

**PENSANDO EL NOMBRE  
PARA MIS VARIABLES**



Crear un algoritmo que permita:

- Al usuario ingresar los nombres de los estudiantes de la asignatura de Programación y las notas de cada uno obtenida en la evaluación1
- almacenarlos en un diccionario llamado ESTUDIANTES.
- **NO PUEDEN SER INGRESADOS NOMBRES CONTENIDOS EN LA TUPLA VETADOS ('Jennifer', 'Luis', 'Rady','Jimmy')**

{'Matias':33, 'Magdiel':44, 'Benjamin':55, 'Oriana':66, 'Randall':20 }

Des pues de finalizar el ingreso de datos a la lista debe mostrar un menú con las siguientes opciones:

- **BUSCAR:** Permite al usuario ingresar un nombre y buscarlo , indicando si existe o no. .
- **MOSTRAR TODO:** Permite Mostrar Todo el contenido .
- **ELIMINAR:** Pedir al usuario el ingreso de un nombre y eliminarlo
- **SALIR:** Única forma de salir de la ejecución del menú

# **FUNCIONES**

# FUNCIONES

Una función es un bloque de código con un nombre asociado, que recibe cero o más argumentos como entrada, sigue una secuencia de sentencias, la cuales ejecuta una operación deseada y devuelve un valor y/o realiza una tarea, este bloque puede ser llamados cuando se necesite.

La palabra reservada `def` se usa para definir funciones. Debe seguirle el nombre de la función y la lista de parámetros formales entre paréntesis. Las sentencias que forman el cuerpo de la función empiezan en la línea siguiente, y deben estar indentado.



Las funciones pueden ser declaradas en el mismo algoritmo:

The image shows a screenshot of a Python script editor and a Python 3.6.6 Shell. The script editor window, titled 'ajem.py', contains the following code:

```
def imprimehola():  
    print ("HOLA")  
  
imprimehola()
```

Annotations with blue arrows point to specific parts of the code:

- A blue arrow points from the text 'Función' to the function definition line: `def imprimehola():`.
- A blue arrow points from the text 'Contenido y llamada principal' to the function call line: `imprimehola()`.

The Python 3.6.6 Shell window shows the execution of the script. It displays the output 'HOLA' and the prompt 'HOLA'.

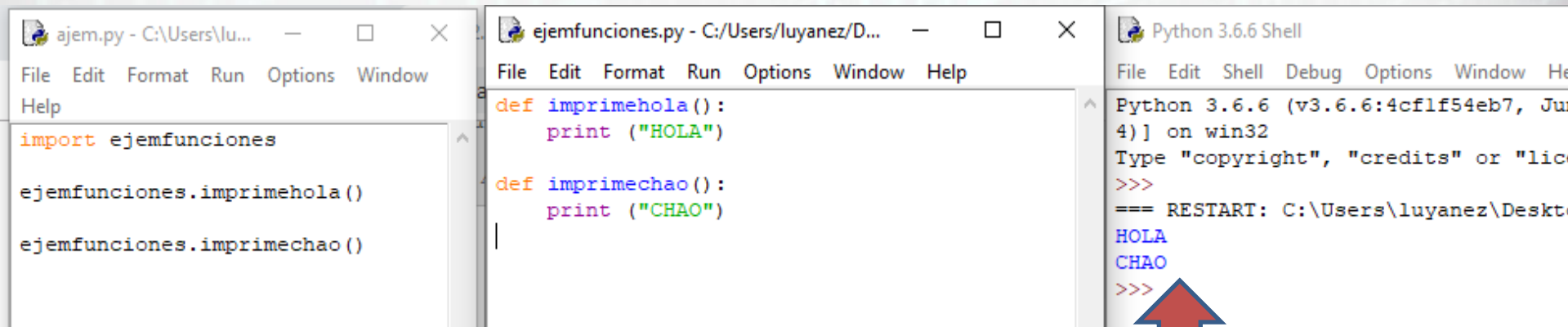
Annotations with red arrows point to the overall context:

- A red arrow points from the text 'Algoritmo Principal' to the entire script editor window.
- A red arrow points from the text 'Ejecución' to the Python 3.6.6 Shell window.

En el ejemplo se crea una función llamada **imprimehola()**, esta función no necesita parámetros o valores externos para obtener algún resultado, por este motivo entre paréntesis no se identifica ningún valor.

Al llamar a la función utilizando la línea **imprimehola()** en cualquier parte de un algoritmo realizara lo que se describe en ella. En el ejemplo imprimirá en pantalla la palabra **HOLA**

Las funciones pueden ser declaradas en un archivo distinto al que contiene el algoritmo principal:



```
ajem.py - C:\Users\lu...
File Edit Format Run Options Window Help
import ejemfunciones

ejemfunciones.imprimehola()

ejemfunciones.imprimechao()

ejemfunciones.py - C:/Users/luyanez/D...
File Edit Format Run Options Window Help
def imprimehola():
    print ("HOLA")

def imprimechao():
    print ("CHAO")

Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
Python 3.6.6 (v3.6.6:4cflf54eb7, Jun 4) on win32
Type "copyright", "credits" or "license()" for more
>>>
=== RESTART: C:\Users\luyanez\Desktop>
HOLA
CHAO
>>>
```



Algoritmo  
Principal en archivo  
**ejem.py**

Para ejecutar las  
funciones debe  
anteponer el nombre  
del archivo importado  
(que contiene las  
funciones)  
**ejemfunciones.imprimehola()**



Archivo  
**ejemfunciones.py**  
Que contiene las  
funciones definidas.  
En este ejemplo  
contiene 2 funciones:  
**imprimehola()** e  
**imprimechao()**

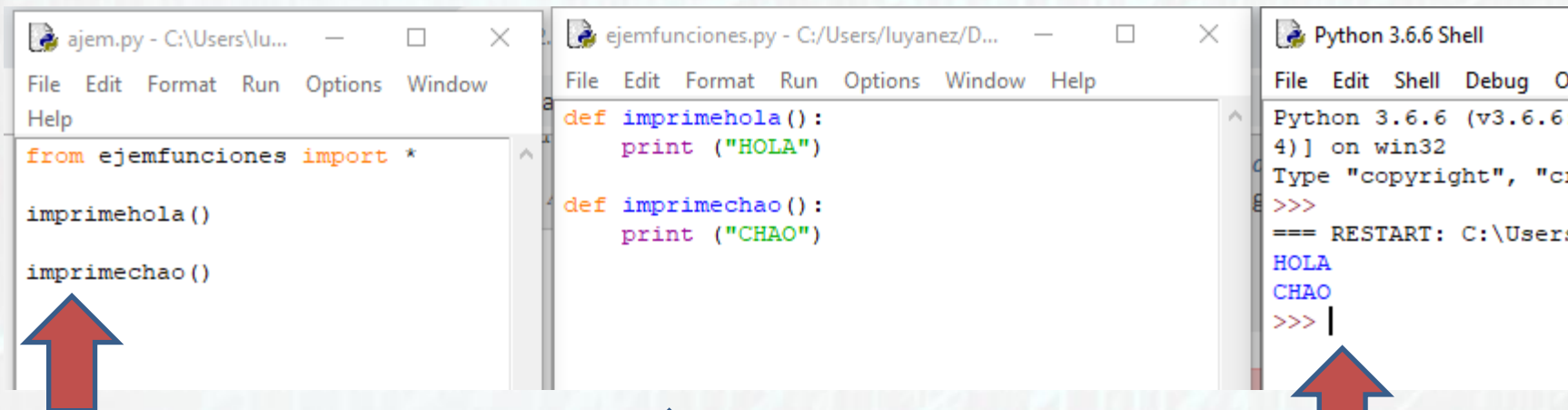


Ejecución

El archivo de funciones también puede ser declarado en el archivo principal utilizando:

```
from ejemfunciones import *
```

Donde \* indica que se importarán todas las funciones contenidas en el archivo **ejemfunciones**.



```
ajem.py - C:\Users\lu...
File Edit Format Run Options Window Help
from ejemfunciones import *
imprimehola()
imprimechao()

ejemfunciones.py - C:/Users/luyanez/D...
File Edit Format Run Options Window Help
def imprimehola():
    print ("HOLA")
def imprimechao():
    print ("CHAO")

Python 3.6.6 Shell
File Edit Shell Debug O
Python 3.6.6 (v3.6.6
4)] on win32
Type "copyright", "c
>>>
=== RESTART: C:\User:
HOLA
CHAO
>>> |
```

Algoritmo  
Principal en archivo  
**ejem.py**

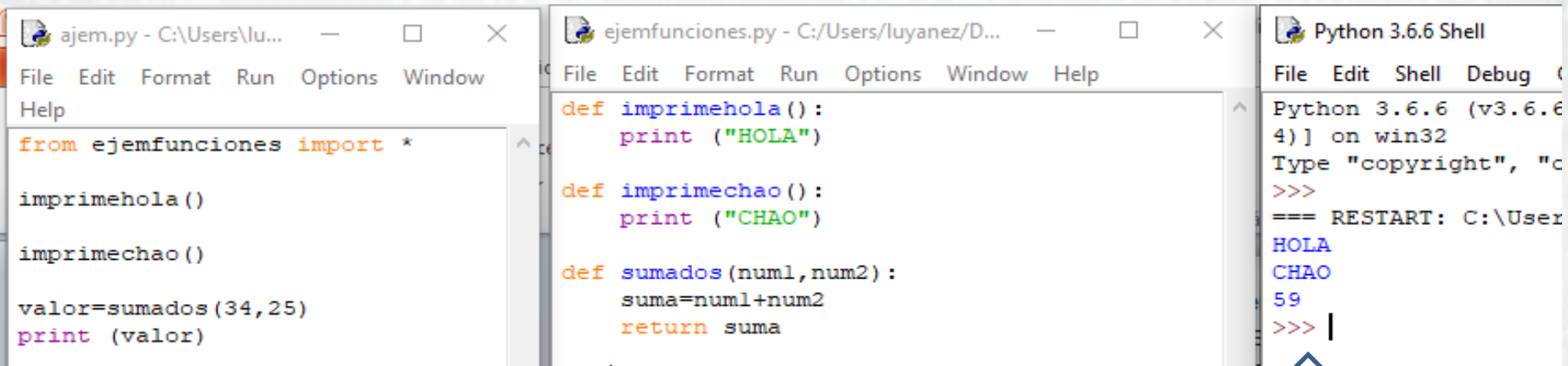
Para ejecutar las funciones  
debe anteponer el nombre  
del archivo importado (que  
contiene las funciones)

Archivo  
**ejemfunciones.py**  
Que contiene las  
funciones definidas.  
En este ejemplo  
contiene 2 funciones:  
**imprimehola()** e  
**imprimechao()**

Ejecución

# Programación

Desde el archivo principal podemos enviar valores  
A la función, los cuales serán recibidos y procesados para obtener algún resultado.  
Estos valores son los parámetros.



```
from ejemfunciones import *

imprimehola()

imprimechao()

valor=sumados(34,25)
print (valor)
```

```
def imprimehola():
    print ("HOLA")

def imprimechao():
    print ("CHAO")

def sumados(num1,num2):
    suma=num1+num2
    return suma
```

```
Python 3.6.6 Shell
Python 3.6.6 (v3.6.6
4)] on win32
Type "copyright", "c
>>>
=== RESTART: C:\User
HOLA
CHAO
59
>>> |
```

Algoritmo  
Principal en archivo  
**ejem.py**

Archivo **ejemfunciones.py**

Ejecución

La función **sumados** contiene la palabra reservada **return** al final del proceso. Esto permite devolver un valor obtenido desde la función al archivo principal donde fue llamada, reemplazando la llamada por el valor devuelto con el **return**.

En el ejemplo en la variable **valor** queda almacenado el resultado de la llamada a la función.

La línea **valor=sumados(34,25)** recibe el valor retornado de la función que suma ambos valores por lo que transforma la línea a :

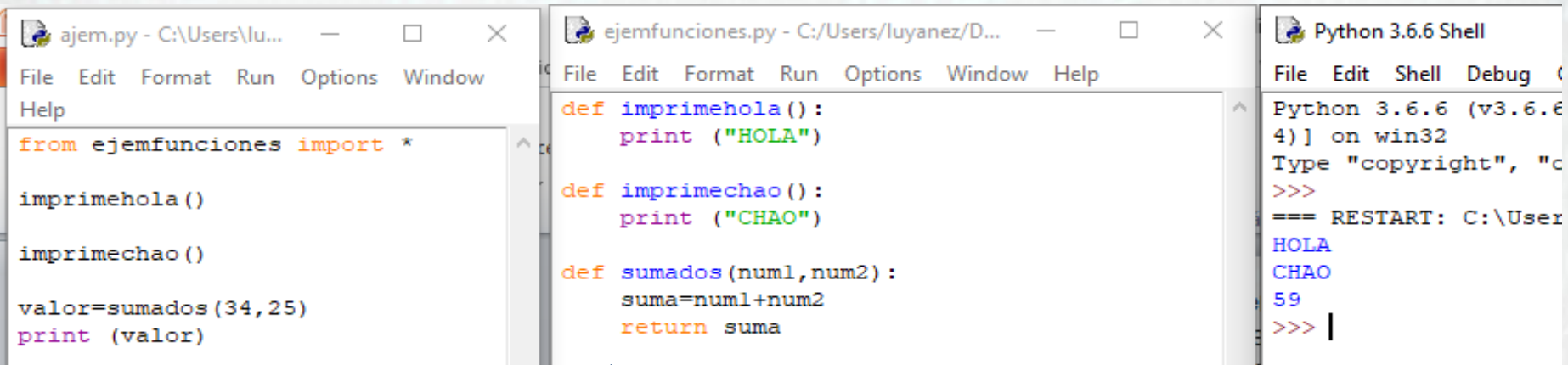
**Valor = 59.**

59 es el resultado devuelto por la función al sumar los valores enviados 34 y 25.



# Programación

Desde el archivo principal podemos enviar valores  
A la función, los cuales serán recibidos y procesados para obtener algún resultado.  
Estos valores son los parámetros.



The image shows three overlapping windows from a Python IDE. The leftmost window, titled 'ejem.py', contains the following code: 

```
from ejemfunciones import *  
  
imprimehola()  
imprimechao()  
  
valor=sumados(34,25)  
print(valor)
```

 The middle window, titled 'ejemfunciones.py', contains the function definitions: 

```
def imprimehola():  
    print("HOLA")  
  
def imprimechao():  
    print("CHAO")  
  
def sumados(num1,num2):  
    suma=num1+num2  
    return suma
```

 The rightmost window, titled 'Python 3.6.6 Shell', shows the output of the execution: 

```
Python 3.6.6 (v3.6.6  
4)] on win32  
Type "copyright", "c  
>>>  
=== RESTART: C:\User  
HOLA  
CHAO  
59  
>>> |
```

↑  
Algoritmo  
Principal en archivo  
**ejem.py**

↑ Archivo **ejemfunciones.py**

↑ Ejecución

El archivo contiene declarada la función **sumados(num1,num2)**  
Donde **num1** y **num2** son las variables que reciben los valores enviados desde el archivo principal, deben ser coherentes los valores enviados desde el archivo principal y los recibidos en la función.

Dentro de la función: num1 toma el valor 34 y num2 el valor 25.  
Ambos valores se suman guardando su resultado en la variable **suma**.  
El contenido de la variable suma es devuelto al archivo principal al realizar:  
**return suma**

### import operaciones

```
varnumero = int(input("ingrese el numero : "))  
print (operaciones.par(varnumero))  
print ("Factorial es: ",operaciones.factorial(varnumero))  
print ("Suma 5: ",operaciones.suma(varnumero,5))
```

Color celeste  
identifica  
archivo  
principal.

### operaciones.py

```
def suma(n1, n2):  
    suma=n1+n2  
    return suma
```

```
def factorial(elemento):  
    fact=1  
    i=1  
    while i<=elemento:  
        fact=i*fact  
        i=i+1  
    return fact
```

```
def par(elemento):  
    resto = elemento % 2  
    if resto > 0:  
        respuesta = "el numero es IMPAR"  
    else:  
        respuesta = "el numero es PAR"  
    return respuesta
```

Color rojo  
identifica  
funciones  
contenidas en  
archivo  
operaciones.py

```
from operaciones import *
```

# Programación

```
varnumero = int(input("ingrese el numero : "))  
print (par(varnumero))  
print ("Factorial es: ",factorial(varnumero))  
print ("Suma 5: ",suma(varnumero,5))
```

Color celeste  
identifica  
archivo  
principal.

## operaciones.py

```
def suma(n1, n2):  
    suma=n1+n2  
    return suma
```

```
def factorial(elemento):  
    fact=1  
    i=1  
    while i<=elemento:  
        fact=i*fact  
        i=i+1  
    return fact
```

```
def par(elemento):  
    resto = elemento % 2  
    if resto > 0:  
        respuesta = "el numero es IMPAR"  
    else:  
        respuesta = "el numero es PAR"  
    return respuesta
```

Color rojo  
identifica  
funciones  
contenidas en  
archivo  
operaciones.py

# Programación

```
def calcula_media(*args):  
    total = 0  
    for i in args:  
        total += i  
    resultado = total / len(args)  
    return resultado
```

Color rojo identifica funciones  
contenidas en archivo  
operaciones.py

```
a, b, c = 3, 5, 10  
media = calcula_media(a, b, c)  
print(f"La media de {a}, {b} y {c} es: {media}")  
print("Programa terminado")
```

Color celeste  
identifica  
archivo  
principal.

Ejemplo donde se envían desde programa principal 3 valores en la línea:

```
media = calcula_media(a, b, c)
```

La función se llama **calcula\_media** y los parámetros son **a, b y c (3, 4 y 10)**

El resultado retornado de la función quedará almacenado en la variable **media**

La función puede recibir los parámetros, enviados desde el programa principal, utilizando la referencia **\*args** que permite que los valores recibidos sean almacenados en una **lista llamada args**.

Dentro de la función se utilizan los datos manipulándolos desde la lista **args** ubicados en la posición de la lista según el orden el que fueron enviados.



```
def sumar(numbers):  
    result = 0  
    for number in numbers:  
        result += number  
    print result
```

Color rojo identifica  
funciones contenidas en  
archivo operaciones.py

```
sumar([4,5])
```

Color celeste identifica  
archivo principal.

Ejemplo donde se envía, desde el programa principal, una lista como parámetro