«Talento Tech»

Automation Testing

Clase 2

Buenos Aires Ciudad

Clase N°2 | Fundamentos de Python - Parte 1

Temario:

- Variables y tipos de datos simples
- Operador de asignación
- Operadores aritméticos, relacionales y lógicos
- Entrada/Salida básica (print(), input())
- Conversión entre tipos de datos simples
- Programas con entrada, procesamiento y salida de datos
- Comentarios en el código
- Estructuras de control:
    - Condicionales (if, elif, else)
    - Bucles (for, while)
    - Contadores y acumuladores
    - Sentencias break y continue
- Calculadora básica en Python (programación lineal)
- Ejercicios prácticos básicos

Objetivos de la clase:

En esta clase vamos a profundizar en los fundamentos de Python, esenciales para automatizar pruebas efectivamente. Exploraremos detalladamente variables, tipos de datos, operaciones básicas, estructuras condicionales, ciclos, y construiremos una calculadora básica mediante programación lineal.

Python.

Python es uno de los lenguajes más populares para la automatización debido a su simplicidad y claridad. Su sintaxis se lee casi como inglés, lo que facilita aprender programación sin distraerse con símbolos complejos. Además, cuenta con una enorme comunidad: si te atascas, es muy probable que alguien ya haya hecho la misma pregunta y exista una solución documentada.

Python viene con una “batería” de librerías estándar que resuelven tareas cotidianas (archivos, fechas, estadísticas) sin instalar nada extra. Cuando necesitas algo más avanzado, como automatizar un navegador o consumir una API, basta con agregar paquetes como **selenium** o **requests** en un solo comando `pip install`. Por todo esto, Python se ha convertido en el “idioma común” entre testers, desarrolladores y científicos de datos, lo que facilita la colaboración dentro de equipos multidisciplinarios.

Repasemos desde cero

Variables y tipos de datos: Las variables son "contenedores" para almacenar datos. Python soporta varios tipos de datos como números enteros (int), decimales (float), cadenas de texto (str), booleanos (bool).

```
nombre = "Matías"
edad = 30
altura = 1.75
es_automation_tester = True
```

Operador de asignación: Se utiliza para almacenar valores en variables:

```
x = 10
y = "Automation Tester”
```

Operadores Aritméticos: Permiten hacer cálculos matemáticos básicos.

```
suma = 5 + 3        # Resultado: 8
resta = 10 - 4      # Resultado: 6
multiplicacion = 2 * 6  # Resultado: 12
division = 8 / 2    # Resultado: 4.0
modulo = 9 % 2      # Resultado: 1
potencia = 3 ** 2   # Resultado: 9
```

Operadores Relacionales: Comparan dos valores y devuelven True o False.

```
mayor = 10 > 5          # True
menor_igual = 3 <= 3    # True
```

```
igual = 7 == 7          # True
distinto = 5 != 3       # True
```

Operadores Lógicos: Combinan condiciones.

```
cond1 = True
cond2 = False
resultado_and = cond1 and cond2    # False
resultado_or = cond1 or cond2      # True
resultado_not = not cond1          # False
```

Entrada/Salida básica.

Interacción con usuarios usando funciones:

Aquí verás que ejecutando este código se le pedirá el nombre de usuario. Luego, ese nombre se almacenará en la variable nombre y por último, por medio de la función print, se mostrará ese valor de la variable nombre por consola.


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

print(f"Hola, {nombre}")
```

Conversión de tipos:

Convertir entre tipos de datos simples: Aquí lo que hacemos es una conversión de tipos. La función input solicita la edad, pero la almacena como un String. Como en este caso nosotros queremos un número, convertimos ese String a un número entero por medio de la función int.

```
edad = int(input("Ingresa tu edad: "))    # Convertimos a entero

altura = float(input("Ingresa tu altura: "))  # Convertimos a float
```

Estructuras de control:

Las estructuras de control nos permiten decidir qué instrucciones se ejecutan y cuántas veces, según condiciones definidas.

Condicionales:

Permiten ejecutar diferentes bloques de código dependiendo de si se cumple o no una condición.

```
if edad >= 18:
   print("Puedes trabajar en TalentoLab")
elif edad >= 16:
   print("Podrías realizar una pasantía")
else:
   print("Eres demasiado joven para trabajar aquí")
```

Bucles:

Sirven para repetir instrucciones múltiples veces.

for: Se utiliza cuando conocemos la cantidad de repeticiones.

```
for i in range(5):
   print(i)
```

while: Se utiliza cuando no sabemos cuántas veces se repetirá el ciclo; depende de una condición.

```
contador = 0
while contador < 5:
   print(contador)
   contador += 1
```

Diferencia entre for y while:

- `for` se usa para iteraciones controladas (por ejemplo, recorrer una lista o repetir algo una cantidad conocida de veces).
- `while` se usa cuando la repetición depende de una condición que puede cambiar dinámicamente.

Contadores y acumuladores:

Variables utilizadas para contar o acumular valores durante una iteración.

```
suma = 0
for i in range(1, 11):
   suma += i
print(f"La suma es: {suma}")
```

Sentencias break y continue:

- `break` termina el bucle actual.
- `continue` salta a la siguiente iteración del bucle.

```
for i in range(10):
   if i == 5:
       break  # Termina el bucle al llegar a 5
   print(i)

for i in range(5):
   if i == 2:
       continue  # Salta la iteración donde i es 2
   print(i)
```

Ejemplo: Calculadora básica en Python

Hagamos uso de todo lo visto y repasado en las clases anteriores para construir una calculadora simple. Utilizaremos entrada por teclado, operadores aritméticos y estructuras condicionales para que el programa pueda realizar operaciones básicas como suma, resta, multiplicación y división.
Este ejercicio nos permitirá reforzar los fundamentos de programación lineal en Python, sin necesidad de definir funciones por ahora.

```
num1 = float(input("Introduce el primer número: "))
num2 = float(input("Introduce el segundo número: "))
operacion = input("Introduce la operación (+, -, *, /): ")

if operacion == '+':
    resultado = num1 + num2
elif operacion == '-':
    resultado = num1 - num2
elif operacion == '*':
    resultado = num1 * num2
elif operacion == '/':
    resultado = num1 / num2
else:
    resultado = "Operación inválida"
print(f"El resultado es: {resultado}")
```

💡 **Nota:** En la siguiente clase haremos la misma calculadora pero con funciones. Y luego utilizaremos esa calculadora para realizar tests automáticos.

Preguntas para Reflexionar

- ¿Por qué es importante dominar estructuras básicas como condicionales y ciclos en Automation Testing?
- ¿De qué manera consideras que se aplicaría lo aprendido hoy en tu rol como Automation Tester?
- ¿Qué dificultades encontraste al escribir tus primeros scripts en Python y cómo las resolviste?

Próximos Pasos

En la siguiente clase profundizaremos aún más en Python, explorando estructuras avanzadas, funciones, manejo de excepciones y comenzaremos a aplicar estos conocimientos en casos prácticos específicos de automatización de pruebas.

🗨️**Storytelling**

¡Trabajando en Talento Lab!

Tras configurar tu entorno básico, Matías, tu mentor en el equipo, te introduce al proyecto que llevarás adelante durante las siguientes semanas:

Silvia (Product Owner) comenta:

"Estamos desarrollando un módulo crítico de gestión de perfiles internos. Necesitamos scripts robustos que automatizarán pruebas recurrentes, asegurando así la calidad continua del sistema."

Matías añade:

"Tu primera tarea real será practicar con Python básico. Esto te preparará para escribir scripts más avanzados próximamente."

Ejercicios Prácticos

Actividad 1:

- Define variables para almacenar el nombre, edad y profesión del usuario.
- Solicita estos datos por teclado utilizando `input()`.
- Imprime en pantalla un mensaje personalizado incluyendo todos estos datos.

Actividad 2:

- Escribe un pequeño script que utilice un bucle para mostrar los primeros 10 números pares.
- Usa condicionales para validar y mostrar sólo los números pares.


Buenos Aires
aprende
Agencia de Habilidades para el Futuro
BA
Buenos
Aires
Ciudad