Programación modular

Programación Modular

Programación Modular

Hasta el momento hemos hecho para **resolver un problema**: Un **programa** que **tiene varias tareas**

Ejemplo:

Dados los lados diferentes de un rectángulo, hallar el área y el perímetro

Tarea1	Tarea2	Tarea3	Tarea 4
Pedir datos: lado 1 y lado 2	Hallar área	Hallar perímetro	Mostrar área y perímetro

Programación Modular

Construcción de programas modulares

Una rutina, módulo o subprograma → Es un programa encargado de realizar una tarea específica

Un programa modular → Está formado por tantas rutinas como tareas diferentes haya que realizar para lograr el objetivo

Función → Segmento de código que tiene un encabezado, un cuerpo, y una instrucción de retorno y en la cual se hace alguna tarea: Recibe 0 o más parámetros de entrada. Las funciones permiten reutilizar código

Programación Modular

Construcción de programas modulares

Ventajas

- Facilita el desarrollo de las soluciones y permite que éste sea gradual
- Facilita la localización de los errores, en qué función ocurren.
- Permite la reutilización de una o más funciones. Una función se puede utilizar más de una vez si es necesario, sin tener que escribirla nuevamente
- Hace el código más legible

Programación Modular

funcion4

funcion3

funcion5

funcion1

funcion6

funcion2

main()

Programación Modular

Estructura de una Función

Definición

```
def  nombre_función (lista de parámetros separados por coma):
        cuerpo de la función
        return – no siempre es necesario
```

Invocación o llamado de la función

```
nombre_función (lista de argumentos):
```

Programación Modular

Comunicación entre funciones

Parámetro ← argumento

Argumentos

- Se pasan cuando se llaman o invocan una función. Van dentro de los paréntesis
- Pueden ser expresiones, constantes o variables

Parámetros

- Aparecen en el encabezado de la función y reciben los argumentos
- Siempre tienen que ser variables

Los argumentos y los parámetros tienen que coincidir en cantidad y tipo. Se asocia el primer argumento al primer parámetro, el segundo argumento con el segundo parámetro, y así sucesivamente.

Argumento inmutable: variables simples, expresiones.
Se pasa una copia o un valor. Su contenido no se cambia en la función que se invoca.

Argumento mutable: vectores, matrices, listas
Su valor se puede cambiar en la función que se invoca

Importante: Sino hay argumentos no hay parámetros, Si se retorna se recibe, sino se retorna no se recibe

Programación Modular

```
def  nombre_función1 (lista de parámetros separados por coma):
        cuerpo de la función 1

def  nombre_función2 (lista de parámetros separados por coma):
        cuerpo de la función 2
…
```

La función principal main debe dar una buena idea del problema que resuelve, sin los detalles, los cuales están en cada función que se invoque.

```
def  main():
        cuerpo función main
main()
```

Programación Modular

Función

Definición

```
def impares (fin):
        for cont in range(1,fin+1,2)
                print(cont)
```

Invocación

```
fin= int( (input("Escriba los impares de 1 hasta donde me diga", hasta dónde?))
impares(fin)
```

Programación Modular

Función

Definición

```
def impares (fin):
        fon cont in range(1,fin+1,2)
                print(cont)
```

Invocación

```
impares(fin)
```

Programación Modular

Función

Definición

```
def funcionEjemplo():
        print(“Hola mundo”)
```

Invocación

```
funcionEjemplo()
```

Programación Modular

#Este programa calcula el factorial de n

Definición)

```
def factorial (n):
    fact= 1
    cont=1
    while cont<=n:
            fact=fact*cont
            cont=cont+1
     return fact
```

cont y fact son variables locales en la función **factorial**. Se crean cuando la función se ejecuta

¿Qué pasa si ponemos instrucciones después de un return?

```
def main ():
    print("Calculo el factorial de un número")

    n =int(input("Ingrese el número"))
    fact= factorial(n)  (Invocación)
    print("El factorial de:",n, "es:",fact)
    print("Chao chao")
main()
```

n y fact son variables locales en el **main**. Se crean cuando la función se ejecuta

Programación Modular

```
#Este programa suma dos números de manera modular

def pedir_numeros():

    n1=int(input("Escriba numero 1"))
    n2=int(input("Escriba numero 2"))

    return [n1, n2]

def sumar (n1,n2):

    sum= n1+n2
    return sum

def mostrar (n1, n2, suma):

    print ("La suma de ", n1, "y", n2, " es ", suma)
```

```
def main ():

    print("Bienvenidos al programa que suma dos números enteros")

    [n1, n2] = pedir_numeros()
    print ("Los valores ingresados son:", n1,n2)
    suma= sumar(n1,n2)
    mostrar(n1,n2,suma)

    print("Chao chao")

main()
```

n1, n2 y suma son variables locales en el main. Se crean cuando la función se ejecuta

Programación Modular

```
def main ():

    print("Bienvenidos al programa que suma dos
números enteros")

    #Pedir números
    [num1, num2] = pedir_numeros()

    #Sumar los números
    suma= sumar(num1,num2)

    #Mostrar los resultados
    mostrar(num1,num2,suma)

    print("Chao chao")

main()
```

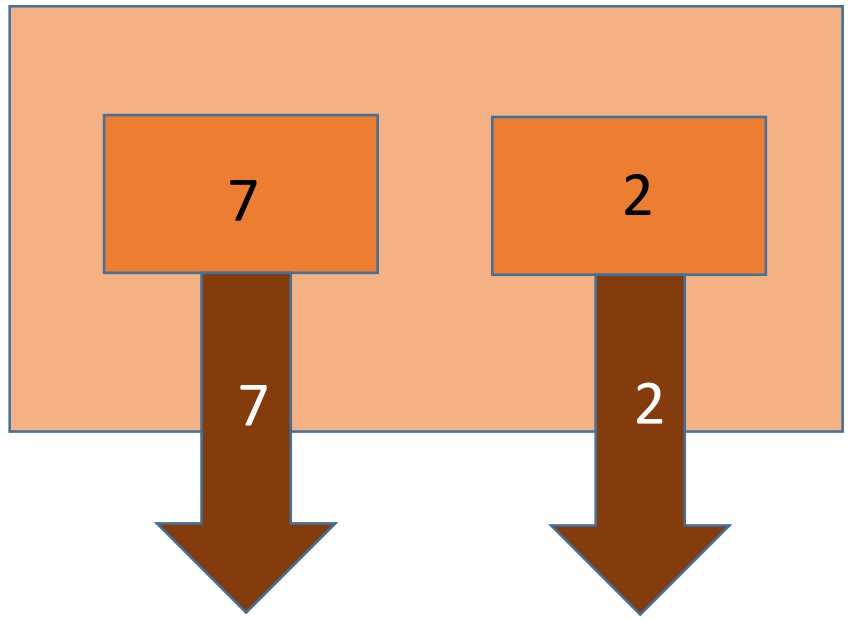
Pantalla

Bienvenidos al programa que suma dos números enteros

Programación Modular

```
def pedir_numeros():

    n1=int(input("Escriba numero 1: "))
    n2=int(input("Escriba numero 2: "))

    return [n1, n2]
```

Entorno de memoria pedir_numeros



Pantalla

Bienvenidos al programa que suma dos números enteros

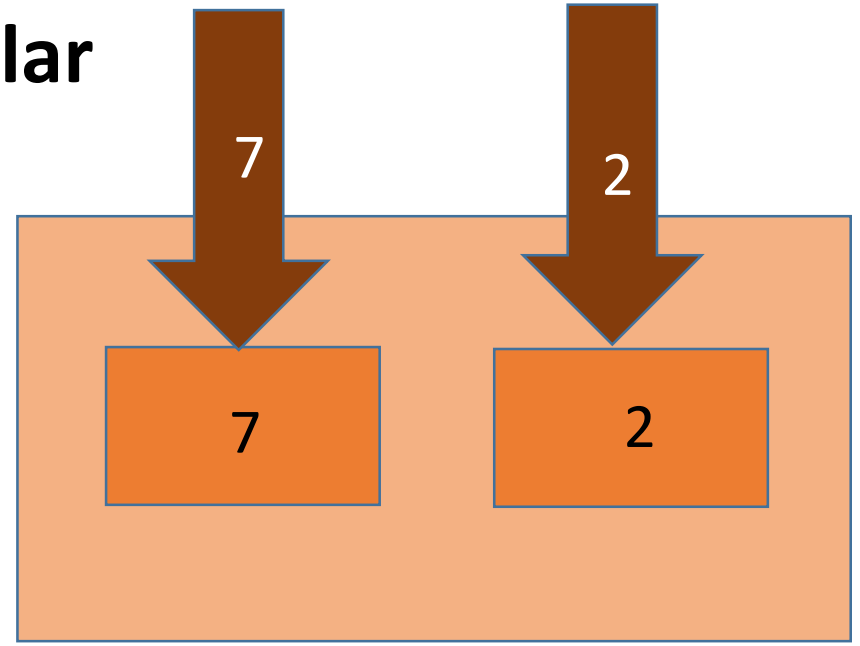
Escriba numero 1

Escriba numero 2

Programación Modular

main()

```
def main ():

    print("Bienvenidos al programa que suma dos
números enteros")

    #Pedir números
    [num1, num2] = pedir_numeros()

    #Sumar los números
    suma= sumar(num1,num2)

    #Mostrar los resultados
    mostrar(num1,num2,suma)

    print("Chao chao")

main()
```



Pantalla

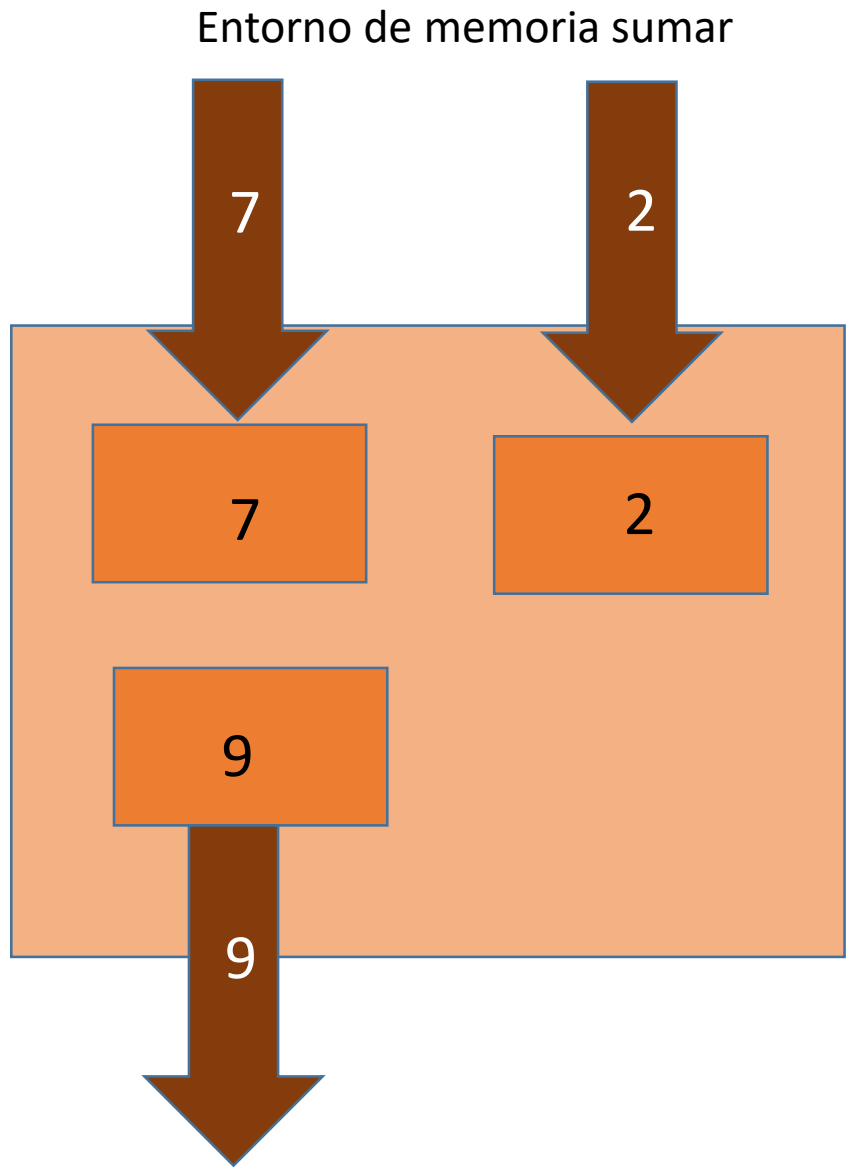
Bienvenidos al programa que suma dos números enteros

Escriba numero 1

Escriba numero 2

Programación Modular

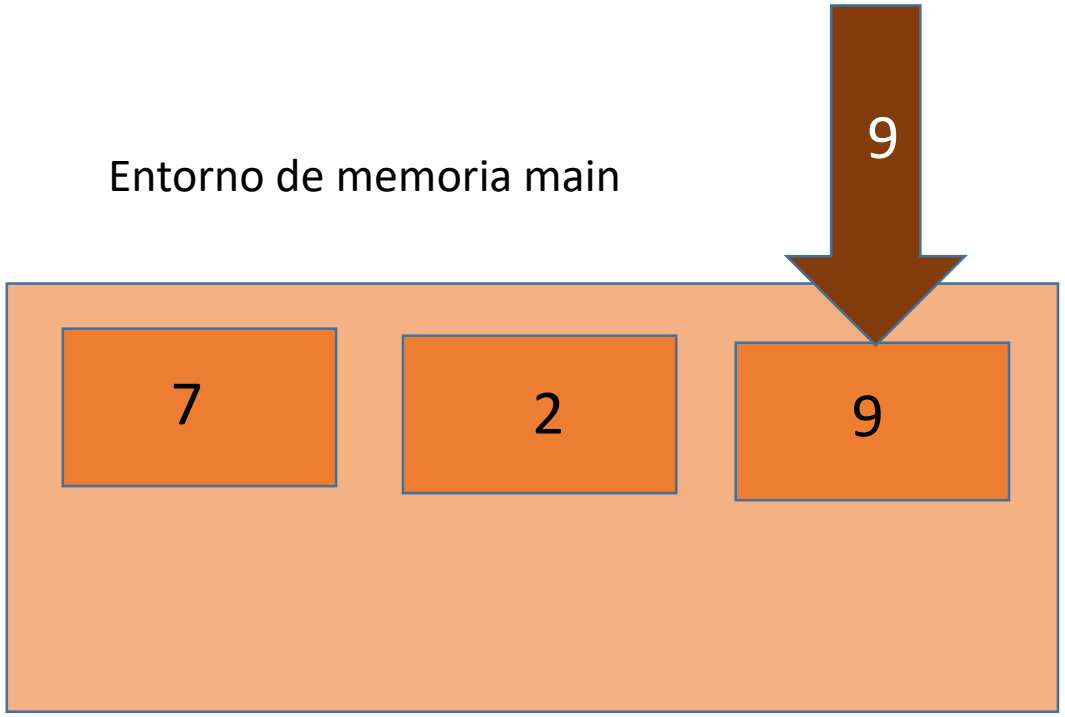
```
def sumar (n1,n2):

    suma= n1+n2
    return suma
```



Programación Modular

main()

```
def main ():

    print("Bienvenidos al programa que suma dos números enteros")

    #Pedir números
    [num1, num2] = pedir_numeros()

    #Sumar los números
    suma= sumar(num1,num2)

    #Mostrar los resultados
    mostrar(num1,num2,suma)

    print("Chao chao")

main()
```



Pantalla

Bienvenidos al programa que suma dos números enteros

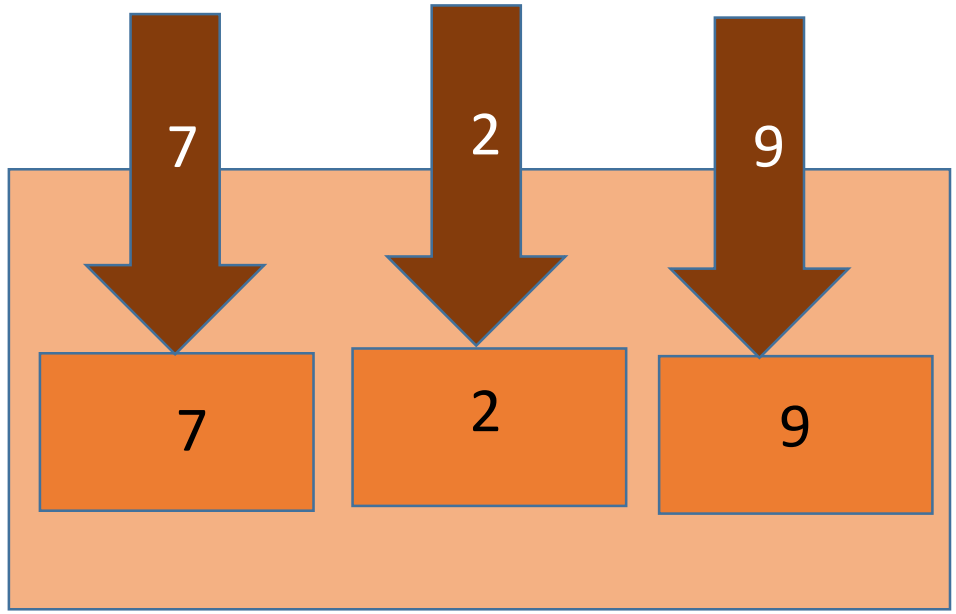
Escriba numero 1

Escriba numero 2

Programación Modular

Entorno de memoria mostrar

```
def mostrar (n1, n2, suma):

    print ("La suma de ", n1, "y", n2, " es ", suma)
```



Pantalla

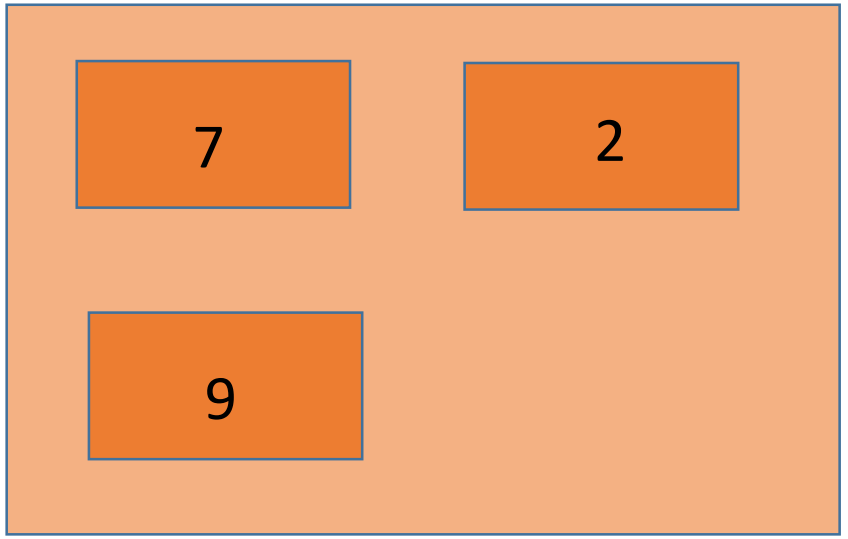
```
Bienvenidos al programa que suma dos números enteros
Escriba numero 1
Escriba numero 2
La suma de 7 y  2  es  9
```

Programación Modular

main()

```
def main ():

    print("Bienvenidos al programa que suma dos
números enteros")

    #Pedir números
    [num1, num2] = pedir_numeros()

    #Sumar los números
    suma= sumar(num1,num2)

    #Mostrar los resultados
    mostrar(num1,num2,suma)

    print("Chao chao")

main()
```

Entorno de memoria main



Pantalla

Bienvenidos al programa que suma dos números enteros

Escriba numero 1

Escriba numero 2

La suma de 7 y 2 es 9

Chao Chao