Tipos de variables y operaciones en Python

Materia: Algoritmos y Programación

Prof. Dr. Aboud BARSEKH-ONJI

Facultad de Ingeniería
Universidad Anáhuac México

26 de enero de 2026

Agenda

¿Qué es una Variable?

Analogía: Una Caja con Etiqueta

Una variable es como una **caja en la memoria** de la computadora donde podemos guardar información. Le ponemos una **etiqueta (el nombre de la variable)** para poder encontrar y usar esa información más tarde.

Tipado Dinámico en Python

A diferencia de otros lenguajes, en Python no necesitas declarar el tipo de una variable. Python lo infiere automáticamente cuando le asignas un valor.

```
1 # Python sabe que 'edad' es un numero
2 edad = 25
3
4 # Python sabe que 'nombre' es texto
5 nombre = 'Juan Perez'
```

Tipos de Datos Numéricos: 'int' y 'float'

Enteros ('int')

Representan números enteros, positivos o negativos, sin parte decimal.

```
# Ejemplos de enteros
edad = 30
temperatura = -5
numero_de_alumnos = 150

```

Flotantes ('float')

Representan números de punto flotante, es decir, números con parte decimal.

```
# Ejemplos de flotantes
precio = 199.99
pi = 3.14159
altura = 1.75

```

Tipos de Datos Numéricos: 'int' y 'float'

Operaciones Matemáticas

Python realiza las operaciones aritméticas de forma natural. ¡Ojo! La división '/' siempre produce un 'float'.

```
suma = 10 + 5        # Resultado: 15 (int)
resta = 20.5 - 10    # Resultado: 10.5 (float)
division = 10 / 2    # Resultado: 5.0 (float!)

```

Tipo de Dato de Texto: 'str'

Cadenas de Caracteres ('str')

Una cadena de caracteres, o *string*, se usa para almacenar texto. El texto debe ir entre comillas simples (' ') o dobles (" ").

```
1 nombre = 'Ana'
2 mensaje = 'Hola Mundo'
3 frase = 'El lenguaje "Python" es muy versatil.'
4
```

Tipo de Dato de Texto: 'str'

Operación Principal: Concatenación (+)

El operador '+' con cadenas de texto no suma, sino que las une (concatena).

```
1 nombre = 'Carlos'
2 apellido = 'Santana'
3 nombre_completo = nombre + ' ' + apellido
4
5 # Resultado: 'Carlos Santana'
6 print(nombre_completo)
7
```

Tipo de Dato Lógico: 'bool'

Booleanos ('bool')

El tipo booleano solo puede tener dos valores: **'True'** (Verdadero) o **'False'** (Falso).

```
1 es_estudiante = True
2 es_mayor_de_edad = False
3
```

Tipo de Dato Lógico: 'bool'

Uso Principal

Los booleanos son el resultado de las comparaciones y son fundamentales para la toma de decisiones en el código (condicionales 'if', bucles 'while', etc.).

```
edad = 20
es_mayor = edad >= 18 # es_mayor sera True

temperatura = 15
hace_frio = temperatura < 10 # hace_frio sera False

```

Agenda

El Problema: Operaciones entre Tipos Incompatibles

Python es de Tipado Fuerte

Aunque Python es flexible, no permite realizar operaciones entre tipos que no son compatibles. Por ejemplo, no se puede 'sumar' un número a un texto directamente.

Ejemplo de Error Típico ('TypeError')

Este código producirá un error:

```
1 edad = 25
2 # TypeError: can only concatenate str (not 'int') to str
3 mensaje = 'Tu edad es: ' + edad
4
```

La solución es convertir explícitamente uno de los tipos para que sean compatibles. A esto se le llama **Casting**.

Conversión de Tipos (Casting)

'str()'

Convierte un valor a una cadena de texto.

```
edad = 25
# Correcto!
mensaje = 'Tu edad es: ' + str(edad)
# mensaje = 'Tu edad es: 25'

numero = 123
texto = str(numero)
# texto = '123'

```

Conversión de Tipos (Casting)

'int()'

Convierte un valor a un entero. **Trunca (corta) los decimales**, no redondea.

```
1 # De str a int
2 puntos_str = '100'
3 puntos_int = int(puntos_str)
4
5 # De float a int
6 precio = 199.99
7 precio_int = int(precio)
8 # precio_int = 199
9
```

Conversión de Tipos (Casting)

'float()'

Convierte un valor a un número de punto flotante.

```
1 # De str a float
2 pi_str = '3.14159'
3 pi_float = float(pi_str)
4
5 # De int a float
6 edad = 30
7 edad_float = float(edad)
8 # edad_float = 30.0
9
```

Agenda

Operadores Aritméticos Básicos

Suma, Resta y Multiplicación

- '+' (Suma)
- '-' (Resta)
- '*' (Multiplicación)

```
a = 15
b = 4

suma = a + b          # 19
resta = a - b         # 11
multiplicacion = a * b # 60

```

Operadores Aritméticos Básicos

División

- '/' (División)

¡Importante!: El operador de división '/' siempre devuelve un resultado de tipo 'float', incluso si la división es exacta.

```
# La division siempre resulta en float
division_1 = 15 / 4  # 3.75
division_2 = 20 / 4  # 5.0

```

Operadores Aritméticos Avanzados

División Entera '//'

Realiza la división y descarta (trunca) la parte decimal, devolviendo solo la parte entera.

```
1 resultado = 15 // 4 # 3
2
```

Operadores Aritméticos Avanzados

Módulo %

Devuelve el **residuo** de una división. Es muy útil para saber si un número es par o impar.

```
# 15 / 4 es 3 con residuo 3
residuo = 15 % 4  # 3

# Si n % 2 == 0, n es par
es_par = 10 % 2   # 0

```

Operadores Aritméticos Avanzados

Exponenciación '**'

Eleva un número a una potencia.

```
# 2 elevado a la 3
potencia = 2 ** 3 # 8

# Raiz cuadrada
raiz = 9 ** 0.5 # 3.0

```

Orden de Operaciones (Precedencia)

PEMDAS

Python sigue el orden estándar de las operaciones matemáticas, a menudo recordado por el acrónimo PEMDAS.

Jerarquía de Operadores

1. **P**aréntesis '()'
2. **E**xponentes '**'
3. **M**ultiplicación '*', **D**ivisión '/', Módulo '%', División Entera '//' (de izq. a der.)
4. **S**uma (Adición) '+', **R**esta '-' (de izq. a der.)

Orden de Operaciones (Precedencia)

Examples

```
# Como lo evalua Python:
# 1. Exponente: 3 ** 2 = 9
# 2. Multiplicacion: 2 * 9 = 18
# 3. Suma: 5 + 18 = 23
resultado = 5 + 2 * 3 ** 2
print(resultado) # Imprime 23

```

Homework: Simulador de Cajero Automático (ATM)

Objetivo

Escribir un programa en Python que simule las operaciones básicas de un cajero automático. Deberás aplicar todo lo aprendido: variables, input, casting, operaciones matemáticas y estructuras de control ('if'/'elif'/'else' y 'while').

Homework: Simulador de Cajero Automático (ATM)

Requisitos del Programa

- **Saldo Inicial:** El programa debe comenzar con un saldo inicial (por ejemplo, `saldo = 1000.0`).
- **Menú Interactivo:** El programa debe mostrar un menú con 4 opciones y repetirse hasta que el usuario elija "Salir".
  - 1. Consultar Saldo
  - 2. Depositar Dinero
  - 3. Retirar Dinero
  - 4. Salir

Homework: Simulador de Cajero Automático (ATM)

Requisitos del Programa

- **Funcionalidad de Opciones:**
  - **Opción 1:** Muestra el saldo actual formateado.
  - **Opción 2:** Pide al usuario una cantidad, la suma al saldo y muestra el nuevo saldo.
  - **Opción 3:** Pide al usuario una cantidad. **Si hay fondos suficientes**, la resta del saldo. Si no, muestra un mensaje de "Fondos insuficientes".
  - **Opción 4:** Muestra un mensaje de despedida y termina el programa.

Homework: Simulador de Cajero Automático (ATM)

Pistas y Consejos

- Usa un bucle `while:` para que el menú se muestre continuamente.
- Recuerda convertir la entrada del usuario a número (`int()` o `float()`) antes de hacer cálculos.
- Usa una estructura `if/elif/else` para manejar las 4 opciones del menú.
- Para salir del bucle `while`, puedes usar la palabra clave `break`.