Última actualización: November 1, 2026

🗺️ Ruta de cuatro años · Año 1, Primavera

Sobre este curso

Este curso gratuito tiene 9 unidades y 4 laboratorios. Son los totales reales, no un programa más largo recortado.

Una comprobación es un número o un resultado de código en una página de curso que un script vuelve a calcular antes de publicar. El número impreso y el recalculado tienen que coincidir.

> Este es el esquema de un curso gratuito de autoestudio escrito por Saving Planets. No es un curso universitario, no otorga créditos académicos y ninguna institución participa.
>
> Su alcance sigue los cursos de programación de primer año como MIT 6.100L (antes 6.0001), CS50 de Harvard, CS106A de Stanford y CS 1371 de Georgia Tech para ingenieros. Enlazamos materiales gratuitos en todo el curso. Ninguna de estas organizaciones respalda esta página.
>
> Lo que añadimos: cada ejemplo es Python real que nuestro script de compilación ejecuta antes de publicar la página, y su salida impresa debe coincidir exactamente con lo que ves aquí. Los datos provienen de fuentes públicas reales: el registro de CO_2 de Mauna Loa de NOAA, el análisis de temperatura GISTEMP de la NASA y los medidores de ríos del USGS.
>
> No necesitas instalar nada: usa Google Colab o JupyterLite, que ejecuta Python dentro de tu navegador con Pyodide. Tu progreso se guarda solo en este navegador.

Cómo ayuda este curso a salvar el planeta

La ciencia del clima y del medio ambiente funciona con código. La curva de CO_2, los registros de temperatura global y los pronósticos de ríos los construyen programas que leen datos públicos, y tú puedes leer los mismos archivos.

Cada unidad termina con una breve sección con fuentes que muestra cómo sus ideas ayudan a las personas y al planeta.

Para docentes

Puedes enseñar este curso o dejar que un estudiante lo haga solo. Es gratis. Saving Planets no pone nota ni da crédito escolar.

- Esta página es el curso completo: 9 unidades y 4 laboratorios (ENGR 130).
- Calcula 15 semanas, unas 6 a 9 horas por semana, incluidas las semanas de laboratorio.
- Sigue el calendario de 15 semanas. Lee la unidad, haz la práctica en papel y luego abre la respuesta. En una semana de laboratorio, haz el laboratorio o la versión en casa.
- Una comprobación es un número o un resultado de código en la página que un script vuelve a calcular antes de publicar. El número que ves y el recalculado tienen que coincidir.
- Usa Imprimir esta página, o descarga la guía de estudio en PDF. La hoja de fórmulas está cerca del final.
- Pregunta: ¿cómo ayuda esta unidad a las personas y al planeta? Cada unidad tiene un recuadro verde con un dato y su fuente.
- Copia la plantilla de guía docente y escribe el nombre de tu clase, el tiempo que tienes y las preguntas que harás. Abrir la plantilla de guía docente

Antes de empezar

Requisitos previos

- Álgebra de bachillerato.
- No se necesita experiencia en programación.
- Física I y Cálculo I ayudan en la unidad de simulación, pero no son obligatorios.

Cómo estudiar

- Escribe tú mismo cada ejemplo en lugar de pegarlo; tus dedos aprenden la sintaxis.
- Predice la salida antes de ejecutar el código.
- Cuando te atasques, lee el mensaje de error de abajo hacia arriba.
- Cambia una cosa a la vez y vuelve a ejecutar.
- Haz cada ejercicio antes de abrir la respuesta.
- Repite los cuestionarios unos días después. La práctica espaciada es mejor que una sesión larga.

Qué necesitas

- Una computadora o tableta con un navegador web moderno.
- Una cuenta gratuita de Google para Colab, o JupyterLite sin ninguna cuenta.
- Opcional: Python 3 instalado localmente con NumPy y Matplotlib.

Calendario sugerido de 15 semanas

Unas 6 a 9 horas por semana: lectura, problemas de práctica y, en las semanas de laboratorio, el laboratorio.

Semana	Unidad	Temas	Lectura
1	Primeros pasos: variables, tipos y expresiones	Notebooks, variables, tipos, expresiones	Tutorial de Python §3
2	Control de flujo: condiciones y bucles	Condiciones y bucles	Tutorial de Python §4
3	Funciones	Funciones	MIT 6.100L
4	Listas, diccionarios y cadenas	Listas, diccionarios, cadenas	Tutorial de Python §5
5	Archivos y datos CSV	Archivos y CSV	Documentación del módulo csv
6	Archivos y datos CSV	Datos reales con errores; práctica A	NOAA GML
7	NumPy y Matplotlib	Arreglos de NumPy; práctica para el parcial	Fundamentos de NumPy
8	NumPy y Matplotlib	Matplotlib; práctica B	Matplotlib pyplot
9	Depuración y pruebas	Depuración y pruebas; práctica C	Documentación de pytest
10	Algoritmos y complejidad	Búsqueda y complejidad	MIT 6.100L
11	Algoritmos y complejidad	Bisección y métodos numéricos	MIT 6.100L
12	Simulación numérica	Simulación de proyectiles	Física I
13	Simulación numérica	Órbitas y poblaciones	Cálculo II
14	Simulación numérica	Resistencia y convergencia; práctica D	Introducción a la ingeniería aeroespacial
15	Repaso	Repaso y serie de práctica final	Todas las unidades

Tu progreso

Marca las unidades y los laboratorios al terminarlos y haz los cuestionarios. El progreso y las puntuaciones se guardan solo en este navegador y nunca se nos envían.

Unidades

Primeros pasos: variables, tipos y expresiones

Ejecuta tu primer código Python en un notebook del navegador, guarda valores en variables y calcula con números y texto.

Objetivos de aprendizaje

Al terminar esta unidad podrás:

- Abrir un notebook de Colab o JupyterLite y ejecutar celdas.
- Usar variables y los tipos int, float, str y bool.
- Usar operadores aritméticos, incluidos //, % y **.
- Dar formato a la salida con f-strings.

Conceptos clave

- Un programa es una secuencia de instrucciones. Un notebook te permite ejecutarlas celda a celda y ver los resultados al instante.
- Una variable es un nombre ligado a un valor: mass_kg = 1100. Elige nombres que digan qué es y en qué unidades.
- Todo valor tiene un tipo. int (números enteros), float (decimales), str (texto) y bool (True/False) son los básicos; type() te dice cuál.
- / siempre da un float; // es la división entera; % es el resto; ** es una potencia.
- Los floats son aproximaciones binarias: 0.1 + 0.2 no es exactamente 0.3, así que compara floats con una tolerancia.

Ecuaciones clave

$$W = mg$$

`W=mg`

El ejemplo 1.1 calcula un peso

$$\text{change} = \frac{x_{new} - x_{old}}{x_{old}}$$

`change=_new-x_oldx_old`

Cambio relativo (ejemplo 1.2)

Ejemplos resueltos paso a paso

Ejemplo 1.1 Calcula el peso de un avión de 1100 kg y comprueba los tipos.

```
mass_kg = 1100
g = 9.81
weight = mass_kg * g
print(weight)
print(type(mass_kg), type(weight))
```

1. mass_kg es un int; multiplicarlo por el float g da un float.

Salida

```
10791.0
<class 'int'> <class 'float'>
```

Respuesta: 10791 N, un float.

Ejemplo 1.2 ¿Cuánto ha subido el CO_2 de Mauna Loa desde la media anual de 1959 hasta la de 2025 (NOAA)?

```
ppm_1959 = 315.98
ppm_2025 = 427.35
rise = ppm_2025 - ppm_1959
print(f"CO2 rose {rise:.2f} ppm, or {rise / ppm_1959:.1%}")
```

1. Resta y luego divide por el valor inicial.
2. La f-string da formato con :.2f (2 decimales) y :.1% (porcentaje).

Salida

```
CO2 rose 111.37 ppm, or 35.2%
```

Respuesta: 111.37 ppm, una subida del 35.2%.

Ejemplo 1.3 Explora los operadores y las sorpresas de los floats.

```
print(7 / 2, 7 // 2, 7 % 2, 2 ** 10)
print(0.1 + 0.2 == 0.3, abs((0.1 + 0.2) - 0.3) < 1e-9)
print(int("42") + 1, str(42) + "1")
```

1. 7/2 = 3.5, 7//2 = 3, 7%2 = 1.
2. 0.1 + 0.2 == 0.3 es False; una prueba con tolerancia es True.
3. int("42") convierte texto en número; str(42) + "1" une texto.

Salida

```
3.5 3 1 1024
False True
43 421
```

Respuesta: Usa tolerancias con floats y convierte tipos a propósito.

Problemas de práctica

Resuelve cada uno primero en papel y luego comprueba.

1. ¿Qué da 17 % 5?

   2.

2. ¿De qué tipo es 3 * 2.0?

   float (6.0).

3. Escribe una f-string que imprima pi con 3 decimales.

   f"{math.pi:.3f}" → 3.142 (después de import math).

4. ¿Por qué mass_kg es mejor nombre que m?

   Indica la magnitud y su unidad, lo que evita errores de unidades.

Ideas erróneas comunes

Mito: = significa «igual».

En realidad: En Python = asigna; == compara.

Mito: 0.1 + 0.2 == 0.3 es True.

En realidad: Es False por el redondeo binario.

Mito: "5" + 5 da 10.

En realidad: Es un TypeError; convierte con int("5").

Tarjetas de estudio

Variable	**int / float**
Nombre ligado a un valor.	Número entero / número decimal.

str

Texto.

bool

True o False.

f-string

Formato f"{x:.2f}".

//, %

División entera, resto.

Cuestionario de autoevaluación

1. 7 // 2 =

3.5

3

4

1

3. División entera.

2. type(2.0) es...

int

float

str

bool

float. Tiene punto decimal.

3. 2 ** 10 =

20

100

1024

12

1024. Potencia.

4. La asignación en Python usa...

==

=

:=only

->

=. = liga un nombre.

Repetir este cuestionario

Cómo ayuda esto a salvar el planeta

Los números del ejemplo 1.2 salen directamente del Global Monitoring Laboratory de NOAA, que publica el registro de CO_2 de Mauna Loa como un archivo CSV gratuito. Unas pocas líneas de Python lo convierten en los datos clave de la ciencia del clima.

En Saving Planets: Cálculo I

Fuentes: NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html) · Google Colab (notebooks gratuitos) (https://colab.research.google.com/)

Recursos de aprendizaje gratuitos

- El tutorial oficial de Python (https://docs.python.org/3/tutorial/): Secciones 3 y 4.
- Google Colab (notebooks gratuitos) (https://colab.research.google.com/): Empieza un notebook.
- JupyterLite: Python en tu navegador (Pyodide) (https://jupyter.org/try-jupyter/lab/): Opción sin cuenta.

☐ **Terminé esta unidad**

Control de flujo: condiciones y bucles

Hacer que los programas decidan con if/elif/else y repitan con bucles for y while.

Objetivos de aprendizaje

Al terminar esta unidad podrás:

- Escribir if/elif/else con operadores de comparación y booleanos.
- Recorrer rangos y secuencias con for.
- Usar bucles while con una condición de parada clara.
- Acumular totales y conteos.

Conceptos clave

- La sangría define los bloques en Python. Lo estándar son cuatro espacios por nivel.
- Las comparaciones (<, >=, ==, !=) producen booleanos; combínalas con and, or, not.
- for x in sequence se repite una vez por elemento; range(a, b) da a, a+1, ..., b−1.
- while se repite hasta que su condición es False. Asegúrate de que algo dentro del bucle la cambie, o nunca terminará.
- El patrón acumulador: empieza un total en 0 y suma dentro del bucle.

Ecuaciones clave

$$\sum_{n=1}^{100} n = \frac{100 \cdot 101}{2} = 5050$$

_n=1^100n=(100·101)/(2)=5050

La suma de Gauss comprueba el ejemplo 2.2

$$x_{k+1} = x_k + \Delta$$

x_k+1=x_k+Δ

Un bucle que avanza un valor

Ejemplos resueltos paso a paso

Ejemplo 2.1 Clasifica la anomalía de temperatura global de 2024 de la NASA.

```
anomaly = 1.28   # NASA GISTEMP, 2024, deg C vs 1951-1980
if anomaly >= 1.0:
    print("at least 1.0 C above the 1951-1980 average")
```

```
elif anomaly > 0:
    print("warmer than average")
else:
    print("at or below average")
```

1. 1.28 ≥ 1.0, así que se ejecuta la primera rama y se omiten las demás.

Salida

```
at least 1.0 C above the 1951-1980 average
```

Respuesta: Las ramas se prueban en orden; solo se ejecuta la primera verdadera.

Ejemplo 2.2 Suma los números del 1 al 100 con un bucle.

```
total = 0
for n in range(1, 101):
    total += n
print(total)
```

1. range(1, 101) se detiene antes de 101.
2. total += n suma cada número.

Salida

```
5050
```

Respuesta: 5050, igual que la fórmula de Gauss.

Ejemplo 2.3 Si el CO_2 siguiera subiendo 2.5 ppm al año desde la media de 2025, ¿cuándo superaría 450 ppm?

```
ppm = 427.35   # 2025 annual mean, Mauna Loa
year = 2025
while ppm < 450:
    ppm += 2.5
    year += 1
print(year, round(ppm, 2))
```

1. El bucle suma 2.5 ppm y un año hasta que ppm ≥ 450.

Salida

```
2035 452.35
```

Respuesta: En 2035, con 452.35 ppm en este escenario sencillo de ritmo constante (no es un pronóstico).

Problemas de práctica

Resuelve cada uno primero en papel y luego comprueba.

1. ¿Cuántas veces se ejecuta for i in range(3, 10, 2)?

   4 veces (3, 5, 7, 9).

2. Escribe un bucle que cuente los números impares menores que 20.

   count = sum(1 for n in range(20) if n % 2): 10.

3. ¿Qué está mal en: while x < 10: print(x)

   x nunca cambia, así que el bucle es infinito.

Ideas erróneas comunes

Mito: range(1, 10) incluye 10.
En realidad: Se detiene en 9.

Mito: else después de if es obligatorio.
En realidad: Es opcional.

Mito: La sangría es solo estilo.
En realidad: En Python define la estructura del programa.

Tarjetas de estudio

if / elif / else

Elegir una rama.

for

Repetir por elemento.

while

Repetir mientras sea verdadero.

range(a, b, s)

De a hasta b−1, con paso s.

Acumulador

total += x en un bucle.

break / continue

Salir del bucle / pasar a la siguiente vuelta.

Cuestionario de autoevaluación

1. list(range(2, 6)) es...

[2, 3, 4, 5, 6]

[2, 3, 4, 5]

[3, 4, 5, 6]

[2, 6]

[2, 3, 4, 5]. Se detiene antes de 6.

2. ¿Qué palabra clave añade otra prueba después de if?

else if

elif

then

case

elif. elif.

3. Un bucle while se detiene cuando...

su condición es False

tras 10 vueltas

nunca

en un print

su condición es False. La condición se comprueba en cada vuelta.

4. not (True and False) es...

True

False

None

un error

True. True and False es False.

Repetir este cuestionario

Cómo ayuda esto a salvar el planeta

Umbrales como 1.0 °C o 450 ppm son justo lo que los programas de monitoreo prueban con sentencias if, marcando récords y alertas automáticamente cuando llegan datos nuevos.

En Saving Planets: Física I

Fuentes: NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/) · NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html)

Recursos de aprendizaje gratuitos

- El tutorial oficial de Python (https://docs.python.org/3/tutorial/): Sección 4: control de flujo.
- MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/): Clases sobre ramificación e iteración.

☐ **Terminé esta unidad**

Funciones

Empaquetar la lógica en funciones reutilizables y comprobables, con parámetros, valores por defecto y valores de retorno.

Objetivos de aprendizaje

Al terminar esta unidad podrás:

- Definir funciones con def, parámetros y return.
- Usar argumentos por defecto y con nombre.
- Escribir docstrings.
- Entender el ámbito local.

Conceptos clave

- Una función da nombre a un fragmento de lógica. Escríbela una vez, pruébala una vez, úsala en todas partes.
- Los parámetros reciben valores; return devuelve un resultado. Una función sin return devuelve None.
- Los argumentos por defecto (cl_max=1.6) acortan los casos comunes; los argumentos con nombre hacen legibles las llamadas.
- Las variables creadas dentro de una función son locales y desaparecen cuando esta retorna.
- Un docstring (la cadena de la primera línea) documenta lo que hace la función.

Ecuaciones clave

$$F = \frac{9}{5}C + 32$$

F=95C+32

Ejemplo 3.1

$$V_s = \sqrt{\frac{2W}{\rho S C_{L,\max}}}$$

V_s=2Wρ SC_L,

Ejemplo 3.2, de Introducción a la ingeniería aeroespacial

Ejemplos resueltos paso a paso

Ejemplo 3.1 Escribe y prueba una función de Celsius a Fahrenheit.

```
def c_to_f(c):
    """Convert Celsius to Fahrenheit."""
    return c * 9 / 5 + 32

for c in (-40, 0, 1.28, 100):
    print(c, "->", round(c_to_f(c), 2))
```

1. −40 es igual en ambas escalas, una comprobación útil.
2. round(..., 2) ordena los floats.

Salida

```
-40 -> -40.0
0 -> 32.0
1.28 -> 34.3
100 -> 212.0
```

Respuesta: Una anomalía de 1.28 °C es una diferencia de 2.3 °F, pero 1.28 °C como temperatura son 34.3 °F; las unidades importan.

Ejemplo 3.2 Convierte la fórmula de la velocidad de pérdida en una función y úsala con dos densidades.

```
def stall_speed(weight_n, rho, area, cl_max=1.6):
    return (2 * weight_n / (rho * area * cl_max)) ** 0.5

print(round(stall_speed(10791, 1.227, 16.2), 1))
print(round(stall_speed(10791, 0.738, 16.2), 1))
```

1. cl_max tiene un valor por defecto, así que quien llama puede omitirlo.
2. ** 0.5 es una raíz cuadrada.

Salida

```
26.0
33.6
```

Respuesta: 26.0 m/s a nivel del mar y 33.6 m/s a 5000 m, igual que en el curso aeroespacial.

Ejemplo 3.3 Escribe una función que devuelva los cambios interanuales de una lista (medias anuales de Mauna Loa 2022–2025).

```
def growth(values):
    """Year-to-year changes of a list."""
    return [b - a for a, b in zip(values, values[1:])]

co2 = [418.53, 421.08, 424.61, 427.35]   # 2022-2025
print([round(d, 2) for d in growth(co2)])
```

1. zip empareja cada valor con el siguiente.
2. Una comprensión de listas construye el resultado.

Salida

```
[2.55, 3.53, 2.74]
```

Respuesta: Aumentos de 2.55, 3.53 y 2.74 ppm.

Problemas de práctica

Resuelve cada uno primero en papel y luego comprueba.

1. Escribe area_circle(r) y pruébala con r = 2.

   return math.pi * r**2 → 12.566....

2. ¿Qué devuelve una función sin return?

   None.

3. ¿Por qué probar c_to_f en −40?

   Ambas escalas coinciden ahí, así que la respuesta esperada es conocida.

Ideas erróneas comunes

Mito: print y return son lo mismo.
En realidad: print muestra un valor; return lo devuelve para usarlo después.

Mito: Cambiar un parámetro cambia la variable de quien llama.
En realidad: Reasignar un parámetro es local (los objetos mutables son una excepción).

Mito: Las funciones deben ser largas y hacerlo todo.
En realidad: Las funciones pequeñas y de un solo propósito son más fáciles de probar.

Tarjetas de estudio

def

Define una función.

return

Devuelve un valor.

Parámetro

Nombre para una entrada.

Argumento por defecto

Valor usado si se omite.

Docstring

Cadena de documentación.

Ámbito

Dónde es visible un nombre.

Cuestionario de autoevaluación

1. Se define una función con...

func

def

function

solo lambda

def. def name(params):

2. Una función sin sentencia return devuelve...

0

None

False

un error

None. None implícito.

3. En def f(x, k=2), k es...

obligatorio

un argumento por defecto

global

una constante

un argumento por defecto. Tiene un valor por defecto.

4. Las variables creadas dentro de una función son...

global

locales

compartidas

permanentes

locales. Ámbito local.

Repetir este cuestionario

Cómo ayuda esto a salvar el planeta

Los códigos de clima e hidrología se construyen con funciones pequeñas y probadas como estas, para conversiones de unidades, tasas de cambio y fórmulas físicas; así es como los grandes modelos siguen siendo fiables.

En Saving Planets: Introducción a la ingeniería aeroespacial

Fuentes: NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html) · El tutorial oficial de Python (https://docs.python.org/3/tutorial/)

Recursos de aprendizaje gratuitos

- El tutorial oficial de Python (https://docs.python.org/3/tutorial/): Sección 4.8: definir funciones.
- Harvard CS50's Introduction to Programming with Python (https://cs50.harvard.edu/python/): Funciones y variables.

☐ **Terminé esta unidad**

Listas, diccionarios y cadenas

Almacenar colecciones de datos: listas para secuencias, diccionarios para búsquedas y cadenas para texto.

Objetivos de aprendizaje

Al terminar esta unidad podrás:

- Indexar, rebanar y modificar listas.
- Usar len, sum, min, max y sorted.
- Construir y consultar diccionarios.
- Dividir cadenas y contar palabras.

Conceptos clave

- Una lista guarda una secuencia ordenada: flows[0] es el primer elemento, flows[-1] el último, flows[2:5] una porción.
- Las listas son mutables: append, insert, remove y sort las cambian en su lugar.
- Un diccionario asigna claves a valores: gistemp[2024] → 1.28. Las búsquedas son rápidas sin importar su tamaño.
- Las comprensiones de listas construyen listas de forma concisa: [x * 2 for x in xs if x > 0].
- Las cadenas son secuencias de caracteres con métodos como split, strip, lower y startswith.

Ecuaciones clave

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

```
x=1n_i=1^n x_i
```

Media de una lista (ejemplo 4.1)

Ejemplos resueltos paso a paso

Ejemplo 4.1 Resume diez días de caudal medio diario del USGS para el Potomac en Little Falls (1–10 de junio de 2025, en pies cúbicos por segundo).

```
flows = [44700, 40200, 29100, 22700, 17300, 14100, 12600, 12200, 14300, 22600]
print(len(flows), max(flows), min(flows))
print(sum(flows) / len(flows))
print(sorted(flows)[:3])
```

1. len, max y min funcionan con cualquier lista.

2. sum/len es la media; sorted devuelve una lista nueva.

Salida

```
10 44700 12200
22980.0
[12200, 12600, 14100]
```

Respuesta: Media 22980 cfs; rango 12200–44700 cfs.

Ejemplo 4.2 Encuentra el año más cálido en un diccionario de anomalías de NASA GISTEMP.

```
gistemp = {2016: 1.01, 2020: 1.01, 2023: 1.17, 2024: 1.28, 2025: 1.18}
warmest = max(gistemp, key=gistemp.get)
print(warmest, gistemp[warmest])
print([y for y, a in gistemp.items() if a > 1.1])
```

1. max con key=gistemp.get compara valores.
2. Una comprensión filtra elementos.

Salida

```
2024 1.28
[2023, 2024, 2025]
```

Respuesta: 2024, con 1.28 °C; 2023–2025 están todos por encima de 1.1 °C.

Ejemplo 4.3 Cuenta palabras con un diccionario.

```
words = "reduce reuse recycle reduce repair reduce".split()
counts = {}
for w in words:
    counts[w] = counts.get(w, 0) + 1
print(counts)
```

1. get(w, 0) devuelve 0 para una palabra nueva.

Salida

```
{'reduce': 3, 'reuse': 1, 'recycle': 1, 'repair': 1}
```

Respuesta: «reduce» aparece 3 veces.

Problemas de práctica

Resuelve cada uno primero en papel y luego comprueba.

1. ¿Qué es [10, 20, 30, 40][1:3]?

   [20, 30].

2. ¿Cómo añades 2026: 1.2 a gistemp?

   gistemp[2026] = 1.2.

3. Eleva al cuadrado cada número de xs con una comprensión.

   [x**2 for x in xs].

Ideas erróneas comunes

Mito: Las listas empiezan en el índice 1.
En realidad: Empiezan en 0.

Mito: sorted(xs) cambia xs.
En realidad: sorted devuelve una lista nueva; xs.sort() la cambia.

Mito: Las claves de un diccionario pueden repetirse.
En realidad: Cada clave es única; asignar de nuevo reemplaza el valor.

Tarjetas de estudio

Lista

Secuencia ordenada y mutable.

Índice / porción

xs[i], xs[a:b].

Diccionario

Mapa clave → valor.

Comprensión

[f(x) for x in xs if cond].

Tupla

str.split

Secuencia inmutable.

Texto → lista de trozos.

Cuestionario de autoevaluación

1. xs[-1] es...

el primer elemento

el último elemento

un error

la longitud

el último elemento. Los índices negativos cuentan desde el final.

2. len({"a": 1, "b": 2}) =

1

2

4

3

2. Dos claves.

3. "a,b,c".split(",") da...

"abc"

["a", "b", "c"]

("a","b","c")

3

["a", "b", "c"]. Una lista de cadenas.

4. Las listas son...

inmutables

mutables

diccionarios

siempre ordenadas

mutables. Pueden cambiar en su lugar.

Repetir este cuestionario

Cómo ayuda esto a salvar el planeta

El USGS publica el caudal diario de miles de medidores de ríos. Los diez números del ejemplo 4.1 son reales: un pulso de caudal alto en el Potomac que baja durante una semana, el tipo de registro que se usa para el abastecimiento de agua y la planificación de inundaciones.

En Saving Planets: Página de Ríos

Fuentes: USGS: Potomac River near Washington, DC (Little Falls), site 01646500 (https://waterdata.usgs.gov/monitoring-location/USGS-01646500/) · NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/)

Recursos de aprendizaje gratuitos

- El tutorial oficial de Python (https://docs.python.org/3/tutorial/): Sección 5: estructuras de datos.
- USGS Water Services (API de datos) (https://waterservices.usgs.gov/): Cómo solicitar datos al USGS.

☐ **Terminé esta unidad**

Archivos y datos CSV

Leer y escribir archivos, analizar datos CSV y manejar filas defectuosas sin que el programa falle.

Objetivos de aprendizaje

Al terminar esta unidad podrás:

- Abrir, leer y escribir archivos de texto con with.
- Analizar CSV con el módulo csv.
- Convertir cadenas en números y manejar errores con try/except.
- Descargar un conjunto de datos real y analizarlo.

Conceptos clave

- with open(name) as f: abre un archivo y lo cierra automáticamente.
- CSV (valores separados por comas) es el formato más común de los datos públicos. csv.DictReader convierte cada fila en un diccionario con las claves del encabezado.
- Todo lo que se lee de un archivo es texto: convierte con int() y float().
- Los datos reales tienen comentarios, vacíos y valores erróneos. try/except ValueError te permite omitirlos o registrarlos.
- El archivo de medias anuales de Mauna Loa de NOAA empieza con líneas de comentario #, y luego year,mean,unc.

Ecuaciones clave

$$\text{rate} = \frac{x_{last} - x_{first}}{t_{last} - t_{first}}$$

```
rate=_last-x_firstt_last-t_first
```

Tasa de cambio media (ejemplo 5.1)

$$1 \text{ cfs} = 0.0283168 \text{ m}^3/\text{s}$$

```
1\ cfs=0.0283168\ m^3/s
```

Conversión de unidades (ejemplo 5.2)

Ejemplos resueltos paso a paso

Ejemplo 5.1 Analiza las últimas cinco filas del CSV de medias anuales de Mauna Loa de NOAA y halla la tasa media de aumento.

```
import csv, io
text = """year,mean,unc
2021,416.41,0.12
2022,418.53,0.12
2023,421.08,0.12
2024,424.61,0.12
2025,427.35,0.12
"""
rows = list(csv.DictReader(io.StringIO(text)))
years = [int(r["year"]) for r in rows]
ppm = [float(r["mean"]) for r in rows]
print(years[0], years[-1], ppm[-1])
print(round((ppm[-1] - ppm[0]) / (years[-1] - years[0]), 2), "ppm per year")
```

1. io.StringIO permite que csv lea una cadena como si fuera un archivo.
2. Convierte los campos de texto a int y float.

Salida

```
2021 2025 427.35
2.73 ppm per year
```

Respuesta: 2.73 ppm al año de 2021 a 2025.

Ejemplo 5.2 Escribe un pequeño CSV de caudales del USGS, vuelve a leerlo y conviértelo a m³/s.

```
with open("flows.csv", "w") as f:
    f.write("date,cfs\n2025-06-01,44700\n2025-06-02,40200\n2025-06-03,29100\n")
with open("flows.csv") as f:
    next(f)                      # skip the header
    for line in f:
        date, cfs = line.strip().split(",")
        print(date, round(int(cfs) * 0.0283168, 1), "m3/s")
```

1. next(f) salta la línea de encabezado.
2. strip quita el salto de línea; split separa los campos.

Salida

```
2025-06-01 1265.8 m3/s
2025-06-02 1138.3 m3/s
2025-06-03 824.0 m3/s
```

Respuesta: 44700 cfs ≈ 1265.8 m^3/s.

Ejemplo 5.3 Lee líneas con comentarios y valores erróneos sin que el programa falle.

```
lines = ["# comment", "year,mean", "2024,424.61", "2025,bad", "2026,"]
good = []
for line in lines:
    if line.startswith("#") or line.startswith("year"):
        continue
    year, value = line.split(",")
    try:
        good.append((int(year), float(value)))
    except ValueError:
        print("skipping", repr(line))
print(good)
```

1. continue salta los comentarios y el encabezado.
2. float("bad") y float("") lanzan ValueError, que se captura.

Salida

```
skipping '2025,bad'
skipping '2026,'
[(2024, 424.61)]
```

Respuesta: Solo se conserva la fila válida; las erróneas se informan.

Problemas de práctica

Resuelve cada uno primero en papel y luego comprueba.

1. ¿Cómo lees el archivo completo de NOAA en Colab?

   Usa urllib.request.urlopen(url) o descárgalo, y salta las líneas que empiezan por #.

2. ¿Qué da csv.DictReader para cada fila?

   Un dict con los nombres de columna como claves y valores de tipo cadena.

3. ¿Por qué usar with open(...)?

   El archivo se cierra automáticamente, incluso si ocurre un error.

Ideas erróneas comunes

Mito: Los números de un CSV son números.

En realidad: Se leen como cadenas hasta que se convierten.

Mito: Las filas defectuosas hacen inútil el conjunto de datos.

En realidad: Omítelas y regístralas, y luego informa cuántas hay.

Mito: Abrir un archivo en modo "w" añade al final.

En realidad: "w" sobrescribe; "a" añade al final.

Tarjetas de estudio

with open()

Abre un archivo y lo cierra automáticamente.

CSV

Valores separados por comas.

csv.DictReader

Filas como dicts.

try / except

Manejar errores.

ValueError

Conversión errónea.

Modo "w" / "a"

Sobrescribir / añadir.

Cuestionario de autoevaluación

1. float("3.5") da...

"3.5"

3.5

3

un error

3.5. Convierte texto en float.

2. float("") lanza...

TypeError

ValueError

KeyError

nada

ValueError. Una cadena vacía no es un número.

3. El modo de archivo "a"...

sobrescribe

añade al final

lee

borra

añade al final. Añadir.

4. Las claves de csv.DictReader vienen de...

los números de fila

la fila de encabezado

el nombre del archivo

al azar

la fila de encabezado. Primera línea.

Repetir este cuestionario

Cómo ayuda esto a salvar el planeta

NOAA, NASA y USGS comparten sus mediciones como archivos CSV gratuitos. Saber leerlos tú mismo te permite contrastar los titulares con los datos originales.

En Saving Planets: Cálculo II

Fuentes: NOAA GML: Mauna Loa CO_2 annual means (CSV) (https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_annmean_mlo.csv) · Documentación de Python: el módulo csv (https://docs.python.org/3/library/csv.html)

Recursos de aprendizaje gratuitos

- Documentación de Python: el módulo csv (https://docs.python.org/3/library/csv.html): Leer y escribir CSV.
- NOAA GML: Mauna Loa CO_2 annual means (CSV) (https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_annmean_mlo.csv): El archivo real.
- El tutorial oficial de Python (https://docs.python.org/3/tutorial/): Sección 7.2: archivos.

☐ **Terminé esta unidad**

NumPy y Matplotlib

Calcular con arreglos enteros a la vez con NumPy, ajustar tendencias y hacer gráficos claros con Matplotlib.

Objetivos de aprendizaje

Al terminar esta unidad podrás:

- Crear arreglos de NumPy y usar aritmética vectorizada.
- Usar mean, diff, máscaras booleanas y polyfit.
- Graficar datos con ejes rotulados y títulos.
- Guardar figuras en archivos.

Conceptos clave

- Un arreglo de NumPy guarda números de un solo tipo y opera con todos a la vez: cfs * 0.0283168 convierte cada valor.
- np.diff da las diferencias sucesivas; una máscara booleana como cfs > 20000 selecciona o cuenta valores.
- np.polyfit(x, y, 1) ajusta una recta y devuelve la pendiente y la ordenada al origen.
- pyplot de Matplotlib dibuja figuras: fig, ax = plt.subplots(); ax.plot(x, y); rotula los ejes con unidades; fig.savefig guarda la imagen.
- Un buen gráfico tiene ejes rotulados con unidades, un título y una fuente de datos citada.

Ecuaciones clave

$$y \approx mx + b$$

Recta de mínimos cuadrados con np.polyfit

$$V = \sum Q_i \, \Delta t$$

```
V= Q_i Δ t
```

Volumen a partir de caudales diarios (ejemplo 6.2)

Ejemplos resueltos paso a paso

Ejemplo 6.1 Usa NumPy con las medias anuales de Mauna Loa 2021–2025: diferencias, media y tendencia.

```
import numpy as np
years = np.array([2021, 2022, 2023, 2024, 2025])
ppm = np.array([416.41, 418.53, 421.08, 424.61, 427.35])
print(np.diff(ppm))
print(round(ppm.mean(), 3), round(np.diff(ppm).mean(), 3))
slope, intercept = np.polyfit(years, ppm, 1)
print(round(slope, 3))
```

1. np.diff da cuatro aumentos anuales.
2. La pendiente de polyfit es la tasa de mejor ajuste.

Salida

```
[2.12 2.55 3.53 2.74]
421.596 2.735
2.796
```

Respuesta: Aumento de mejor ajuste ≈ 2.80 ppm al año en 2021–2025.

Ejemplo 6.2 Convierte diez días de caudal del Potomac a m³/s, cuenta los días de caudal alto y halla el volumen total.

```
import numpy as np
cfs = np.array([44700, 40200, 29100, 22700, 17300, 14100, 12600, 12200, 14300, 22600])
m3s = cfs * 0.0283168
print(np.round(m3s[:3], 1))
print(int((cfs > 20000).sum()), "days above 20,000 cfs")
print(round(m3s.sum() * 86400 / 1e9, 3), "km3 in 10 days")
```

1. Multiplica todo el arreglo de una vez.
2. (cfs > 20000).sum() cuenta los valores True.
3. Suma de m³/s × 86400 s por día.

Salida

```
[1265.8 1138.3  824. ]
5 days above 20,000 cfs
0.562 km3 in 10 days
```

Respuesta: Cinco días por encima de 20000 cfs; pasaron unos 0.56 km³ de agua en diez días.

Ejemplo 6.3 Grafica una selección de anomalías anuales de NASA GISTEMP y guarda la figura.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
years = [1880, 1951, 1980, 2000, 2016, 2020, 2023, 2024, 2025]
anom = [-0.17, -0.07, 0.26, 0.39, 1.01, 1.01, 1.17, 1.28, 1.18]
fig, ax = plt.subplots()
ax.plot(years, anom, "o-")
ax.set_xlabel("Year"); ax.set_ylabel("Anomaly (C)")
ax.set_title("NASA GISTEMP v4, selected years")
fig.savefig("gistemp.png")
print(len(ax.lines[0].get_xdata()), "points saved")
```

1. matplotlib.use("Agg") permite que el código guarde imágenes sin pantalla (Colab muestra los gráficos en línea de todos modos).
2. Rotula los ejes y el título; guarda con savefig.

Salida

```
9 points saved
```

Respuesta: Un gráfico de líneas guardado con 9 puntos; abre gistemp.png para verlo.

Problemas de práctica

Resuelve cada uno primero en papel y luego comprueba.

1. ¿Qué da np.array([1, 2, 3]) * 2?

   array([2, 4, 6]).

2. ¿Cómo seleccionas los valores mayores que 1.0 en un arreglo a?

   a[a > 1.0].

3. ¿Qué llamada añade un rótulo de eje?

   ax.set_xlabel("Year").

Ideas erróneas comunes

Mito: Los arreglos de NumPy y las listas se suman igual.

En realidad: [1, 2] + [3, 4] concatena; los arreglos se suman elemento a elemento.

Mito: Una recta ajustada demuestra una causa.

En realidad: Resume una tendencia; las causas requieren comprensión física.

Mito: Los gráficos sin unidades están bien.

En realidad: Rotula siempre las unidades y cita la fuente.

Tarjetas de estudio

np.array

Arreglo numérico rápido.

Vectorizado

Operaciones con arreglos completos.

Máscara booleana

a[a > x].

np.polyfit

Ajuste polinómico por mínimos cuadrados.

plt.subplots

Crea una figura y ejes.

savefig

Guarda un gráfico.

Cuestionario de autoevaluación

1. np.diff([1, 4, 9]) =

[3, 5]

[1, 4, 9]

[0, 3, 5]

14

[3, 5]. Diferencias sucesivas.

2. np.polyfit(x, y, 1) devuelve...

la media

la pendiente y la ordenada al origen

un gráfico

el máximo

la pendiente y la ordenada al origen. Ajuste de grado 1.

3. (a > 0).sum() cuenta…

los valores positivos

todos los valores

los ceros

nada

los valores positivos. True cuenta como 1.

4. fig.savefig("x.png")…

imprime

guarda la figura

la borra

muestra el código

guarda la figura. Escribe un archivo de imagen.

Repetir este cuestionario

Cómo ayuda esto a salvar el planeta

El análisis GISTEMP de la NASA se publica con sus datos para que cualquiera pueda reproducir el registro de temperatura global. Graficarlo tú mismo, con unidades rotuladas y la fuente, es la forma honesta de compartir datos climáticos.

En Saving Planets: Datos climáticos en este sitio: CO2M

Fuentes: NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/) · NumPy: the absolute basics for beginners (https://numpy.org/doc/stable/user/absolute_beginners.html)

Recursos de aprendizaje gratuitos

- NumPy: the absolute basics for beginners (https://numpy.org/doc/stable/user/absolute_beginners.html): Arreglos para principiantes.
- Matplotlib: pyplot tutorial (https://matplotlib.org/stable/tutorials/pyplot.html): Fundamentos de gráficos.
- NASA GISTEMP v4 global means (CSV) (https://data.giss.nasa.gov/gistemp/tabledata_v4/GLB.Ts+dSST.csv): Los datos reales.

☐ **Terminé esta unidad**

Depuración y pruebas

Encontrar errores de forma sistemática, leer tracebacks y escribir pruebas que detecten los fallos antes de que importen.

Objetivos de aprendizaje

Al terminar esta unidad podrás:

- Leer un traceback y localizar la línea que falla.
- Usar print y aserciones para acotar los errores.
- Lanzar errores claros ante entradas incorrectas.
- Escribir pruebas unitarias con assert y pytest.

Conceptos clave

- Lee los mensajes de error desde abajo: la última línea nombra el tipo de error y las de arriba muestran dónde ocurrió.
- Depura de forma científica: formula una hipótesis, pruébala con una entrada pequeña, cambia una sola cosa.
- Los casos límite (listas vacías, cero, números negativos, datos faltantes) causan la mayoría de los errores.
- Un assert declara lo que debe ser cierto; un assert que falla detiene el programa en el error.
- Las pruebas son funciones pequeñas que comprueban respuestas conocidas; pytest ejecuta automáticamente toda función llamada test_....

Ecuaciones clave

$$|x_{computed} - x_{expected}| < \varepsilon$$

Prueba con tolerancia para floats

Ejemplos resueltos paso a paso

Ejemplo 7.1 Encuentra el error de una función de media llamada con una lista vacía.

```
def mean(xs):
    return sum(xs) / len(xs)

try:
    print(mean([]))
except ZeroDivisionError as e:
    print("bug found:", e)
```

1. Dividir por len([]) = 0 lanza ZeroDivisionError.
2. Capturarlo muestra el mensaje.

Salida

```
bug found: division by zero
```

Respuesta: La función debe manejar una entrada vacía.

Ejemplo 7.2 Corrígela con un error claro y añade pruebas con aserciones.

```
def mean(xs):
    if not xs:
        raise ValueError("mean() of empty list")
    return sum(xs) / len(xs)

assert mean([1, 2, 3]) == 2
assert abs(mean([0.1, 0.2]) - 0.15) < 1e-12
try:
    mean([])
except ValueError:
    print("all tests passed")
```

1. Lanza ValueError con un mensaje ante una entrada vacía.
2. Los resultados float se comparan con una tolerancia.

Salida

```
all tests passed
```

Respuesta: Todas las pruebas pasan.

Ejemplo 7.3 Escribe una prueba al estilo pytest para f_to_c con puntos conocidos.

```
def f_to_c(f):
    return (f - 32) * 5 / 9

def test_f_to_c():
    assert f_to_c(32) == 0
    assert f_to_c(212) == 100
    assert abs(f_to_c(-40) - (-40)) < 1e-12
```

```
test_f_to_c()
print("ok")
```

1. Pares conocidos: 32 °F = 0 °C, 212 °F = 100 °C, −40 = −40.
2. En un archivo, pytest descubriría test_f_to_c automáticamente; aquí la llamamos.

Salida

```
ok
```

Respuesta: La función pasa las tres comprobaciones.

Problemas de práctica

Resuelve cada uno primero en papel y luego comprueba.

1. ¿Qué suele significar «IndexError: list index out of range»?

   Usaste un índice ≥ len(list), a menudo un error por uno.

2. Escribe un assert de que x es positivo.

   assert x > 0, "x must be positive".

3. Nombra tres casos límite para una función que promedia datos.

   Lista vacía, un solo valor y valores faltantes o no numéricos.

Ideas erróneas comunes

Mito: Si se ejecuta, es correcto.

En realidad: El código puede ejecutarse y dar respuestas erróneas; pruébalo contra resultados conocidos.

Mito: Compara floats con ==.

En realidad: Usa una tolerancia.

Mito: Las pruebas son solo para proyectos grandes.

En realidad: Las pruebas pequeñas ahorran tiempo desde el primer día.

Tarjetas de estudio

Traceback	**Caso límite**
Informe de error; se lee desde abajo.	Entrada inusual en el borde.

assert

Se detiene si una condición es falsa.

Prueba unitaria

Comprueba una función contra respuestas conocidas.

pytest

Ejecutor de pruebas.

raise

Señala un error deliberadamente.

Cuestionario de autoevaluación

1. La línea más útil de un traceback suele ser...

- la primera
- la última
- la del medio
- ninguna

la última. Nombra el error.

2. pytest ejecuta las funciones llamadas...

- run_*
- test_*
- check_*
- main

test_*. Convención.

3. Un assert que falla lanza...

- ValueError
- AssertionError

nada

TypeError

AssertionError. AssertionError.

4. Un buen primer paso de depuración es...

reescribirlo todo

reproducirlo con una entrada pequeña

añadir print al azar por todas partes

ignorarlo

reproducirlo con una entrada pequeña. Reduce el problema.

Repetir este cuestionario

Cómo ayuda esto a salvar el planeta

El código científico que orienta políticas debe comprobarse. Este curso predica con el ejemplo: cada ejemplo de código de esta página lo ejecuta nuestro script de compilación, y su salida debe coincidir exactamente antes de publicar la página.

En Saving Planets: Cursos

Fuentes: Documentación de pytest (https://docs.pytest.org/) · El tutorial oficial de Python (https://docs.python.org/3/tutorial/)

Recursos de aprendizaje gratuitos

- Documentación de pytest (https://docs.pytest.org/): Escribir pruebas.
- El tutorial oficial de Python (https://docs.python.org/3/tutorial/): Sección 8: errores y excepciones.

Ver la clase

Lecture 7 testing debugging exceptions and assertions (https://ocw.mit.edu/courses/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/resources/lecture-7-testing-debugging-exceptions-and-assertions/) (en inglés)

MIT 6.0001 Introduction to Computer Science and Programming in Python (Fall 2016), MIT OpenCourseWare.
Licencia: CC BY-NC-SA 4.0 (https://creativecommons.org/licenses/by-nc-sa/4.0/) · Condiciones de uso (https://ocw.mit.edu/pages/pri

vacy-and-terms-of-use/). Enlace solo para estudio gratuito; Saving Planets no está afiliado a esta universidad ni cuenta con su respaldo.

- [ ] **Terminé esta unidad**

Algoritmos y complejidad

Búsqueda, cálculo de raíces y cómo crece el tiempo de ejecución con el tamaño de la entrada: pensar en O grande.

Objetivos de aprendizaje

Al terminar esta unidad podrás:

- Implementar la búsqueda lineal y la búsqueda binaria.
- Usar la bisección para resolver ecuaciones.
- Estimar la complejidad: O(1), O(log n), O(n), O(n²).
- Elegir estructuras de datos por velocidad.

Conceptos clave

- Un algoritmo es un método preciso paso a paso. Su costo es cómo crece el número de pasos con el tamaño de la entrada n.
- La búsqueda lineal revisa los elementos uno a uno: O(n). La búsqueda binaria parte en dos una lista ordenada en cada paso: O(log n).
- La bisección halla una raíz de f en [a, b], donde f cambia de signo, partiendo el intervalo en dos; tras n pasos el intervalo es (b − a)/2ⁿ.
- Los bucles anidados sobre n elementos son O(n²): bien para 1000 elementos, inviable para un millón.
- Las búsquedas en diccionarios y conjuntos son O(1) en promedio; en listas, O(n).

Ecuaciones clave

$$\text{steps}_{binary} \approx \lceil \log_2 n \rceil$$

Pasos de la búsqueda binaria

$$n = \left\lceil \log_2 \frac{b-a}{\text{tol}} \right\rceil$$

```
n=_2-atol
```

Iteraciones de la bisección

Ejemplos resueltos paso a paso

Ejemplo 8.1 Busca el último elemento de una lista ordenada de un millón de números pares, de las dos formas.

```
def linear_search(xs, target):
    for i, x in enumerate(xs):
        if x == target:
            return i
    return -1

def binary_search(xs, target):
    lo, hi, steps = 0, len(xs) - 1, 0
    while lo <= hi:
        steps += 1
        mid = (lo + hi) // 2
        if xs[mid] == target:
            return mid, steps
        if xs[mid] < target:
            lo = mid + 1
        else:
            hi = mid - 1
    return -1, steps

data = list(range(0, 2_000_000, 2))
print(linear_search(data, 1_999_998))
print(binary_search(data, 1_999_998))
```

1. La búsqueda lineal revisa 1000000 elementos para llegar al índice 999999.
2. La búsqueda binaria solo necesita 20 pasos.

Salida

```
999999
(999999, 20)
```

Respuesta: 20 pasos frente a un millón: $\log_2(10^6) \approx 20$.

Ejemplo 8.2 Halla $\sqrt{2}$ por bisección en $x^2 - 2$ desde $[0, 2]$ hasta 10^{-6}.

```
def bisect_root(f, a, b, tol=1e-6):
    n = 0
    while b - a > tol:
        m = (a + b) / 2
        if f(a) * f(m) <= 0:
            b = m
        else:
            a = m
        n += 1
    return (a + b) / 2, n
```

```
root, n = bisect_root(lambda x: x**2 - 2, 0, 2)
print(round(root, 6), n)
```

1. Cada paso parte el intervalo en dos.
2. $2/2^{21} < 10^{-6}$ tras 21 pasos.

Salida

```
1.414214 21
```

Respuesta: 1.414214 tras 21 iteraciones.

Ejemplo 8.3 Compara el crecimiento de $\log_2 n$ y n^2.

```
import math
for n in (10, 1_000, 1_000_000):
    print(n, math.ceil(math.log2(n)), n * n)
```

1. math.ceil(math.log2(n)) es el número de pasos de la búsqueda binaria.

Salida

```
10 4 100
1000 10 1000000
1000000 20 1000000000000
```

Respuesta: Para $n = 10^6$, $\log_2 n$ es 20 mientras que n^2 es 10^{12}.

Problemas de práctica

Resuelve cada uno primero en papel y luego comprueba.

1. ¿Cuántos pasos de búsqueda binaria para 1 mil millones de elementos ordenados?

   $\lceil \log_2 10^9 \rceil = 30$.

2. ¿Cuál es la complejidad de sumar una lista?

   $O(n)$.

3. Bisección en $[0, 1]$ hasta 10^{-3}: ¿cuántos pasos?

   $\lceil \log_2 1000 \rceil = 10$.

Ideas erróneas comunes

Mito: Las computadoras más rápidas hacen irrelevante la complejidad.

En realidad: Un método $O(n^2)$ sobre un millón de elementos son 10^{12} pasos; ninguna computadora lo arregla.

Mito: La búsqueda binaria funciona con cualquier lista.

En realidad: Solo con datos ordenados.

Mito: La O grande da el tiempo exacto.

En realidad: Describe el crecimiento, ignorando constantes.

Tarjetas de estudio

Algoritmo

Método preciso paso a paso.

Big-O

Ritmo de crecimiento del costo.

Búsqueda lineal

O(n).

Búsqueda binaria

O(log n), datos ordenados.

Bisección

Cálculo de raíces partiendo en dos.

$O(n^2)$

Bucles anidados.

Cuestionario de autoevaluación

1. La búsqueda binaria es...

O(n)

O(log n)

$O(n^2)$

O(1)

O(log n). Parte en dos en cada paso.

2. La bisección requiere...

una derivada

un cambio de signo en [a, b]

datos ordenados

enteros

un cambio de signo en [a, b]. Teorema del valor intermedio.

3. Un bucle dentro de otro sobre n elementos es...

O(n)

$O(n^2)$

O(log n)

O(1)

$O(n^2)$. n × n.

4. Una búsqueda media en un dict es...

O(n)

O(1)

$O(n^2)$

O(log n)

O(1). Hashing.

Repetir este cuestionario

Cómo ayuda esto a salvar el planeta

Los algoritmos eficientes permiten procesar la avalancha de datos de observación de la Tierra, desde satélites hasta miles de medidores de ríos. Elegir O(n log n) en lugar de $O(n^2)$ puede convertir

una semana de cálculo en segundos, y menos cálculo también significa menos energía.

En Saving Planets: Cálculo I

Fuentes: USGS Water Services (API de datos) (https://waterservices.usgs.gov/) · MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/)

Recursos de aprendizaje gratuitos

- MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/): Clases sobre búsqueda, bisección y complejidad.
- Harvard CS50's Introduction to Programming with Python (https://cs50.harvard.edu/python/): Práctica de resolución de problemas.

☐ **Terminé esta unidad**

Simulación numérica

Avanzar modelos físicos en el tiempo: un proyectil, una órbita y una población, el núcleo de todo modelo climático y de ingeniería.

Objetivos de aprendizaje

Al terminar esta unidad podrás:

- Usar pasos de tiempo (Euler y Euler-Cromer) para simular el movimiento.
- Comprobar las simulaciones con respuestas exactas.
- Simular una órbita circular y una población logística.
- Explorar cómo el tamaño del paso afecta a la precisión.

Conceptos clave

- Una simulación actualiza el estado (posición, velocidad, población) en pequeños pasos de tiempo dt usando las tasas de cambio: x_new = x + v·dt.
- Euler-Cromer actualiza primero la velocidad y luego la posición con la velocidad nueva; mantiene las órbitas estables mucho mejor que el Euler simple.
- Valida siempre: compara con una solución exacta en un caso donde exista (sin resistencia del aire, órbita circular, fórmula logística).
- Un dt más pequeño es más preciso pero más lento; divide dt a la mitad y mira si cambia la respuesta.
- Los modelos meteorológicos y climáticos son versiones enormes de estos bucles, que avanzan las ecuaciones de la física sobre una malla global.

Ecuaciones clave

$$v_{k+1} = v_k + a_k\,\Delta t, \quad x_{k+1} = x_k + v_{k+1}\,\Delta t$$

```
v_k+1=v_k+a_k Δ t, x_k+1=x_k+v_k+1 Δ t
```

Paso de Euler-Cromer

$$R = \frac{v_0^2 \sin 2\theta}{g}$$

```
R=(v_0^22θ)/(g)
```

Alcance de un proyectil sin resistencia

$$\vec{a} = -\frac{GM}{r^3}\vec{r}$$

a=-(GM)/(r^3) r

Gravedad para una órbita

$$P_{k+1} = P_k + h\,r P_k \left(1 - \frac{P_k}{K}\right)$$

P_k+1=P_k+h rP_k(1-_kK)

Modelo logístico, paso de Euler

Ejemplos resueltos paso a paso

Ejemplo 9.1 Simula un proyectil a 20 m/s lanzado a 45° y compáralo con el alcance exacto.

```
import math
v0, angle, g, dt = 20.0, math.radians(45), 9.81, 0.001
x, y = 0.0, 0.0
vx, vy = v0 * math.cos(angle), v0 * math.sin(angle)
while y >= 0:
    vy -= g * dt
    x += vx * dt
    y += vy * dt
print(round(x, 2), "m simulated;", round(v0**2 / g, 2), "m exact")
```

1. Avanza la velocidad y luego la posición hasta que aterrice.
2. Alcance exacto = v_0^2 sin 90°/g.

Salida

```
40.77 m simulated; 40.77 m exact
```

Respuesta: Ambos dan 40.77 m con dos decimales con dt = 0.001 s.

Ejemplo 9.2 Simula una órbita a 400 km sobre la Tierra con pasos de 1 segundo.

```
import math
GM = 3.986e14
r0 = 6.771e6
x, y = r0, 0.0
vx, vy = 0.0, math.sqrt(GM / r0)
dt, t = 1.0, 0
period = 2 * math.pi * math.sqrt(r0**3 / GM)
while t < period:
    r3 = (x * x + y * y) ** 1.5
```

```
    vx -= GM * x / r3 * dt          # Euler-Cromer: update velocity first
    vy -= GM * y / r3 * dt
    x += vx * dt
    y += vy * dt
    t += dt
print(round(period / 60, 1), "min;", round(math.hypot(x, y) / 1000), "km from Earth's center
```

1. Empieza en r_0 con la velocidad circular $\sqrt{(GM/r_0)}$.
2. Actualiza la velocidad a partir de la gravedad y luego la posición.

Salida

```
92.4 min; 6771 km from Earth's center
```

Respuesta: Tras un periodo (92.4 minutos), la nave sigue a 6771 km del centro de la Tierra; la órbita se mantiene circular.

Ejemplo 9.3 Avanza una población logística de peces (r = 0.5, K = 1000, P_0 = 100) durante cuatro años.

```
r, K, P, h = 0.5, 1000, 100.0, 1.0
for year in range(1, 5):
    P += h * r * P * (1 - P / K)
    print(year, round(P, 1))
```

1. Cada año suma rP(1 − P/K).

Salida

```
1 145.0
2 207.0
3 289.1
4 391.8
```

Respuesta: 145.0, 207.0, 289.1, 391.8, como en la práctica A de Cálculo II (allí 391.9, una diferencia de redondeo).

Problemas de práctica

Resuelve cada uno primero en papel y luego comprueba.

1. ¿Qué le pasa al error del ejemplo 9.1 si dt es 0.1 s?

   El aterrizaje se detecta tarde, así que el error de alcance crece; pruébalo y compara.

2. ¿Por qué el Euler simple hace que una órbita se abra en espiral hacia afuera?

   Añade un poco de energía en cada paso; Euler-Cromer casi la conserva.

3. Añade una resistencia −k v v⃗ al ejemplo 9.1. ¿Qué le pasa al mejor ángulo?

   Baja de 45° (práctica D).

Ideas erróneas comunes

Mito: Una simulación que se ejecuta es correcta.

En realidad: Valida con casos exactos y comprueba que dividir dt a la mitad apenas cambia la respuesta.

Mito: Un dt más pequeño siempre es mejor.

En realidad: Cuesta tiempo y puede añadir error de redondeo; elige dt con pruebas de convergencia.

Mito: Los modelos predicen el futuro con exactitud.

En realidad: Son aproximaciones cuyos supuestos deben declararse.

Tarjetas de estudio

Paso de tiempo dt

Tamaño de cada actualización.

Método de Euler

x += v·dt.

Euler-Cromer

Primero la velocidad, luego la posición.

Validación

Comprobar con respuestas conocidas.

Convergencia

La respuesta deja de cambiar al reducir dt.

Estado

Variables que describen el sistema ahora.

Cuestionario de autoevaluación

1. Euler-Cromer actualiza...

primero la posición

primero la velocidad y luego la posición con la velocidad nueva

solo la velocidad

nada

primero la velocidad y luego la posición con la velocidad nueva. Estable para órbitas.

2. Para validar una simulación...

confiar en ella

compararla con una solución exacta

usar un dt enorme

quitar la física

compararla con una solución exacta. Casos conocidos.

3. Alcance exacto a 20 m/s, 45°, g = 9.81:

20.4 m

40.8 m

81.5 m

9.8 m

40.8 m. 400/9.81.

4. Dividir dt a la mitad y no ver cambios significa...

un error

el resultado ha convergido

el modelo está mal

nada

el resultado ha convergido. Prueba de convergencia.

Repetir este cuestionario

Cómo ayuda esto a salvar el planeta

Los modelos de población como el ejemplo 9.3 forman parte de cómo los científicos pesqueros evalúan las poblaciones y fijan límites de captura sostenibles.

La misma idea de pasos de tiempo, ampliada enormemente, es como se construyen los modelos meteorológicos y climáticos.

En Saving Planets: Cálculo II, práctica A · Guía de órbitas

Fuentes: NOAA Fisheries: evaluaciones de poblaciones (https://www.fisheries.noaa.gov/topic/population-assessments) · MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/)

Recursos de aprendizaje gratuitos

- MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/): Clases sobre simulación y modelado.
- NumPy: the absolute basics for beginners (https://numpy.org/doc/stable/user/absolute_beginners.html): Vectorizar simulaciones.

☐ **Terminé esta unidad**

Problemas

Estos problemas están dentro de cada unidad. Hazlos en papel y luego abre la respuesta.

1. Primeros pasos: variables, tipos y expresiones
   4 problemas
2. Control de flujo: condiciones y bucles
   3 problemas
3. Funciones
   3 problemas
4. Listas, diccionarios y cadenas
   3 problemas
5. Archivos y datos CSV
   3 problemas
6. NumPy y Matplotlib
   3 problemas
7. Depuración y pruebas
   3 problemas
8. Algoritmos y complejidad

3 problemas

9. Simulación numérica

   3 problemas

Prácticas de programación

Cuatro prácticas de programación usan datos públicos reales y simulación: el registro de CO_2 de Mauna Loa, el registro mundial de temperatura de la NASA, una estación de aforo del USGS y un proyectil con resistencia del aire.

El código inicial de cada práctica funciona tal como se muestra; los resultados impresos se obtuvieron ejecutándolo. Después descarga el conjunto de datos completo y amplía el análisis.

Referencia de seguridad: El tutorial oficial de Python (https://docs.python.org/3/tutorial/)

Práctica A: el registro de CO_2 de Mauna Loa (NOAA)

SEMANA 6

Objetivo: Carga el CSV de medias anuales de CO_2 de la NOAA, calcula el aumento total y la tasa de crecimiento reciente, y grafica la curva.

$$\text{rate} = \text{slope of } \text{np.polyfit}(\text{year}, \text{ppm}, 1)$$

```
rate=slope of np.polyfit(year, ppm, 1)
```

Tasa de crecimiento con un ajuste lineal

Procedimiento

1. Ejecuta el código inicial (unas cuantas filas reales del archivo de la NOAA).
2. Descarga el CSV completo de la NOAA (enlace abajo), omite las líneas #, y carga todos los años.
3. Grafica las ppm frente al año con ejes rotulados y la fuente.
4. Ajusta rectas a 1960–1969 y 2016–2025. ¿Cómo ha cambiado la tasa?

```
import csv, io
import numpy as np
text = """year,mean,unc
1959,315.98,0.12
1960,316.91,0.12
2020,414.21,0.12
2021,416.41,0.12
2022,418.53,0.12
2023,421.08,0.12
2024,424.61,0.12
2025,427.35,0.12
"""
rows = list(csv.DictReader(io.StringIO(text)))
yr = np.array([int(r["year"]) for r in rows])
ppm = np.array([float(r["mean"]) for r in rows])
print(round(ppm[-1] - ppm[0], 2), "ppm since 1959")
recent = yr >= 2020
slope = np.polyfit(yr[recent], ppm[recent], 1)[0]
print(round(slope, 2), "ppm per year, 2020-2025")
```

Salida

```
111.37 ppm since 1959
2.65 ppm per year, 2020-2025
```

Tabla de datos

Año	CO_2 medio anual (ppm)
1959	315.98
1960	316.91
2020	414.21
2023	421.08
2024	424.61
2025	427.35

Valores reales del archivo de medias anuales de Mauna Loa de NOAA GML, consultado el 24 de oct. de 2026.

Cálculo de ejemplo

1. 427.35 − 315.98 = 111.37 ppm desde 1959.
2. Tasa ajustada 2020–2025 ≈ 2.65 ppm por año.

> **Seguridad**
>
> - Ninguna más allá del uso normal de una computadora.

Alternativa en casa o virtual

- Colab o JupyterLite en un navegador.

NOAA GML: Mauna Loa CO_2 annual means (CSV) (https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_annmean_mlo.csv) · Google Colab (notebooks gratuitos) (https://colab.research.google.com/)

Vínculo con el planeta: Es el registro detrás de la curva de Keeling, la medición continua más larga del aumento del CO_2. NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html)

☐ **Completé este laboratorio (real o virtual)**

Práctica B: temperatura global con NASA GISTEMP

SEMANA 8

Objetivo: Analiza las anomalías anuales de temperatura global de la NASA: promedios, el año más cálido y la tendencia.

$$\Delta T \text{ per decade} = 10 \times \text{slope}$$

```
Δ T\ per decade=10×slope
```

Tendencia por década

Procedimiento

1. Ejecuta el código inicial con las anomalías anuales (J–D) de 2016–2025.
2. Descarga el CSV global completo de GISTEMP; omite su primera línea de título; gestiona los valores faltantes ***.
3. Grafica 1880–2025 y calcula la tendencia desde 1975.
4. Compara un ajuste de 10 años con uno de 50 años y explica por qué las tendencias cortas tienen más ruido.

```
import numpy as np
years = np.arange(2016, 2026)
anom = np.array([1.01, 0.91, 0.85, 0.98, 1.01, 0.85, 0.89, 1.17, 1.28, 1.18])
print(round(anom.mean(), 3))
print(int(years[anom.argmax()]), anom.max())
print(round(np.polyfit(years, anom, 1)[0] * 10, 2), "C per decade (10-year fit)")
```

Salida

```
1.013
2024 1.28
0.32 C per decade (10-year fit)
```

Tabla de datos

Año	Anomalía (°C respecto a 1951–1980)
2016	1.01
2020	1.01
2023	1.17
2024	1.28
2025	1.18

Valores reales de NASA GISTEMP v4 (GLB.Ts+dSST, columna J–D), consultados el 24 de oct. de 2026. La NASA actualiza el análisis, así que descargas posteriores pueden diferir ligeramente.

Cálculo de ejemplo

1. Media 2016–2025 = 1.013 °C.
2. Un ajuste de diez años da unos 0.32 °C por década, pero diez años es poco y hay mucho ruido; usa periodos más largos para la tendencia a largo plazo.

> **Seguridad**
>
> - Ninguna más allá del uso normal de una computadora.

Alternativa en casa o virtual

- Colab o JupyterLite.

NASA GISTEMP v4 global means (CSV) (https://data.giss.nasa.gov/gistemp/tabledata_v4/GLB.Ts+dSST.csv) · NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/)

Vínculo con el planeta: Reproducir tú mismo el registro de la NASA es la mejor respuesta a «¿cómo lo sabemos?» NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/)

☐ **Completé este laboratorio (real o virtual)**

Práctica C: un río a través de una estación del USGS

SEMANA 9

Objetivo: Analiza el caudal diario de la estación del USGS en el río Potomac en Little Falls: unidades, estadísticas, recesión y volumen.

$$V = \sum_{i} Q_i \times 86{,}400 \text{ s}$$

V=_iQ_i×86,400\ s

Volumen a partir de caudales medios diarios

Procedimiento

1. Ejecuta el código inicial con diez días reales (1–10 de junio de 2025).
2. Solicita un año completo a USGS Water Services (format=rdb) para el sitio 01646500 y el parámetro 00060, y analiza el archivo separado por tabulaciones.
3. Grafica el caudal diario, encuentra el pico y calcula el volumen anual en km³.
4. Escribe pruebas para tu función de conversión de unidades.

```
import numpy as np
cfs = np.array([44700, 40200, 29100, 22700, 17300, 14100, 12600, 12200, 14300, 22600])
m3s = cfs * 0.0283168
print(round(m3s.mean(), 1), round(np.median(m3s), 1))
drops = np.diff(cfs)
print(int(drops.min()), "cfs biggest one-day drop")
print(round(m3s.sum() * 86400 / 1e6, 1), "million m3 in 10 days")
```

Salida

```
650.7 564.9
-11100 cfs biggest one-day drop
562.2 million m3 in 10 days
```

Tabla de datos

Fecha (2025)	Caudal medio diario (cfs)
1 de junio	44,700
3 de junio	29,100
5 de junio	17,300
8 de junio	12,200

Fecha (2025)	Caudal medio diario (cfs)
10 de junio	22,600

Valores diarios reales y aprobados del sitio USGS 01646500, consultados el 24 de oct. de 2026.

Cálculo de ejemplo

1. Media 650.7 m³/s, mediana 564.9 m³/s (la media sube por el pico del 1 de junio).
2. El mayor descenso en un día fue de 11100 cfs; pasaron unos 562 millones de m³ en diez días.

> **Seguridad**
>
> - Ninguna más allá del uso normal de una computadora.

Alternativa en casa o virtual

- Colab o JupyterLite.

USGS: Potomac River near Washington, DC (Little Falls), site 01646500 (https://waterdata.usgs.gov/monitoring-location/USGS-01646500/) · USGS Water Services (API de datos) (https://waterservices.usgs.gov/)

Vínculo con el planeta: Las estaciones de aforo como esta sirven para el abastecimiento de agua potable, las alertas de inundación y los caudales ecológicos. USGS: Potomac River near Washington, DC (Little Falls), site 01646500 (https://waterdata.usgs.gov/monitoring-location/USGS-01646500/)

☐ **Completé este laboratorio (real o virtual)**

Práctica D: proyectil con resistencia del aire

SEMANA 14

Objetivo: Amplía la simulación del proyectil con resistencia cuadrática y encuentra el mejor ángulo de lanzamiento a medida que aumenta la resistencia.

$$\vec{a} = -g\hat{y} - k\,|\vec{v}|\,\vec{v}$$

`a=-g y-k | v| v`

Resistencia cuadrática (k por metro)

Procedimiento

1. Ejecuta el código inicial, que prueba ángulos enteros de 20° a 60°.
2. Comprueba que k = 0 da 45° y el alcance exacto.
3. Divide dt a la mitad y confirma que los resultados apenas cambian.
4. Grafica el alcance frente al ángulo para cada k.

```
import math
def fly(v0, deg, k=0.0, dt=0.001, g=9.81):
    vx, vy = v0 * math.cos(math.radians(deg)), v0 * math.sin(math.radians(deg))
    x = y = 0.0
    while y >= 0:
        v = math.hypot(vx, vy)
        vx -= k * v * vx * dt
        vy -= (g + k * v * vy) * dt
        x += vx * dt
        y += vy * dt
    return x
for k in (0.0, 0.01, 0.05):
    best = max(range(20, 61), key=lambda a: fly(20, a, k))
    print(k, best, round(fly(20, best, k), 1))
```

Salida

```
0.0 45 40.8
0.01 43 31.4
0.05 39 17.8
```

Tabla de datos

Resistencia k (1/m)	Mejor ángulo	Alcance (m)
0	45°	40.8

Resistencia k (1/m)	Mejor ángulo	Alcance (m)
0.01	43°	31.4
0.05	39°	17.8

Calculado con el código inicial (v_0 = 20 m/s, dt = 0.001 s).

Cálculo de ejemplo

1. Sin resistencia, el mejor ángulo es 45° y el alcance 40.8 m (exacto 40.77 m).
2. La resistencia acorta el alcance y baja el mejor ángulo.

> **Seguridad**
>
> - Ninguna más allá del uso normal de una computadora.

Alternativa en casa o virtual

- Colab o JupyterLite.

Google Colab (notebooks gratuitos) (https://colab.research.google.com/) · JupyterLite: Python en tu navegador (Pyodide) (https://jupyter.org/try-jupyter/lab/)

Vínculo con el planeta: La resistencia es la misma física que determina el consumo de combustible de un avión (Introducción a la ingeniería aeroespacial).

☐ **Completé este laboratorio (real o virtual)**

Series de práctica para exámenes

Cronométrate: unos 10 minutos por problema. Luego compara con las soluciones completas.

Ejercicios de práctica para el parcial (unidades 1–5)

1. ¿Qué imprime print(7 // 2, 7 % 2)?

    1. División entera y resto.

    Respuesta: 3 1

2. Escribe un bucle que sume de 1 a 100.

    1. total = 0; for n in range(1, 101): total += n.

    Respuesta: 5050

3. Escribe stall_speed(W, rho, S, cl_max=1.6) y evalúala en 10791 N, 1.227 kg/m³, 16.2 m².

    1. (2W/(ρSC))**0.5.

    Respuesta: 26.0 m/s

4. Dado co2 = [418.53, 421.08, 424.61, 427.35], calcula los cambios de un año a otro.

    1. [b - a for a, b in zip(co2, co2[1:])].

    Respuesta: [2.55, 3.53, 2.74]

5. ¿Por qué float("") necesita try/except al leer datos?

    1. Lanza ValueError.

    Respuesta: Para omitir o registrar las filas erróneas en lugar de fallar.

Serie de práctica final (todas las unidades)

1. Convierte 44700 cfs a m³/s.

    1. × 0.0283168.

    Respuesta: ≈ 1,265.8 m³/s

2. ¿Cuántos pasos de búsqueda binaria para un millón de elementos ordenados?

    1. $\lceil \log_2 10^6 \rceil$.

    Respuesta: 20

3. ¿Cuántos pasos de bisección llevan [0, 2] por debajo de 10^{-6}?

    1. $2/2^n < 10^{-6}$.

    Respuesta: 21

4. ¿Alcance exacto de un proyectil a 20 m/s a 45° (g = 9.81)?

    1. v_0^2/g.

    Respuesta: 40.77 m

5. ¿Cuatro pasos de Euler de P += 0.5P(1 − P/1000) desde 100?

    1. 145.0, 207.0, 289.1, …

    Respuesta: 391.8

6. Escribe una prueba para f_to_c.

1. assert f_to_c(212) == 100, y otras.

Respuesta: Pares conocidos: 32 → 0, 212 → 100, −40 → −40.

Hoja de fórmulas

$$7//2 = 3,\ 7\,\%\,2 = 1,\ 2^{10} = 1024$$

$$\bar{x} = \tfrac{1}{n}\sum x_i$$

$$\text{rate} = \Delta y / \Delta x$$

$$y \approx mx + b \ (\text{np.polyfit})$$

$$1\ \text{cfs} = 0.0283168\ \text{m}^3/\text{s}$$

$$\text{binary search} \approx \lceil \log_2 n \rceil$$

$$n_{bisect} = \lceil \log_2((b-a)/\text{tol}) \rceil$$

$$v\ +\!= a\,\Delta t,\ x\ +\!= v\,\Delta t$$

$$R = v_0^2 \sin 2\theta / g$$

$$P\ +\!= hrP(1 - P/K)$$

Constantes

Constante	Valor
1 pie cúbico por segundo (cfs)	0.0283168 m³/s
Segundos por día	86,400
CO_2 de Mauna Loa, media anual de 1959	315.98 ppm
CO_2 de Mauna Loa, media anual de 2025	427.35 ppm
Anomalía GISTEMP 2024 (respecto a 1951–1980)	1.28 °C
GM de la Tierra	3.986 × 10¹⁴ m³/s²
g	9.81 m/s²

Fuente: NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html)

Herramientas y referencias gratuitas

Ejecuta Python gratis en tu navegador y consulta la documentación aquí.

- Google Colab (notebooks gratuitos) (https://colab.research.google.com/): Cuadernos gratuitos con NumPy y Matplotlib preinstalados.

- JupyterLite: Python en tu navegador (Pyodide) (https://jupyter.org/try-jupyter/lab/): Python en el navegador, sin cuenta, impulsado por Pyodide.
- El tutorial oficial de Python (https://docs.python.org/3/tutorial/): El tutorial de referencia.
- MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/): El curso introductorio completo del MIT.
- Harvard CS50's Introduction to Programming with Python (https://cs50.harvard.edu/python/): El curso gratuito de Python de Harvard.

Glosario

Algoritmo
Método preciso paso a paso.

Arreglo
Contenedor de NumPy para cálculo numérico rápido.

Aserción
Comprobación que detiene el programa si es falsa.

Big-O
Cómo crece el coste con el tamaño de la entrada.

Bisección
Búsqueda de raíces dividiendo un intervalo a la mitad.

CSV
Archivo de valores separados por comas.

Diccionario
Asociación clave → valor.

Caso límite
Entrada inusual en un límite.

Euler-Cromer
Paso de tiempo estable: primero la velocidad.

Excepción
Error lanzado durante la ejecución.

Función
Bloque de código con nombre y reutilizable.

Lista
Secuencia ordenada y mutable.

Bucle
Ejecución repetida.

Cuaderno
Documento de celdas de código ejecutables.

Pyodide
Python compilado para ejecutarse en un navegador web.

Simulación
Hacer avanzar un modelo en el tiempo.

Traceback
Informe de dónde ocurrió un error.

Prueba unitaria
Pequeña comprobación de una sola función.

Variable
Nombre ligado a un valor.

f-string
Literal de cadena con formato.

Rúbrica de autoevaluación

No se otorga nota ni crédito. Úsala para valorar con honestidad dónde estás en cada unidad.

Nivel	Cómo se ve
4 · Dominio	Escribe funciones claras y probadas; valida los resultados con respuestas conocidas; explica sus decisiones.
3 · Competente	Los programas funcionan, con errores ocasionales encontrados mediante pruebas.
2 · En desarrollo	Puede modificar ejemplos, pero le cuesta escribir desde cero.
1 · Inicial	Copia código sin entenderlo; repasar la unidad.

Ponderación sugerida si quieres una nota

Parte	Peso
Problemas de práctica	20%
Prácticas de programación	35%
Serie de práctica de mitad de curso	20%
Serie de práctica final	25%

Apunta al nivel 3 o superior en cada unidad antes de pasar al siguiente curso.

Cursos relacionados

- Dibujo técnico / CAD ENGR 120
- Cálculo III MATH 210
- Química General I + Laboratorio CHEM 110 + CHEM 110L
- Todos los cursos