Clase 006 — Python: tipos, estructuras, control de flujo

Parte: 0 — Prerrequisitos · Fuente: Python Tutorial oficial (caps. 3-5) · Fluent Python (Ramalho, 2ª ed.) cap. 1. · Duración estimada: 120 min.

Clase 006 — Python: tipos, estructuras, control de flujo

> *Parte: 0 — Prerrequisitos · Fuente: Python Tutorial oficial (caps. 3-5) · Fluent Python (Ramalho, 2ª ed.) cap. 1. · Duración estimada: 120 min.*

Objetivo

Refrescar (o instalar) los cimientos de Python que el resto del programa asume: tipos primitivos, las 4 estructuras built-in (list, tuple, set, dict), control de flujo (if/for/while), unpacking, truthiness y la diferencia entre mutables e inmutables — la fuente del 90% de bugs sutiles.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Diferenciar tipos mutables (list, dict, set) vs inmutables (tuple, str, int, frozenset) y predecir el efecto en asignaciones.
2. Usar las 4 estructuras eligiendo bien: list (orden + duplicados), tuple (inmutable, rápida), set (unicidad), dict (lookup O(1)).
3. Aplicar unpacking en for, returns múltiples y args/*kwargs.
4. Evaluar truthiness correctamente ([], {}, 0, '', None son falsy; el resto es truthy).
5. Identificar el bug del default mutable en funciones (def f(x, lst=[])) y por qué es trampa.

Temas

#	Tema	Por qué importa
1	Mutables vs inmutables	Define qué pasa con a = b.
2	list, tuple, set, dict — cuándo cada uno	Complejidad y semántica distintas.
3	Iteración: for, enumerate, zip	Idiomático > C-style.
4	Unpacking y starred expressions	a, *b, c = [1,2,3,4,5].
5	Truthiness y operadores and/or	Evalúan al objeto, no al booleano.
6	Default mutables: el clásico	def f(x, lst=[]) comparte la lista entre l

Definiciones y características

Mutable

: Objeto cuyo estado puede modificarse después de crearlo (list, dict, set, bytearray). Consecuencia: si dos variables apuntan al mismo objeto mutable, cambiar una cambia la otra (alias).

Inmutable

: Objeto que NO se puede modificar; cualquier 'modificación' produce un objeto nuevo (int, float, str, tuple, frozenset). Característica clave: son hashables y pueden ser claves de dict o miembros de set.

Hashable

: Objeto con __hash__ definido y constante durante su vida. Los inmutables built-in son hashables; las listas y

dicts NO lo son. Es lo que permite el O(1) lookup de set/dict.

Unpacking

: Sintaxis para asignar múltiples variables desde un iterable (a, b = (1, 2) o a, resto = [1, 2, 3, 4]). El captura el resto en una lista. Funciona en for, return, llamadas a funciones (f(lista), g(*dict)).

Truthy / Falsy

: Cómo evalúa Python un objeto en contexto booleano (if x:). Falsy: False, None, 0, 0.0, '', [], {}, set(). Truthy: todo lo demás (incluso ' ' con espacio, [0], {None: None}).

Dataset / recursos

Datos sintéticos pequeños generados en el notebook (lista de diccionarios simulando estudiantes). No requiere descarga.

Ejercicios

1. Cuenta palabras. Dado un texto, devuelve un dict[str, int] con frecuencias. Sin usar Counter.

2. Unique con orden. Recibe list[int], devuelve la lista de únicos manteniendo el orden de primera aparición.

3. Reproduce el bug del default mutable. Escribe def add(item, target=[]), llámala 3 veces con add('x'). Observa. Explica por qué y arregla.

4. Top-K palabras. Mismo texto del ejercicio 1, devuelve las 5 más frecuentes ordenadas por frecuencia descendente.

5. Grupos por inicial. Dado list[str], devuelve dict[str, list[str]] agrupando por primera letra (case-insensitive).

Homework verificable

Notebook homework.ipynb con las 5 funciones de los ejercicios, cada una con: (a) implementación, (b) 3 casos de prueba (incluyendo edge cases — lista vacía, string vacío), (c) docstring corto explicando complejidad.

Criterio de aceptación: Las 5 funciones pasan sus casos de prueba; los edge cases manejados sin excepción.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
TypeError: unhashable type: 'list'	Intentaste usar una lista como clave de di
Modifiqué una lista y otra variable tambié	Las dos apuntan al mismo objeto (alias). F
Función con default =[] comparte la lista	El default se evalúa una vez al definir la
KeyError al acceder a dict que no tiene la	d['x'] lanza KeyError si no existe. Fix: u
if items != None: en vez de if items is no	Para singletons (None, True, False) usa si

Preguntas frecuentes

¿Cuándo tupla y cuándo lista?

Tupla para records heterogéneos de tamaño fijo (punto = (3.5, 7.1), (nombre, edad)). Lista para colecciones homogéneas que crecen (temperaturas = [22.1, 22.5, ...]). Tupla además sirve como clave de dict; lista no.

¿Set o dict para chequear pertenencia?

Ambos son O(1). Set si solo necesitas saber si está. Dict si además necesitas asociar valor. Una list para esto es O(n) — usa set/dict si haces if x in coleccion con N grande.

¿copy() me da una copia completa? ¿Y con listas anidadas?

.copy() es shallow: copia el contenedor pero comparte las referencias internas. Para anidamiento, import copy; copy.deepcopy(x) — más lento pero independiente.

¿Por qué 0.1 + 0.2 != 0.3?

Floats son IEEE 754 binarios; 0.1 y 0.2 no tienen representación exacta. Fix: para igualdad usa math.isclose(a, b). Para precisión decimal financiera, from decimal import Decimal.

¿Qué pasa con dict si insertas la misma clave dos veces?

El segundo valor sobrescribe al primero. El orden de inserción se preserva (Python 3.7+), así que list(d) da las keys en el orden en que se insertaron por primera vez.

Referencias

- Python Tutorial — Data Structures
- Ramalho, Fluent Python 2e — cap. 1 The Python Data Model.
- Python Tutorial — Control flow

Siguiente clase

Clase 007 — Comprehensions y generadores

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
import sys
from typing import Iterable
print('python:', sys.version.split()[0])
```

Archivos complementarios

- notebook.ipynb