30 EJERCICIOS ESPECIALIZADOS

CAPÍTULO 4: FUNCIONES EN PYTHON

Curso Base:	Python para Todos - Universidad de Michigan
Capítulo:	4 - Funciones Definidas por el Usuario
Nivel:	Principiante → Intermedio Avanzado
Distribución:	6 Básicos (20%) + 12 Intermedios (40%) + 12 Avanzados (40%)
Enfoque:	Solo Funciones (sin bucles) + Integración Caps 2-3
Objetivo:	Dominio experto de funciones Python

CONCEPTOS CLAVE CUBIERTOS

✓ Definición de funciones con **def**
✓ Parámetros vs argumentos
✓ Funciones productivas vs estériles (**return** vs **print**)
✓ Variables locales vs globales (**scope**)
✓ Conversiones de tipos en funciones
✓ Funciones anidadas y composición
✓ Argumentos posicionales y por defecto
✓ Return múltiple y desempaquetado
✓ Validación y control de errores
✓ Integración con condicionales del Capítulo 2
✓ Aplicaciones reales y sistemas complejos

FUENTES DEL PROYECTO UTILIZADAS

• **PythonParaTodos.pdf** - Capítulo 4 Funciones
• **capitulo4.ipynb** - Ejemplos prácticos y código
• **Funciones.ipynb** - Conceptos avanzados
• **caiptulo4_Funciones.txt** - Explicaciones detalladas

• **199ejerciciosEstefania.txt** - Ejercicios complementarios

EJERCICIOS BÁSICOS (1-6)

Fundamentos esenciales de definición y uso de funciones

EJERCICIO 1: PRIMERA FUNCIÓN

Nivel: Básico | Tipo: Definición básica | Conceptos: def, invocación

Crea una función llamada **presentacion()** que imprima exactamente: "¡Hola! Soy una función de Python" "Estoy aquí para ayudarte a programar" Luego invócala para que muestre el mensaje.

> ***Punto clave:*** *Distingue entre DEFINIR (def) e INVOCAR (llamar) la función.*

EJERCICIO 2: FUNCIÓN CON PARÁMETRO ÚNICO

Nivel: Básico | Tipo: Parámetros | Conceptos: argumentos, interpolación

Define una función **saludar_estudiante(nombre)** que reciba un nombre y muestre: "Bienvenido al curso de Python, [nombre]" Pruébala con al menos 3 nombres diferentes.

> ***Punto clave:*** *El parámetro 'nombre' es como una variable temporal dentro de la función.*

EJERCICIO 3: FUNCIÓN CON DOS PARÁMETROS

Nivel: Básico | Tipo: Múltiples parámetros | Conceptos: orden de argumentos

Crea **mostrar_operacion(numero1, numero2)** que imprima: "Los números son: [numero1] y [numero2]" "Su suma es: [resultado]" "Su multiplicación es: [resultado]" Prueba con diferentes pares de números. ¿Qué pasa si cambias el orden de los argumentos?

> ***Punto clave:*** *El orden de los argumentos debe coincidir con el orden de los parámetros.*

EJERCICIO 4: PRIMERA FUNCIÓN CON RETURN

Nivel: Básico | Tipo: Return básico | Conceptos: función productiva

Define **obtener_mensaje_motivacional()** que RETORNE (no imprima) la cadena: "Python es increíble y tú lo estás dominando" Luego, en el código principal: 1. Llama a la función y asigna el resultado a una variable 2. Imprime esa variable 3. Comprueba el tipo de dato que retorna

> ***Punto clave:*** *return DEVUELVE un valor, print solo muestra en pantalla.*

EJERCICIO 5: FUNCIÓN CALCULADORA BÁSICA

Nivel: Básico | Tipo: Cálculos con return | Conceptos: operaciones matemáticas

Crea cuatro funciones separadas: • **sumar(a, b)** que retorne la suma • **restar(a, b)** que retorne la resta • **multiplicar(a, b)** que retorne la multiplicación • **dividir(a, b)** que retorne la división Prueba cada función y muestra los resultados de operar 15 y 3.

> ***Punto clave:*** *Cada función debe ser específica y retornar el resultado del cálculo.*

EJERCICIO 6: ANÁLISIS FUNCIÓN ESTÉRIL VS PRODUCTIVA

Nivel: Básico | Tipo: Conceptual | Conceptos: void vs fruitful functions

Analiza estas dos funciones: `def funcion_a(texto): print(f"Procesando: {texto}")` `def funcion_b(texto): return f"Procesado: {texto}"` Preguntas para responder: 1. ¿Cuál es estéril y cuál productiva? ¿Por qué? 2. ¿Qué pasa si haces: resultado = funcion_a("hola")? 3. ¿Qué pasa si haces: resultado = funcion_b("hola")? 4. ¿Cuándo usarías cada tipo?

Punto clave: *Las funciones estériles ejecutan acciones, las productivas devuelven valores.*

EJERCICIOS INTERMEDIOS (7-18)

Integración con conceptos previos y casos prácticos

EJERCICIO 7: FUNCIÓN CON CONDICIONALES SIMPLES

Nivel: Intermedio | Tipo: Integración Cap.2 | Conceptos: if/elif/else en funciones

Define **clasificar_temperatura(grados)** que retorne: • "Muy frío" si es menor a 0 • "Frío" si está entre 0 y 15 • "Templado" si está entre 16 y 25 • "Calor" si está entre 26 y 35 • "Muy calor" si es mayor a 35 Prueba con: -5, 10, 20, 30, 40

> ***Punto clave:*** *Los condicionales dentro de funciones permiten lógica de decisión reutilizable.*

EJERCICIO 8: CONVERSIÓN DE TIPOS CON VALIDACIÓN

Nivel: Intermedio | Tipo: Tipos de datos | Conceptos: int(), float(), str(), validación

Crea **convertir_seguro(valor, tipo_destino)** que: • Si tipo_destino es "entero": intenta convertir a int • Si tipo_destino es "decimal": intenta convertir a float • Si tipo_destino es "texto": convierte a str • Si la conversión falla, retorna "Error de conversión" • Si el tipo_destino no es válido, retorna "Tipo no soportado" Prueba con: ("123", "entero"), ("12.5", "decimal"), (456, "texto"), ("abc", "entero")

> ***Punto clave:*** *Las funciones pueden manejar errores y validar entradas.*

EJERCICIO 9: FUNCIÓN CON INPUT DEL USUARIO

Nivel: Intermedio | Tipo: Interactividad | Conceptos: input(), return, validación

Define **recopilar_datos_estudiante()** que: 1. Pida el nombre del estudiante 2. Pida su edad 3. Pida su nota (0-10) 4. Retorne una cadena formateada: "Estudiante: [nombre], Edad: [edad], Nota: [nota]" La función debe validar que la edad sea un número entero y la nota esté entre 0 y 10. Si algo está mal, debe retornar "Datos inválidos".

> ***Punto clave:*** *Las funciones pueden encapsular toda la lógica de recopilación de datos.*

EJERCICIO 10: FUNCIONES ANIDADAS (FUNCIÓN LLAMANDO FUNCIÓN)

Nivel: Intermedio | Tipo: Composición | Conceptos: funciones anidadas, reutilización

Define estas funciones: 1. **calcular_area_rectangulo(base, altura)** - retorna base * altura 2. **calcular_perimetro_rectangulo(base, altura)** - retorna 2 * (base + altura) 3. **analisis_rectangulo(base, altura)** - usa las dos anteriores y retorna: "Área: [area], Perímetro: [perimetro], Relación área/perímetro: [relacion]" Prueba con un rectángulo de 5x8.

> ***Punto clave:*** *Las funciones pueden usar otras funciones para crear soluciones más complejas.*

EJERCICIO 11: RETURN MÚLTIPLE Y DESEMPAQUETADO

Nivel: Intermedio | Tipo: Return avanzado | Conceptos: return múltiple, desempaquetado

Crea **estadisticas_basicas(num1, num2, num3)** que retorne simultáneamente: • El mayor de los tres números • El menor de los tres números • El promedio de los tres números Luego muestra cómo capturar los tres valores en variables separadas y también cómo capturarlos en una sola variable (tupla). Prueba con: 15, 8, 22

Punto clave: *Una función puede retornar múltiples valores usando comas.*

EJERCICIO 12: FUNCIÓN DE VALIDACIÓN BOOLEANA

Nivel: Intermedio | Tipo: Validación | Conceptos: return bool, lógica de validación

Define **es_email_valido(email)** que retorne True si el email cumple: • Contiene exactamente un símbolo @ • Tiene al menos un carácter antes del @ • Tiene al menos un punto después del @ • Termina con al menos 2 caracteres después del último punto Prueba con: "usuario@dominio.com", "usuario@dominio", "usuariodominio.com", "@dominio.com"

Punto clave: *Las funciones de validación retornan True/False para facilitar el control de flujo.*

EJERCICIO 13: FUNCIÓN CON PARÁMETROS POR DEFECTO

Nivel: Intermedio | Tipo: Parámetros opcionales | Conceptos: valores por defecto

Crea **generar_saludo(nombre, tratamiento="Sr./Sra.", idioma="español")** que retorne: • En español: "[tratamiento] [nombre], ¡bienvenido!" • En inglés: "[tratamiento] [nombre], welcome!" • En francés: "[tratamiento] [nombre], bienvenue!" Prueba estas llamadas: • generar_saludo("Ana") • generar_saludo("Carlos", "Dr.") • generar_saludo("Maria", "Dra.", "inglés") • generar_saludo("Pierre", idioma="francés")

Punto clave: *Los parámetros por defecto permiten flexibilidad en las llamadas.*

EJERCICIO 14: SCOPE DE VARIABLES (LOCAL VS GLOBAL)

Nivel: Intermedio | Tipo: Conceptual avanzado | Conceptos: scope, variables locales/globales

Analiza y predice qué imprimirá este código: `contador = 100 def incrementar(): contador = 1 contador += 5 return contador def mostrar_contador(): return contador` Preguntas: 1. ¿Qué devuelve incrementar()? 2. ¿Qué devuelve mostrar_contador() antes y después de llamar incrementar()? 3. ¿Por qué las variables dentro de las funciones no afectan las globales?

Punto clave: *Las variables dentro de funciones son locales por defecto.*

EJERCICIO 15: FUNCIÓN PROCESADORA DE TEXTO

Nivel: Intermedio | Tipo: Manipulación strings | Conceptos: métodos string, condicionales

Define **procesar_texto(texto, operacion)** que según la operación retorne: • "mayus": texto en mayúsculas • "minus": texto en minúsculas • "titulo": texto con formato de título • "longitud": número de caracteres • "palabras": número de palabras • "invertir": texto al revés Si la operación no es válida, retorna "Operación no soportada". Prueba con: "Hola Mundo Python" y todas las operaciones.

Punto clave: *Una función puede ser un procesador versátil según parámetros.*

EJERCICIO 16: CALCULADORA DE EDAD PRECISA

Nivel: Intermedio | Tipo: Cálculos complejos | Conceptos: lógica de fechas, condicionales

Crea **calcular_edad_exacta(dia_nac, mes_nac, año_nac, dia_actual, mes_actual, año_actual)** que: • Calcule la edad exacta en años • Considere si ya pasó el cumpleaños este año • Retorne también cuántos días faltan para el próximo cumpleaños • El resultado debe ser: "Edad: [años] años, Próximo cumpleaños en: [días] días" Prueba con una fecha de nacimiento conocida y la fecha actual.

Punto clave: *Las funciones pueden encapsular lógica compleja de cálculos.*

EJERCICIO 17: FUNCIÓN GENERADORA DE CONTRASEÑAS

Nivel: Intermedio | Tipo: Algoritmos | Conceptos: lógica condicional, strings

Define **generar_password(longitud, incluir_numeros=True, incluir_simbolos=False)** que: • Genere una contraseña de la longitud especificada • Use letras mayúsculas y minúsculas siempre • Incluya números (0-9) si incluir_numeros es True • Incluya símbolos (!@#$%^&*) si incluir_simbolos es True • Retorne la contraseña generada Simula la "aleatoriedad" usando posiciones basadas en la longitud.

Punto clave: *Las funciones pueden generar contenido dinámico basado en parámetros.*

EJERCICIO 18: FUNCIÓN DE ANÁLISIS NUMÉRICO

Nivel: Intermedio | Tipo: Análisis matemático | Conceptos: múltiples cálculos, return múltiple

Crea **analizar_numero(numero)** que retorne un diccionario con: • "es_par": True/False • "es_positivo": True/False • "es_primo": True/False (para números positivos hasta 100) • "cuadrado": número al cuadrado • "digitos": cantidad de dígitos • "suma_digitos": suma de todos los dígitos Prueba con: 17, -8, 25, 2

Punto clave: *Las funciones pueden realizar análisis completos y estructurados.*

EJERCICIOS AVANZADOS (19-30)

Casos complejos y aplicaciones reales

EJERCICIO 19: SISTEMA DE CALIFICACIONES UNIVERSITARIO

Nivel: Avanzado | Tipo: Sistema completo | Conceptos: múltiples funciones, lógica compleja

Desarrolla un sistema con estas funciones: 1. **calcular_nota_final(parcial1, parcial2, final, trabajos)** Fórmula: (parcial1*0.25) + (parcial2*0.25) + (final*0.4) + (trabajos*0.1) 2. **obtener_letra_calificacion(nota_numerica)** A: 90-100, B: 80-89, C: 70-79, D: 60-69, F: 0-59 3. **determinar_estado(letra)** A,B,C: "Aprobado", D: "Condicional", F: "Reprobado" 4. **reporte_estudiante(nombre, parcial1, parcial2, final, trabajos)** Que use todas las anteriores y retorne un reporte completo. Prueba con: Ana, 85, 78, 92, 88

> ***Punto clave:*** *Los sistemas reales se construyen combinando múltiples funciones especializadas.*

EJERCICIO 20: CONVERSOR DE UNIDADES AVANZADO

Nivel: Avanzado | Tipo: Múltiples conversiones | Conceptos: diccionarios, escalabilidad

Crea un sistema de conversión con: 1. **convertir_temperatura(valor, desde, hacia)** Soporta: celsius, fahrenheit, kelvin 2. **convertir_distancia(valor, desde, hacia)** Soporta: metros, kilómetros, millas, pies 3. **convertir_peso(valor, desde, hacia)** Soporta: gramos, kilogramos, libras, onzas 4. **conversor_universal(valor, tipo_magnitud, desde, hacia)** Que use las tres anteriores según el tipo Cada función debe validar que las unidades sean válidas y retornar error si no.

> ***Punto clave:*** *Los sistemas modulares permiten extensibilidad y mantenimiento fácil.*

EJERCICIOS AVANZADOS RESTANTES (21-30)

Para ver los enunciados completos de estos ejercicios, consulta el archivo Python adjunto:

- **EJERCICIO 21: ANALIZADOR DE TEXTO AVANZADO**
- **EJERCICIO 22: CALCULADORA FINANCIERA**
- **EJERCICIO 23: VALIDADOR DE DATOS EMPRESARIAL**
- **EJERCICIO 24: SIMULADOR DE JUEGO DE CARTAS**
- **EJERCICIO 25: GENERADOR DE REPORTES EMPRESARIALES**
- **EJERCICIO 26: ALGORITMO DE RECOMENDACIÓN SIMPLE**
- **EJERCICIO 27: SISTEMA DE INVENTARIO INTELIGENTE**
- **EJERCICIO 28: ANALIZADOR DE RENDIMIENTO WEB**
- **EJERCICIO 29: CALCULADORA DE CARBON FOOTPRINT**
- **EJERCICIO 30: PROYECTO INTEGRADOR - SISTEMA DE GESTIÓN ACADÉMICA**

SECUENCIA RECOMENDADA DE TRABAJO

1. **Fundamentos (Ejercicios 1-7):** Domina definición, llamada y return básico
2. **Parámetros y argumentos (Ejercicios 8-12):** Practica paso de datos
3. **Integración de conceptos (Ejercicios 13-17):** Combina con capítulos anteriores
4. **Conceptos avanzados (Ejercicios 18-22):** Funciones complejas y validación
5. **Proyectos integradores (Ejercicios 23-30):** Aplicaciones reales y sistemas completos

CRITERIOS DE EVALUACIÓN PARA DOMINIO EXPERTO

FUNDAMENTOS SÓLIDOS (Ejercicios 1-6):

• Comprende perfectamente def vs invocación
• Domina parámetros vs argumentos
• Distingue funciones estériles vs productivas
• Maneja return correctamente

INTEGRACIÓN Y APLICACIÓN (Ejercicios 7-18):

• Combina funciones con condicionales fluidamente
• Implementa validación robusta
• Usa funciones anidadas efectivamente
• Maneja scope de variables correctamente
• Aplica parámetros por defecto apropiadamente

MAESTRÍA Y SISTEMAS COMPLEJOS (Ejercicios 19-30):

• Diseña sistemas modulares y escalables
• Implementa algoritmos complejos
• Crea validaciones empresariales
• Desarrolla lógica de negocio sofisticada
• Integra múltiples conceptos cohesivamente

INDICADORES DE DOMINIO EXPERTO

✓ Código limpio y bien estructurado
✓ Funciones con responsabilidad única
✓ Manejo apropiado de errores
✓ Documentación clara de funciones
✓ Reutilización efectiva de código
✓ Diseño modular y escalable
✓ Aplicación de mejores prácticas

¡Éxito en tu camino hacia la maestría en funciones Python!