Notas de la Conferencia 1: Fundamentos de Programación con Python

¡Bienvenido al mundo de la programación! Este documento resume los conceptos clave que vimos en nuestra primera clase. Úsalo para repasar y afianzar tu conocimiento.

1. ¿Qué es Programar?

Programar es darle a la computadora una serie de **instrucciones precisas y ordenadas** para resolver un problema. Piensa en ello como una receta de cocina: cada paso debe ser claro y el orden es fundamental para que el resultado sea el esperado.

Elegimos **Python** porque su sintaxis es limpia y legible, lo que lo hace ideal para aprender, sin dejar de ser una herramienta extremadamente poderosa utilizada en toda la industria tecnológica.

2. Nuestro Entorno de Trabajo

Interactuamos con Python de dos maneras principales:

- **Intérprete Interactivo:** Es una consola donde escribes una línea de código y se ejecuta al instante. Es perfecta para hacer pruebas rápidas y experimentar. No guarda tu trabajo.
- **Archivos de Script (`.py`):** Aquí es donde escribimos nuestros programas completos. Guardamos una secuencia de instrucciones en un archivo (por ejemplo, `mi_programa.py`) y luego le pedimos a Python que lo ejecute.

3. Conceptos Fundamentales

"Hola, Mundo": Tu Primer Programa

El primer paso en cualquier lenguaje es el "Hola, Mundo". Usamos la función `print()` para mostrar texto en la pantalla.

```
print("¡Hola, Mundo!")
```

Variables: Guardando Información

Una variable es como una **caja con una etiqueta** donde guardamos un valor para usarlo más tarde. El signo `=` se usa para **asignar** un valor a una variable.

```
saludo = "¡Hola, clase de programación!"
print(saludo)
```

El Orden Secuencial: ¡Concepto Clave!

El código se ejecuta estrictamente de arriba hacia abajo. El valor de una variable es una "foto" del estado del programa en un momento específico. No se actualiza automáticamente.

Considera este código:

```
a = 5
b = a + 1
print(b)  # Imprime 6

a = 10
print(b)  # Sigue imprimiendo 6
```

¿Por qué `b` sigue siendo 6?

1. En la línea 2, Python calculó `a + 1` (que era `5 + 1`) y guardó el **resultado**, `6`, en `b`.
2. En la línea 5, aunque `a` cambió a `10`, ninguna instrucción le dijo a Python que volviera a calcular `b`. Por lo tanto, `b` conserva el valor que se le asignó.

La asignación es una acción, no una relación continua.

Tipos de Datos Primitivos

Todo valor en Python tiene un tipo, que define qué podemos hacer con él. Los básicos son:

- `int`: Números enteros (ej. `10`, `-3`, `0`).
- `float`: Números con decimales (ej. `3.14`, `-0.5`).
- `str`: Cadenas de texto (siempre entre comillas `""` o `''`).
- `bool`: Representa verdad o falsedad (`True` o `False`).

Podemos usar la función `type()` para saber el tipo de un dato: `type(25)` devuelve `int`.

4. Operaciones

Operadores Aritméticos

Se usan con `int` y `float`.

- `+` (suma), `-` (resta), `*` (multiplicación)
- `/` (división, siempre devuelve `float`, ej. `10 / 3` → `3.333...`)
- `//` (división entera, ej. `10 // 3` → `3`)
- `%` (módulo o residuo, ej. `10 % 3` → `1`)
- `**` (potencia, ej. `2 ** 3` → `8`)

Operadores de Comparación

Comparan dos valores y siempre devuelven un `bool` (`True` o `False`).

- `==` (igual a)
- `!=` (no igual a)
- `>` (mayor que), `<` (menor que)
- `>=` (mayor o igual que), `<=` (menor o igual que)

Operadores Lógicos

Combinan valores `bool` para crear expresiones más complejas.

- **`and`**: Devuelve `True` solo si **ambos** lados son `True`.
  - `(10 > 5) and (3 == 3)` → `True and True` → `True`
- **`or`**: Devuelve `True` si **al menos uno** de los lados es `True`.
  - `(10 < 5) or (3 == 3)` → `False or True` → `True`
- **`not`**: Invierte el valor booleano.
  - `not (10 < 5)` → `not False` → `True`

Operadores de Strings

- `+` (concatenación): `"Hola" + " " + "Mundo"` → `"Hola Mundo"`
- `*` (repetición): `"Eco..." * 3` → `"Eco...Eco...Eco..."`

5. Interactuando con el Usuario

`input()`: Recibiendo Datos

La función `input()` pausa el programa y espera que el usuario escriba algo. **Importante:** `input()` siempre devuelve los datos como un `str`.

```
nombre = input("¿Cómo te llamas? ")
print("Hola, " + nombre)
```

Conversión de Tipos (Casting)

Si recibimos números con `input()`, debemos convertirlos de `str` a `int` o `float` antes de poder hacer cálculos.

```
edad_str = input("¿Qué edad tienes? ") # Usuario escribe "25"
edad_num = int(edad_str)               # Convertimos el str "25" al int 25
print("El año que viene tendrás", edad_num + 1)
```

f-strings: Formateando Salidas

La mejor manera de incluir variables en un `print` es con f-strings. Solo pon una `f` antes de las comillas y las variables entre llaves `{}`.

```
print(f"Hola, {nombre}. El año que viene tendrás {edad_num + 1} años.")
```

6. Ejemplo Final: Calculadora Completa

Este script une todos los conceptos que vimos (revisa `code01.py` para ver el script completo):

```
# --- Calculadora Completa ---
# Este programa demuestra los conceptos de entrada de datos,
# conversión de tipos y operaciones.

# --- 1. ENTRADA DE DATOS ---
print("--- Calculadora de Operaciones Básicas ---")
input_a_str = input("Introduce el primer número (a): ")
input_b_str = input("Introduce el segundo número (b): ")
```

```
# --- 2. PROCESO (Conversión de Tipos) ---
a = float(input_a_str)
b = float(input_b_str)

# --- 3. SALIDA (Mostrar Resultados) ---
print("\n--- Operaciones Aritméticas ---")
print(f"{a} + {b} = {a + b}")
print(f"{a} * {b} = {a * b}")

print("\n--- Operaciones de Comparación ---")
print(f"{a} == {b} (¿Es igual?) = {a == b}")
print(f"{a} > {b}  (¿Es mayor que?) = {a > b}")
```

7. Ejercicios Propuestos

¡Ahora te toca a ti! Intenta resolver estos problemas para practicar.

Ejercicio 1: Calculadora de Ecuación Cuadrática

Escribe un programa que pida al usuario los valores de `a`, `b` y `c` para una ecuación cuadrática (`ax² + bx + c = 0`) y calcule sus dos raíces.

Fórmula: Las raíces se calculan como

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Pista: Para calcular la raíz cuadrada (√), puedes elevar un número a la potencia de `0.5`. Por ejemplo, `25 ** 0.5` da como resultado `5.0`. ¡Más adelante veremos librerías con funciones matemáticas más avanzadas!

Tu programa deberá calcular las dos raíces (una usando `+` y la otra `-`) e imprimirlas. (Nota: si el `discriminante`, es decir, `b² - 4ac`, es negativo, el programa dará un error. ¡No te preocupes! Aprenderemos a manejar eso en la próxima clase).

Ejercicio 2: La Desigualdad Triangular

Escribe un programa que lea tres números (`a`, `b`, `c`) que representan las longitudes de los lados de un posible triángulo.

Para que tres lados puedan formar un triángulo, se debe cumplir la **desigualdad triangular**: la suma de las longitudes de dos lados cualesquiera debe ser siempre mayor que la longitud del tercer lado. Es decir, las **tres** condiciones siguientes deben ser ciertas a la vez:

1. `a + b > c`
2. `a + c > b`
3. `b + c > a`

Tu programa debe verificar si las tres condiciones se cumplen simultáneamente y mostrar un único resultado booleano (`True` o `False`). Para combinar las condiciones, necesitarás el operador lógico `and`.

La salida debe ser exactamente así: `Puede formar un triangulo: True` (o `False`)