CLASE 8
Funciones I: Fundamentos

Resumen de Estudio

Curso de Python - Fundamentos

1. Que es una Funcion?

Una **funcion** es un bloque de codigo reutilizable con un nombre que realiza una tarea especifica. Las funciones nos permiten organizar el codigo, evitar repeticion y hacer programas mas mantenibles.

> *Analogia: Es como una maquina que recibe entrada, procesa y produce salida.*

Ya conocemos funciones built-in: print(), len(), input(), type(), range()

2. Sintaxis Basica

Estructura de una funcion:

```
def nombre_funcion():
    # codigo aqui
    print('Hola!')

# Llamar la funcion
nombre_funcion()
```

Componentes:

- **def**: palabra clave para definir funcion

- **nombre**: en snake_case (minusculas con guion bajo)

- **()**: parentesis obligatorios (para parametros)

- **:**: dos puntos al final de la linea

- **cuerpo**: indentado con 4 espacios

3. Definir vs Llamar

DEFINIR = Crear la funcion (escribir la receta)
LLAMAR = Ejecutar la funcion (cocinar con la receta)

```
# DEFINIR
def saludar():
    print('Hola!')

# LLAMAR
saludar()  # Ejecuta la funcion
```

> *IMPORTANTE: Definir una funcion NO la ejecuta. Debes llamarla con () para que corra.*

4. Parametros y Argumentos

Concepto	Descripcion	Donde aparece
PARAMETRO	Variable que recibe el valor	En la DEFINICION
ARGUMENTO	Valor real que se pasa	En la LLAMADA

```
def saludar(nombre):  # 'nombre' es PARAMETRO
    print(f'Hola {nombre}!')

saludar('Ana')  # 'Ana' es ARGUMENTO
```

Multiples parametros:

```
def presentar(nombre, edad, ciudad):
    print(f'{nombre} tiene {edad} anios')
    print(f'Vive en {ciudad}')

presentar('Maria', 25, 'Madrid')
```

El orden de los argumentos debe coincidir con el orden de los parametros.

5. Return - Devolver Valores

```
def calcular_area(base, altura):
    area = base * altura
    return area

resultado = calcular_area(5, 3)
print(resultado)  # 15
```

Caracteristicas de return:

- DEVUELVE un valor al lugar donde se llamo la funcion
- La funcion TERMINA inmediatamente al llegar a return
- El valor retornado puede guardarse en una variable o usarse directamente

6. Print vs Return - DIFERENCIA CRITICA

print()	return
Solo MUESTRA en pantalla	DEVUELVE valor al programa
No puedes usar el resultado	Puedes guardar y usar el resultado
Para mostrar informacion	Para calculos y procesamiento
Resultado = None	Resultado = valor retornado

```
def con_print(a, b):
    print(a + b)  # Solo muestra

def con_return(a, b):
    return a + b  # Devuelve

x = con_print(2, 3)   # x = None
y = con_return(2, 3)  # y = 5
```

REGLA: Si necesitas USAR el resultado despues, usa RETURN. Si solo quieres MOSTRAR, usa PRINT.

7. Scope - Ambito de Variables

SCOPE = Donde existe una variable. Define en que partes del codigo una variable es accesible.

Tipos de scope:

Tipo	Descripcion	Accesibilidad
LOCAL	Variables creadas DENTRO de funcion	Solo dentro de esa funcion
GLOBAL	Variables creadas FUERA de funciones	En todo el programa

```
mensaje = 'Soy global'  # GLOBAL

def mi_funcion():
    local = 'Soy local'  # LOCAL
    print(mensaje)  # OK - puede leer global
    print(local)    # OK

mi_funcion()
print(mensaje)  # OK
print(local)    # ERROR! No existe aqui
```

Buena practica:

Pasar valores como PARAMETROS en lugar de usar variables globales. Esto hace el codigo mas predecible y facil de mantener.

8. Return de Diferentes Tipos

Una funcion puede retornar CUALQUIER tipo de dato:

```
def obtener_numero(): return 42
def obtener_lista(): return [1, 2, 3]
def obtener_dict(): return {'nombre': 'Ana'}

# Return multiple (tupla)
def obtener_datos():
    return 'Ana', 25, 'Madrid'

nombre, edad, ciudad = obtener_datos()  # Desempaquetado
```

9. Early Return - Validaciones

```
def dividir_seguro(a, b):
    if b == 0:
        return 'Error: division por cero'
    return a / b
```

Early return: terminar la funcion temprano si hay un problema o condicion especial.

10. Docstrings - Documentacion

```
def calcular_promedio(numeros):
    """
    Calcula el promedio de una lista.
    Args: numeros - lista de numeros
    Returns: float - promedio
    """
    return sum(numeros) / len(numeros)
```

11. Errores Comunes

1. Olvidar llamar la funcion: saludar vs saludar()
2. Confundir print con return
3. Numero incorrecto de argumentos
4. Usar variable local fuera de su scope

12. Tabla de Referencia Rapida

Accion	Sintaxis	Ejemplo
Definir funcion	`def nombre():`	`def saludar():`
Con parametros	`def nombre(p1, p2):`	`def sumar(a, b):`
Retornar valor	`return valor`	`return resultado`
Llamar funcion	`nombre()`	`saludar()`
Con argumentos	`nombre(v1, v2)`	`sumar(5, 3)`
Early return	`if cond: return`	`if b==0: return "Error"`
Return multiple	`return v1, v2`	`return nombre, edad`

EJERCICIOS ADICIONALES

NIVEL PRINCIPIANTE

Ejercicio P1: Saludo Personalizado

Crear una funcion saludar_hora(nombre, hora) que reciba un nombre y una hora (0-23) y retorne un saludo apropiado: 'Buenos dias' (6-11), 'Buenas tardes' (12-19), 'Buenas noches' (20-5).

Ejercicio P2: Celsius a Fahrenheit

Crear funcion celsius_a_fahrenheit(celsius) que convierta temperatura. Formula: F = C * 9/5 + 32. Retornar el resultado redondeado a 1 decimal.

Ejercicio P3: Es Par o Impar

Crear funcion verificar_paridad(numero) que retorne 'Par' si el numero es par, 'Impar' si es impar, y 'No es entero' si no es un numero entero.

Ejercicio P4: Calcular Descuento

Crear funcion calcular_precio_final(precio, descuento_porcentaje) que retorne el precio despues de aplicar el descuento. Si el descuento es mayor a 100 o negativo, retornar 'Descuento invalido'.

Ejercicio P5: Contar Vocales

Crear funcion contar_vocales(texto) que retorne la cantidad de vocales (a, e, i, o, u) en un texto dado. Debe funcionar con mayusculas y minusculas.

NIVEL INTERMEDIO

Ejercicio I1: Validador de Password

Crear funcion validar_password(password) que retorne un diccionario con: 'valido': True/False, 'longitud_ok': True/False (min 8), 'tiene_numero': True/False, 'tiene_mayuscula': True/False. La password es valida si cumple los 3 criterios.

Ejercicio I2: Estadisticas de Lista

Crear funcion estadisticas_lista(numeros) que reciba una lista de numeros y retorne un diccionario con: 'minimo', 'maximo', 'suma', 'promedio', 'cantidad'. Si la lista esta vacia, retornar None.

Ejercicio I3: Calculadora de Propinas

Crear funcion calcular_cuenta(subtotal, porcentaje_propina, num_personas) que retorne una tupla con: (total_con_propina, propina_total, monto_por_persona). Validar que todos los valores sean positivos.

Ejercicio I4: Analizador de Calificaciones

Crear funcion analizar_calificaciones(calificaciones) que reciba un diccionario {nombre: [notas]} y retorne otro diccionario con cada estudiante, su promedio y su letra (A>=90, B>=80, C>=70, D>=60, F<60).

Ejercicio I5: Generador de Recibo

Crear funcion generar_recibo(productos) que reciba una lista de tuplas (nombre, precio, cantidad) y retorne un diccionario con: 'items' (lista de strings formateados), 'subtotal', 'iva' (16%), 'total'.

NIVEL AVANZADO

Ejercicio A1: Sistema de Inventario

Crear 4 funciones: agregar_producto(inventario, nombre, cantidad, precio), vender_producto(inventario, nombre, cantidad), valor_total_inventario(inventario), productos_bajo_stock(inventario, minimo). El inventario es un diccionario donde cada producto tiene 'cantidad' y 'precio'. Manejar errores de stock insuficiente.

Ejercicio A2: Analizador de Texto Avanzado

Crear funcion analizar_texto_completo(texto) que retorne diccionario con: 'total_caracteres', 'total_palabras', 'total_oraciones', 'palabra_mas_larga', 'promedio_longitud_palabra', 'frecuencia_palabras' (dict), 'palabras_unicas'. Ignorar signos de puntuacion para el conteo de palabras.

Ejercicio A3: Calculadora de Prestamos

Crear funcion simular_prestamo(monto, tasa_anual, meses) que retorne diccionario con: 'pago_mensual', 'total_a_pagar', 'total_intereses', 'tabla_amortizacion' (lista de dicts con mes, pago, interes, capital, saldo). Usar formula de amortizacion francesa. Incluir validaciones de parametros positivos.

SOLUCIONES

SOLUCIONES NIVEL PRINCIPIANTE

Solucion P1:

```
def saludar_hora(nombre, hora):
    if 6 <= hora <= 11:
        saludo = 'Buenos dias'
    elif 12 <= hora <= 19:
        saludo = 'Buenas tardes'
    else:
        saludo = 'Buenas noches'
    return f'{saludo}, {nombre}!'
```

Solucion P2:

```
def celsius_a_fahrenheit(celsius):
    fahrenheit = celsius * 9/5 + 32
    return round(fahrenheit, 1)
```

Solucion P3:

```
def verificar_paridad(numero):
    if not isinstance(numero, int):
        return 'No es entero'
    if numero % 2 == 0:
        return 'Par'
    return 'Impar'
```

Solucion P4:

```
def calcular_precio_final(precio, descuento):
    if descuento < 0 or descuento > 100:
        return 'Descuento invalido'
    return precio * (1 - descuento/100)
```

Solucion P5:

```
def contar_vocales(texto):
    vocales = 'aeiouAEIOU'
    contador = 0
    for char in texto:
        if char in vocales:
            contador += 1
    return contador
```

SOLUCIONES NIVEL INTERMEDIO

Solucion I1:

```
def validar_password(password):
    longitud_ok = len(password) >= 8
```

```
    tiene_numero = any(c.isdigit() for c in password)
    tiene_mayuscula = any(c.isupper() for c in password)
    valido = longitud_ok and tiene_numero and tiene_mayuscula
    return {
        'valido': valido,
        'longitud_ok': longitud_ok,
        'tiene_numero': tiene_numero,
        'tiene_mayuscula': tiene_mayuscula
    }
```

Solucion I2:

```
def estadisticas_lista(numeros):
    if not numeros:
        return None
    return {
        'minimo': min(numeros),
        'maximo': max(numeros),
        'suma': sum(numeros),
        'promedio': sum(numeros)/len(numeros),
        'cantidad': len(numeros)
    }
```

Solucion I3:

```
def calcular_cuenta(subtotal, propina_pct, personas):
    if subtotal <= 0 or propina_pct < 0 or personas <= 0:
        return None
    propina = subtotal * propina_pct / 100
    total = subtotal + propina
    por_persona = total / personas
    return (round(total,2), round(propina,2), round(por_persona,2))
```

Solucion I4:

```
def analizar_calificaciones(calificaciones):
    resultado = {}
    for nombre, notas in calificaciones.items():
        promedio = sum(notas) / len(notas)
        if promedio >= 90: letra = 'A'
        elif promedio >= 80: letra = 'B'
        elif promedio >= 70: letra = 'C'
        elif promedio >= 60: letra = 'D'
        else: letra = 'F'
        resultado[nombre] = {'promedio': promedio, 'letra': letra}
    return resultado
```

Solucion I5:

```
def generar_recibo(productos):
    items = []
    subtotal = 0
    for nombre, precio, cantidad in productos:
        total_item = precio * cantidad
```

```
        items.append(f'{nombre} x{cantidad}: ${total_item:.2f}')
        subtotal += total_item
    iva = subtotal * 0.16
    return {'items': items, 'subtotal': subtotal,
            'iva': round(iva,2), 'total': round(subtotal+iva,2)}
```

SOLUCIONES NIVEL AVANZADO

Solucion A1:

```
def agregar_producto(inv, nombre, cantidad, precio):
    if nombre in inv:
        inv[nombre]['cantidad'] += cantidad
    else:
        inv[nombre] = {'cantidad': cantidad, 'precio': precio}
    return inv

def vender_producto(inv, nombre, cantidad):
    if nombre not in inv:
        return 'Producto no existe'
    if inv[nombre]['cantidad'] < cantidad:
        return 'Stock insuficiente'
    inv[nombre]['cantidad'] -= cantidad
    return inv

def valor_total_inventario(inv):
    total = 0
    for prod in inv.values():
        total += prod['cantidad'] * prod['precio']
    return total

def productos_bajo_stock(inv, minimo):
    return [n for n, p in inv.items() if p['cantidad'] < minimo]
```

Solucion A2:

```
def analizar_texto_completo(texto):
    import re
    palabras = re.findall(r'\b\w+\b', texto.lower())
    oraciones = re.split(r'[.!?]+', texto)
    oraciones = [o for o in oraciones if o.strip()]
    frecuencia = {}
    for p in palabras:
        frecuencia[p] = frecuencia.get(p, 0) + 1
    palabra_larga = max(palabras, key=len) if palabras else ''
    prom_long = sum(len(p) for p in palabras)/len(palabras) if palabras else 0
    return {
        'total_caracteres': len(texto),
        'total_palabras': len(palabras),
        'total_oraciones': len(oraciones),
        'palabra_mas_larga': palabra_larga,
        'promedio_longitud_palabra': round(prom_long, 2),
        'frecuencia_palabras': frecuencia,
        'palabras_unicas': len(frecuencia)
    }
```

Solucion A3:

```
def simular_prestamo(monto, tasa_anual, meses):
    if monto <= 0 or tasa_anual < 0 or meses <= 0:
```

```
        return None
    tasa_mensual = tasa_anual / 100 / 12
    if tasa_mensual > 0:
        pago = monto * (tasa_mensual * (1+tasa_mensual)**meses)
        pago = pago / ((1+tasa_mensual)**meses - 1)
    else:
        pago = monto / meses
    tabla = []
    saldo = monto
    for mes in range(1, meses + 1):
        interes = saldo * tasa_mensual
        capital = pago - interes
        saldo -= capital
        tabla.append({'mes': mes, 'pago': round(pago,2),
            'interes': round(interes,2), 'capital': round(capital,2),
            'saldo': round(max(0, saldo),2)})
    total = pago * meses
    return {'pago_mensual': round(pago,2), 'total_a_pagar': round(total,2),
            'total_intereses': round(total-monto,2), 'tabla_amortizacion':
tabla}
```