



FUNDAMENTOS DE INFORMATICA

INGENIERA SILVIA PATRICIA BARDELLI

PROFESORA ING. SILVIA PATRICIA BARDELLI

CLASE NRO 5

Temas:

- § Programación modular
- § Funciones
- § Valor de retorno
- § Ámbito de las variables

INTEGRACIÓN VERSUS ESPECIALIZACIÓN

§ En la vida cotidiana es común que las tareas se distribuyan entre varias personas o secciones de una empresa con el fin de optimizar el proceso.

INTEGRACIÓN VERSUS ESPECIALIZACIÓN

- § Por ejemplo, una fábrica automotriz tiene secciones de estampado, soldadura, pintura, montaje y control final.
- § Sería difícil poder integrar todos estos procesos en uno solo.
- § Lo mismo ocurre con los programas.

INTEGRACIÓN VERSUS ESPECIALIZACIÓN

§ Aunque sería posible desarrollar todo el programa como una sola unidad, en general se acostumbra a dividir el trabajo en módulos más pequeños llamados *funciones*.

FUNCIONES

Una **función** es un módulo independiente de programa que realiza una tarea específica.

EJEMPLO 1: CÁLCULO DE UN PROMEDIO

Función

```
def calcularpromedio(a, b):  
    total = (a + b) / 2  
    return total
```

Programa principal

```
x = int(input("Ingrese un numero entero: "))  
y = int(input("Ingrese otro numero entero: "))  
resultado = calcularpromedio(x, y)  
print("El promedio de", x, "y", y, "es", resultado)
```

EJEMPLO 1: CÁLCULO DE UN PROMEDIO

Función

```
def calcularpromedio(a, b):  
    total = (a + b) / 2  
    return total
```

Programa principal

```
x = int(input("Ingrese un numero entero: "))  
y = int(input("Ingrese otro numero entero: "))  
resultado = calcularpromedio(x, y)  
print("El promedio de", x, "y", y, "es", resultado)
```



EJEMPLO 1: CÁLCULO DE UN PROMEDIO

Función

```
def calcularpromedio(a, b):  
    total = (a + b) / 2  
    return total
```

Programa principal

```
x = int(input("Ingrese un numero entero: "))  
y = int(input("Ingrese otro numero entero: "))  
resultado = calcularpromedio(x, y)  
print("El promedio de", x, "y", y, "es", resultado)
```

VENTAJAS DE TRABAJAR CON FUNCIONES

1. Es posible dividir el programa en módulos más pequeños, para que cada función realice una tarea concreta. Esto facilita el desarrollo y la comprensión del programa.

VENTAJAS DE TRABAJAR CON FUNCIONES

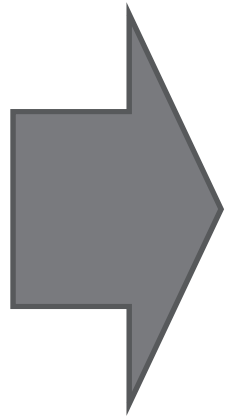
2. Las funciones pueden invocarse muchas veces dentro del mismo programa, evitando la reiteración innecesaria de código.

VENTAJAS DE TRABAJAR CON FUNCIONES

3. Una misma función puede ser utilizada en otros programas, evitando tener que volver a escribir código ya escrito.

ELEMENTOS DE UNA FUNCIÓN

1. Palabra reservada *def*.



Función

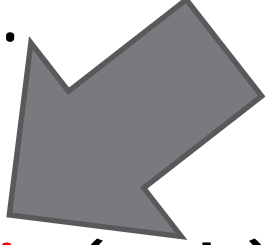
```
def calcularpromedio(a, b):  
    total = (a + b) / 2  
    return total
```

ELEMENTOS DE UNA FUNCIÓN

2. Nombre de la función.

Función

```
def calcularpromedio(a, b):  
    total = (a + b) / 2  
    return total
```

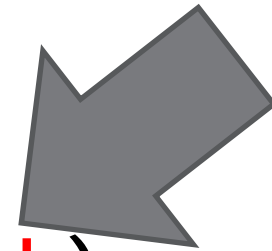


ELEMENTOS DE UNA FUNCIÓN

3. Los parámetros o argumentos.

Función

```
def calcularpromedio(a, b):  
    total = (a + b) / 2  
    return total
```

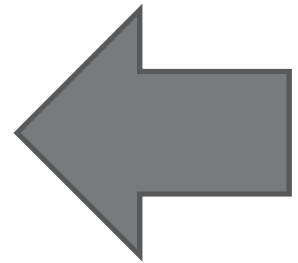


ELEMENTOS DE UNA FUNCIÓN

4. El carácter "dos puntos".

Función

```
def calcularpromedio(a, b):  
    total = (a + b) / 2  
    return total
```



ELEMENTOS DE UNA FUNCIÓN

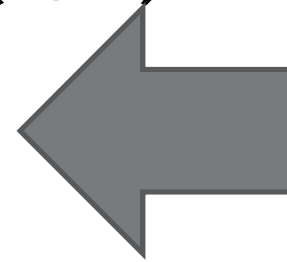
5. El cuerpo de la función, definido por la sangría.

Función

```
def calcularpromedio(a, b):
```

```
    total = (a + b) / 2
```

```
    return total
```



ELEMENTOS DE UNA FUNCIÓN

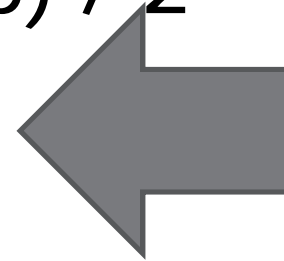
6. La instrucción *return*.

Función

```
def calcularpromedio(a, b):
```

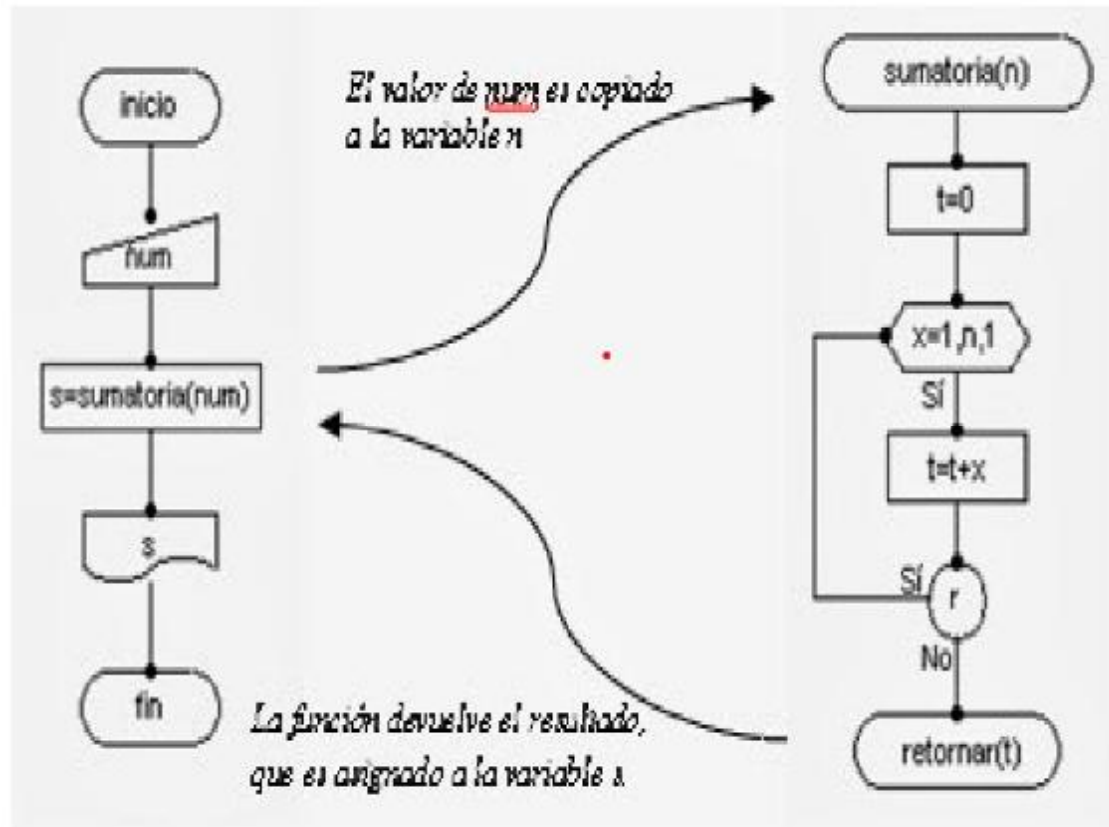
```
    total = (a + b) / 2
```

```
    return total
```



FLUJO DURANTE EL LLAMADO A LA FUNCIÓN

Una llamada a una función es como un desvío en el flujo de la ejecución. En vez de pasar a la siguiente sentencia, el flujo salta al cuerpo de la función, ejecuta todas las sentencias que hay allí, y después vuelve al punto donde lo dejó.



PARA TENER EN CUENTA

- § El nombre de la función debe respetar las mismas reglas de los nombres de las variables.
- § Se recomienda que contenga un verbo en infinitivo que describa la tarea que realiza.

PARA TENER EN CUENTA

- § Los parámetros se escriben entre paréntesis y separados por comas, colocando "dos puntos" al final de la línea.
- § Puede haber 0 o más parámetros.

PARÁMETROS

- § Los parámetros son la puerta de entrada a la función.
- § A través de ellos la función recibe datos para poder trabajar.

TIPOS DE PARÁMETROS

- § Existen parámetros *formales* y parámetros *reales*.
- § Los formales son los que se escriben en el encabezado de la función.

PARÁMETROS

- § Los parámetros reales son los que se escriben en la llamada o invocación.
- § Pueden llevar el mismo nombre o nombres diferentes.

PARÁMETROS

Función

```
def calcularpromedio(a, b):  
    total = (a + b) / 2  
    return total
```

*Parámetros
formales
a y b*



Programa principal

```
x = int(input("Ingrese un numero entero: "))  
y = int(input("Ingrese otro numero entero: "))  
resultado = calcularpromedio(x, y)  
print("El promedio de", x, "y", y, "es", resultado)
```

*Parámetros
reales
X e y*



PARÁMETROS

§ Los parámetros formales actúan dentro de la función como *representantes* de los parámetros reales.

DEVOLUCIÓN DE RESULTADOS


- § La función realiza su trabajo y obtiene uno o más resultados que debe entregarle al programa principal.
- § Esto se hace a través de la instrucción *return*, que además finaliza la ejecución de la función.

DEVOLUCIÓN DE RESULTADOS

Función

```
def calcularpromedio(a, b):  
    total = (a + b) / 2  
    return total
```

Programa principal



```
x = int(input("Ingrese un numero entero: "))  
y = int(input("Ingrese otro numero entero: "))  
resultado = calcularpromedio(x, y)  
print("El promedio de", x, "y", y, "es", resultado)
```

ÁMBITO DE LAS VARIABLES

§ Este proceso de enviar datos a través de los parámetros y recibir resultados enviados con un return es necesario porque las funciones no deben utilizar las variables del programa principal y viceversa.

ÁMBITO DE LAS VARIABLES

total sólo
puede usarse
dentro de la
función

Función

```
def calcularpromedio(a, b):  
    total = (a + b) / 2  
    return total
```

x e y sólo
pueden
usarse
en el
programa
principal

Programa principal

```
x = int(input("Ingrese un numero entero: "))  
y = int(input("Ingrese otro numero entero: "))  
resultado = calcularpromedio(x, y)  
print("El promedio de", x, "y", y, "es", resultado)
```

CONCLUSIÓN

- § Todo dato que la función necesite debe pasarse como parámetro.
- § Las funciones no deben utilizar las variables del programa principal, y viceversa.

CONCLUSIÓN

Función

```
def calcularpromedio():  
    total = (x + y) / 2  
    return total
```

Programa principal

```
x = int(input("Ingrese un numero entero: "))  
y = int(input("Ingrese otro numero entero: "))  
resultado = calcularpromedio()  
print("El promedio de", x, "y", y, "es", resultado)
```

incorrecto

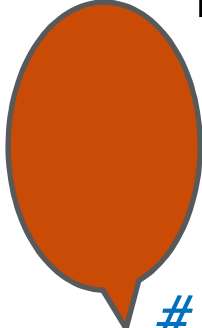
IMPORTANTE

Las funciones deben escribirse **antes** que el programa principal.

Ni durante, ni después.

El programa principal debe ubicarse **siempre** al final del código.

IMPORTANTE



Todo programa que utilice funciones debe incluir un comentario que indique dónde comienza el programa principal.

Programa principal

```
x = int(input("Ingrese un numero entero: "))  
y = int(input("Ingrese otro numero entero: "))  
resultado = calcularpromedio(x, y)  
print("El promedio de", x, "y", y, "es", resultado)
```

EJEMPLO 2

Objetivo:

Escribir una función para ingresar un número entero a través del teclado, verificando que éste se encuentre dentro de los límites permitidos.

EJEMPLO 2

```
def leerentero(minimo, maximo):  
    print("Ingrese un numero entero entre", minimo, "y",  
          maximo, ":")  
    a = int(input()) # ¿Por qué el mensaje no se muestra  
aquí?  
    while a<minimo or a>maximo:  
        print("Valor incorrecto.")  
        print("Debe estar entre", minimo, "y", maximo)  
        a = int(input("Ingrese un numero entero: "))  
    return a
```

EJEMPLO 3

Objetivo:

Escribir una función que reciba como parámetros la fecha de nacimiento de una persona y la fecha actual, y devuelva la edad de la persona.

EJEMPLO 3

```
def calcularedad(dnac, mnac, anac, dact, mact, aact):  
    edad = aact - anac  
    if mact < mnac:  
        edad = edad - 1  
    elif mact == mnac and dact < dnac:  
        edad = edad - 1  
    return edad
```

EJEMPLO 4

Objetivo:

Ingresa por teclado tres números enteros correspondientes a una fecha y devolver la fecha ingresada como valor de retorno.

EJEMPLO 5

Programa principal

```
print("Ingrese el día de nacimiento:")
dnac = leerentero(1, 31)

print("Ingrese el mes de nacimiento:")
mnac = leerentero(1, 12)

print("Ingrese el año de nacimiento:")
anac= leerentero(1583,2100)

print("Ingrese el día de hoy:")
dhoy = leerentero(1, 31)

print("Ingrese el mes de hoy:")
mhoy = leerentero(1, 12)

print("Ingrese el año de hoy:")
ahoy= leerentero(1583,2100)

edad=calcularedad(dnac,mnac,anac,dhoy,mhoy,ahoy)
print("Ud. tiene",edad,"años")
```


Ahora vamos a combinar
estas últimas dos funciones
con el programa principal.

PROGRAMA COMPLETO

Programa principal

```
print("Ingrese el día de nacimiento:")
dnac = leerentero(1, 31)
print("Ingrese el mes de nacimiento:")
mnac = leerentero(1, 12)
print("Ingrese el año de nacimiento:")
anac= leerentero(1583,2100)
print("Ingrese el día de hoy:")
dhoy = leerentero(1, 31)
print("Ingrese el mes de hoy:")
mhoy = leerentero(1, 12)
print("Ingrese el año de hoy:")
ahoy= leerentero(1583,2100)
edad=calcularedad(dnac,mnac,anac,d
hoy,mhoy,ahoy)
print("Ud. tiene",edad,"años")
```

```
def leerentero(min, max):
    print("Ingrese un número entre entre",
min, "y", max, end="")
    a = int(input()) # ¿Por qué el mensaje no
se muestra aquí?
    while a<min or a>max:
        print("Valor incorrecto. Debe estar
entre", min, "y", max)
        a=int(input("Ingrese un numero
entero: "))
    return a
```

```
def calcularedad(dnac, mnac, anac, dact,
mact, aact):
    edad = aact - anac
    if mact < mnac:
        edad = edad - 1
    elif mact == mnac and dact < dnac:
        edad = edad - 1
    return edad
```

EJEMPLO 5

Este ejemplo incorpora tres novedades:

- § La función puede no recibir parámetros, cuando éstos no sean necesarios.
- § Es posible devolver más de un valor de retorno.
- § Desde una función se puede llamar a otra función.

EJEMPLO 6

Objetivo:

Imprimir una fecha
recibida como parámetro.

¿Qué valor tenemos que devolver?

EJEMPLO 6 RESOLUCION

```
def imprimirfecha(d, m, a):  
    print("La fecha es", d, "/", m, "/", a)
```

...que se invoca como...

```
imprimirfecha(dia, mes, año)
```

EJEMPLO 7

Imprimir una columna de asteriscos, donde su altura se recibe como parámetro usando una función

Por ejemplo: ingrese una altura 5

*

*

*

*

*

EJEMPLO 7 RESOLUCIÓN

```
def imprimiraltura(a):
```

```
    n=0
```

```
    while n<a:
```

```
        print ("*")
```

```
        n=n+1
```



```
altura=int(input("ingrese una altura"))
```

```
imprimiraltura(altura)
```

EJEMPLO 8

Realiza una función llamada relacion(a, b) que a partir de dos números cumpla lo siguiente:

Si el primer número es mayor que el segundo, debe devolver 1.

Si el primer número es menor que el segundo, debe devolver -1.

Si ambos números son iguales, debe devolver un 0.

Solo se pueden ingresar valores positivos

EJEMPLO 8 RESOLUCION

```
def relacion(a,b):
```

```
    if a>b:
```

```
        return 1
```

```
    elif a<b:
```

```
        return -1
```

```
    else:
```

```
        return 0
```

```
nro1=int(input("Ingrese un numero positivo"))
while nro1<0:
    nro1=int(input("Error, Ingrese un numero positivo"))
nro2=int(input("Ingrese un numero positivo"))
while nro2<0:
    nro2=int(input("Error, Ingrese un numero positivo"))
resultado=relacion(nro1, nro2)
print(resultado)
if resultado==1:
    print("el primer número es mayor que el segundo", nro1,nro2)
elif resultado==-1:
    print("el primer número es menor que el segundo", nro1,nro2)
else:
    print("ambos números son iguales", nro1,nro2)
```

EJEMPLO 8 RESOLUCION PROGRAMA PRINCIPAL

EJEMPLO 9

Escribir una función que, dado un numero de DNI, retorne True si el número es válido y False si no lo es.

Para que un numero de DNI sea válido debe tener entre 7 y 8 dígitos.

Diseñar el pseudocódigo para ver si el algoritmo es valido

Luego en lenguaje python.

EJEMPLO 9 SEUDOCÓDIGO

Necesitamos una variable que cuente los digitos

cantidad=0

mientras haya digitos en el numero

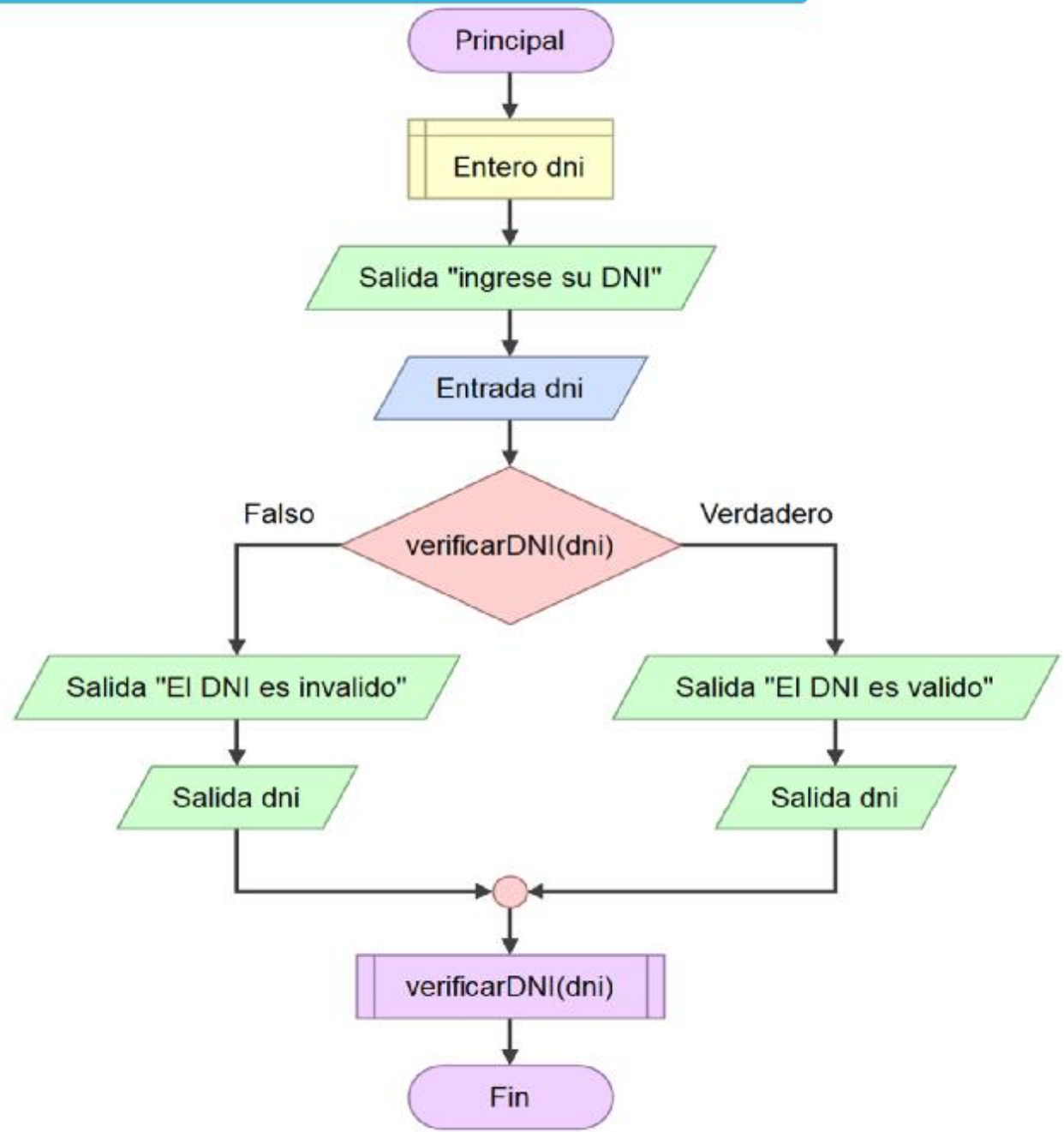
- sumar 1 a cantidad

- eliminar 1 digito del numero

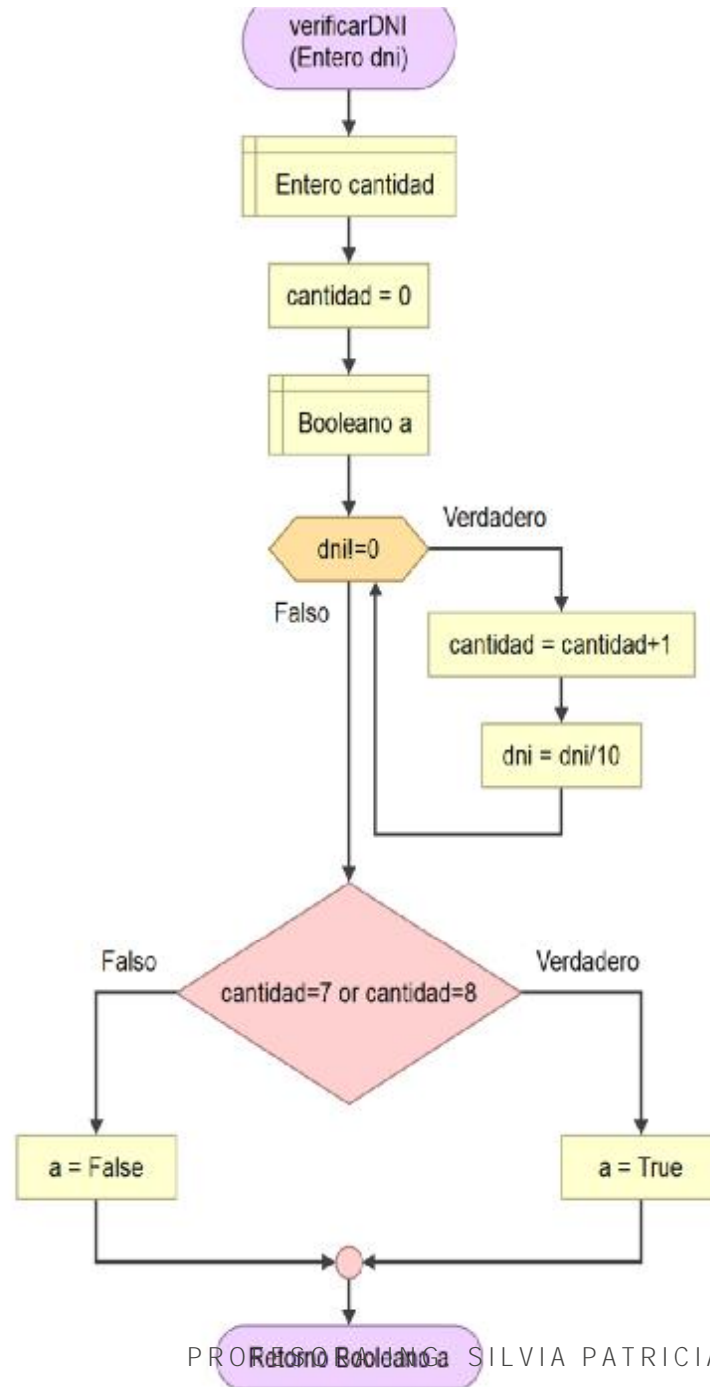
Si cantidad es 7 u 8: el DNI es valido

sino: es invalido

EJEMPLO 9 SEUDOCÓDIGO



EJEMPLO 9 SEUDOCÓDIGO



EJEMPLO 9 PYTHON

```
def verificarDNI(dni):  
    cantidad=0  
    while dni!=0:  
        cantidad=cantidad+1  
        dni=dni//10  
    if cantidad==7 or cantidad==8:  
        return True  
    else:  
        return False
```

```
dni=int(input("Ingrese su DNI"))  
if verificarDNI(dni):  
    print("El DNI", dni, "es valido")  
else:  
    print("El DNI", dni, "es invalido")
```

Ejercitación

- Práctica 6 Funciones: Completa