Análisi teórico

Python desde cero, fundamentos claros

Unidad seleccionada: Estructura de control (condiciones y bucles)

Las estructuras de control son esenciales en cualquier lenguaje de programación. En python, estas estructuras permiten alterar la secuencia usual de ejecución de un programa, lo que hace posible que ciertas condiciones o que se repitan mientras se mantenga un criterio específico. Esta sección es importante porque presenta la lógica necesaria para abordar problemas del mundo real de forma flexible y eficaz.

Las estructuras condicionales facilitan la toma de decisiones en el codigo. Python usa principalmente la declaración if, que puede estar acompañada de opcionalmente por elif y else.

```
if Condition: # type: ignore
# bloque de código
elif another_condition: # type: ignore
# bloque alternativo
else: # type: ignore
# bloque final
```

Características principales:

- Las condiciones se consideran verdaderas o falsas.
- Python no emplea llaves {}, sino que requiere indentación, lo que facilita la lectura del código.
- Es posible unir condiciones mediante operadores lógicos como and, or, not.

Ejemplo:

```
edad = 18
if edad >= 18:
print("Mayor de edad")
else:
print("Menor de edad")
```

Los bucles permiten ejecutar un bloque de código varias veces. Python ofrece dos estructuras principales while y for.

El bucle while repite un bloque de código mientras una condición sea verdadera. Este tien ventajas y desventajas, es útil cuando anteriormente no se conoce cuantas veces se debe repetir el ciclo, pero si la condición nunca se vuelve falsa, se genera un bucle infinito.

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

El bucle for se utiliza para iterar sobre secuencias como listas, tuplas, cadenas o rangos. Las ventajas son que es más seguro que el bucle while en la mayoría de los casos y es ideal para recorrer estructuras de datos.

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

Bucles:

```
for i in range(1, 11, 2): # inicio, fin, paso
print(i)

while True:
if condicion:
break
```

Palabras claves útiles:

Palabra clave	Utilidad
break	Sale del bucle inmediatamente
continue	Salta a la siguiente iteración
pass	No hace nada, útil como marcador
else (en bucles)	Se ejecuta si el bucle termina sin break

```
for n in range(5):
if n == 3:
break
else:
print("Terminó sin interrupciones")
```

El Zen de Python

Principio seleccionado: “Simple is better than complex”

Este concepto del Zen de Python, elaborado por Tim Peters, enfatiza la relevancia de la sencillez en la estructura del código. Python fue diseñado con el principio de que el código debe ser comprensible y accesible, incluso para aquellos que no lo han programado.

La sencillez no implica escasez de funcionalidades, sino que se trata de optar por soluciones evidentes y lógicas en vez de enfoques innecesariamente enrevesados. Un código sencillo minimiza fallos, simplifica el mantenimiento y favorece la colaboración.
Ejemplo de mal uso:

```
resultado = True if x > 10 else False
```

Ejemplo de buen uso:

```
resultado = x > 10
```

Conclusión:
Para resumir, implementar este principio ayuda a crear códigos más comprensibles, que se ajustan a la filosofía de Python y son más sencillos de mantener en el tiempo.