

# Programas básicos en Python

## Adolfo Sánchez Burón

- E-1 Calculadora simple
- E-2 Calculadora de Interés Simple
- E-3 Comprobar si un número es par o impar
- E- 4 Encontrar el número más grande entre tres números
- E- 5 Imprimir la tabla de multiplicar
- E-6 Convertir Celsius a Fahrenheit
- E-7 Operaciones simples de cadenas
- E-8 Comprobar si un año es bisiesto
- E-9 Contar vocales en una cadena
- E-10 Comprobar si un número es positivo, negativo o cero
- E-11 Generar una lista de números primos dentro de un rango
- E-12 Revertir una cadena
- E-13 Comprobar si un número es un número perfecto
- E-14 Contar el número de palabras en una cadena
- E-15 Concatenar dos cadenas
- E-16 Comprobar si un número es un cuadrado perfecto
- E-17 Ordenar una lista de cadenas
- E-18 Contar el número de dígitos en un entero
- E-19 Generar una contraseña aleatoria
- E-20 Calcular el valor exponencial
- E-21 Validar una dirección IP
- E-22 Imprimir el calendario de un mes y año dado
- E-23 Encontrar la mediana de tres valores
- E-24 Encontrar la suma de los dígitos en un número
- E-25 Encontrar el mayor entre tres números
- E-26 Convertir un número decimal a binario
- E-27 Encontrar el Máximo Común Divisor (MCD) y el Mínimo Común Múltiplo (MCM) de dos números
- E-28 Encontrar la suma de elementos en una lista
- E-29 Verificar si una cadena es una dirección de correo electrónico válida
- E-30 Generar una lista aleatoria de números

- E-31 Calcular la desviación estándar de una lista de números
- E-32 Generar una contraseña aleatoria con requisitos específicos
- E-33 Implementar una calculadora simple
- E-34 Ordenar una lista de diccionarios por una clave específica
- E-35 Generar una matriz aleatoria
- E-36 Verificar si una cadena es una URL válida
- E-37 Calcular el número primo
- E-38 Calcular la secuencia de Fibonacci
- E-39 Identificar el número más pequeño de una lista
- E-40 Identificar el número más grande de una lista
- E-41 Encontrar el segundo número mayor de una lista
- E-42 Identificar los N elementos más grandes de una lista

## E-1 Calculadora simple

```
# Solicita al usuario que ingrese dos números enteros.
a = int(input("Ingrese el primer número: "))
b = int(input("Ingrese el segundo número: "))

# Muestra la suma, la diferencia, el producto y el cociente de los números
# ingresados.
print("Suma:", a + b)
print("Diferencia:", a - b)
print("Producto:", a * b)
print("Cociente:", a / b) # Observación: Esto puede devolver un valor
# decimal incluso si los números ingresados son enteros.
```

Ingrese el primer número: 5345

Ingrese el segundo número: 345

Suma: 5690

Diferencia: 5000

Producto: 1844025

Cociente: 15.492753623188406

## E-2 Calculadora de Interés Simple

```
# Calculadora de Interés Simple:

# Solicita al usuario que ingrese el monto principal, la tasa de interés y el
período de tiempo.

p = float(input("Ingrese el monto principal: "))
r = float(input("Ingrese la tasa de interés: "))
t = float(input("Ingrese el período de tiempo: "))

# Calcula el interés simple utilizando la fórmula: interés = (principal *
tasa * tiempo) / 100

interest = (p * r * t) / 100

# Muestra el interés simple calculado.
print("Interés Simple:", interest)

Ingrese el monto principal: 3454
Ingrese la tasa de interés: 3
Ingrese el período de tiempo: 12

Interés Simple: 1243.44
```

## E-3 Comprobar si un número es par o impar

```
# Solicita al usuario que ingrese un número.

num = int(input("Ingrese un número: "))

# Comprueba si el número es divisible por 2 para determinar si es par o
impar.

if num % 2 == 0:
    print("Par")
else:
    print("Impar")

Ingrese un número: 432
```

## E-4 Encontrar el número más grande entre tres números

```
# Solicita al usuario que ingrese tres números.
a = float(input("Ingrese el primer número: "))
b = float(input("Ingrese el segundo número: "))
c = float(input("Ingrese el tercer número: "))

# Encuentra el número más grande utilizando la función max().
max_num = max(a, b, c)

# Muestra el número más grande encontrado.
print("Número más grande:", max_num)
```

Ingrese el primer número: 545

Ingrese el segundo número: 534

Ingrese el tercer número: 645

Número más grande: 645.0

## E-5 Imprimir la tabla de multiplicar

```
# Solicita al usuario que ingrese un número.
num = int(input("Ingrese un número: "))

# Utiliza un bucle for para iterar desde 1 hasta 10 e imprime cada multiplicación.
for i in range(1, 11):
    print(f"{num} x {i} = {num * i}")
```

Ingrese un número: 7

7 x 1 = 7

```
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

## E-6 Convertir Celsius a Fahrenheit

```
# Solicita al usuario que ingrese una temperatura en grados Celsius.
celsius = float(input("Ingrese la temperatura en grados Celsius: "))

# Convierte la temperatura de Celsius a Fahrenheit utilizando la fórmula
correspondiente.
fahrenheit = (celsius * 9 / 5) + 32

# Muestra la temperatura convertida en Fahrenheit.
print("Temperatura en grados Fahrenheit:", fahrenheit)

Ingrese la temperatura en grados Celsius: 29

Temperatura en grados Fahrenheit: 84.2
```

## E-7 Operaciones simples de cadenas

```
# Define una cadena.
cadena = "¡Hola, Mundo!"

# Imprime la longitud de la cadena.
print("Longitud de la cadena:", len(cadena))

# Imprime la cadena en mayúsculas.
```

```
print("En mayúsculas:", cadena.upper())

# Imprime la cadena en minúsculas.
print("En minúsculas:", cadena.lower())

# Imprime la cadena al revés.
print("Cadena invertida:", cadena[::-1])
```

Longitud de la cadena: 13

En mayúsculas: ¡HOLA, MUNDO!

En minúsculas: ¡hola, mundo!

Cadena invertida: !odnuM ,aloH¡

## E-8 Comprobar si un año es bisiesto

```
def es_año_bisiesto(año):
    # Comprueba si el año es divisible por 4 pero no por 100, o si es
    # divisible por 400.

    if (año % 4 == 0 and año % 100 != 0) or (año % 400 == 0):
        return True
    else:
        return False

# Solicita al usuario que ingrese un año.
año = int(input("Ingrese un año: "))

# Llama a la función es_año_bisiesto y muestra el resultado.
if es_año_bisiesto(año):
    print("Año bisiesto")
else:
    print("No es un año bisiesto")
```

Ingrese un año: 2024

Año bisiesto

## E-9 Contar vocales en una cadena

```
def contar_vocales(cadena):  
    # Define las vocales en mayúsculas y minúsculas.  
  
    vocales = 'aeiouAEIOU'  
    contador = 0  
  
    # Itera sobre cada carácter en la cadena y cuenta las vocales.  
  
    for char in cadena:  
        if char in vocales:  
            contador += 1  
  
    return contador  
  
# Solicita al usuario que ingrese una cadena.  
cadena = input("Ingrese una cadena: ")  
  
# Llama a la función contar_vocales y muestra el resultado.  
print("Número de vocales:", contar_vocales(cadena))  
  
Ingrese una cadena:  Francia  
  
Número de vocales: 3
```

## E-10 Comprobar si un número es positivo, negativo o cero

```
num = float(input("Ingrese un número: "))  
  
if num > 0:  
    print("Número positivo")  
  
elif num < 0:  
    print("Número negativo")  
  
else:  
    print("Cero")  
  
Ingrese un número:  452
```

Número positivo

## E-11 Generar una lista de números primos dentro de un rango

```
def generar_primos(inicio, fin):
    primos = []
    for num in range(inicio, fin + 1):
        if num > 1:
            for i in range(2, num):
                if num % i == 0:
                    break
            else:
                primos.append(num)
    return primos

# Solicita al usuario que ingrese el rango inicial y final.
inicio_rango = int(input("Ingrese el rango inicial: "))
fin_rango = int(input("Ingrese el rango final: "))

# Llama a la función generar_primos y muestra los números primos en el rango especificado.
print("Números primos:", generar_primos(inicio_rango, fin_rango))
```

Ingrese el rango inicial: 100

Ingrese el rango final: 1000

Números primos: [101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821,



```
823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919,
929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]
```

## E-12 Revertir una cadena

```
# Solicita al usuario que ingrese una cadena.
cadena = input("Ingrese una cadena: ")

# Utiliza la técnica de rebanado para revertir la cadena.
cadena_invertida = cadena[::-1]

# Muestra la cadena invertida.
print("Cadena invertida:", cadena_invertida)

Ingrese una cadena:  Francia

Cadena invertida: aicnarF
```

## E-13 Comprobar si un número es un número perfecto

```
def es_numero_perfecto(n):
    suma = 0

    # Itera a través de todos los números menores que n y suma los divisores.
    for i in range(1, n):
        if n % i == 0:
            suma += i

    # Comprueba si la suma de los divisores es igual al número original.
    return suma == n

numero = int(input("Ingrese un número: "))

if es_numero_perfecto(numero):
    print("Número perfecto")
else:
    print("No es un número perfecto")

Ingrese un número: 5678
```

No es un número perfecto

## E-14 Contar el número de palabras en una cadena

```
# Contar el número de palabras en una cadena:
# Solicita al usuario que ingrese una cadena.
cadena = input("Ingrese una cadena: ")

# Utiliza el método split() para dividir la cadena en palabras y luego cuenta
el número de elementos en la lista resultante.
cantidad_palabras = len(cadena.split())

# Muestra el número de palabras.
print("Número de palabras:", cantidad_palabras)

Ingrese una cadena:  Noruega

Número de palabras: 1
```

## E-15 Concatenar dos cadenas

```
# Solicita al usuario que ingrese dos cadenas.
cadena1 = input("Ingrese la primera cadena: ")
cadena2 = input("Ingrese la segunda cadena: ")

# Utiliza el operador de concatenación para unir las dos cadenas con un
espacio entre ellas.
cadena_concatenada = cadena1 + ' ' + cadena2

# Muestra la cadena concatenada.
print("Cadena concatenada:", cadena_concatenada)

Ingrese la primera cadena:  Francia
Ingrese la segunda cadena:  Noruega
```

Cadena concatenada: Francia Noruega

## E-16 Comprobar si un número es un cuadrado perfecto

```
import math

def es_cuadrado_perfecto(n):
    # Calcula la raíz cuadrada del número.

    raiz = math.isqrt(n)

    # Comprueba si el cuadrado de la raíz es igual al número original.

    return raiz * raiz == n

# Solicita al usuario que ingrese un número.
numero = int(input("Ingrese un número: "))

if es_cuadrado_perfecto(numero):
    print("Cuadrado perfecto")
else:
    print("No es un cuadrado perfecto")
```

Ingrese un número: 87

No es un cuadrado perfecto

## E-17 Ordenar una lista de cadenas

```
strings = ['apple', 'banana', 'cherry', 'date', 'elderberry']

# Utiliza la función sorted() para ordenar la lista de cadenas.
strings_ordenadas = sorted(strings)

# Muestra la lista ordenada de cadenas.
print("Cadenas ordenadas:", strings_ordenadas)

Cadenas ordenadas: ['apple', 'banana', 'cherry', 'date', 'elderberry']
```

# E-18 Contar el número de dígitos en un entero

```
# Solicita al usuario que ingrese un entero.

numero = int(input("Ingrese un entero: "))

# Convierte el entero en su valor absoluto y luego a cadena para contar los dígitos.

# Usando len(), determina la cantidad de dígitos en el número.

num_digitos = len(str(abs(numero)))

# Muestra el número de dígitos.

print("Número de dígitos:", num_digitos)

Ingrese un entero: 6346

Número de dígitos: 4
```

# E-19 Generar una contraseña aleatoria

```
import random

import string

def generar_contraseña(longitud):

    # Definir los caracteres a utilizar en la contraseña (letras mayúsculas y minúsculas, dígitos y signos de puntuación).

    caracteres = string.ascii_letters + string.digits + string.punctuation

    # Generar una contraseña aleatoria utilizando los caracteres definidos.

    contraseña = ''.join(random.choice(caracteres) for _ in range(longitud))

    return contraseña

# Longitud de la contraseña a generar.

longitud_contraseña = 12

# Generar la contraseña y mostrarla.

print("Contraseña generada:", generar_contraseña(longitud_contraseña))

Contraseña generada: E`KE&KEln:M\
```

## E-20 Calcular el valor exponencial

```
# Solicita al usuario que ingrese la base y el exponente.

base = float(input("Ingrese la base: "))
exponente = float(input("Ingrese el exponente: "))

# Calcula el valor exponencial utilizando el operador de potencia (**).
resultado = base ** exponente

# Muestra el resultado.
print("Resultado:", resultado)

Ingrese la base: 2342
Ingrese el exponente: 23

Resultado: 3.165909433515004e+77
```

## E-21 Validar una dirección IP

```
import socket

def es_direccion_ip_valida(ip):
    try:
        # Intenta convertir la dirección IP a formato binario.
        socket.inet_aton(ip)
        return True
    except socket.error:
        # Si hay un error, la dirección IP es inválida.
        return False

# Solicita al usuario que ingrese una dirección IP.
direccion_ip = input("Ingrese una dirección IP: ")

# Llama a la función es_direccion_ip_valida y muestra el resultado.
if es_direccion_ip_valida(direccion_ip):
```

```
print("Dirección IP válida")  
  
else:  
    print("Dirección IP inválida")
```

Ingrese una dirección IP: 5364536

Dirección IP válida

## E-22 Imprimir el calendario de un mes y año dado

```
import calendar  
  
# Solicita al usuario que ingrese el año y el mes.  
año = int(input("Ingrese el año: "))  
mes = int(input("Ingrese el mes: "))  
  
# Utiliza la función month() de la biblioteca calendar para imprimir el  
calendario del mes y año especificados.  
print(calendar.month(año, mes))
```

Ingrese el año: 2024

Ingrese el mes: 4

```
April 2024  
Mo Tu We Th Fr Sa Su  
1  2  3  4  5  6  7  
8  9 10 11 12 13 14  
15 16 17 18 19 20 21  
22 23 24 25 26 27 28  
29 30
```

## E-23 Encontrar la mediana de tres valores

```
def encontrar_mediana(a, b, c):  
    # Ordena los tres valores en una lista y devuelve el valor del medio.
```

```
return sorted([a, b, c])[1]
```

```
# Solicita al usuario que ingrese los tres números.
```

```
num1 = float(input("Ingrese el primer número: "))
```

```
num2 = float(input("Ingrese el segundo número: "))
```

```
num3 = float(input("Ingrese el tercer número: "))
```

```
# Llama a la función encontrar_mediana y muestra el resultado.
```

```
print("Mediana:", encontrar_mediana(num1, num2, num3))
```

```
Ingrese el primer número: 423
```

```
Ingrese el segundo número: 6546
```

```
Ingrese el tercer número: 344
```

```
Mediana: 423.0
```

## E-24 Encontrar la suma de los dígitos en un número

```
def suma_de_digitos(n):
```

```
# Convierte el número a una cadena y luego suma cada dígito convertido a entero.
```

```
return sum(int(digito) for digito in str(n))
```

```
# Solicita al usuario que ingrese un número.
```

```
numero = int(input("Ingrese un número: "))
```

```
# Llama a la función suma_de_digitos y muestra el resultado.
```

```
print("Suma de los dígitos:", suma_de_digitos(numero))
```

```
Ingrese un número: 4353
```

```
Suma de los dígitos: 15
```

## E-25 Encontrar el mayor entre tres números

```
# Solicita al usuario que ingrese los tres números.

a = float(input("Ingrese el primer número: "))
b = float(input("Ingrese el segundo número: "))
c = float(input("Ingrese el tercer número: "))

# Utiliza la función max() para encontrar el número máximo entre los tres
números ingresados.

maximo = max(a, b, c)

# Muestra el número máximo encontrado.

print("El número más grande es:", maximo)

Ingrese el primer número: 6453
Ingrese el segundo número: 876
Ingrese el tercer número: 35

El número más grande es: 6453.0
```

## E-26 Convertir un número decimal a binario

```
# Solicita al usuario que ingrese un número decimal.

decimal = int(input("Ingrese un número decimal: "))

# Utiliza la función bin() para convertir el número decimal a binario.
# Se elimina '0b' al principio del resultado para obtener solo la
representación binaria.

binario = bin(decimal)[2:]

# Muestra la representación binaria del número decimal.

print("Binario:", binario)

Ingrese un número decimal: 34

Binario: 100010
```



## E-27 Encontrar el Máximo Común Divisor (MCD) y el Mínimo Común Múltiplo (MCM) de dos números

```
import math

# Definir dos números.

num1 = int(input("Ingrese el primer número: "))
num2 = int(input("Ingrese el segundo número: "))

# Calcular el Máximo Común Divisor (MCD) utilizando la función gcd() de la
# biblioteca math.

mcd = math.gcd(num1, num2)
print("El Máximo Común Divisor (MCD) de", num1, "y", num2, "es:", mcd)

# Calcular el Mínimo Común Múltiplo (MCM) utilizando la fórmula:  $MCM(a, b) = (a * b) // MCD(a, b)$ .

mcm = (num1 * num2) // mcd
print("El Mínimo Común Múltiplo (MCM) de", num1, "y", num2, "es:", mcm)

Ingrese el primer número: 5345
Ingrese el segundo número: 7567

El Máximo Común Divisor (MCD) de 5345 y 7567 es: 1
El Mínimo Común Múltiplo (MCM) de 5345 y 7567 es: 40445615
```

## E-28 Encontrar la suma de elementos en una lista

```
# Definir una lista.

mi_lista = [1, 2, 3, 4, 5]

# Utilizar la función sum() para encontrar la suma de los elementos en la
# lista.

suma_de_elementos = sum(mi_lista)

# Mostrar la suma de los elementos.

print("Suma de los elementos:", suma_de_elementos)
```

Suma de los elementos: 15

## E-29 Verificar si una cadena es una dirección de correo electrónico válida

```
import re

def es_direccion_email_valida(email):
    return bool(re.match(r"^[^@]+@[^@]+\.[^@]+", email))

email_ingresado = input("Ingrese una dirección de correo electrónico: ")
if es_direccion_email_valida(email_ingresado):
    print("Dirección de correo electrónico válida")
else:
    print("Dirección de correo electrónico inválida")
```

Ingrese una dirección de correo electrónico: asburon@gmail.com

Dirección de correo electrónico válida

## E-30 Generar una lista aleatoria de números

```
import random

lista_aleatoria = random.sample(range(1, 100), 5)
print("Lista aleatoria:", lista_aleatoria)

Lista aleatoria: [17, 3, 86, 44, 89]
```

## E-31 Calcular la desviación estándar de una lista de números

```
import statistics

datos = [1, 2, 3, 4, 5]

desviacion_estandar = statistics.stdev(datos)
```

```
print("Desviación estándar:", desviacion_estandar)
```

```
Desviación estándar: 1.5811388300841898
```

## E-32 Generar una contraseña aleatoria con requisitos específicos

```
import random
import string

def generar_contraseña(longitud, incluir_digitos=True,
incluir_caracteres_especiales=True):
    caracteres = string.ascii_letters

    if incluir_digitos:
        caracteres += string.digits

    if incluir_caracteres_especiales:
        caracteres += string.punctuation

    contraseña = ''.join(random.choice(caracteres) for _ in range(longitud))

    return contraseña

longitud_contraseña = 12
print("Contraseña generada:", generar_contraseña(longitud_contraseña))

Contraseña generada: uwqEiQ!Q!NCd
```

## E-33 Implementar una calculadora simple

```
def sumar(x, y):
    return x + y

def restar(x, y):
    return x - y

def multiplicar(x, y):
    return x * y

def dividir(x, y):
```

```
if y == 0:

    return "No se puede dividir por cero"

return x / y
```

```
num1 = float(input("Ingrese el primer número: "))
num2 = float(input("Ingrese el segundo número: "))
```

```
print("Suma:", sumar(num1, num2))
print("Diferencia:", restar(num1, num2))
print("Producto:", multiplicar(num1, num2))
print("Cociente:", dividir(num1, num2))
```

Ingrese el primer número: 764

Ingrese el segundo número: 45645

Suma: 46409.0

Diferencia: -44881.0

Producto: 34872780.0

Cociente: 0.016737868331690216

## E-34 Ordenar una lista de diccionarios por una clave específica

```
# Definir una lista de diccionarios.
```

```
lista_de_diccionarios = [{'name': 'John', 'age': 30}, {'name': 'Jane', 'age': 25},
                          {'name': 'Bob', 'age': 35}]
```

```
# Utilizar la función sorted() para ordenar la lista de diccionarios por la clave 'age'.
```

```
# La clave 'key' especifica una función de ordenamiento personalizada que extrae la edad de cada diccionario.
```

```
lista_ordenada = sorted(lista_de_diccionarios, key=lambda x: x['age'])
```

```
# Mostrar la lista de diccionarios ordenada.
```

```
print("Lista de diccionarios ordenada:", lista_ordenada)
```

```
Lista de diccionarios ordenada: [{'name': 'Jane', 'age': 25}, {'name': 'John', 'age': 30}, {'name': 'Bob', 'age': 35}]
```

## E-35 Generar una matriz aleatoria

```
import numpy as np
```

```
filas = 3
```

```
columnas = 3
```

```
matriz_aleatoria = np.random.rand(filas, columnas)
```

```
print("Matriz aleatoria:")
```

```
print(matriz_aleatoria)
```

```
Matriz aleatoria:
```

```
[[0.04886981 0.0166007 0.87062023]
```

```
 [0.471779 0.89327991 0.89705373]
```

```
 [0.1501042 0.1030853 0.76082135]]
```

## E-36 Verificar si una cadena es una URL válida

```
import re
```

```
def es_url_valida(url):
```

```
    regex = re.compile(
```

```
        r'^(?:http|ftp)s?:/' # Protocolo
```

```
        r'(?:(?:[A-Z0-9](?:[A-Z0-9-]{0,61}[A-Z0-9])?\.)+(?:[A-Z]{2,6}\.|[A-Z0-9-]{2,}\.?)|' # Dominio
```

```
        r'localhost|' # Localhost
```

```
        r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}|' # Dirección IP (IPv4)
```

```
        r'\[?[A-F0-9]*:[A-F0-9:]+\]|' # Dirección IP (IPv6)
```

```
        r'(?::\d+)?' # Puerto
```

```
        r'(?:/?|[/?]\S+)$', re.IGNORECASE) # Rutas y parámetros opcionales
```

```
    return re.match(regex, url) is not None
```

```
input_url = input("Ingrese una URL: ")

if es_url_valida(input_url):
    print("URL válida")
else:
    print("URL inválida")
```

Ingrese una URL: http://www.google.es

URL válida

## E-37 Calcula el número primo

*#Comprueba si el número dado x es primo o no*

```
def es_primo(x):
    if x == 2:
        return True
    if x % 2 == 0:
        return False
    for i in range(3, int(x**0.5)+1, 2):
        if x % i == 0:
            return False
    return True
```

*#Devuelve el siguiente número primo después del primo\_actual*

```
def generar_primo(primo_actual):
    nuevo_primo = primo_actual + 1
    while True:
        if not es_primo(nuevo_primo):
            nuevo_primo += 1
        else:
            break
    return nuevo_primo
```

```
def main():
    # Función principal

    primo_actual = 2

    while True:

        respuesta = input('¿Desea ver el siguiente número primo? (S/N) ')

        if respuesta.lower().startswith('s'):

            print(primo_actual)

            primo_actual = generar_primo(primo_actual)

        else:

            break

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

¿Desea ver el siguiente número primo? (S/N) 2344

*# Solicitar al usuario que ingrese una distancia en kilómetros*

```
kilometros = float(input("Ingresa la distancia en kilómetros: "))
```

*# Factor de conversión: 1 kilómetro = 0.621371 millas*

```
factor_de_conversion = 0.621371
```

*# Calcular las millas multiplicando los kilómetros por el factor de conversión*

```
millas = kilometros * factor_de_conversion
```

*# Imprimir el resultado de la conversión*

```
print(f"{kilometros} kilómetros es igual a {millas} millas")
```

Ingresa la distancia en kilómetros: 345434

345434.0 kilómetros es igual a 214642.670014 millas

## E-38 Calcula la secuencia de Fibonacci

La secuencia de Fibonacci es una serie de números en la que cada número es la suma de los dos anteriores, comenzando típicamente con 0 y 1.

```

# Solicitar al usuario que ingrese el número de términos que desea para la
secuencia de Fibonacci

nterms = int(input("¿Cuántos términos? "))

# Primeros dos términos de la secuencia de Fibonacci
n1, n2 = 0, 1
count = 0

# Verificar si el número de términos es válido
if nterms <= 0:
    print("Por favor, ingrese un número entero positivo")
# Si solo hay un término, retornar n1
elif nterms == 1:
    print("Secuencia de Fibonacci hasta", nterms, ":")
    print(n1)
# Generar la secuencia de Fibonacci
else:
    print("Secuencia de Fibonacci:")
    while count < nterms:
        print(n1)
        nth = n1 + n2
        # Actualizar valores
        n1 = n2
        n2 = nth
        count += 1

```

¿Cuántos términos? 10

Secuencia de Fibonacci:

0

1

1

2

3



5  
8  
13  
21  
34

## E-39 Identifica el número más pequeño de una lista

```
# Lista de muestra de números

numeros = [30, 10, -45, 5, 20]


# Inicializar una variable para almacenar el valor mínimo, inicialmente
establecido al primer elemento de la lista

minimo = numeros[0]


# Iterar a través de la lista y actualizar el valor mínimo si se encuentra un
número más pequeño

for i in numeros:
    if i < minimo:
        minimo = i


# Imprimir el valor mínimo

print("El número más pequeño en la lista es:", minimo)

El número más pequeño en la lista es: -45
```

## E-40 Identifica el número más grande de una lista

```
# Lista de muestra de números

numeros = [30, 10, -45, 5, 20]


# Inicializar una variable para almacenar el valor máximo, inicialmente
establecido al primer elemento de la lista

maximo = numeros[0]


# Iterar a través de la lista y actualizar el valor máximo si se encuentra un
número más grande
```

```
for i in numeros:

    if i > maximo:

        maximo = i

# Imprimir el valor máximo
print("El número más grande en la lista es:", maximo)

El número más grande en la lista es: 30
```

## E-41 Encuentra el segundo número mayor de una lista

```
# Lista de muestra de números
numeros = [30, 10, 45, 5, 20]

# Ordenar la lista en orden descendente
numeros.sort(reverse=True)

# Verificar si hay al menos dos elementos en la lista
if len(numeros) >= 2:

    # Obtener el segundo número más grande
    segundo_mayor = numeros[1]

    print("El segundo número más grande en la lista es:", segundo_mayor)
else:

    print("La lista no contiene un segundo número más grande.")

El segundo número más grande en la lista es: 30
```

## E-42 Identifica los N elementos más grandes de una lista

```
# Definir la función para encontrar los N elementos más grandes en una lista
def encontrar_n_elementos_mas_grandes(lista, n):

    # Ordenar la lista en orden descendente

    lista_ordenada = sorted(lista, reverse=True)

    # Obtener los primeros N elementos

    elementos_mas_grandes = lista_ordenada[:n]
```

```
return elementos_mas_grandes
```

```
# Lista de muestra de números
```

```
numeros = [30, 10, 45, 5, 20, 50, 15, 3, 345, 54, 67, 87, 98, 100, 34]
```

```
# Número de elementos más grandes a encontrar
```

```
N = int(input("N = "))
```

```
# Encontrar los N elementos más grandes de la lista
```

```
resultado = encontrar_n_elementos_mas_grandes(numeros, N)
```

```
# Imprimir los N elementos más grandes
```


```
print(f"Los {N} elementos más grandes en la lista son:", resultado)
```

```
N = 56
```

```
Los 56 elementos más grandes en la lista son: [345, 100, 98, 87, 67, 54, 50, 45, 34, 30, 20, 15, 10, 5, 3]
```

## Bibliografía

Página web: [https://rpubs.com/AdSan-R/Python\\_Basic](https://rpubs.com/AdSan-R/Python_Basic)

 by RStudio

[Sign in](#) [Register](#)

## Getting Started with RPubs


RStudio lets you harness the power of [R Markdown](#) to create documents that weave together your writing and the output of your R code. And now, with RPubs, you can publish those documents on the web with the click of a button!

### Prerequisites

You'll need [R](#) itself, [RStudio](#) (v0.96.230 or later), and the [knitr](#) package (v0.5 or later).

### Instructions

1. In RStudio, create a new R Markdown document by choosing **File | New | R Markdown**.
2. Click the **Knit HTML** button in the doc toolbar to preview your document.
3. In the preview window, click the **Publish** button.

 by RStudio

[Sign in](#) [Register](#)

### Easy web publishing from R

Write [R Markdown](#) documents in RStudio.  
Share them here on RPubs. (It's free, and couldn't be simpler!)

[Get Started](#)