Introducción a Python y Variables

1. ¿Qué es Python y por qué aprenderlo?

1.1 Historia breve

Python es un lenguaje de programación de propósito general creado en 1989 por Guido van Rossum. El nombre no proviene de la serpiente, sino de la serie británica de comedia "Monty Python". Python se ha convertido en uno de los lenguajes más populares del mundo debido a su simplicidad, legibilidad y versatilidad.

1.2 Características principales de Python

Python destaca por varias razones que lo hacen ideal para principiantes:

- **Sintaxis sencilla y legible**: El código de Python se parece mucho al lenguaje natural (inglés), lo que hace que sea fácil de entender incluso para quienes no tienen experiencia en programación.
- **Interpretado**: No necesita compilación; el código se ejecuta línea por línea, lo que facilita la depuración.
- **Versátil**: Se usa en desarrollo web (Django, Flask), ciencia de datos (Pandas, NumPy), inteligencia artificial (TensorFlow), automatización, y mucho más.
- **Comunidad activa**: Existe una gran cantidad de librerías, tutoriales y soporte en línea.

1.3 ¿Dónde se usa Python?

Python se utiliza en empresas como:

- **Google**: para procesamiento de datos y automatización.
- **Netflix**: para sistemas de recomendación.
- **Spotify**: para análisis de datos de usuarios.
- **Tesla**: para programación de sistemas autónomos.
- **Instagram**: en su backend.
- **Instituciones académicas**: para enseñanza de programación y investigación.

2. Configuración inicial: Cómo ejecutar código Python

2.1 Opciones para ejecutar Python

Existen varias formas de trabajar con Python sin necesidad de instalación complicada:

Opción 1: Entornos en línea (recomendado para aprender)

- **Replit** (https://replit.com/): Plataforma en línea muy fácil de usar.
- **Google Colab** (https://colab.research.google.com/): Basado en Jupyter, excelente para aprender.
- **Trinket** (https://trinket.io/): Interfaz simple y directa.
- **OneCompiler** (https://onecompiler.com/python): Muy rápido y sin registro requerido.

Opción 2: Instalación local

- Descargar Python desde https://www.python.org/
- Instalar un editor como **Visual Studio Code** o **PyCharm Community Edition**.

Para este curso usaremos cualquiera de los entornos en línea mencionados.

2.2 Tu primer programa: "Hola, Mundo"

El primer programa que escribirá cualquier programador es uno que muestre un mensaje en pantalla. Abre tu entorno en línea y copia el siguiente código:

```
print("Hola, Mundo")
```

Luego presiona el botón de ejecución (generalmente un triángulo verde ▶️). Deberías ver en la pantalla:

```
Hola, Mundo
```

¡Felicidades! Acabas de ejecutar tu primer programa en Python.

3. Conceptos fundamentales: Print, comentarios y entrada de datos

3.1 La función print()

La función `print()` es la más básica en Python. Se usa para mostrar información en la pantalla (consola).

Sintaxis básica:

```
print("Texto que queremos mostrar")
```

Ejemplos:

```
print("Mi nombre es Juan")
print("Estoy aprendiendo Python")
print("123")  # Aunque sea un número, se muestra como texto
```

Salida:

```
Mi nombre es Juan
Estoy aprendiendo Python
123
```

Mostrando múltiples elementos:

Puedes mostrar varios elementos separados por comas:

```
print("Mi nombre es", "Juan", "y tengo", 20, "años")
```

Salida:

```
Mi nombre es Juan y tengo 20 años
```

Saltos de línea:

Por defecto, `print()` añade un salto de línea al final. Si quieres múltiples `print()`, cada uno aparecerá en una línea diferente:

```
print("Primera línea")
print("Segunda línea")
print("Tercera línea")
```

Salida:

```
Primera línea
Segunda línea
Tercera línea
```

3.2 Comentarios en Python

Los comentarios son notas que escribimos en el código pero que Python **no ejecuta**. Sirven para explicar qué hace el código y ayudar a otros (o a ti mismo en el futuro) a entenderlo.

Comentarios de una línea con #:

```
# Este es un comentario
print("Este código se ejecuta")  # Este también es un comentario
# print("Este código NO se ejecuta porque está comentado")
```

Salida:

```
Este código se ejecuta
```

Buenas prácticas con comentarios:

- Usa comentarios para explicar el "por qué", no el "qué" (el código debe ser claro por sí solo para explicar qué hace).
- Mantén los comentarios actualizados con el código.
- No abuses de los comentarios; código claro es mejor que muchas notas.

Ejemplo de buen comentario:

```
# Calculamos el total con impuesto del 19%
total_con_impuesto = precio * 1.19
```

Ejemplo de mal comentario:

```
# Multiplicar precio por 1.19
total_con_impuesto = precio * 1.19
```

3.3 Entrada de datos con input()

Hasta ahora solo hemos mostrado información. Ahora aprenderemos a **recibir información del usuario** usando `input()` .

Sintaxis básica:

```
variable = input("Pregunta al usuario: ")
```

Ejemplo simple:

```
nombre = input("¿Cuál es tu nombre? ")
print("Hola,", nombre)
```

Ejecución (el usuario digita "Carlos"):

```
¿Cuál es tu nombre? Carlos
Hola, Carlos
```

Punto importante: `input()` siempre devuelve texto (una cadena de caracteres). Si el usuario digita un número, seguirá siendo texto, no un número.

```
edad = input("¿Cuántos años tienes? ")
print("Tu respuesta fue:", edad)
print("El tipo de dato es:", type(edad))
```

Salida (si el usuario digita 18):

```
¿Cuántos años tienes? 18
Tu respuesta fue: 18
El tipo de dato es: <class 'str'>
```

4. Variables: Almacenando información

4.1 ¿Qué es una variable?

Una variable es un "contenedor" en la memoria del computador donde podemos guardar información. Es como una caja etiquetada donde ponemos valores para usarlos después.

Analogía del mundo real:

Imagina que tienes una caja de almacenamiento. En el exterior escribes una etiqueta (el nombre de la variable) y dentro pones un objeto (el valor). Cuando necesitas ese objeto, simplemente buscas la caja por su etiqueta.

4.2 Creando variables (asignación)

Para crear una variable en Python, usamos el operador `=` :

```
nombre_de_variable = valor
```

Ejemplos:

```
nombre = "Carlos"
edad = 20
altura = 1.75
es_estudiante = True
```

En estos ejemplos:

- `nombre` es una variable que contiene el texto "Carlos"
- `edad` contiene el número entero 20
- `altura` contiene el número decimal 1.75
- `es_estudiante` contiene el valor booleano True (verdadero)

Importante: Cuando usamos `=` en programación, significa "asignar el valor del lado derecho a la variable del lado izquierdo". **No es una ecuación matemática.**

4.3 Reglas para nombrar variables

Python tiene algunas reglas para los nombres de variables:

✅ **Permitido:**

- Pueden contener letras (a-z, A-Z), números (0-9) y guiones bajos (_).
- Deben comenzar con una letra o un guión bajo, nunca con un número.
- Son sensibles a mayúsculas/minúsculas (`edad` , `Edad` , `EDAD` son tres variables diferentes).
- Pueden ser tan largos como quieras.

❌ **No permitido:**

- No pueden contener espacios.
- No pueden comenzar con números.
- No pueden ser palabras reservadas de Python (como `print` , `if` , `for` , etc.).
- No pueden contener caracteres especiales (excepto el guión bajo).

Ejemplos de nombres válidos:

```
mi_nombre = "Juan"
nombre_completo = "Juan García"
edad_2025 = 20
_edad_privada = 20
edadEnAños = 20  # Estilo camelCase
EDAD_CONSTANTE = 20  # Estilo CONSTANT_CASE
```

Ejemplos de nombres inválidos:

```
2nombre = "Juan"  # ❌ Comienza con número
mi-nombre = "Juan"  # ❌ Contiene guión (no guión bajo)
mi nombre = "Juan"  # ❌ Contiene espacio
mi nombre! = "Juan"  # ❌ Contiene carácter especial
for = 20  # ❌ Es una palabra reservada
```

4.4 Convención de nombres (buenas prácticas)

En Python, la convención estándar es usar **snake_case** (palabras separadas por guiones bajos):

```
nombre_del_usuario = "Carlos"
edad_actual = 25
saldo_bancario = 1500.50
```

Evita:

```
nombreDelUsuario = "Carlos"  # camelCase (más usado en otros lenguajes)
NombreDelUsuario = "Carlos"  # PascalCase
nombre = "Carlos"  # Muy corto, poco descriptivo
```

5. Tipos de datos en Python

Python tiene varios tipos de datos fundamentales. Ahora aprenderemos los más básicos y comunes.

5.1 Números enteros (int)

Los números enteros son números sin decimales: -5, 0, 42, 1000, etc.

```
edad = 25
cantidad_estudiantes = 150
temperatura_bajo_cero = -10
numero_grande = 1000000
```

Operaciones con enteros:

```
a = 10
b = 3

suma = a + b  # 13
resta = a - b  # 7
multiplicacion = a * b  # 30
division = a / b  # 3.3333... (resultado es float)
division_entera = a // b  # 3 (divide y redondea hacia abajo)
modulo = a % b  # 1 (el residuo de la división)
potencia = a ** b  # 1000 (10 elevado a 3)

print("Suma:", suma)
print("Resta:", resta)
print("Multiplicación:", multiplicacion)
print("División:", division)
```

```
print("División entera:", division_entera)
print("Módulo:", modulo)
print("Potencia:", potencia)
```

Salida:

```
Suma: 13
Resta: 7
Multiplicación: 30
División: 3.3333333333333335
División entera: 3
Módulo: 1
Potencia: 1000
```

5.2 Números decimales (float)

Los números decimales (números con punto) se llaman "float" en Python.

```
altura = 1.75
precio = 29.99
pi = 3.14159
temperatura = -3.5
```

Operaciones con floats:

```
x = 10.5
y = 3.2

print(x + y)  # 13.7
print(x - y)  # 7.3
print(x * y)  # 33.6
print(x / y)  # 3.28125
```

Nota importante: Cuando divides dos enteros con `/`, el resultado siempre es un float:

```
resultado = 10 / 2  # 5.0 (es float, no int)
print(type(resultado))  # <class 'float'>
```

5.3 Cadenas de texto (str)

Una cadena de caracteres (string) es cualquier texto envuelto en comillas (simples o dobles).

```
nombre = "Carlos"
apellido = 'García'
```

```
mensaje = "Hola, ¿cómo estás?"
linea_vacia = ""
```

Las comillas simples y dobles son equivalentes:

```
print("Esto es lo mismo")
print('Esto también es lo mismo')
```

Usando comillas dentro de strings:

Si quieres incluir comillas dentro de tu texto, tienes dos opciones:

```
# Opción 1: Usar la otra tipo de comilla
texto1 = "Él dijo 'Hola'"
texto2 = 'Ella respondió "¿Qué tal?"'

# Opción 2: Escapar con barra invertida \
texto3 = "Él dijo \"Hola\""
texto4 = 'Ella respondió \'¿Qué tal?\''

print(texto1)
print(texto2)
print(texto3)
print(texto4)
```

Salida:

```
Él dijo 'Hola'
Ella respondió "¿Qué tal?"
Él dijo "Hola"
Ella respondió '¿Qué tal?'
```

Concatenación (unión) de strings:

Puedes unir cadenas de texto usando el operador `+` :

```
nombre = "Juan"
apellido = "Pérez"
nombre_completo = nombre + " " + apellido

print(nombre_completo)  # Juan Pérez
```

Repetición de strings:

Puedes repetir una cadena usando `*` :

```
palabra = "Ha"
risa = palabra * 3
print(risa)  # HaHaHa
```

Métodos útiles para strings:

```
texto = "Python es genial"

# Convertir a mayúsculas
print(texto.upper())  # PYTHON ES GENIAL

# Convertir a minúsculas
print(texto.lower())  # python es genial

# Cantidad de caracteres
print(len(texto))  # 16

# Reemplazar texto
print(texto.replace("genial", "increíble"))  # Python es increíble

# Dividir en palabras
palabras = texto.split()
print(palabras)  # ['Python', 'es', 'genial']
```

5.4 Booleanos (bool)

Un booleano es un tipo de dato que solo puede tener dos valores: `True` (verdadero) o `False` (falso). Son útiles para tomar decisiones en el código.

```
es_mayor_de_edad = True
es_fin_de_semana = False
esta_lloviendo = True
tiene_dinero = False
```

Comparaciones que devuelven booleanos:

```
edad = 20

print(edad > 18)  # True
print(edad < 18)  # False
print(edad == 20)  # True
print(edad != 18)  # True
```

6. Conversión de tipos de datos

A veces necesitamos convertir datos de un tipo a otro. Por ejemplo, `input()` siempre devuelve texto, pero si queremos números, debemos convertir.

6.1 Convertir a entero (int)

```
numero_texto = "25"
numero_entero = int(numero_texto)

print(numero_entero + 5)  # 30 (ahora sí es una suma numérica)
```

Con input:

```
edad = int(input("¿Cuántos años tienes? "))
edad_en_10_anos = edad + 10
print("En 10 años tendrás", edad_en_10_anos, "años")
```

Si intentas convertir texto que no es un número, obtendrás un error:

```
numero = int("abc")  # ❌ Error: ValueError
```

6.2 Convertir a decimal (float)

```
numero_texto = "3.14"
numero_decimal = float(numero_texto)
print(numero_decimal * 2)  # 6.28

# También funciona con enteros
numero_entero = 5
numero_decimal2 = float(numero_entero)
print(numero_decimal2)  # 5.0
```

6.3 Convertir a texto (str)

```
numero = 42
numero_como_texto = str(numero)

print("El número es: " + numero_como_texto)  # El número es: 42

edad = 25
print("Tengo " + str(edad) + " años")  # Tengo 25 años
```

6.4 Conocer el tipo de dato con type()

Para saber qué tipo de dato tiene una variable, usamos `type()`:

```
print(type(25))  # <class 'int'>
print(type(3.14))  # <class 'float'>
print(type("Hola"))  # <class 'str'>
print(type(True))  # <class 'bool'>
```

7. Ejercicio guiado: Calculadora simple

Ahora que conocemos los conceptos básicos, crearemos un programa pequeño pero completo que funciona como calculadora.

7.1 Objetivo del ejercicio

Crear un programa que:

1. Pida al usuario dos números.
2. Realice las cuatro operaciones básicas (suma, resta, multiplicación, división).
3. Muestre los resultados de forma clara.

7.2 Planificación del programa

Antes de escribir código, siempre es bueno pensar en los pasos:

1. **Entrada**: Pedir dos números al usuario.
2. **Procesamiento**: Calcular suma, resta, multiplicación y división.
3. **Salida**: Mostrar los resultados en pantalla.

7.3 Código paso a paso

```
# PASO 1: Pedir números al usuario
print("=== CALCULADORA SIMPLE ===")
print()

numero1 = float(input("Ingresa el primer número: "))
numero2 = float(input("Ingresa el segundo número: "))

# PASO 2: Calcular las operaciones
suma = numero1 + numero2
resta = numero1 - numero2
multiplicacion = numero1 * numero2
division = numero1 / numero2  # Asumimos que numero2 no es 0
```

```
# PASO 3: Mostrar resultados
print()
print("=== RESULTADOS ===")
print(f"Suma: {numero1} + {numero2} = {suma}")
print(f"Resta: {numero1} - {numero2} = {resta}")
print(f"Multiplicación: {numero1} * {numero2} = {multiplicacion}")
print(f"División: {numero1} / {numero2} = {division}")
```

7.4 Explicación línea por línea

- Línea 1-2: Comentarios que explican el propósito.
- Línea 5-6: Usamos `float(input())` para pedir números decimales al usuario.
- Línea 8-11: Guardamos los resultados en variables.
- Línea 13-18: Mostramos los resultados. Nota el uso de `f"..."` (f-strings) para insertar variables dentro del texto.

7.5 Salida del programa

Si el usuario ingresa 10 y 3:

```
=== CALCULADORA SIMPLE ===

Ingresa el primer número: 10
Ingresa el segundo número: 3

=== RESULTADOS ===
Suma: 10.0 + 3.0 = 13.0
Resta: 10.0 - 3.0 = 7.0
Multiplicación: 10.0 * 3.0 = 30.0
División: 10.0 / 3.0 = 3.3333333333333335
```

7.6 Mejora: Redondear resultados

El resultado de la división tiene muchos decimales. Podemos redondear usando `round()`:

```
# PASO 1: Pedir números al usuario
print("=== CALCULADORA SIMPLE ===")
print()

numero1 = float(input("Ingresa el primer número: "))
numero2 = float(input("Ingresa el segundo número: "))

# PASO 2: Calcular las operaciones
suma = numero1 + numero2
resta = numero1 - numero2
multiplicacion = numero1 * numero2
```

```
division = round(numero1 / numero2, 2)  # Redondear a 2 decimales

# PASO 3: Mostrar resultados
print()
print("=== RESULTADOS ===")
print(f"Suma: {numero1} + {numero2} = {suma}")
print(f"Resta: {numero1} - {numero2} = {resta}")
print(f"Multiplicación: {numero1} * {numero2} = {multiplicacion}")
print(f"División: {numero1} / {numero2} = {division}")
```

Nueva salida (más limpia):

```
=== CALCULADORA SIMPLE ===

Ingresa el primer número: 10
Ingresa el segundo número: 3

=== RESULTADOS ===
Suma: 10.0 + 3.0 = 13.0
Resta: 10.0 - 3.0 = 7.0
Multiplicación: 10.0 * 3.0 = 30.0
División: 10.0 / 3.0 = 3.33
```

8. Ejercicio de evaluación

8.1 Enunciado

Crea un programa en Python que:

1. Pida al usuario su **nombre completo** (nombre y apellido por separado).
2. Pida su **edad actual**.
3. Muestre un mensaje personalizado que diga:
   - Su nombre completo.
   - La edad que tendrá el próximo año.
   - Una línea adicional que muestre cuántos años le faltan para cumplir 60 años.

Ejemplo de salida esperada (si el usuario ingresa "Juan García" con 25 años):

```
=== BIENVENIDA ===

Ingresa tu nombre: Juan
Ingresa tu apellido: García
Ingresa tu edad: 25

=== INFORMACIÓN PERSONAL ===
```

```
Nombre completo: Juan García
Edad actual: 25 años
El próximo año tendrás: 26 años
Años para cumplir 60: 35 años
```

8.2 Pistas para resolver

- Necesitarás 4 variables para guardar: nombre, apellido, edad actual.
- Deberás concatenar (unir) nombre y apellido.
- Necesitarás hacer cálculos: edad + 1, y 60 - edad.
- Usa `print()` y f-strings para mostrar los resultados de forma clara.

9. Solución del ejercicio de evaluación

```
# PASO 1: Pedir información al usuario
print("=== BIENVENIDA ===")
print()

nombre = input("Ingresa tu nombre: ")
apellido = input("Ingresa tu apellido: ")
edad = int(input("Ingresa tu edad: "))

# PASO 2: Realizar cálculos
nombre_completo = nombre + " " + apellido
edad_proximo_ano = edad + 1
anos_para_60 = 60 - edad

# PASO 3: Mostrar resultados
print()
print("=== INFORMACIÓN PERSONAL ===")
print(f"Nombre completo: {nombre_completo}")
print(f"Edad actual: {edad} años")
print(f"El próximo año tendrás: {edad_proximo_ano} años")
print(f"Años para cumplir 60: {anos_para_60} años")
```

9.1 Explicación de la solución

- **Línea 4-7**: Pedimos la información usando `input()`. Para la edad usamos `int()` para convertir a número.
- **Línea 9-11**: Guardamos los resultados de las operaciones:
  - Concatenamos nombre y apellido con un espacio.
  - Calculamos la edad del próximo año sumando 1.
  - Calculamos los años faltantes restando la edad actual de 60.

- **Línea 13-18**: Mostramos los resultados de forma clara y organizada.

9.2 Alternativa: Usando f-strings más avanzados

Si quieres una versión más compacta sin tantas variables intermedias:

```
print("=== BIENVENIDA ===")
print()

nombre = input("Ingresa tu nombre: ")
apellido = input("Ingresa tu apellido: ")
edad = int(input("Ingresa tu edad: "))

print()
print("=== INFORMACIÓN PERSONAL ===")
print(f"Nombre completo: {nombre} {apellido}")
print(f"Edad actual: {edad} años")
print(f"El próximo año tendrás: {edad + 1} años")
print(f"Años para cumplir 60: {60 - edad} años")
```

Esta versión hace lo mismo pero sin variables intermedias; los cálculos se hacen directamente en el f-string.

10. Ejercicios adicionales para practicar

Si quieres practicar más los conceptos de este PDF, intenta resolver estos ejercicios adicionales:

10.1 Ejercicio: Conversión de monedas

Crea un programa que:

- Pida un cantidad de dinero en dólares.
- Pida la tasa de cambio actual (cuántos pesos son 1 dólar).
- Muestre cuántos pesos son esa cantidad de dólares.

Pista: Necesitarás multiplicar.

10.2 Ejercicio: Área y perímetro de un rectángulo

Crea un programa que:

- Pida el ancho y alto de un rectángulo.
- Calcule el área (ancho × alto).
- Calcule el perímetro (2 × ancho + 2 × alto).

- Muestre ambos resultados.

Pista: Usa multiplicación y suma.

10.3 Ejercicio: Promedio de tres calificaciones

Crea un programa que:

- Pida tres calificaciones del estudiante.
- Calcule el promedio (suma / 3).
- Muestre el promedio con 2 decimales.

Pista: Usa `round()` para redondear.

11. Resumen de conceptos clave

Concepto	Descripción	Ejemplo
Variable	Contenedor que guarda un valor	`nombre = "Carlos"`
print()	Muestra texto en pantalla	`print("Hola")`
input()	Recibe datos del usuario	`edad = input("Edad: ")`
int	Número entero	`25` , `-10` , `1000`
float	Número decimal	`3.14` , `-2.5` , `19.99`
str	Cadena de texto	`"Hola"` , `'Mundo'`
bool	Verdadero o falso	`True` , `False`
Comentario	Nota que Python ignora	`# Esto es un comentario`
Conversión	Cambiar tipo de dato	`int("25")` , `str(42)`
F-string	Insertar variables en texto	`f"Hola {nombre}"`

12. Preguntas frecuentes

P: ¿Cuál es la diferencia entre `int` y `float` ?

R: `int` son números sin decimales (25, -10), mientras que `float` tienen decimales (3.14, -2.5). La división `/` siempre devuelve `float` .

P: ¿Qué pasa si escribo `print(10 + "5")` ?

R: Python dará un error porque no se pueden sumar un número y un texto. Necesitas convertir: `print(10 + int("5"))` .

P: ¿Pueden tener dos variables el mismo nombre?

R: No. Si creas una segunda variable con el mismo nombre, sobrescribirá la primera.

P: ¿Los comentarios ocupan espacio en el programa final?

R: No. Los comentarios se ignoran cuando el programa se ejecuta y no afectan el tamaño del programa.

P: ¿Es mejor usar `"texto"` o `'texto'` ?

R: Python los trata igual. Elige uno y sé consistente. La convención es usar comillas dobles.

Conclusión

Felicidades por completar el PDF 1. Ahora entiendes:

- Cómo ejecutar código Python.
- Cómo usar `print()` e `input()` .
- Qué son las variables y cómo usarlas.
- Los tipos de datos básicos: `int` , `float` , `str` y `bool` .
- Cómo convertir entre tipos de datos.
- Cómo crear un programa simple pero completo.

En el siguiente PDF (Condicionales), aprenderás a tomar decisiones en tu código usando `if` , `elif` y `else` .