

Números

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 Numéricos en Python	2
0.4.2 2. Enteros (int)	2
0.4.3 3. Floats (decimales)	3
0.4.4 4. Números Complejos (complex)	3
0.4.5 5. Conversión entre Tipos (Casting)	3
0.4.6 6. Operaciones Numéricas Comunes	4
0.4.7 7. Números Aleatorios	4
0.4.8 8. Precisión de Floats	4
0.5 Ejemplos	5
0.6 Errores comunes	6
0.7 Buenas prácticas	6
0.8 Ejercicio	7
0.9 Síntesis	8
0.10 Qué sigue	8
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

Números

0.1 Motivación

Los números son fundamentales en programación: desde operaciones aritméticas básicas hasta cálculos científicos y análisis de datos. Python maneja tres tipos numéricos principales, cada uno con propósitos específicos. Entender sus diferencias te permite elegir el tipo correcto para cada situación y evitar errores de precisión o conversión.

0.2 Intuición

Imagina tres tipos de contenedores para números:

- **int (enteros)**: como una hoja de papel donde escribes números sin decimales: 5, -3, 42.
- **float (decimales)**: como una calculadora científica que maneja decimales y notación exponencial: 3.14, -0.5, 1.5e10.
- **complex (complejos)**: como un plano cartesiano con dos ejes (real e imaginario): 3+5j.

Cada uno tiene sus propias reglas y usos, y Python te permite convertir entre ellos cuando es necesario.

0.3 Modelo mental

- **int**: precisión exacta, sin límite de tamaño (puede ser tan grande como la memoria lo permita).
- **float**: precisión limitada (aproximación decimal, típicamente 15-17 dígitos significativos).
- **complex**: dos floats (parte real e imaginaria) combinados.
- **Conversión**: posible entre int y float (puede perder precisión), pero no directamente desde complex.

0.4 Desarrollo

0.4.1 1. Tipos Numéricos en Python

Tipo	Ejemplo	Descripción	Cuándo usar
int	5, -3, 1000000	Números enteros sin decimales	Conteos, índices, cantidades exactas
float	3.14, -0.5, 1.5e10	Números con decimales (precisión limitada)	Mediciones, cálculos científicos, dinero (cuidado con precisión)
complex	3+5j, 1j, -5j	Números complejos (real + imaginario)	Matemáticas avanzadas, ingeniería, física

0.4.2 2. Enteros (int)

Los enteros son números sin decimales, sin límite de tamaño:

```
# Enteros positivos y negativos
x = 42
y = -7
z = 12345678901234567890 # Puede ser enorme

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

0.4.3 3. Floats (decimales)

Los floats representan números con decimales:

```
# Floats básicos
x = 3.14
y = -0.5
z = 1.0

print(type(x)) # <class 'float'>

# Notación científica
a = 35e3 # 35 * 10^3 = 35000.0
b = 12E4 # 12 * 10^4 = 120000.0
c = -87.7e100 # Número muy pequeño (científico)

print(a) # 35000.0
```

0.4.4 4. Números Complejos (complex)

Los números complejos tienen parte real e imaginaria:

```
# Números complejos
x = 3 + 5j
y = 5j # Solo parte imaginaria
z = -5j # Imaginario negativo

print(type(x)) # <class 'complex'>
print(x) # (3+5j)
print(x.real) # 3.0 (parte real)
print(x.imag) # 5.0 (parte imaginaria)
```

0.4.5 5. Conversión entre Tipos (Casting)

Puedes convertir entre tipos numéricos:

```
# Conversiones básicas
x = 5 # int
y = 3.14 # float
z = 1j # complex
```

```
# int → float
a = float(x)      # 5.0
print(type(a))    # <class 'float'>

# float → int (trunca decimales)
b = int(y)         # 3 (pierde .14)
print(type(b))    # <class 'int'>

# int → complex
c = complex(x)     # (5+0j)
print(type(c))    # <class 'complex'>

# No se puede convertir complex → int/float
# d = int(z)       # TypeError: can't convert complex to int
```

0.4.6 6. Operaciones Numéricas Comunes

```
a = 10
b = 3

# Operaciones básicas
print(a + b)  # 13 (suma)
print(a - b)  # 7  (resta)
print(a * b)  # 30 (multiplicación)
print(a / b)  # 3.333... (float)
print(a // b) # 3  (división entera)
print(a % b)  # 1  (resto/modulo)
print(a ** b) # 1000 (potencia)
```

0.4.7 7. Números Aleatorios

Python tiene un módulo random para generar números aleatorios:

```
import random

# Número aleatorio entre 1 y 9 (inclusive)
numero = random.randrange(1, 10)
print(numero)

# Número aleatorio entre 0.0 y 1.0
print(random.random())

# Número aleatorio entre 1 y 10 (float)
print(random.uniform(1, 10))
```

0.4.8 8. Precisión de Floats

Importante: los floats tienen precisión limitada:

```
# Problemas de precisión
print(0.1 + 0.2) # 0.30000000000000004 (no es exactamente 0.3)

# Comparar floats
# No recomendado
if 0.1 + 0.2 == 0.3:
    print("Igual") # No se ejecuta

# Mejor usar tolerancia
tolerancia = 1e-10
if abs((0.1 + 0.2) - 0.3) < tolerancia:
    print("Aproximadamente igual")
```

0.5 Ejemplos

Mínimo — Tipos numéricos básicos:

```
entero = 42
decimal = 3.14
complejo = 2 + 3j

print(type(entero)) # <class 'int'>
print(type(decimal)) # <class 'float'>
print(type(complejo)) # <class 'complex'>
```

Progresivo — Conversiones y operaciones:

```
precio = 19.99
cantidad = 3
total = precio * cantidad
print(total) # 59.97 (float)

# Convertir a entero (trunca)
total_entero = int(total)
print(total_entero) # 59

# Redondear
total_redondeado = round(total)
print(total_redondeado) # 60
```

Real — Cálculo de IMC (Índice de Masa Corporal):

```
# Datos del usuario
peso_kg = 72.5 # float
altura_m = 1.75 # float

# Cálculo del IMC
imc = peso_kg / (altura_m ** 2)
```

```
print(f"IMC: {imc:.2f}") # 23.67 (con 2 decimales)

# Categorización
if imc < 18.5:
    categoria = "Bajo peso"
elif imc < 25:
    categoria = "Peso normal"
elif imc < 30:
    categoria = "Sobrepeso"
else:
    categoria = "Obesidad"

print(f"Categoría: {categoria}")
```

0.6 Errores comunes

Error	Síntoma	Solución
Dividir por cero	ZeroDivisionError	Verifica que el denominador no sea 0
Precisión de floats	Resultados inesperados (0.1 + 0.2)	Usa Decimal para dinero o compara con tolerancia
Conversión de complex	TypeError	No se puede convertir complex a int/float
Concatenar número + string	TypeError	Convierte: str(numero) o usa f-strings
División con ints	5 / 2 da 2.5 (float), no 2	Usa // para división entera

0.7 Buenas prácticas

1. Usa **int** para conteos e **índices**, **float** para mediciones y cálculos.
2. **Para dinero**, considera `decimal.Decimal` en lugar de `float`:

```
from decimal import Decimal
precio = Decimal('19.99')
```

3. Usa **//** para división entera y **%** para obtener el resto:

```
minutos = 125
horas = minutos // 60 # 2
minutos_restantes = minutos % 60 # 5
```

4. **Redondea explícitamente** cuando necesites controlar la precisión:

```
resultado = 3.14159265
print(round(resultado, 2)) # 3.14
```

5. **Prefiere `isinstance()` para verificar tipos:**

```
if isinstance(valor, (int, float)):
    print("Es numérico")
```

6. **Usa `_` para separar números grandes (Python 3.6+):**

```
poblacion = 1_000_000_000 # Más legible que 1000000000
```

0.8 Ejercicio

Intermedio — Calculadora de Prestación

Escribe un programa que:

1. Pida al usuario (simula con variables):
 - Salario mensual (float)
 - Años trabajados (int)
 - Edad (int)
2. Calcule:
 - Salario anual (salario * 12)
 - Prestación por antigüedad: 5% del salario anual por cada año trabajado
 - Edad en 10 años
3. Muestre todos los resultados con 2 decimales
4. Convierta el resultado de la prestación a entero para mostrar la cantidad “redondeada”

Solución (no mires hasta haberlo intentado):

```
# Datos
salario_mensual = 2500.50
años_trabajados = 8
edad = 32

# Cálculos
salario_anual = salario_mensual * 12
prestacion_porcentaje = 0.05 * años_trabajados
prestacion = salario_anual * prestacion_porcentaje
edad_futura = edad + 10

# Mostrar resultados
print("=== Cálculo de Prestación ===")
print(f"Salario mensual: ${salario_mensual:.2f}")
print(f"Salario anual: ${salario_anual:.2f}")
print(f"Años trabajados: {años_trabajados}")
print(f"Prestación: ${prestacion:.2f}")
print(f"Prestación redondeada: ${int(prestacion)}")
print(f"Edad actual: {edad}")
print(f"Edad en 10 años: {edad_futura}")
```

0.9 Síntesis

- **int**: números enteros sin decimales, sin límite de tamaño.
- **float**: números con decimales, precisión limitada.
- **complex**: números complejos (real + imaginario, con j).
- **type()**: verifica el tipo de un número.
- **Conversiones**: `int()`, `float()`, `complex()`.
- **Operaciones**: `+`, `-`, `*`, `/`, `//`, `%`, `**`.
- **Precisión**: los floats son aproximados (problemas con decimales).
- **random**: módulo para generar números aleatorios.

0.10 Qué sigue

- **Relacionado**: operadores matemáticos, módulo `math`, formateo de números
- **Siguiente tema**: Strings — aprenderás a trabajar con texto y todas sus operaciones.

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. [!\[\]\(0f48f43ebd21f231a458c96216dbf4d1_img.jpg\) Instalacion De Anaconda](#)
2. [!\[\]\(ba0878532603d6e0b20c60ffb7475d12_img.jpg\) Configurar Entorno Virtual Python Anaconda](#)
3. [!\[\]\(0a70dbd9915c9ea99c6f238a2c711f53_img.jpg\) 01 Introducion A La Programacion Con Python](#)
4. [!\[\]\(76511d025e9c8b0592325ddba91331d4_img.jpg\) 02 Variables Expresiones Y Statements Con Python](#)
5. [!\[\]\(4ecf357067dff2cd4f65e9b71acfab07_img.jpg\) 03 Objetos De Python](#)
6. [!\[\]\(bdcc6b43570ac03740a9b9fceebadffe_img.jpg\) 04 Ejecucion Condicional Con Python](#)
7. [!\[\]\(19e7888cfa74401aed481c3fe14d7758_img.jpg\) 05 Iteraciones Con Python](#)
8. [!\[\]\(f030a72b34d8705fb3a85bf642ca0dcf_img.jpg\) 06 Funciones Con Python](#)
9. [!\[\]\(302a53aa762c1c37ba80a26a4d8595c8_img.jpg\) 07 Dataframes Con Python](#)
10. [!\[\]\(cc6426c2424de94bccb97122f1010e5f_img.jpg\) 08 Prediccion Y Metrica De Performance Con Python](#)
11. [!\[\]\(e6c9a425b59d6581d047c5bc4ecaae35_img.jpg\) 09 Metodos De Machine Learning Para Clasificacion Con Python](#)
12. [!\[\]\(27d3caa558663f0123fe8843a9223c11_img.jpg\) 10 Metodos De Machine Learning Para Regresion Con Python](#)
13. [!\[\]\(3069913754f6d5f2ee05ee08eb0b17c0_img.jpg\) Visualizacion De Datos Con Python](#)

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