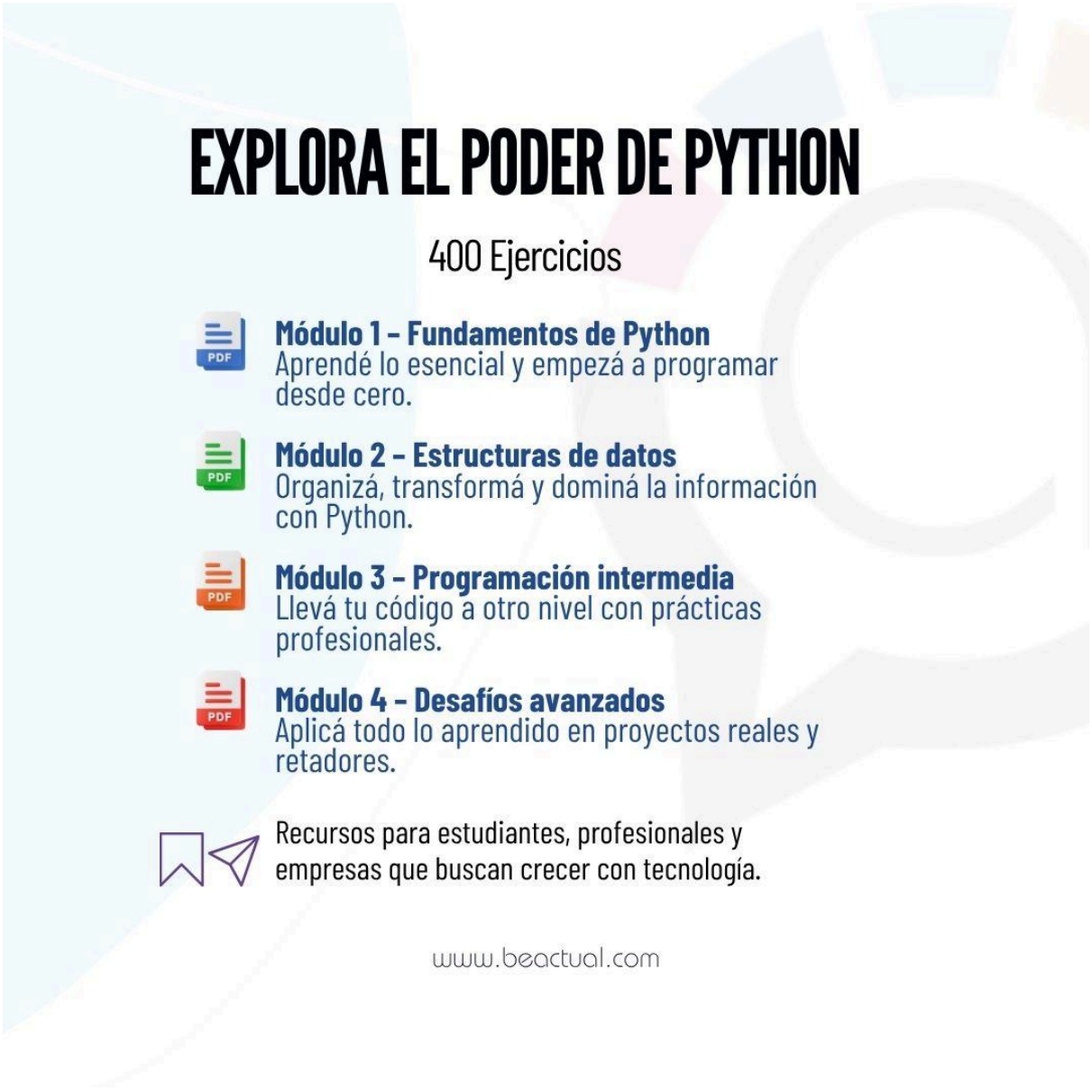
Aprendé Python desde cero con esta guía de ejercicios prácticos.

Aprendé Python paso a paso con una guía práctica y optimizada. Esta recopilación incluye 4 módulos con 400 ejercicios resueltos, desde nivel inicial hasta avanzado, diseñados para que practiques de forma progresiva y entiendas cada concepto con claridad.

A lo largo de la guía vas a encontrar:

- Ejercicios básicos para dominar la sintaxis y la lógica de programación.
- Problemas intermedios con funciones, listas, diccionarios y estructuras de datos.
- Desafíos avanzados en recursividad, programación orientada a objetos, grafos y algoritmos.
- Aplicaciones prácticas en ciencia de datos, machine learning e inteligencia artificial.
- Cada ejercicio está acompañado de su solución explicada, lo que te permitirá no solo resolverlo, sino también comprender el razonamiento detrás del código.
- Al finalizar esta guía, estarás en condiciones de dar el salto hacia proyectos más complejos en inteligencia artificial, análisis de datos y desarrollo de software profesional.



Optimizamos este material para que sea una herramienta clara, útil y accesible, pensada para estudiantes, programadores y profesionales que buscan crecer en el mundo de la tecnología.

Al terminar, vas a estar listo para empezar con Inteligencia Artificial y sus aplicaciones.Ideal para programadores, estudiantes y perfiles técnicos que quieren crecer en tecnología.

Módulo 1 – Fundamentos de Python (Ejercicios 1–100)
Entrada/salida, condicionales, bucles, funciones básicas.

En este módulo vas a descubrir la verdadera potencia de Python, sus estructuras de datos. A través de ejercicios con listas, tuplas, diccionarios y conjuntos, aprenderás a organizar, transformar y analizar información de manera eficiente.

Estos 100 ejercicios te ayudarán para enfrentar desafíos de programación más avanzados, como la manipulación de datos en ciencia de datos o la gestión de información en aplicaciones web. 👉 Perfecto para quienes ya dominan lo básico y quieren dar el siguiente paso.

Ejercicio 1 – Saludo personalizado
Enunciado: Hacer un programa que pida un nombre por teclado y muestre: “Hola [nombre]”.

```
nombre = input("Ingresá tu nombre: ")
print("Hola", nombre)
```

Ejercicio 2 – Operaciones básicas
Enunciado: Solicitar dos números y mostrar la suma, resta, multiplicación y división.

```
a = float(input("Ingresá el primer número: "))
b = float(input("Ingresá el segundo número: "))

print("Suma:", a + b)
print("Resta:", a - b)
print("Multiplicación:", a * b)
print("División:", a / b if b != 0 else "No se puede dividir por cero")
```

Ejercicio 3 – Mayor de edad
Enunciado: Pedir una edad y determinar si es mayor de edad.

```
edad = int(input("Ingresá tu edad: "))
if edad >= 18:
    print("Es mayor de edad")
else:
    print("Es menor de edad")
```

Ejercicio 4 – Condicional simple
Enunciado: Solicitar una edad e imprimir “Es mayor” si es mayor de edad, de lo contrario “Es menor”.

```
edad = int(input("Ingresá tu edad: "))
print("Es mayor" if edad >= 18 else "Es menor")
```

Ejercicio 5 – Semáforo
Enunciado: Pedir un color y mostrar el mensaje correspondiente: verde = “Puede pasar”, amarillo = “Precaución”, rojo = “No pasar”.

```
color = input("Ingresá un color (rojo, amarillo, verde): ").lower()

if color == "verde":
    print("Puede pasar")
elif color == "amarillo":
    print("Precaución")
elif color == "rojo":
    print("No pasar")
else:
    print("Color inválido")
```

Ejercicio 6 – Contar del 1 al 100
Enunciado: Mostrar los números del 1 al 100.

```
for i in range(1, 101):
    print(i)
```

Ejercicio 7 – Números pares del 1 al 100
Enunciado: Mostrar solo los números pares del 1 al 100.

```
for i in range(2, 101, 2):
    print(i)
```

Ejercicio 8 – Números divisibles por 2 y 5
Enunciado: Mostrar los números del 1 al 100 que sean divisibles por 2 y por 5.

```
for i in range(1, 101):
    if i % 2 == 0 and i % 5 == 0:
        print(i)
```

Ejercicio 9 – Tablas de multiplicar (2 al 9)
Enunciado: Mostrar las tablas de multiplicar del 2 al 9.

```
for i in range(2, 10):
    print(f"Tabla del {i}")
    for j in range(1, 10):
        print(f"{i} x {j} = {i*j}")
    print("-" * 10)
```

Ejercicio 10 – Dibujo con asteriscos (cuadro)
Enunciado: Mostrar un cuadrado con asteriscos.

```
print("*****")
print("*   *")
print("*   *")
print("*****")
```

Ejercicio 11 – Dibujo con asteriscos (línea vertical)
Enunciado: Mostrar una línea vertical de asteriscos.

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

Ejercicio 12 – Dibujo con asteriscos (triángulo rectángulo)
Enunciado: Mostrar un triángulo rectángulo con asteriscos.

```
for i in range(1, 6):
    print("*" * i)
```

Ejercicio 13 – Dibujo con asteriscos (triángulo invertido)
Enunciado: Mostrar un triángulo invertido con asteriscos.

```
for i in range(5, 0, -1):
    print("*" * i)
```

Ejercicio 14 – Función: texto centrado
Enunciado: Crear una función que reciba un texto y lo muestre centrado en 80 columnas.

```
def escribir_centrado(texto):
    print(texto.center(80, "="))

escribir_centrado("Bienvenido a Python")
```

Ejercicio 15 – Función: múltiplos
Enunciado: Crear una función que reciba dos números y diga si uno es múltiplo del otro.

```
def es_multiplo(a, b):
    return a % b == 0 or b % a == 0

print(es_multiplo(10, 5))  # True
```

Ejercicio 16 – Temperatura media
Enunciado: Crear una función que calcule la temperatura media de un día a partir de la máxima y mínima.

```
def temperatura_media(maxima, minima):
    return (maxima + minima) / 2

print(temperatura_media(30, 18))
```

Ejercicio 17 – Función: espaciado de texto
Enunciado: Crear una función que reciba un texto y devuelva cada letra separada por un espacio.

```
def convertir_espaciado(texto):
    return " ".join(texto)
```

```
print(convertir_espaciado("Hola"))
```

Ejercicio 18 – Máximo y mínimo en lista
Enunciado: Crear una función que reciba una lista de números y devuelva el máximo y el mínimo.

```
def calcular_max_min(lista):
    return max(lista), min(lista)

print(calcular_max_min([3, 7, 1, 9]))
```

Ejercicio 19 – Área y perímetro de circunferencia
Enunciado: Crear una función que calcule el área y perímetro de una circunferencia dado su radio.

```
import math

def circunferencia(radio):
    area = math.pi * radio**2
    perimetro = 2 * math.pi * radio
    return area, perimetro

print(circunferencia(5))
```

Ejercicio 20 – Login simple
Enunciado: Crear una función que valide usuario y contraseña con 3 intentos.

```
def login(usuario, clave):
    return usuario == "usuario1" and clave == "asdasd"

intentos = 0
while intentos < 3:
    u = input("Usuario: ")
    c = input("Contraseña: ")
    if login(u, c):
        print("Acceso concedido")
        break
    else:
        print("Datos incorrectos")
        intentos += 1
```

Ejercicio 21 – Factorial con recursividad
Enunciado: Crear una función recursiva que calcule el factorial de un número.

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    return n * factorial(n - 1)

print(factorial(5))  # 120
```

Ejercicio 22 – Máximo Común Divisor (MCD)
Enunciado: Crear una función que calcule el MCD de dos números usando el método de Euclides.

```
def mcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

print(mcd(48, 18))  # 6
```

Ejercicio 23 – Conversión de tiempo
Enunciado: Escribir dos funciones:

1. Convertir horas, minutos y segundos a segundos.
2. Convertir segundos a horas, minutos y segundos.

def a_segundos(h, m, s): return h*3600* + *m*60 + s

def a_horas(segundos): h = segundos // 3600 m = (segundos % 3600) // 60 s = segundos % 60 return h, m, s

print(a_segundos(1, 30, 15)) # 5415 print(a_horas(5415)) # (1, 30, 15)

Ejercicio 24 – Día juliano
Enunciado: Crear un programa que calcule el día juliano de una fecha dada.

```
import datetime

def dia_juliano(dia, mes, anio):
    fecha = datetime.date(anio, mes, dia)
    return fecha.timetuple().tm_yday

print(dia_juliano(20, 10, 2025))
```

Ejercicio 25 – Validación de fecha
Enunciado: Crear una función que valide si una fecha ingresada es correcta.

```
import datetime

def validar_fecha(dia, mes, anio):
    try:
        datetime.date(anio, mes, dia)
        return True
    except ValueError:
        return False

print(validar_fecha(31, 2, 2023))  # False
```

Ejercicio 26 – Operaciones con fracciones
Enunciado: Crear un programa que trabaje con fracciones representadas como numerador y denominador.

```
from fractions import Fraction

f1 = Fraction(1, 2)
f2 = Fraction(3, 4)
print(f1 + f2)  # 5/4
```

Ejercicio 27 – Función max()
Enunciado: Definir una función que devuelva el mayor de dos números.

```
def maximo(a, b):
    return a if a > b else b

print(maximo(10, 7))  # 10
```

Ejercicio 28 – Función max_de_tres()
Enunciado: Definir una función que devuelva el mayor de tres números.

```
def max_de_tres(a, b, c):
    return max(a, b, c)

print(max_de_tres(5, 9, 2))  # 9
```

Ejercicio 29 – Longitud de lista o cadena
Enunciado: Definir una función que calcule la longitud de una lista o cadena.

```
def longitud(objeto):
    contador = 0
    for _ in objeto:
        contador += 1
    return contador

print(longitud("Python"))  # 6
```

Ejercicio 30 – Vocales
Enunciado: Definir una función que indique si un carácter es vocal.

```
def es_vocal(c):
    return c.lower() in "aeiou"

print(es_vocal("a"))  # True
```

Ejercicio 31 – Suma y multiplicación de lista
Enunciado: Definir funciones que sumen y multipliquen todos los números de una lista.

```
def suma(lista):
    total = 0
    for n in lista:
        total += n
    return total

def multip(lista):
    total = 1
```

```
    for n in lista:
        total *= n
    return total

print(suma([1,2,3,4]))   # 10
print(multip([1,2,3,4])) # 24
```

Ejercicio 32 – Inversa de cadena
Enunciado: Definir una función que devuelva la cadena invertida.

```
def inversa(cadena):
    return cadena[::-1]

print(inversa("Python"))  # nohtyP
```

Ejercicio 33 – Palíndromo
Enunciado: Definir una función que reconozca palíndromos.

```
def es_palindromo(cadena):
    cadena = cadena.lower().replace(" ", "")
    return cadena == cadena[::-1]

print(es_palindromo("radar"))  # True
```

Ejercicio 34 – Superposición de listas
Enunciado: Definir una función que indique si dos listas tienen al menos un elemento en común.

```
def superposicion(lista1, lista2):
    for i in lista1:
        if i in lista2:
            return True
    return False

print(superposicion([1,2,3], [3,4,5]))  # True
```

Ejercicio 35 – Generar caracteres
Enunciado: Definir una función que repita un carácter n veces.

```
def generar_n_caracteres(n, caracter):
    return caracter * n

print(generar_n_caracteres(5, "x"))  # xxxxx
```

Ejercicio 36 – Histograma
Enunciado: Definir una función que imprima un histograma a partir de una lista de enteros.

```
def histograma(lista):
    for n in lista:
        print("*" * n)

histograma([4, 9, 7])
```

Ejercicio 37 – Operaciones con listas
Enunciado: Crear un programa que permita ingresar números en una lista y realizar operaciones (sumatoria, filtrado, conteo).

```
lista = []
while True:
    n = int(input("Ingresá un número (0 para salir): "))
    if n == 0:
        break
    lista.append(n)

print("Lista:", lista)
print("Suma:", sum(lista))
```

Ejercicio 38 – Pasajeros y destinos
Enunciado: Procesar datos de pasajeros y destinos usando listas de tuplas.

```
pasajeros = [("Manuel", 19823451, "Liverpool"), ("Silvana", 22709128, "Buenos Aires")]
for p in pasajeros:
    print(f"{p[0]} viaja a {p[2]}")
```

Ejercicio 39 – Alumnos de primaria y secundaria
Enunciado: Ingresar nombres de alumnos de primaria y secundaria y mostrar coincidencias.

```
primaria = {"Ana", "Luis", "Pedro"}
secundaria = {"Pedro", "María", "Ana"}

print("Nombres en común:", primaria & secundaria)
```

Ejercicio 40 – Compras de clientes
Enunciado: Procesar datos de compras de clientes y mostrar domicilios únicos.

```
ventas = [("Nuria", 5, 12780.78, "Calle Flores 355"),
        ("Jorge", 7, 699, "Mirasol 218"),
        ("Nuria", 7, 532.90, "Calle Flores 355")]

domicilios = {venta[3] for venta in ventas}
print("Domicilios únicos:", domicilios)
```

Ejercicio 41 – Contar ocurrencias de caracteres
Enunciado: Procesar 50 strings ingresados por el usuario e informar cuántas veces aparece cada carácter.

```
texto_total = ""
for i in range(50):
    texto_total += input("Ingresá un texto: ")

conteo = {}
for c in texto_total:
    conteo[c] = conteo.get(c, 0) + 1

print(conteo)
```

Ejercicio 42 – Gestión de socios de un club
Enunciado: Crear un programa que administre un diccionario de socios con número, nombre, fecha de ingreso y cuota al día.

```
socios = {
    1: {"nombre": "Amanda Núñez", "ingreso": "17/03/2009", "cuota": True},
    2: {"nombre": "Bárbara Molina", "ingreso": "17/03/2009", "cuota": True},
    3: {"nombre": "Lautaro Campos", "ingreso": "17/03/2009", "cuota": True}
}

print("Cantidad de socios:", len(socios))
```

Ejercicio 43 – Contar elementos en lista
Enunciado: Escribir una función que cuente cuántas veces aparece cada elemento en una lista.

```
def contar_elementos(lista):
    conteo = {}
    for elem in lista:
        conteo[elem] = conteo.get(elem, 0) + 1
    return conteo

print(contar_elementos([1,2,2,3,3,3]))
```

Ejercicio 44 – Promedio de calificaciones
Enunciado: Calcular el promedio de calificaciones de un diccionario de estudiantes.

```
notas = {"Ana": 8, "Luis": 6, "Pedro": 9}
promedio = sum(notas.values()) / len(notas)
print("Promedio:", promedio)
```

Ejercicio 45 – Filtrar aprobados
Enunciado: Filtrar un diccionario de estudiantes para obtener solo los que aprobaron.

```
notas = {"Ana": 8, "Luis": 4, "Pedro": 9}
aprobados = {k:v for k,v in notas.items() if v >= 6}
print(aprobados)
```

Ejercicio 46 – Unir diccionarios
Enunciado: Combinar dos diccionarios en uno solo.

```
d1 = {"a": 1, "b": 2}
d2 = {"c": 3, "d": 4}
d3 = {**d1, **d2}
print(d3)
```

Ejercicio 47 – Contar palabras en texto
Enunciado: Contar cuántas veces aparece cada palabra en un texto.

```
texto = "hola mundo hola python"
palabras = texto.split()
conteo = {}
for p in palabras:
    conteo[p] = conteo.get(p, 0) + 1
print(conteo)
```

Ejercicio 48 – Eliminar duplicados de lista
Enunciado: Eliminar duplicados de una lista y guardarlos en un diccionario.

```
lista = [1,2,2,3,4,4,5]
unicos = list(set(lista))
print(unicos)
```

Ejercicio 49 – Valor mínimo y máximo en diccionario
Enunciado: Encontrar el valor mínimo y máximo en un diccionario de números.

```
datos = {"a": 10, "b": 3, "c": 25}
print("Mínimo:", min(datos.values()))
print("Máximo:", max(datos.values()))
```

Ejercicio 50 – Eliminar claves por condición
Enunciado: Eliminar claves de un diccionario si su valor es mayor a un umbral.

```
datos = {"a": 10, "b": 3, "c": 25}
umbral = 15
filtrado = {k:v for k,v in datos.items() if v <= umbral}
print(filtrado)
```

Ejercicio 51 – Invertir diccionario
Enunciado: Crear una función que invierta claves y valores de un diccionario.

```
def invertir_diccionario(dic):
    return {v:k for k,v in dic.items()}
```

```
print(invertir_diccionario({"a":1, "b":2}))
```

Ejercicio 52 – Histograma de letras
Enunciado: Crear un histograma de letras a partir de una cadena.

```
texto = "python"
conteo = {}
for c in texto:
    conteo[c] = conteo.get(c, 0) + 1
print(conteo)
```

Ejercicio 53 – Diccionario anidado (tienda)
Enunciado: Crear un diccionario anidado con productos y precios.

```
tienda = {
    "manzana": {"precio": 100, "stock": 50},
    "banana": {"precio": 80, "stock": 30}
}
print(tienda)
```

Ejercicio 54 – Contar vocales
Enunciado: Contar cuántas vocales hay en una cadena.

```
texto = "programacion"
conteo = {v: texto.count(v) for v in "aeiou"}
print(conteo)
```

Ejercicio 55 – Ordenar diccionario
Enunciado: Ordenar un diccionario por sus claves.

```
datos = {"c":3, "a":1, "b":2}
ordenado = dict(sorted(datos.items()))
print(ordenado)
```

Ejercicio 56 – Verificar diccionario vacío
Enunciado: Verificar si un diccionario está vacío.

```
dic = {}
print("Vacío" if not dic else "No vacío")
```

Ejercicio 57 – Buscar clave en diccionario
Enunciado: Buscar una clave en un diccionario y devolver su valor.

```
datos = {"a":1, "b":2}
clave = "a"
print(datos.get(clave, "No encontrado"))
```

Ejercicio 58 – Eliminar clave de diccionario
Enunciado: Eliminar una clave específica de un diccionario si existe.

```
datos = {"a":1, "b":2}
datos.pop("a", None)
print(datos)
```

Ejercicio 59 – Diccionario de contactos
Enunciado: Crear un diccionario de contactos con nombres y teléfonos.

```
contactos = {"Ana": "12345", "Luis": "67890"}
print(contactos["Ana"])
```

Ejercicio 60 – Unir listas en diccionario
Enunciado: Convertir dos listas en un diccionario, una como claves y otra como valores.

```
claves = ["a", "b", "c"]
valores = [1, 2, 3]
dic = dict(zip(claves, valores))
print(dic)
```

Ejercicio 61 – Diccionario de frecuencias
Enunciado: Dada una lista de números, crear un diccionario que muestre cuántas veces aparece cada número.

```
lista = [5, 2, 5, 7, 2, 5]
```

```
frecuencias = {}
for n in lista:
    frecuencias[n] = frecuencias.get(n, 0) + 1
print(frecuencias)
```

Ejercicio 62 – Diccionario de traducción
Enunciado: Crear un diccionario que traduzca palabras de un idioma a otro.

```
traduccion = {"cat": "gato", "dog": "perro", "house": "casa"}
print(traduccion["dog"])
```

Ejercicio 63 – Registro de ventas
Enunciado: Crear un programa que registre ventas diarias de una tienda usando un diccionario.

```
ventas = {}
ventas["2025-10-20"] = [("pan", 2, 100), ("leche", 1, 200)]
print(ventas)
```

Ejercicio 64 – Puntuaciones de jugadores
Enunciado: Crear un diccionario que almacene jugadores y sus puntuaciones.

```
jugadores = {"Ana": [10, 20], "Luis": [15, 30]}
jugadores["Ana"].append(25)
print(jugadores)
```

Ejercicio 65 – Diccionario de estudiantes
Enunciado: Crear un diccionario que almacene cursos y calificaciones de estudiantes.

```
estudiantes = {
    "Ana": {"cursos": ["Python", "IA"], "notas": [9, 10]},
    "Luis": {"cursos": ["Python"], "notas": [7]}
}
print(estudiantes)
```

Ejercicio 66 – Inventario de tienda
Enunciado: Crear un programa que administre el inventario de una tienda.

```
inventario = {"manzana": {"precio": 100, "stock": 50}}
inventario["banana"] = {"precio": 80, "stock": 30}
print(inventario)
```

Ejercicio 67 – Registro de películas
Enunciado: Crear un diccionario que contenga información de películas.

```
peliculas = {
    "Matrix": {"año": 1999, "actores": ["Keanu Reeves", "Carrie-Anne Moss"]}
}
print(peliculas)
```

Ejercicio 68 – Recetas de cocina
Enunciado: Crear un diccionario que almacene recetas con ingredientes y pasos.

```
recetas = {
    "Tarta": {"ingredientes": ["harina", "huevo", "leche"], "pasos": "Mezclar y hornear"}
}
print(recetas)
```

Ejercicio 69 – Directorio telefónico
Enunciado: Crear un directorio telefónico con nombres, teléfonos y correos.

```
directorio = {"Ana": ("12345", "ana@mail.com")}
print(directorio["Ana"])
```

Ejercicio 70 – Registro de compras
Enunciado: Crear un programa que registre compras de clientes en una tienda online.

```
compras = {"cliente1": [("producto1", 2), ("producto2", 1)]}
print(compras)
```

Ejercicio 71 – Registro de viajes
Enunciado: Crear un diccionario que almacene información de viajes.

```
viajes = {"viaje1": {"destino": "Madrid", "lugares": [("Museo", "20/10/2025")]}}
print(viajes)
```

Ejercicio 72 – Registro de pedidos en restaurante
Enunciado: Crear un sistema de pedidos con número, platos y estado.

```
pedidos = {1: {"platos": ["Pizza", "Ensalada"], "estado": "pendiente"}}
print(pedidos)
```

Ejercicio 73 – Clave con valor máximo
Enunciado: Encontrar la clave con el valor máximo en un diccionario.

```
datos = {"a": 10, "b": 25, "c": 7}
clave_max = max(datos, key=datos.get)
print(clave_max)
```

Ejercicio 74 – Fusionar diccionarios
Enunciado: Fusionar dos diccionarios sin usar update().

```
d1 = {"a": 1, "b": 2}
d2 = {"c": 3, "d": 4}
fusion = {**d1, **d2}
print(fusion)
```

Ejercicio 75 – Eliminar claves pares
Enunciado: Eliminar todas las claves con valores pares de un diccionario.

```
datos = {"a": 2, "b": 3, "c": 4}
filtrado = {k:v for k,v in datos.items() if v % 2 != 0}
print(filtrado)
```

Ejercicio 76 – Claves con valores iguales en dos diccionarios
Enunciado: Encontrar claves que tengan valores iguales en dos diccionarios.

```
d1 = {"a": 1, "b": 2}
d2 = {"c": 2, "d": 3}
comunes = [k for k,v in d1.items() if v in d2.values()]
print(comunes)
```

Ejercicio 77 – Suma de valores en diccionario anidado
Enunciado: Calcular la suma de valores en un diccionario anidado.

```
datos = {"a": {"x": 2}, "b": {"x": 3}}
suma = sum(v["x"] for v in datos.values())
print(suma)
```

Ejercicio 78 – Ordenar lista de tuplas
Enunciado: Ordenar una lista de tuplas por el segundo elemento.

```
lista = [(1,3), (2,1), (3,2)]
lista.sort(key=lambda x: x[1])
print(lista)
```

Ejercicio 79 – Tupla con mayor suma
Enunciado: Encontrar la tupla con mayor suma de elementos en una lista.

```
lista = [(1,2), (3,4), (5,1)]
mayor = max(lista, key=lambda x: sum(x))
print(mayor)
```

Ejercicio 80 – Combinar listas de tuplas
Enunciado: Combinar dos listas de tuplas eliminando duplicados por el primer elemento.

```
lista1 = [(1,"a"), (2,"b")]
lista2 = [(2,"c"), (3,"d")]
dic = dict(lista1 + lista2)
print(list(dic.items()))
```

Ejercicio 81 – Dividir lista en pares e impares (tuplas)
Enunciado: Dividir una lista de números en dos listas de tuplas: una con pares y otra con impares.

```
lista = [1,2,3,4,5,6]

pares = [(n, "par") for n in lista if n % 2 == 0]

impares = [(n, "impar") for n in lista if n % 2 != 0]

print(pares)

print(impares)
```

Ejercicio 82 – Tupla más común
Enunciado: Encontrar la tupla más común en una lista de tuplas.

```
lista = [(1,2), (2,3), (1,2), (4,5)]

mas_comun = max(set(lista), key=lista.count)

print(mas_comun)
```

Ejercicio 83 – Intersección de conjuntos
Enunciado: Calcular la intersección de dos conjuntos sin usar el operador &.

```
a = {1,2,3,4}

b = {3,4,5,6}

interseccion = {x for x in a if x in b}

print(interseccion)
```

Ejercicio 84 – Subconjunto
Enunciado: Verificar si un conjunto es subconjunto de otro.

```
a = {1,2}

b = {1,2,3,4}

print(a.issubset(b))
```

Ejercicio 85 – Elementos únicos en lista
Enunciado: Obtener todos los elementos únicos de una lista en un conjunto.

```
lista = [1,2,2,3,4,4,5]

unicos = set(lista)

print(unicos)
```

Ejercicio 86 – Unión de múltiples conjuntos
Enunciado: Calcular la unión de varios conjuntos.

```
a = {1,2}

b = {2,3}

c = {3,4}

union = a | b | c

print(union)
```

Ejercicio 87 – Diferencia simétrica
Enunciado: Calcular la diferencia simétrica entre dos conjuntos.

```
a = {1,2,3}

b = {3,4,5}

dif = a ^ b
```

```
print(dif)
```

Ejercicio 88 – Eliminar duplicados de lista
Enunciado: Eliminar duplicados de una lista sin cambiar el orden.

```
lista = [1,2,2,3,4,4,5]

resultado = []

for n in lista:

    if n not in resultado:

        resultado.append(n)

print(resultado)
```

Ejercicio 89 – Elemento más repetido en lista
Enunciado: Encontrar el elemento que más se repite en una lista.

```
lista = [1,2,2,3,3,3,4]

mas_repetido = max(set(lista), key=lista.count)

print(mas_repetido)
```

Ejercicio 90 – Dividir lista en sublistas
Enunciado: Dividir una lista en sublistas de tamaño fijo.

```
lista = [1,2,3,4,5,6,7,8]

tamaño = 3

sublistas = [lista[i:i+tamaño] for i in range(0, len(lista), tamaño)]

print(sublistas)
```

Ejercicio 91 – Rotación a la izquierda en lista
Enunciado: Rotar los elementos de una lista hacia la izquierda.

```
lista = [1,2,3,4,5]

rotada = lista[1:] + lista[:1]

print(rotada)
```

Ejercicio 92 – Subsecuencia creciente más larga
Enunciado: Encontrar la subsecuencia creciente más larga en una lista.

```
lista = [10, 22, 9, 33, 21, 50]

subsecuencia = []

for n in lista:

    if not subsecuencia or n > subsecuencia[-1]:

        subsecuencia.append(n)

print(subsecuencia)
```

Ejercicio 93 – Lista de usuarios (diccionarios)
Enunciado: Crear una lista de usuarios, cada uno con nombre, correo y pedidos.

```
usuarios = [

    {"nombre": "Ana", "correo": "ana@mail.com", "pedidos": ["pan", "leche"]},

    {"nombre": "Luis", "correo": "luis@mail.com", "pedidos": ["café"]}

]

print(usuarios)
```

Ejercicio 94 – Salario promedio
Enunciado: Calcular el salario promedio de una lista de empleados.

```
empleados = [{"nombre": "Ana", "salario": 1000}, {"nombre": "Luis", "salario": 1500}]

promedio = sum(e["salario"] for e in empleados) / len(empleados)

print(promedio)
```

Ejercicio 95 – Información de películas
Enunciado: Crear un diccionario con películas, actores y fecha de lanzamiento.

```
peliculas = {

    "Matrix": {"año": 1999, "actores": ["Keanu Reeves"], "duracion": 136}

}

print(peliculas)
```

Ejercicio 96 – Empleado con mayor salario
Enunciado: Encontrar el empleado con mayor salario en una lista.

```
empleados = [{"nombre": "Ana", "salario": 1000}, {"nombre": "Luis", "salario": 1500}]

mayor = max(empleados, key=lambda e: e["salario"])

print(mayor)
```

Ejercicio 97 – Productos y revisiones
Enunciado: Crear un diccionario de productos con revisiones de usuarios.

```
productos = {

    "Laptop": [{"comentario": "Muy buena", "calificacion": 5}]
```

}

print(productos)

Ejercicio 98 – Mínimo y máximo en lista
Enunciado: Devolver una tupla con el valor mínimo y máximo de una lista.

lista = [3,7,1,9]

resultado = (min(lista), max(lista))

print(resultado)

Ejercicio 99 – Ordenar estudiantes por calificación
Enunciado: Ordenar una lista de estudiantes según sus calificaciones.

estudiantes = [{"nombre": "Ana", "nota": 9}, {"nombre": "Luis", "nota": 7}]

estudiantes.sort(key=lambda e: e["nota"], reverse=True)

print(estudiantes)

Ejercicio 100 – Correos válidos e inválidos
Enunciado: Separar correos válidos de inválidos en dos listas.

correos = ["ana@mail.com", "luis@", "test@gmail.com"]

validos = [c for c in correos if "@" in c and "." in c]

invalidos = [c for c in correos if c not in validos]

print("Válidos:", validos)

print("Inválidos:", invalidos)