Funciones

Informática I - 2547100

Departamento de Ingeniería Electrónica
y de Telecomunicaciones

Facultad de Ingeniería

2016-2

Problem statement

Reutilizar un código ya hecho en otro programa más grande:

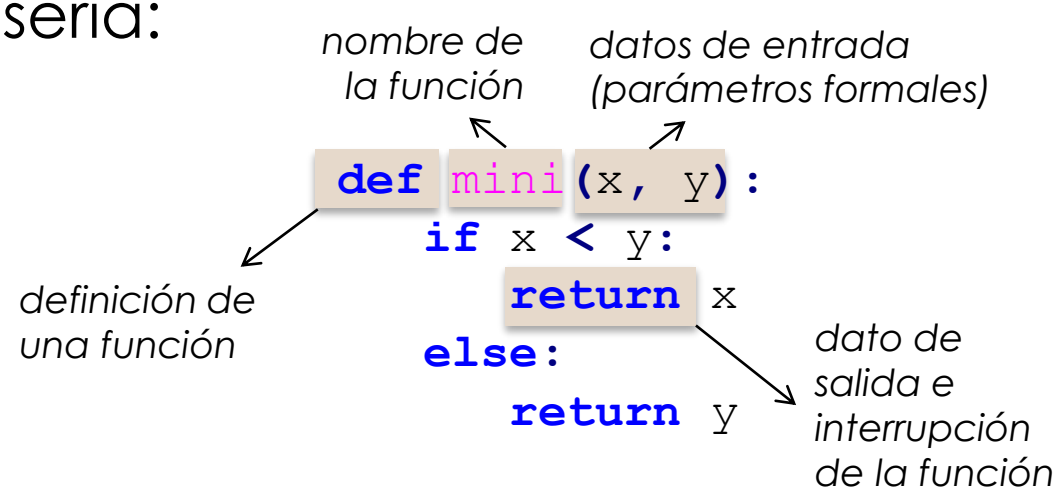
- los nombres de las variables deben ajustarse al nuevo programa
- copiar el mismo código en todos los lugares donde se necesite
- si se quiere modificar el código que se copió, será necesario modificar todas las copias

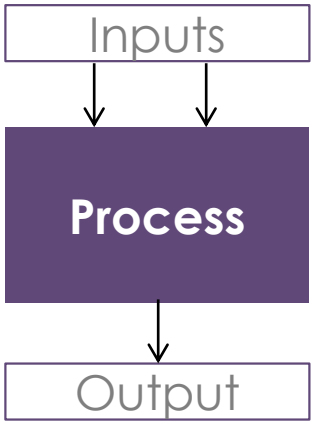
Solución: encapsular el código a reutilizar en un subprograma o **función**

Functions in Python

Una función es un segmento de código que ha sido encapsulado con un nombre para ser fácilmente reutilizado en diferentes partes de un programa o en otros programas.

Una función que calcula el mínimo de dos números sería:





Using functions

```
age1 = int(input('Enter age of first person: '))
weight1 = int(input('Enter weight of first person: '))
age2 = int(input('Enter age of second person: '))
weight2 = int(input('Enter weight of second person: '))
if age1 < age2:
   young = age1
else:
   young = age2
if weight1 < weight2:
   slim = weight1
else:
   slim = weight2
print('The young one is', young, 'years old')
print('and the slim one weights', slim, 'kilos')
```

definición de la función

parámetros formales

invocación o llamado de la función

argumentos

```
def mini(x, y):
   if x<y:
      return x
   else:
      return y
age1 = int(input('Enter age of first person: '))
weight1 = int(input('Enter weight of first person: '))
age2 = int(input('Enter age of second person: '))
weight2 = int(input('Enter weight of second person: '))
young = mini(age1, age2)
slim = mini(weight1, weight2)
print('The young one is', young, 'years old')
print('and the slim one weights', slim, 'kilos')
```

More on function arguments

```
def strIsIn(str1, str2, sen=True):
    if not sen:
        str1 = str1.lower()
        str2 = str2.lower()
    if str1 in str2:
        return 'Yes'
    else:
        return 'No'

s1 = 'Hola'
s2 = 'holas'

print(strIsIn(s1, s2, False)) # imprime Yes

print(strIsIn(s1, s2, True))  # imprime No

print(strIsIn(s2, s1, False))

print(strIsIn(sen=False, str2=s1, str1=s2))

print(strIsIn(str2=s1, str1=s2))
```

parámetros con valores por defecto

argumentos asignados por nombre

argumento omitido por tener valor por defecto

Specification of a function

```
def findRoot(x, power, epsilon=0.01):
    ''' Asume que x y epsilon son int o float,
    que power es int, epsilon > 0 y power >= 1
    Retorna un dato float tal que dato**power
    está dentro de una distancia epsilon de x
    Si no existe tal dato, retorna None'''
    if x<0 and power%2==0:
        return None
    low = 0.0
    high = max(1.0, x)
    ans = (high + low)/2.0
    while abs(ans**power - x) >= epsilon:
        if ans**power < x:
            low = ans
        else:
            high = ans
    ans = (high + low)/2.0
    return ans
```

Especificación *de una función: condiciones de uso de una función.*

- ***Suposiciones****: requisitos de los argumentos*
- ***Garantías****: lo que la función garantiza hacer*

Docstring: *comentario entre comillas triples que resume la especificación de la función*

Las especificaciones de las funciones permiten desarrollar un programa entre varias personas

Scoping

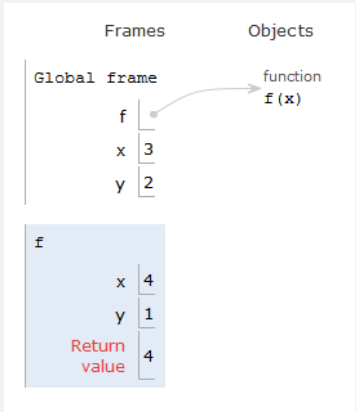
Es el ámbito en el que es válido el nombre de una variable o función.

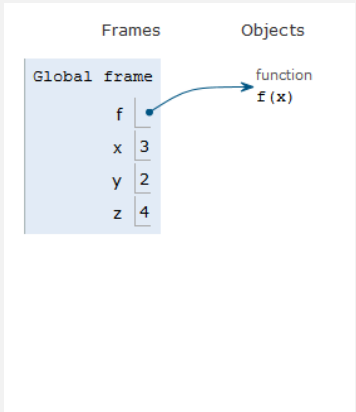
variables locales a f

variables globales

```
def f(x):
    y = 1
    x = x + y
    print('x =', x)
    return x

x = 3
y = 2
z = f(x) # valor de x usado como argumento
print('z =', z)
print('x =', x)
print('y =', y)
```

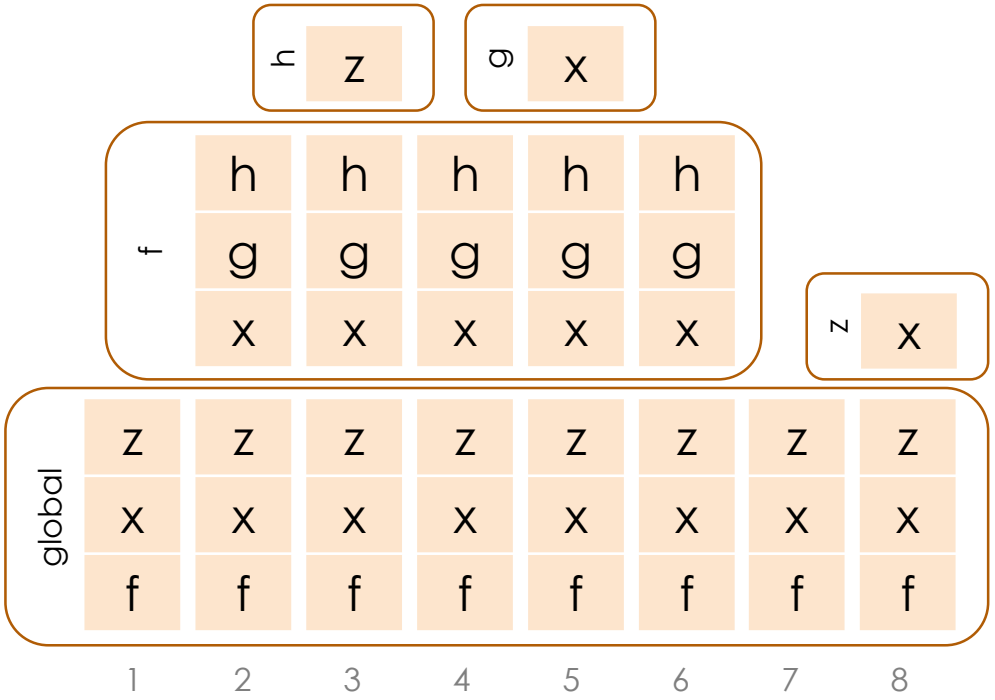




Stack frames

```
def f(x):
    def g():
        x = 'abc'
8, 5 ➡  print('x =', x)
    def h():
        z = x
3 ➡     print('z =', z)
2 ➡ x = x + 1
    print('x =', x)
    h()
4 ➡ g()
    print('x =', x)
6 ➡ return g

x = 3
1 ➡ z = f(x)
7 ➡ print('x =', x)
print('z =', z)
z()
```



Nested functions

```
def compare():
    def mini(x, y):
        if x<y:
            return x
        else:
            return y
    age1 = int(input('Enter age of first person: '))
    weight1 = int(input('Enter weight of first person: '))
    age2 = int(input('Enter age of second person: '))
    weight2 = int(input('Enter weight of second person: '))
    young = mini(age1, age2)
    slim = mini(weight1, weight2)
    print('The young one is', young, 'years old')
    print('and the slim one weights', slim, 'kilos')

while(True):
    #print(mini(7,8))
    wish = input('Do you want to compare people? ')
    if wish == 'no':
        break
    compare()
```

función anidada

función desconocida en este ámbito

Global vs local variables

```
def f1():
    global a
    a = a + 1
    b = b + 1
    h = k + 1
    global c
    c = 7
    d = 23
    print(a, b, c, d)

a=4
b=15
k=15
f1()
print(a, b)
print(c)
print(d)
```

Error: b no ha sido inicializada

variables globales

variables locales

Error: d no existe en este ámbito

¡Las variables globales hacen que los programas sean muy difíciles de entender y depurar!

Modules

main.py

```
from geom import *

r = input('Ingrese el radio: ')

r = float(r)

print('pi: ', pi)

print('área: ', area(r))

print('perímetro: ', perimetro(r))

print('superficie: ', sfEsfera(r))

print('volumen: ', volEsfera(r))
```

geom.py

```
pi = 3.14159

def area(radio):
    return pi*(radio**2)

def perimetro(radio):
    return 2*pi*radio

def sfEsfera(radio):
    return 4*pi*(radio**2)

def volEsfera(radio):
    return (4/3)*pi*(radio**3)
```

módulos

Files

```
file = open('names.txt', 'r')
for line in file:
   print(line)
file.close()

file = open('names.txt', 'w')
file.write('Juan\n')
file.write('Ana\n')
file.close()
file = open('names.txt', 'r')
for line in file:
   print(line[:-1])
file.close()

file = open('names.txt', 'a')
file.write('Clara\n')
file.write('Julia\n')
file.close()
file = open('names.txt', 'r')
for line in file:
   print(line[:-1])
file.close()
```

Imprime:

```
Carlos

Pedro
```

Imprime:

```
Juan
Ana
```

Imprime:

```
Juan
Ana
Clara
Julia
```