Manual completo de Python

Aprender a programar de 6 a 18 años · Sin necesidad de plataforma

Versión: manual autónomo · 3 módulos por edad
Para: alumnos, padres, ludotecas, centros educativos y tutores

Cómo usar este manual

Este documento está pensado para que **cualquier adulto responsable** (padre, madre, monitor de ludoteca, entrenador en un centro deportivo con aula de informática, etc.) pueda **enseñar Python** sin depender de una plataforma online.

Estructura

Módulo	Edades	Duración orientativa	Contenido
A	6–12	8–12 sesiones de 30–45 min	Primeros pasos, variables, input
B	12–14	10–14 sesiones de 45–60 min	Condicionales, bucles, listas
C	14–18	12–16 sesiones de 60 min	Funciones, diccionarios, proyectos

Cada lección incluye: explicación, ejemplo, práctica y **solución al final del módulo**.

Para el adulto que enseña: No hace falta ser programador. Lee la lección antes, ejecuta el código tú mismo y deja que el alumno escriba con sus dedos (no copies tú el código por él). Celebra los errores: son parte del aprendizaje.

Qué necesitas

1. Un ordenador (Windows, Mac o Linux).
2. **Python 3.10 o superior** instalado desde python.org.
3. Un editor de texto: **IDLE** (viene con Python), **Thonny** (muy recomendado para niños) o VS Code.
4. Papel y boli para dibujar «cajas» de variables (módulo A).

Cómo ejecutar un programa

1. Abre el editor y crea un archivo `programa.py`.
2. Escribe el código y guarda.
3. Ejecuta:
   - En **Thonny** o **IDLE**: botón ▶ Ejecutar.
   - En terminal: `python3 programa.py` (Mac/Linux) o `python programa.py` (Windows).

> **Alternativa online (sin instalar):** replit.com → nuevo Repl → Python. Útil en ludotecas sin permiso de instalación.

Reglas de oro en el aula

1. **Un concepto por sesión.** No corras si no está asimilado.
2. **Escribir > leer.** El alumno debe teclear el código.
3. **Preguntar antes de corregir:** «¿Qué crees que hace esta línea?»
4. Las soluciones están al final de cada módulo; intenta los ejercicios primero.

Módulo A · 6–12 años

Módulo A · Primeros pasos

Objetivos del módulo

Al terminar, el alumno será capaz de:

- Explicar qué es un programa en palabras sencillas.
- Usar `print()` para mostrar mensajes.
- Guardar datos en variables.
- Pedir datos con `input()` y mostrarlos.

Lección A1 · ¿Qué es programar?

Duración: 20–30 min

Programar es **dar instrucciones muy claras** a un ordenador. Es como una receta: paso 1, paso 2, paso 3.

Actividad sin ordenador (5 min): El adulto dice: «Párate, da tres pasos, levanta la mano derecha». Si las instrucciones son vagas, el niño hace algo distinto. Conclusión: el ordenador necesita precisión.

Tu primer programa:

```
print("¡Hola! Soy un programador.")
```

Práctica A1: Escribe un programa que muestre tres líneas: tu nombre, tu edad y tu comida favorita (como texto).

Lección A2 · Mensajes con print()

Duración: 30 min

`print()` muestra texto en pantalla. Puedes imprimir varias cosas separadas por comas:

```
print("Me llamo", "Luna")
print("Tengo", 10, "años")
print(2 + 3)
```

Los textos van entre comillas `"..."`. Los números no.

Caracteres especiales: `\n` salta de línea:

```
print("Línea 1\nLínea 2")
```

Práctica A2: Imprime un «cartel» de 4 líneas para una fiesta de cumpleaños (título, fecha inventada, lugar, «¡Te esperamos!»).

Lección A3 · Variables (cajas con nombre)

Duración: 30–40 min

Una **variable** es una caja con etiqueta donde guardas un valor:

```
nombre = "Alex"
edad = 10
print(nombre)
print(edad)
```

Puedes cambiar el contenido:

```
puntos = 0
puntos = puntos + 5
print(puntos)  # muestra 5
```

Dibuja en papel tres cajas etiquetadas `nombre`, `edad`, `color_favorito` y pide al niño que escriba valores dentro.

Práctica A3: Crea variables para el nombre de tu mascota (real o inventada), su tipo (perro, gato...) y su edad. Imprime una frase que las use todas.

Lección A4 · Números y operaciones

Duración: 30 min

Python calcula como una calculadora:

```
suma = 7 + 3
resta = 10 - 4
producto = 6 * 2
division = 15 / 3
print(suma, resta, producto, division)
```

Práctica A4: Crea un programa «Calculadora de golosinas»: cantidad de caramelos y de chicles, imprime el total de dulces y cuántos caramelos hay de más que chicles.

Lección A5 · Preguntar al usuario con input()

Duración: 35–45 min

`input()` muestra un mensaje y **espera** a que el usuario escriba:

```
nombre = input("¿Cómo te llamas? ")
print("Encantado,", nombre)
```

Importante: `input()` siempre devuelve **texto** (string). Si quieres un número:

```
edad_texto = input("¿Cuántos años tienes? ")
edad = int(edad_texto)
print("El año que viene tendrás", edad + 1)
```

Práctica A5: Pregunta nombre y color favorito. Muestra: «A [nombre] le gusta el color [color]».

Proyecto A · Tarjeta de presentación interactiva

Duración: 45–60 min (puede dividirse en 2 sesiones)

Programa que:

1. Pida nombre, edad y hobby.

2. Muestre una tarjeta con bordes de asteriscos:

```
**************************
*  Hola, me llamo Alex   *
*  Tengo 10 años         *
*  Me gusta: fútbol      *
**************************
```

Pistas: usa varios `print()` y variables. No hace falta que los bordes queden perfectos.

Autoevaluación módulo A

El alumno responde sin mirar el código:

1. ¿Qué hace `print()`?
2. ¿Qué guarda una variable?
3. ¿`input()` devuelve número o texto?
4. Escribe de memoria un programa que pregunte tu ciudad y la imprima.

> **Para el tutor:** Si falla la pregunta 4, repasa A5 antes de pasar al módulo B.

Soluciones módulo A

A1 (ejemplo):

```
print("María")
print("9 años")
print("Pizza")
```

A2 (ejemplo):

```
print("¡Cumple de Sara!")
print("Sábado 15 de junio")
print("Casa de Sara")
print("¡Te esperamos!")
```

A3 (ejemplo):

```
mascota = "Toby"
tipo = "perro"
edad_mascota = 3
print("Mi mascota se llama", mascota, "es un", tipo, "y tiene", edad_mascota, "añ
```

A4 (ejemplo):

```
caramelos = 12
chicles = 8
print("Total:", caramelos + chicles)
print("Más caramelos que chicles:", caramelos - chicles)
```

A5 (ejemplo):

```
nombre = input("¿Cómo te llamas? ")
color = input("¿Cuál es tu color favorito? ")
print("A", nombre, "le gusta el color", color)
```

Proyecto A (ejemplo):

```
nombre = input("¿Cómo te llamas? ")
edad = input("¿Cuántos años tienes? ")
hobby = input("¿Qué te gusta hacer? ")
print("**************************")
print("* Hola, me llamo", nombre, "  *")
print("* Tengo", edad, "años", "        *")
print("* Me gusta:", hobby, "     *")
print("**************************")
```

Módulo B · 12–14 años

Módulo B · Lógica y estructuras

Objetivos del módulo

- Usar `if`, `elif`, `else` para decisiones.
- Repetir con `for` y `while`.

- Trabajar con listas.
- Resolver problemas dividiéndolos en pasos.

Requisito: haber completado el módulo A o saber variables, `print` e `input`.

Lección B1 · Condicionales (if / else)

Duración: 45 min

```
nota = int(input("¿Qué nota sacaste (0-10)? "))

if nota ⩾ 5:
    print("¡Aprobado!")
else:
    print("Hay que seguir practicando")
```

`elif` para más de dos caminos:

```
if nota ⩾ 9:
    print("Sobresaliente")
elif nota ⩾ 7:
    print("Notable")
elif nota ⩾ 5:
    print("Aprobado")
else:
    print("Suspenso")
```

Indentación: el código dentro del `if` va **indentado** (4 espacios). Sin indentación, Python da error.

Práctica B1: Pide la edad. Si es >= 14, imprime «Puedes usar redes con supervisión»; si no, «Tiempo de juegos educativos».

Lección B2 · Bucles for

Duración: 45 min

Repetir un bloque un número conocido de veces:

```python
for i in range(5):
    print("Repetición número", i)

frutas = ["manzana", "pera", "plátano"]
for fruta in frutas:
    print("Me gusta la", fruta)
```

`range(3, 8)` cuenta del 3 al 7.

Práctica B2: Imprime la tabla de multiplicar del 7 (del 1 al 10): `7 x 1 = 7`, etc.

Lección B3 · Bucles while

Duración: 45 min

Repite **mientras** se cumpla una condición:

```python
contador = 3
while contador > 0:
    print(contador)
    contador = contador - 1
print("¡Despegue!")
```

`break` sale del bucle; `continue` salta a la siguiente vuelta.

Práctica B3: Pide números al usuario y súmalos hasta que escriba `0`. Muestra la suma total.

Lección B4 · Listas

Duración: 50 min

```python
misiones = ["Hacer deberes", "Practicar Python", "Leer"]
print(misiones[0])   # primer elemento (índice 0)
misiones.append("Dormir bien")
print(len(misiones))
```

Recorrer con índice:

```
for i in range(len(misiones)):
    print(i + 1, "-", misiones[i])
```

Práctica B4: Lista de 5 películas o juegos favoritos. Imprime la lista numerada y cuántos hay en total.

Lección B5 · Combinar todo: adivina el número

Duración: 60 min

Juego completo (estudiar línea a línea):

```
import random

secreto = random.randint(1, 10)
intentos = 0
max_intentos = 3

print("Adivina un número del 1 al 10")

while intentos < max_intentos:
    intentos = intentos + 1
    respuesta = int(input("Tu intento: "))

    if respuesta == secreto:
        print("¡Correcto en", intentos, "intentos!")
        break
    elif respuesta < secreto:
        print("Más alto...")
    else:
        print("Más bajo...")
else:
    print("Se acabaron los intentos. Era", secreto)
```

Práctica B5: Cambia el juego para números del 1 al 20 y 5 intentos.

Proyecto B · Lista de la compra

Duración: 2 sesiones

Programa por menú en consola:

1. Añadir producto a la lista.
2. Ver lista numerada.
3. Contar productos.
4. Salir.

Pista estructura:

```
compra = []

while True:
    print("\n1. Añadir  2. Ver  3. Contar  4. Salir")
    opcion = input("Elige: ")
    # completar con if/elif...
```

Autoevaluación módulo B

1. ¿Cuándo usarías `if` en la vida real? (ejemplo propio)
2. Diferencia entre `for` y `while`.
3. ¿Qué índice tiene el primer elemento de una lista?
4. Escribe un `for` que imprima los números pares del 2 al 10.

Soluciones módulo B

B1:

```
edad = int(input("¿Cuántos años tienes? "))
if edad ⩾ 14:
    print("Puedes usar redes con supervisión")
else:
    print("Tiempo de juegos educativos")
```

B2:

```
for i in range(1, 11):
    print("7 x", i, "=", 7 * i)
```

B3:

```python
total = 0
while True:
    n = int(input("Número (0 para terminar): "))
    if n == 0:
        break
    total = total + n
print("Suma total:", total)
```

B4: (solución libre con lista de 5 strings y bucle)

Proyecto B (solución completa):

```python
compra = []

while True:
    print("\n--- Lista de la compra ---")
    print("1. Añadir producto")
    print("2. Ver lista")
    print("3. Contar productos")
    print("4. Salir")
    opcion = input("Elige una opción: ")

    if opcion == "1":
        producto = input("¿Qué producto añades? ")
        compra.append(producto)
        print("Añadido:", producto)
    elif opcion == "2":
        if len(compra) == 0:
            print("La lista está vacía")
        else:
            for i in range(len(compra)):
                print(i + 1, "-", compra[i])
    elif opcion == "3":
        print("Hay", len(compra), "productos")
    elif opcion == "4":
        print("¡Hasta luego!")
        break
    else:
        print("Opción no válida")
```

Módulo C · 14–18 años

Módulo C · Programación estructurada

Objetivos del módulo

- Escribir funciones reutilizables con parámetros y `return`.
- Usar diccionarios para datos estructurados.
- Manejar errores con `try` / `except`.
- Completar un proyecto CLI con varias funciones.

Requisito: módulo B o equivalente (condicionales, bucles, listas).

Lección C1 · Funciones

Duración: 60 min

```
def saludar(nombre):
  print("Hola,", nombre)

saludar("Alex")
saludar("Luna")
```

Con valor de retorno:

```
def area_rectangulo(base, altura):
    return base * altura

resultado = area_rectangulo(5, 3)
print("Área:", resultado)
```

Práctica C1: Función `es_mayor_de_edad(edad)` que devuelva `True` o `False`.

Lección C2 · Diccionarios

Duración: 60 min

```python
alumno = {
    "nombre": "Alex",
    "edad": 16,
    "cursos": ["Python", "Mates"],
}

print(alumno["nombre"])
alumno["xp"] = 120

for clave, valor in alumno.items():
    print(clave, "→", valor)
```

Práctica C2: Diccionario de un videojuego (nombre, nivel, vidas, inventario como lista). Imprime un «perfil» formateado.

Lección C3 · Errores y try/except

Duración: 45 min

```python
def pedir_entero(mensaje):
    while True:
        try:
            return int(input(mensaje))
        except ValueError:
            print("Escribe un número entero válido")

edad = pedir_entero("Edad: ")
print("Tienes", edad, "años")
```

Práctica C3: Función `pedir_float` para decimales con el mismo patrón.

Lección C4 · List comprehensions y buenas prácticas

Duración: 45 min

Forma compacta de crear listas:

```python
cuadrados = [n * n for n in range(1, 6)]
print(cuadrados)

pares = [n for n in range(20) if n % 2 == 0]
```

Buenas prácticas:

- Nombres claros: `total_puntos` mejor que `tp`.
- Una función = una tarea.
- Comentarios solo cuando el «por qué» no es obvio.

Práctica C4: Lista de palabras; crea otra lista solo con las que tengan más de 4 letras.

Lección C5 · Módulos estándar (random, datetime)

Duración: 45 min

```python
import random
from datetime import date

print(random.choice(["cara", "cruz"]))
print("Hoy:", date.today())
```

Práctica C5: Simulador de dado: función `tirar_dado()` que devuelva 1–6.

Proyecto C · Gestor de tareas (CLI)

Duración: 3–4 sesiones

Requisitos funcionales:

1. Lista de tareas como diccionarios: `{"titulo": "...", "hecha": False}`.
2. Menú: añadir, listar (pendientes/hechas), marcar hecha, eliminar, salir.
3. Cada acción en su **función**.
4. Validar entradas con `try/except`.

Ver **solución completa en apéndice** al final del manual.

Autoevaluación módulo C

1. ¿Para qué sirve `return`?

2. ¿Cuándo usar lista y cuándo diccionario?
3. Explica `try/except` con un ejemplo propio.
4. Diseña en papel las funciones de tu gestor de tareas (nombres y parámetros).

Soluciones módulo C (ejercicios)

C1:

```
def es_mayor_de_edad(edad):
    return edad ⩾ 18

print(es_mayor_de_edad(16))
print(es_mayor_de_edad(20))
```

C3 y C5: (el alumno adapta el patrón de C3)

Apéndice · Solución proyecto C (gestor de tareas)

```
tareas = []

def mostrar_menu():
    print("\n═══ Gestor de tareas ═══")
    print("1. Añadir tarea")
    print("2. Listar todas")
    print("3. Listar pendientes")
    print("4. Marcar como hecha")
    print("5. Eliminar tarea")
    print("6. Salir")

def añadir_tarea():
    titulo = input("Título de la tarea: ").strip()
    if titulo:
        tareas.append({"titulo": titulo, "hecha": False})
        print("Tarea añadida")
    else:
        print("Título vacío")

def listar_tareas(solo_pendientes=False):
    if not tareas:
        print("No hay tareas")
        return
    for i, t in enumerate(tareas):
        if solo_pendientes and t["hecha"]:
            continue
        estado = "✓" if t["hecha"] else " "
        print(f"{i + 1}. [{estado}] {t['titulo']}")

def marcar_hecha():
    listar_tareas()
    try:
        n = int(input("Número de tarea: ")) - 1
        if 0 ⩽ n < len(tareas):
            tareas[n]["hecha"] = True
            print("Marcada como hecha")
        else:
            print("Número inválido")
    except ValueError:
        print("Escribe un número")

def eliminar_tarea():
    listar_tareas()
    try:
        n = int(input("Número a eliminar: ")) - 1
        if 0 ⩽ n < len(tareas):
```

```
            print("Eliminada:", tareas.pop(n)["titulo"])
        else:
            print("Número inválido")
    except ValueError:
        print("Escribe un número")

def main():
    while True:
        mostrar_menu()
        opcion = input("Elige: ")
        if opcion == "1":
            añadir_tarea()
        elif opcion == "2":
            listar_tareas()
        elif opcion == "3":
            listar_tareas(solo_pendientes=True)
        elif opcion == "4":
            marcar_hecha()
        elif opcion == "5":
            eliminar_tarea()
        elif opcion == "6":
            print("¡Buen trabajo!")
            break
        else:
            print("Opción no válida")

if __name__ == "__main__":
    main()
```

Apéndice · Errores frecuentes

Error	Causa	Qué hacer
`SyntaxError`	Falta `:`, paréntesis o comilla	Lee el mensaje; revisa la línea anterior también
`IndentationError`	Espacios mal puestos en `if` / `for`	Usa 4 espacios siempre en el mismo bloque
`NameError`	Variable no definida	¿Escribiste bien el nombre? ¿La creaste antes?
`ValueError` en `int()`	`input` no es un número	Usa `try/except` o valida antes

Apéndice · Glosario

Término	Significado
Variable	Nombre que guarda un valor
Función	Bloque de código reutilizable
Lista	Colección ordenada de valores
Diccionario	Pares clave → valor
Condicional	Decisión con `if`
Bucle	Repetición con `for` o `while`
CLI	Programa que se usa escribiendo en la consola

Siguiente paso opcional: Si tienes acceso a la Academia de Programación, el curso `python-starter` y el Editor refuerzan este manual con ejercicios interactivos. **No es obligatorio** para aprender con este documento.