

Tecnológico de Costa Rica

Funciones en Python

Ing. Laura Coto Sarmiento. MSc.



Agenda



Definición de Funciones
F. Predefinidas
Paso de parámetros
Valores por omisión en los
argumentos
F. fructíferas

Funciones

- Secuencia de instrucciones que resuelven un problema.
- Código que será **reutilizado**: Se define una vez y se llama la cantidad de veces que se requiera(invocararlo).
- **Divide y vencerás**
- **Programación en equipo**

Funciones predeterminadas

Ya están definidas en el ambiente de programación

| Función | Retorna |
|---------|-----------------------------------|
| int() | int(a) a convertido en entero |
| str() | str(a) a convertido en string |
| float() | float(a) a convertido en flotante |
| print() | Devuelve a pantalla lo deseado |
| input() | Captura un dato y devuelve |
| upper() | ??? |
| lower() | ??? |

Ejemplos

Las siguientes son ejemplos de funciones pre-definidas en Python:

```
>>> min(52, 36)
```

```
36
```

```
>>> max(0, -5, 89, -4)
```

```
89
```

```
>>> round(78.456, 2)
```

```
78.46
```

```
>>> len("Hola mundo!")
```

```
11
```

```
>>> int("250")
```

```
250
```

```
>>> str(250)
```

```
'250'
```

Aprendamos un poco...



Funciones en python

1. ¿Cómo se declaran?

La palabra "def" introduce el nombre de la función

Lista de parámetros son opcionales pero los paréntesis NO

```
def nombre_de_la_función (Parámetros):  
    INSTRUCCIÓN_1  
    INSTRUCCIÓN_2  
    INSTRUCCIÓN_3
```

El código de la función DEBE de estar indentado con respecto al def

Los dos puntos (:) siguen el paréntesis cerrado e indica el comienzo del conjunto de instrucciones de la función

Ver más ta... Compartir

Uso de funciones que están en otro archivo...

- Un **módulo o librería** es una colección de funciones relacionadas, que ocupamos llamarlas para usarlas.
- La podemos llamar indicando “**import**”

```
>>> import math
>>> math.sqrt(9)
3.0
>>> math.pow(4,4)
256.0
>>> x=5
>>> math.sqrt(x*5)
5.0
>>>
```

```
#importación de librerías
from Funciones import *
```

| Función | Retorna |
|---------|---|
| sqrt() | int(81) calcula la raíz cuadrada de un número |
| pow() | pow(a,b) a elevado a la potencia de b |

Considere necesario...

- 1. Los archivos deben estar juntos en la misma carpeta.
- 2. Recuerda la sencibilidad de Python de mayúsculas y minúsculas, el nombre del archivo llamado a importar debe ser exactamente el mismo.
- 3. Puedes usar 2 alternativas para importar un archivo:
 - a) `import math`
 - o
 - b) `from funciones import *`

PYTHON: BIBLIOTECAS (LIBRARY)



¿Qué son las librerías Python?

Las librerías Python son amplias, con gran cantidad de producciones en contenidos. Consta de módulos que permiten el acceso de funcionalidades del sistema como entrada y salida de archivos, soluciones estandarizadas a problemas de programación, etc.

Los 7 mejores librerías Python



Algunos ejemplos son: histogramas, diagramas de barras, espectros de potencia, series temporales, diagramas de errores, etc.



seaborn

Seaborn es una librería para Python que permite generar fácilmente elegantes gráficos.



Puedes crear gráficos elegantes y versátiles, gracias a sus desarrolladores de buen rendimiento.



NumPy es una librería para realizar cálculo numérico en python. La usaremos principalmente porque nos permite crear y modificar matrices, y hacer operaciones sobre ellas con facilidad.



SciPy Te permite crear rutinas numéricas con estructura de datos. Incorpora las siguientes funciones: optimización, integración numérica, álgebra lineal, estadística, transformadas de Fourier, etc.



pandas Pandas es una librería de Python especializada en el manejo y análisis de estructuras de datos. Pandas proporciona herramientas que permiten: leer y escribir datos en diferentes formatos: CSV, Microsoft Excel, bases SQL y formato HDF5.



Numba es un compilador para Python, el nombre del módulo diseñado especialmente para acelerar tus funciones numéricas, generando código máquina optimizado a partir de código Python



Vamos paso a paso

Componentes de la función

Una función tiene 4 componentes importantes:

- El **nombre** de la función, verbo en infinitivo y complemento, que refleje la única acción que obedece a una única responsabilidad;
- **los parámetros**, que son los valores que recibe la función como entrada, puede no recibir nada, uno o varios;
- el **código de la función o estatutos**, secuencia de instrucciones que resuelve un problema; y
- el **resultado** (o **valor de retorno**), que es el valor final que entrega la función.

• Estructura....


Ejemplo....

```
def nombreFuncion (listaParametros):  
    ...estatutos
```

```
def obtenerCuadrado (px) :  
    resultado= px * px  
    return resultado
```

Creación de funciones fructíferas

- Usted crea sus propias funciones con:
def nombreFuncion (listaParametros):
...estatutos

- Ejemplo: `def imprimirMensaje (pmensaje) :`
 `return pmensaje`  Definición de la función


- Pero no sirve de nada, si no se llama...

```
>>> imprimirMensaje("Hola Mundo!!!")
```

 Llamada 1

```
'Hola Mundo!!!'
```

```
>>> imprimirMensaje("Si se puede...")
```

 Llamada 2

```
'Si se puede...'
```

Estatuto Return

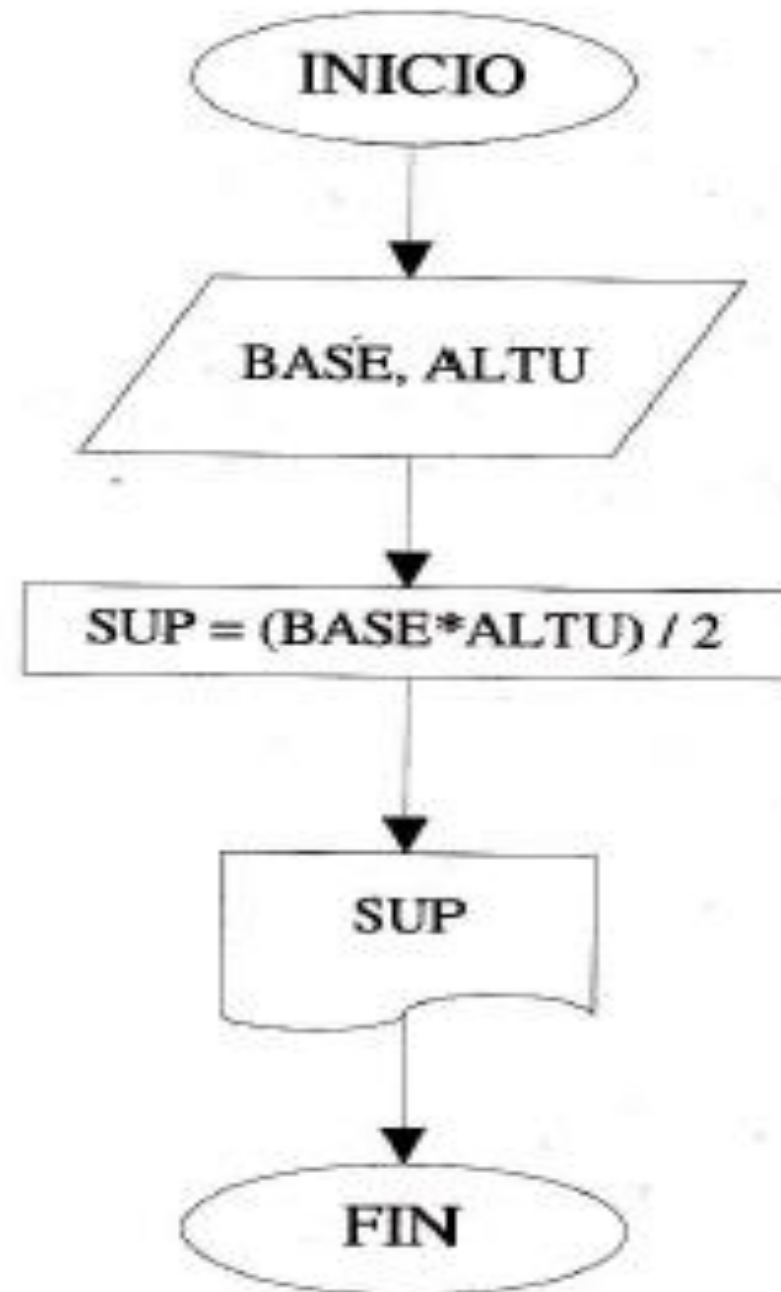
- La definición de una función de forma correcta debe tener al menos un “**return**”
- El “**return**” puede devolver:
 - El valor de un variable
 - Un valor booleano directo
 - O simplemente el return solo

```
#Función que retorna el cuadrado de un número
def obtenerCuadrado(px):
    resultado = px * px
    return resultado
```

```
#Programa Principal
a = int(input("Introduzca un número nuevo: "))
b = obtenerCuadrado(a)
print("El cuadrado de "+str(a)+" es "+str(b))
```

```
Introduzca un número nuevo: 2
El cuadrado de 2 es 4
```

*Cree la función
que resuelve:*



Código sin función:

File Edit Format Run Options Window Help

```
#Elaborado por: Laura Coto Sarmiento
#Fecha de creación: X de X del 20XX, 7:30am
#Ultima modificación: X de X del 20XX, 8:30am
#Versión: X.Y.Z
```

```
#Programa principal
```

```
base=int(input("Indique el valor de la base: "))
altura=int(input("Indique el valor de la altura: "))
area=(base*altura)/2
print("El resultado del área es: "+str(area))
```

Código con función:

File Edit Format Run Options Window Help

```
#Elaborado por: Laura Coto Sarmiento
#Fecha de creación: X de X del 20XX, 7:30am
#Ultima modificación: X de X del 20XX, 8:30am
#Versión: X.Y.Z
```

```
#Definición de funciones
```

```
def calcularArea(pbase, paltura):
    area=(pbase*paltura)/2
    return area
```

```
#Programa principal
```

```
base=int(input("Indique el valor de la base: "))
altura=int(input("Indique el valor de la altura: "))
#llamada de la función
print("El resultado del área es: "+str(calcularArea(base, altura)))
```

Comprenda....

File Edit Format Run Options Window Help

```
#Elaborado por: Laura Coto Sarmiento
#Fecha de creación: X de X del 20XX, 7:30am
#Ultima modificación: X de X del 20XX, 8:30am
#Versión: X.Y.Z
```

```
#Definición de funciones
```

```
def calcularArea(pbase, paltura):
    area=(pbase*paltura)/2
    return area
```



Definición de la función

```
#Programa principal
```

```
base=int(input("Indique el valor de la base: "))
```

```
altura=int(input("Indique el valor de la altura: "))
```

```
#llamada de la función
```

```
print("El resultado del área es: "+str(calcularArea(base, altura)))
```



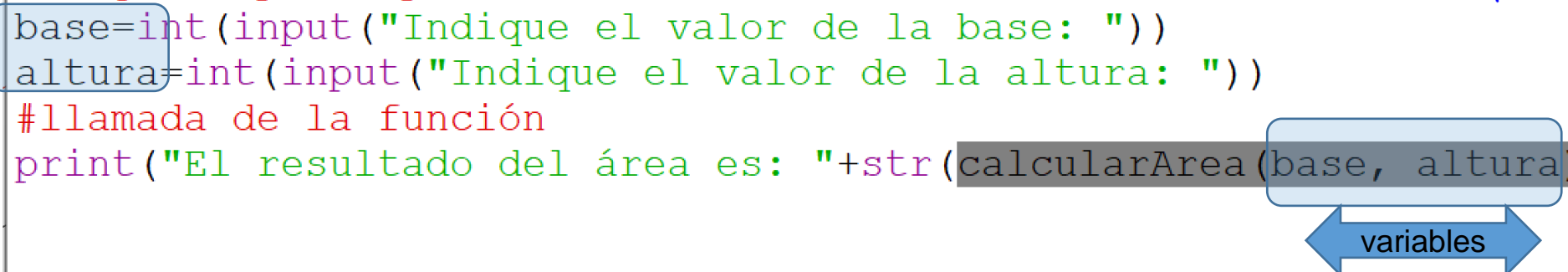
Llamada de la función

Comprenda....

```
File Edit Format Run Options Window Help
#Elaborado por: Laura Coto Sarmiento
#Fecha de creación: X de X del 20XX, 7:30am
#Ultima modificación: X de X del 20XX, 8:30am
#Versión: X.Y.Z

#Definición de funciones
def calcularArea(pbase, paltura):
    area=(pbase*paltura)/2
    return area

#Programa principal
base=int(input("Indique el valor de la base: "))
altura=int(input("Indique el valor de la altura: "))
#llamada de la función
print("El resultado del área es: "+str(calcularArea(base, altura)))
```



Comprenda....

```
File Edit Format Run Options Window Help
#Elaborado por: Laura Coto Sarmiento
#Fecha de creación: X de X del 20XX, 7:30am
#Ultima modificación: X de X del 20XX, 8:30am
#Versión: X.Y.Z

#Definición de funciones          Parámetros de la función, los llamamos con "p" para diferenciarlos
def calcularArea(pbase, paltura):
    area=(pbase*paltura)/2
    return area

#Programa principal
base=int(input("Indique el valor de la base: "))
altura=int(input("Indique el valor de la altura: "))
#llamada de la función
print("El resultado del área es: "+str(calcularArea(base, altura)))
```

Comprenda....

```
File Edit Format Run Options Window Help
#Elaborado por: Laura Coto Sarmiento
#Fecha de creación: X de X del 20XX, 7:30am
#Ultima modificación: X de X del 20XX, 8:30am
#Versión: X.Y.Z

#Definición de funciones
def calcularArea(pbase, paltura):
    area=(pbase*paltura)/2
    return area

#Programa principal
base=int(input("Indique el valor de la base: "))
altura=int(input("Indique el valor de la altura: "))
#llamada de la función
print("El resultado del área es: "+str(calcularArea(base, altura)))
```

En tiempo de ejecución:

- el contenido de la variable “base” se asigna al parámetro “pbase”
- y
- El contenido de la variable “altura” se asigna al parámetro “paltura”

Note: el orden importa...

1

Estatuto Return

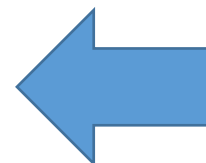
- Detiene (salida abrupta) la función en el momento en que se ejecuta “**return**”.
- Entonces... analice el siguiente ejemplo:

```
#Elaborado por: Laura Coto Sarmiento, versión 3.5.1  
#Fecha de creación: 30/08/2019 1:00pm  
#Fecha de última modificación: 30/08/2019 2:00pm
```

```
#importaciones de las librerías
```

```
#Defición de funciones
```

```
def repetirTexto(ptexto):  
    i=0  
    while i<=10:  
        print(ptexto)  
        return True  
        i+=1
```



¿Qué está mal en esta definición de función?

```
#programa principal  
texto=input("Indique un texto: ")  
repetirTexto(texto)
```

Estatuto Return

- Detiene (salida abrupta) la función en el momento en que se ejecuta “**return**”

```
#Elaborado por: Laura Coto Sarmiento, versión 3.5.1
#Fecha de creación: 30/08/2019 1:00pm
#Fecha de última modificación: 30/08/2019 2:00pm
```

```
#importaciones de las librerías
```

```
#Defición de funciones
```

```
def repetirTexto(ptexto):
    i=0
    while i<=10:
        print(ptexto)
        return True
        i+=1
```

```
#programa principal
texto=input("Indique un texto: ")
repetirTexto(texto)
```

¿Qué está mal en esta definición de función?

Respuesta/

**Detiene (salida abrupta) la función en el momento en que se ejecuta “return”
Nunca se hará el incremento.**

Varios “return”, sólo 1 se ejecuta...

- A pesar de tener varios “return”, analice que sólo se ejecuta uno...

```
El valor absoluto de: 5 es 5  
El valor absoluto de: -10 es 10
```

return.py - /home/laura/ejemplos/funciones/return.py

File Edit Format Run Options Windows Help

```
#Función que recibe un número y retorna el valor absoluto  
def valorAbs(x):  
    if x<0:  
        return -x  
    else:  
        return x  
  
#Programa principal  
a=5  
res=valorAbs(a)  
print('El valor absoluto de: '+str(a)+' es ', res)  
b=-10  
print('El valor absoluto de: '+str(b)+' es ', valorAbs(b))
```

Son
equivalentes

Documentación de una Función

- Cada “def” debe tener su propia documentación, dentro de la función.
- Cada función debe documentarse con:

Funcionalidad:...

Entradas: ...

Salidas: ...

- Puede usar # o

'''

Funcionalidad: Calcula la potencia de un número

Entrada: Número(int) y base(int)

Salida: Potencia del número(int)

'''

Ejemplo de Documentación de una Función

File Edit Format Run Options Window Help

```
#Elaborado por: Laura Coto Sarmiento
#Fecha de creación: X de X del 20XX, 7:30am
#Ultima modificación: X de X del 20XX, 8:30am
#Versión: X.Y.Z

#Definición de funciones
def calcularArea(pbase, paltura):
    """
    Funcionalidad: Calcular el área del triángulo
    Entradas:
    -base(int): medida de la base del triángulo,
    -altura(int): medida de la altura del triángulo
    Salidas:
    -area(float): Superficie del triángulo
    """
    area=(pbase*paltura)/2
    return area

#Programa principal
base=int(input("Indique el valor de la base: "))
altura=int(input("Indique el valor de la altura: "))
print("El resultado del área es: "+str(calcularArea(base, altura)))
```


Ello permite preguntarle al Shell por una función

```
funcArea.py - C:\Users\usuario\Documents\Ingeniería en Computación\Elementos de Computación CA-2125\Ejemplos de ...
File Edit Format Run Options Window Help

#Elaborado por: Laura Coto Sarmiento
#Fecha de creación: X de X del 20XX, 7:30am
#Ultima modificación: X de X del 20XX, 8:30am
#Versión: X.Y.Z

#Definición de funciones
def calcularArea(pbase, paltura):
    """
    Funcionalidad: Calcular el área del triángulo
    Entradas:
    -base(int): medida de la base del triángulo
    -altura(int): medida de la altura del triángulo
    Salidas:
    -area(float): Superficie del triángulo
    """
    area = (pbase * paltura) / 2
    return area

#Programa principal
base = int(input("Indique el valor de la base: "))
altura = int(input("Indique el valor de la altura: "))
print("El resultado del área es: ", calcularArea(base, altura))
```

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help

Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:11) on win32
Type "copyright", "credits" or "license()" for more
>>>
RESTART: C:\Users\usuario\Documents\Ingeniería en Computación CA-2125\Ejemplos de Código\Funciones\funcArea.py
Indique el valor de la base: 10
Indique el valor de la altura: 2
El resultado del área es: 10.0
>>> help(calcularArea)
Help on function calcularArea in module __main__:

calcularArea(pbase, paltura)
    Funcionalidad: Calcular el área del triángulo
    Entradas:
    -base(int): medida de la base del triángulo,
    -altura(int): medida de la altura del triángulo
    Salidas:
    -area(float): Superficie del triángulo

>>>
```

Preguntas por la función y la ayuda te dice:

- Lo que ocupas y el código te dice:
- qué te devuelve...

Para que no te pase...

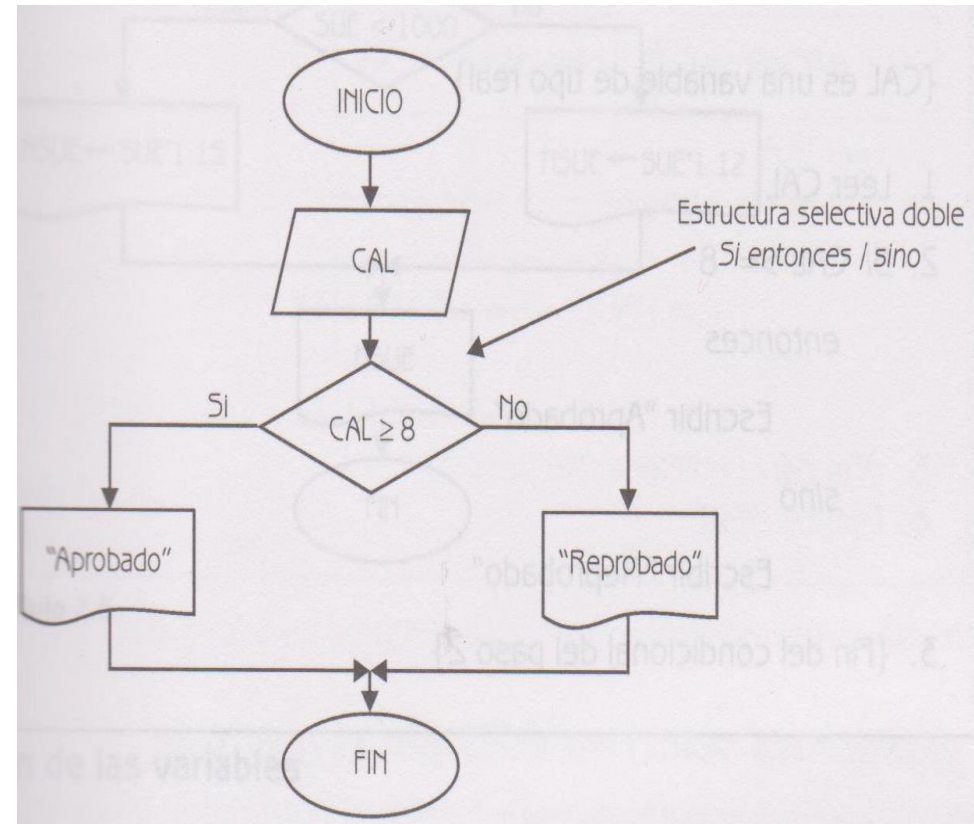
**CUANDO ESCRIBÍ ESTE CÓDIGO,
SÓLO DIOS Y YO SABÍAMOS
CÓMO Y PARA QUÉ LO HICE**



AHORA, SÓLO DIOS LO SABE

Práctica

- Cree la función completa que solicita al usuario el dato de entrada y devuelva la salida indicada.
- Documente la función:

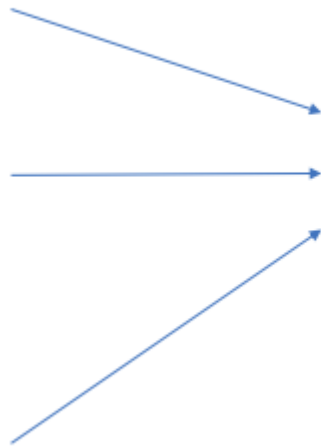


Debes pasar todos los DF a funciones

- Te recomendamos estratégicamente hacer al menos...
- DF 2.25 de la página 94
- DF 2.28 de la página 102
- DF 3.11 de la página 127

Programa 1

```
inst1  
inst2  
inst3  
inst4  
inst1  
inst2  
inst3  
inst5  
inst6  
inst7  
inst1  
inst2  
inst3
```



```
def mi_funcion():  
    inst1  
    inst2  
    inst3
```

Programa 2

```
def mi_funcion():  
    inst1  
    inst2  
    inst3  
  
mi_funcion()  
inst4  
mi_funcion()  
inst5  
inst6  
inst7  
mi_funcion()
```



Continuemos aprendiendo



Parámetros opcionales

- Si es opcional, se da un **valor por defecto**:

```
def imprimirMensaje(pmensaje, pcharacter, pannoActual=2019) :  
    return pmensaje+" "+pcharacter+" "+str(pannoActual)
```

Definición de
la función

```
>>> imprimirMensaje("¿Eres Feliz?", "S", 2019)  
'¿Eres Feliz? S 2019'  
>>> imprimirMensaje("¿Eres Inteligente?", "S")  
'¿Eres Inteligente? S 2019'
```

Llamada con
3 parámetros

Llamada con
2 parámetros

Cuando no se pone el tercer parámetro,
la función usa el valor por omisión o
defecto.

Variables locales y parámetros

- Las **variables locales** se crean y existen únicamente dentro de la función, luego de salir de ella no existe la variable.
- Los parámetros se tratan como **variables locales**

```
def sumarValores(pvalor1,pvalor):  
    resultado=pvalor1+pvalor  
    return resultado  
  
def imprimirMensaje(pmensaje):  
    return pmensaje
```

```
>>> numero1=2  
>>> numero2=3  
>>> sumarValores(numero1,numero2)  
5  
>>> imprimirMensaje(resultado)  
Traceback (most recent call last):  
  File "<pyshell#11>", line 1, in <module>  
    imprimirMensaje(resultado)  
NameError: name 'resultado' is not defined
```

• ¿Qué está malo?




Variables locales y parámetros

- Las **variables locales** se crean y existen únicamente dentro de la función, luego de salir de ella no existe la variable.
- Los parámetros se tratan como **variables locales**

```
def sumarValores(pvalor1,pvalor):  
    resultado=pvalor1+pvalor  
    return resultado  
  
def imprimirMensaje(pmensaje):  
    return pmensaje
```

```
>>> numero1=2  
>>> numero2=3  
>>> sumarValores(numero1,numero2)  
5  
>>> imprimirMensaje(resultado)  
Traceback (most recent call last):  
  File "<pyshell#11>", line 1, in <module>  
    imprimirMensaje(resultado)  
NameError: name 'resultado' is not defined
```



- ¿Qué está malo?

Respuesta/

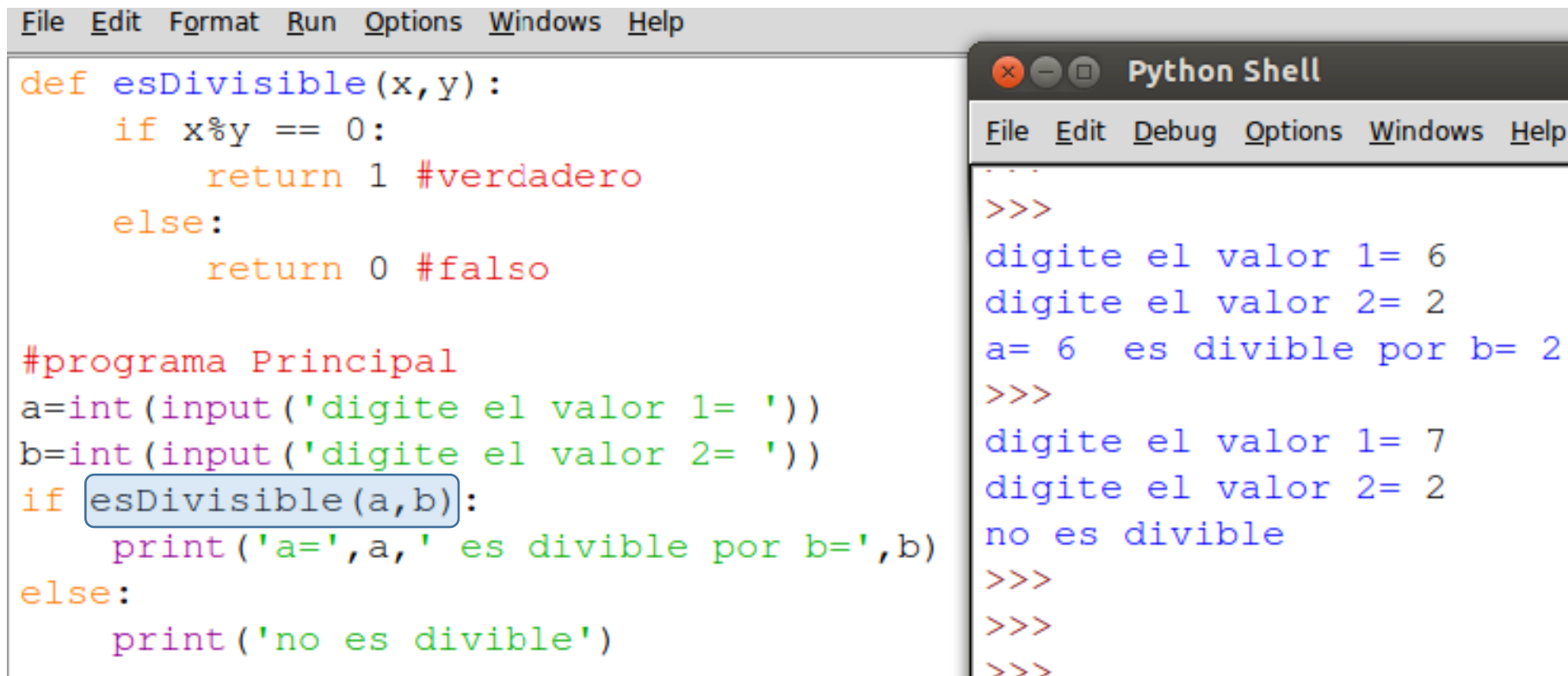
- “resultado” existe únicamente dentro de la función “sumarValores”.
- Y allí se está llamando fuera, por ende, fuera no existe

Debes ser consciente del
alcance de las
variables....



Funciones Booleanas

- Observe la **comparación implícita** del último IF, cuando no hay comparación, se asume se compara con “==True”



The image shows a Python IDE window with a menu bar (File, Edit, Format, Run, Options, Windows, Help) and a code editor. The code defines a function `esDivisible(x, y)` that returns 1 if `x%y == 0` and 0 otherwise. Below the function is a main program that prompts the user for two values, `a` and `b`, and checks if `a` is divisible by `b` using the `esDivisible` function. The `if` statement `if esDivisible(a,b):` is highlighted with a blue box. To the right, a 'Python Shell' window shows the execution of the code. It displays the prompts and user input for the first test case (6 and 2), showing that 6 is divisible by 2. It then shows the prompts for the second test case (7 and 2), showing that 7 is not divisible by 2.

```
def esDivisible(x,y):  
    if x%y == 0:  
        return 1 #verdadero  
    else:  
        return 0 #falso  
  
#programa Principal  
a=int(input('digite el valor 1= '))  
b=int(input('digite el valor 2= '))  
if esDivisible(a,b):  
    print('a=',a, ' es divisible por b=',b)  
else:  
    print('no es divisible')
```

```
>>>  
digite el valor 1= 6  
digite el valor 2= 2  
a= 6  es divisible por b= 2  
>>>  
digite el valor 1= 7  
digite el valor 2= 2  
no es divisible  
>>>  
>>>  
>>>
```

Funciones que ayudan a *validar*...

```
if type(numero) != int:  
    return "Debe indicar un numero entero"
```

```
if not isinstance(numero, int):  
    return 'El número debe ser entero'  
elif numero < 0:  
    return 'El número debe ser mayor a cero'  
else:  
    return sumarDigitosMultiplosAux(numero, digito)
```

```
if type(lista) != list:  
    return "Debe indicar una lista"  
else:  
    return vocalesAux(lista, [], 0)
```

Validación en funciones

```
def esPar(valor):  
    """  
    Función: Indica si un valor es booleano o no.  
    Entrada: valor(int) Recibe un número entero mayor que cero  
    Salida: Devuelve True/False si corresponde a un número binario  
    """  
    if (valor%2)==0:  
        return True  
    else:  
        return False  
  
def esParAux(datos):  
    if isinstance(datos, int):  
        if datos>0:  
            return esPar(datos)  
        else:  
            return "El valor debe ser mayor a cero."  
    else:  
        return "El valor debe ser un número."
```

Ejemplos de: ¿cómo validar?

```
def sumarDigitosMultiplosAux(num, dig):  
    """  
    Funcionalidad: validar que sean un numeros, y que num sea mayor a 0  
    Entradas: num(int), dig(int)  
    Salidas: Posibles mensajes de error o llamada a la funcion principal  
    """  
    if isinstance(num, int) and isinstance(dig, int):  
        if num > 0:  
            return sumarDigitosMultiplos(num, dig, 0)  
        else:  
            return "El numero debe ser mayor que 0"  
    else:  
        return "Los valores deben ser numeros"
```

Comprenda...

Siempre cree una
responsabilidad por función...

```
File Edit Format Run Options Window Help
#Elaborado por: Laura Coto Sarmiento
#Fecha de creación: X de X del 20XX, 7:30am
#Ultima modificación: X de X del 20XX, 8:30am
#Versión: X.Y.Z

#Definición de funciones
def calcularArea(pbase, paltura): #responsabilidad = determinar el área
    """
    Funcionalidad:Calcular el área del triángulo
    Entradas:
    -base(int): medida de la base del triángulo,
    -altura(int): medida de la altura del triángulo
    Salidas:
    -area(float):Superficie del triángulo
    """
    area=(pbase*paltura)/2
    return area

def calcularAreaAux(pbase, paltura): #responsabilidad = validaciones
    if isinstance(pbase,int) and isinstance(paltura,int):
        if (pbase!=0)and(paltura!=0):
            return "El resultado del área es: "+str(calcularArea(pbase, paltura))
        else:
            return "Los valores de base y altura deben ser mayores a cero."
    else:
        return "Los datos de entrada deben ser únicamente enteros."

def mostrarAlUsuario(): #responsabilidad = pedir datos y mostrar salidas al usuario
    base=int(input("Indique el valor de la base: "))
    altura=int(input("Indique el valor de la altura: "))
    print(calcularAreaAux(base, altura))
    return ""

#Programa principal
print(mostrarAlUsuario())
```

Argumentos: Manejo en funciones

Argumentos recibidos por valor: la función recibe una copia del parámetro, y modificaciones a tal parámetro son realizadas en una nueva copia local de la función

Argumentos recibidos por referencia: la función recibe la dirección en memoria del valor original, por lo que si el parámetro es modificado, también la variable original es modificada

Los tipos de datos básicos (números, hileras) y los objetos inmutables (tuplas) son pasados por **valor**, mientras que las variables con tipo de datos mutables (listas) son pasados por **referencia**

Python lo hace por referencia

```
def funcionDemo(num, lista):  
    num += 1  
    # modificación del  
    # valor de num  
    lista[0] = 20  
    # modificación de la  
    # lista  
    return num;  
  
>>> lista = [1, 2, 3]  
>>> num = 1  
>>> funcionDemo(num, lista)  
>>> num # el valor de num no se  
# ha modificado  
1  
>>> lista  
[20, 2, 3] # el valor de lista se  
# ha modificado
```

El paso de la variable de tipo de dato primitivo entero se hace por copia, y el de el tipo de dato lista (mutable), se hace por referencia



SINTAXIS BÁSICA DE PYTHON



Python no usa punto
coma o llaves. Usa saltos
de línea e indentación

Las funciones se definen con def

```
#Ejemplo de código en Python  
import datetime as dt
```

```
def saludo(nombre):  
    """Función de bienvenida al team DSRP"""
```

```
    print( "Hola" , nombre, "!" )  
    print( "Fecha de registro:" , dt.date.today())
```

```
    saludos = ["Welcome" , "Bienvenido" , "Bem-vindo"]  
    for saludo in saludos :  
        print(saludo)
```

```
    saludo("Javier")
```

```
    help (saludo)
```

Comentarios de una
línea con el numeral (#)

Importación de bibliotecas
que necesitamos

No olvidemos los dos
puntos que acompañan
un: if, def, while, for, etc.

Documentación
de la función

Mostramos nuestras
variables en la consola

El bucle for nos indica el
recorrido de una variable
compuesta (lista, tupla, etc.),
también podemos recorrer
un rango de elementos
usando range().

Llamamos a nuestra
función enviando un
parámetro de entrada

Pedimos información
de la función



Functions in Python

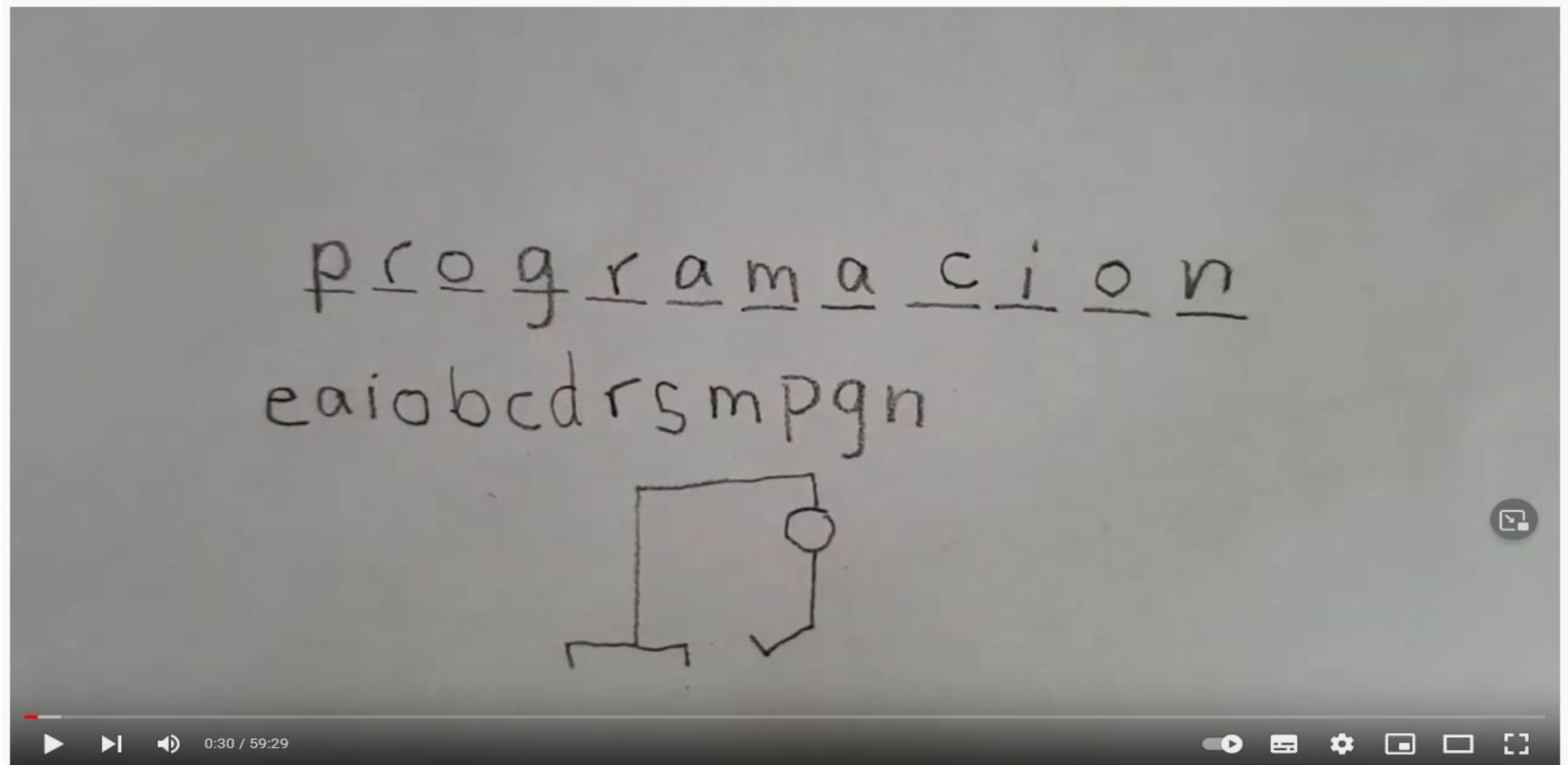
- `abs()`
- `all()`
- `any()`
- `ascii()`
- `bin()`
- `bool()`
- `breakpoint()`
- `bytearray()`
- `bytes()`
- `callable()`
- `chr()`
- `classmethod()`
- `compile()`
- `complex()`
- `delattr()`
- `dict()`
- `dir()`
- `divmod()`
- `enumerate()`
- `eval()`
- `exec()`
- `filter()`
- `float()`
- `format()`
- `frozenset()`
- `getattr()`
- `globals()`
- `hasattr()`
- `hash()`
- `help()`
- `hex()`
- `id()`
- `input()`
- `int()`
- `isinstance()`
- `issubclass()`
- `iter()`
- `len()`
- `list()`
- `locals()`
- `map()`
- `max()`

Te recomiendo veas el siguiente video completo...



<https://www.youtube.com/watch?v=A-iKX8Shge4>

Implementando todo en una complejidad mayor...



INTRO - Programando un juego de Ahorcado en Python

<https://www.youtube.com/watch?v=jtg9NO-GEcU>

Bibliografía

- Solano Soto, J. Introducción a la programación Python. Editorial Tecnológica de Costa Rica.
- Explicaciones:
 - [Funciones propias y definidas](#)
 - [Funciones con parámetros](#)

Bibliografía

- Funciones internas:
<http://pyspanishdoc.sourceforge.net/lib/built-in-funcs.html>
- Presentaciones:
 - Ing. Saúl Calderón. MSc.
 - Ing. Luis Pablo Soto. MEd.