


# Kata: Funciones en Python




## Descripción General

¡Bienvenido a esta kata de programación! 

En esta práctica, dominarás el uso de **funciones** en Python , aprendiendo a:





- Modularizar código para reutilizar lógica.
- Mejorar la legibilidad y mantenimiento de programas.
- Manipular parámetros y valores de retorno.

Cada ejercicio incluye:


-  **Objetivo:** Qué lograrás.
-  **Instrucciones:** Pasos claros para resolverlo.
-  **Preguntas de reflexión:** Para profundizar en el aprendizaje.

## Requisitos

Antes de comenzar, asegurate de tener lo siguiente listo:

-  Python instalado (versión 3.8 o superior)
  -  Editor de código (recomendado: VS Code)
  -  Muchas ganas de aprender y divertirse resolviendo desafíos 
- 

## Ejercicio 1: Tabla de multiplicar

 **Objetivo:** Generar una tabla de multiplicar usando funciones.


 **Instrucciones:**

1. Crea una función `tabla_multiplicar` que reciba un número entero positivo.
2. Devuelve una lista con su tabla de multiplicar del 1 al 10.
  - Ejemplo: Para 3 → [3, 6, 9, ..., 30].

 **Preguntas de reflexión:**

- ¿Cómo adaptarías la función para recibir el rango (ej: hasta 12)?
  - ¿Qué ocurre si se ingresa un número negativo?
- 

## Ejercicio 2: Suma de números pares

 **Objetivo:** Sumar elementos pares de una lista con una función.

 **Instrucciones:**

1. Define una función `suma_pares` que reciba una lista de enteros.
2. Retorna la suma de los números pares.
  - Ejemplo: Para `[1, 2, 3, 4, 5, 6]`  $\rightarrow 12$ .

 **Preguntas de reflexión:**

- ¿Cómo manejarías listas vacías o con decimales?
  - ¿Qué ventaja tiene usar una función en lugar de código inline?
- 

### **Ejercicio 3: Área y perímetro de un rectángulo**

 **Objetivo:** Calcular múltiples valores en una función.

 **Instrucciones:**

1. Crea una función `rectángulo` que reciba **longitud** y **anchura**.
2. Retorna una tupla con el área y el perímetro.
  - Fórmulas:
    - $\text{Área} = \text{longitud} * \text{anchura}$ .
    - $\text{Perímetro} = 2 * (\text{longitud} + \text{anchura})$ .

 **Preguntas de reflexión:**

- ¿Por qué usar una tupla en lugar de una lista?
  - ¿Cómo validarías que las dimensiones sean positivas?
- 

### **Ejercicio 4: Conversión de temperatura**

 **Objetivo:** Convertir temperaturas con condiciones.


 **Instrucciones:**

1. Define una función `convertir_temperatura` que reciba:
  - Temperatura en Celsius.
  - Unidad de destino ("F" o "K").
2. Retorna la temperatura convertida usando:
  - **Fahrenheit:**  $F = C * 9/5 + 32$ .
  - **Kelvin:**  $K = C + 273.15$ .

### Preguntas de reflexión:

- ¿Qué pasa si se ingresa una unidad no válida?
  - ¿Cómo extenderías la función para convertir entre otras unidades?
- 

### Ejercicio 5: Verificador de números primos

 **Objetivo:** Implementar una función para detectar primos.

 **Instrucciones:**

1. Crea una función `es_primo` que reciba un entero positivo.
2. Retorna `True` si es primo (solo divisible por 1 y sí mismo), `False` en caso contrario.
  - Ejemplo: 7 → `True`, 8 → `False`.

### Preguntas de reflexión:

- ¿Cómo optimizarías la función para números grandes?
  - ¿Qué estructura de control es más eficiente aquí: `for` o `while`?
- 

### Ejercicio 6: Promedio de calificaciones

 **Objetivo:** Calcular promedios con funciones.

 **Instrucciones:**

1. Define una función `promedio_calificaciones` que reciba una lista de notas (0 a 10).
2. Retorna el promedio. Si la lista está vacía, retorna 0.
  - Ejemplo: [8.5, 9.0, 7.5, 8.0] → 8.25.

### Preguntas de reflexión:

- ¿Cómo manejarías notas fuera del rango 0-10?
  - ¿Qué ventaja tiene retornar 0 en lugar de `None` para listas vacías?
- 

### Ejercicio 7: Factorial con validación

 **Objetivo:** Combinar funciones para validar y calcular.

 **Instrucciones:**


1. Usa dos funciones:
  - `validar_entrada`: Verifica si un número es entero no negativo.

- factorial: Calcula el factorial (ej:  $5! = 120$ ).
2. El programa principal debe:
    - Pedir un número al usuario.
    - Validarlo y mostrar el factorial o un error.

#### Preguntas de reflexión:

- ¿Por qué separar la validación del cálculo?
  - ¿Cómo manejarías el desbordamiento para números grandes?
- 

### Ejercicio 8: Números primos con funciones auxiliares

 **Objetivo:** Modularizar la lógica de primos.

 **Instrucciones:**

1. Usa dos funciones:
  - es\_divisible: Retorna True si un número divide a otro.
  - es\_primo: Usa es\_divisible para verificar si es primo.
2. El programa principal pide un número y muestra si es primo.

#### Preguntas de reflexión:

- ¿Cómo reutilizarías es\_divisible en otros contextos?
  - ¿Qué optimizaciones aplicarías a es\_primo?
- 

### Ejercicio 9: Conversor de temperatura con menú

 **Objetivo:** Integrar funciones con interacción de usuario.


 **Instrucciones:**

1. Usa tres funciones:
  - convertir\_a\_fahrenheit: Convierte Celsius a Fahrenheit.
  - convertir\_a\_kelvin: Convierte Celsius a Kelvin.
  - menu\_conversion: Muestra un menú para elegir la unidad.
2. El programa principal:
  - Pide la temperatura en Celsius.
  - Muestra el resultado según la unidad elegida.

#### Preguntas de reflexión:

- ¿Cómo mejorarías la experiencia de usuario del menú?
  - ¿Qué pasa si el usuario ingresa una opción inválida?
- 

### **Ejercicio 10: Rectángulo con validación**

 **Objetivo:** Validar entradas antes de calcular.

 **Instrucciones:**

1. Usa tres funciones:
  - `validar_dimensiones`: Verifica que longitud y anchura sean positivas.
  - `calcular_area`: Retorna el área del rectángulo.
  - `calcular_perimetro`: Retorna el perímetro.
2. El programa principal:
  - Pide las dimensiones al usuario.
  - Valida y muestra resultados o un error.

 **Preguntas de reflexión:**

- ¿Por qué es importante validar las entradas?
  - ¿Cómo extenderías el programa para otras figuras geométricas?
- 

 **Bonus:**

- **Pruebas:** Escribe casos de prueba para cada función.