. Documenta los tipos en funciones (type hints):

```
def saludar(nombre: str) -> str:
    return f"Hola, {nombre}"
```

0.8 Ejercicio

Básico — Exploración de tipos

Escribe un programa que:

1. Cree variables de los siguientes tipos: str, int, float, bool, list, dict, NoneType
2. Muestre el tipo de cada variable con `type()`
3. Convierta las siguientes variables:
 - El str a int
 - El int a float
 - El float a str
 - El bool a int
4. Muestre los resultados de cada conversión y sus tipos

Solución (no mires hasta haberlo intentado):

```
# 1. Variables de diferentes tipos
texto = "42"
numero_entero = 25
numero_decimal = 3.14
logico = True
lista_ejemplo = [1, 2, 3]
diccionario_ejemplo = {"nombre": "Ana"}
nulo = None

# 2. Mostrar tipos originales
print("Tipos originales:")
print(f"texto: {type(texto)}")
print(f"numero_entero: {type(numero_entero)}")
print(f"numero_decimal: {type(numero_decimal)}")
print(f"logico: {type(logico)}")
print(f"lista: {type(lista_ejemplo)}")
print(f"diccionario: {type(diccionario_ejemplo)}")
print(f"nulo: {type(nulo)}")

print("\n" + "-" * 40)

# 3. Conversiones
texto_a_int = int(texto)
entero_a_float = float(numero_entero)
decimal_a_str = str(numero_decimal)
bool_a_int = int(logico)

# 4. Mostrar resultados
print("Resultados de conversiones:")
print(f"'{texto}' → {texto_a_int} (tipo: {type(texto_a_int)})")
print(f"{numero_entero} → {entero_a_float} (tipo: {type(entero_a_float)})")
print(f"{numero_decimal} → '{decimal_a_str}' (tipo: {type(decimal_a_str)})")
print(f"{logico} → {bool_a_int} (tipo: {type(bool_a_int)})")
```

0.9 Síntesis

- **Tipos principales:** str, int, float, bool, list, tuple, dict, set, NoneType
- **type():** verifica el tipo de una variable
- **Tipado dinámico:** Python infiere el tipo automáticamente
- **Casting:** conversión explícita con funciones constructoras
- **Operaciones:** dependen del tipo de dato
- **isinstance():** mejor que type() para verificar tipos en herencia
- **Type hints:** documentación opcional de tipos (Python 3.5+)

0.10 Qué sigue

- **Relacionado:** variables (asignación y alcance), strings y métodos de string
- **Siguiente tema:** Strings — aprenderás a trabajar con texto en Python, incluyendo métodos, formateo y manipulación avanzada.

1 Publicaciones Similares

Si te interesó este artículo, te recomendamos que explores otros blogs y recursos relacionados que pueden ampliar tus conocimientos. Aquí te dejo algunas sugerencias:

1.  [Instalacion De Anaconda](#)
2.  [Configurar Entorno Virtual Python Anaconda](#)
3.  [01 Introducion A La Programacion Con Python](#)
4.  [02 Variables Expresiones Y Statements Con Python](#)
5.  [03 Objetos De Python](#)
6.  [04 Ejecucion Condicional Con Python](#)
7.  [05 Iteraciones Con Python](#)
8.  [06 Funciones Con Python](#)
9.  [07 Dataframes Con Python](#)
10.  [08 Prediccion Y Metrica De Performance Con Python](#)
11.  [09 Metodos De Machine Learning Para Clasificacion Con Python](#)
12.  [10 Metodos De Machine Learning Para Regresion Con Python](#)
13.  [Visualizacion De Datos Con Python](#)

Esperamos que encuentres estas publicaciones igualmente interesantes y útiles. ¡Disfruta de la lectura!