El futuro digital es de todos

MinTIC

Entorno de variables

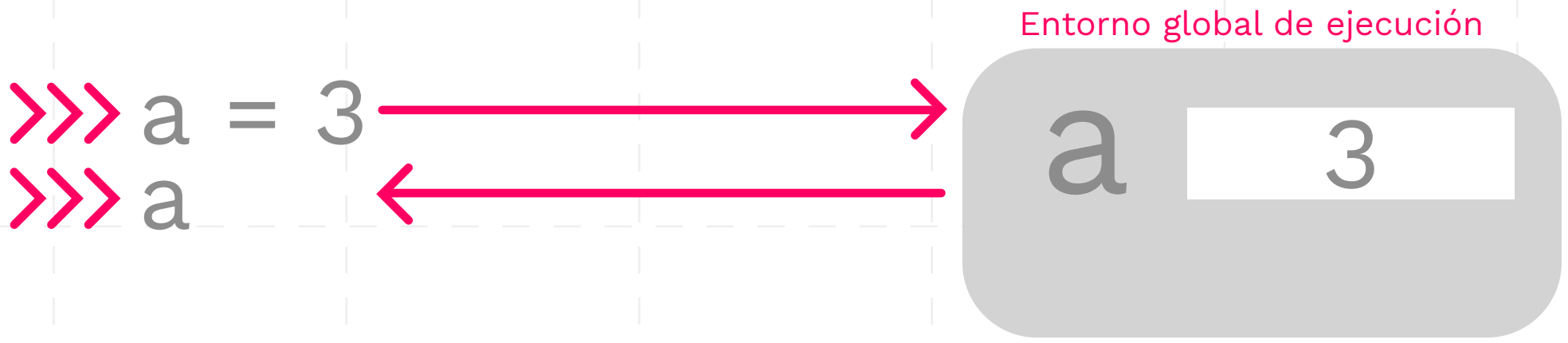
Universidad de Caldas

Hola

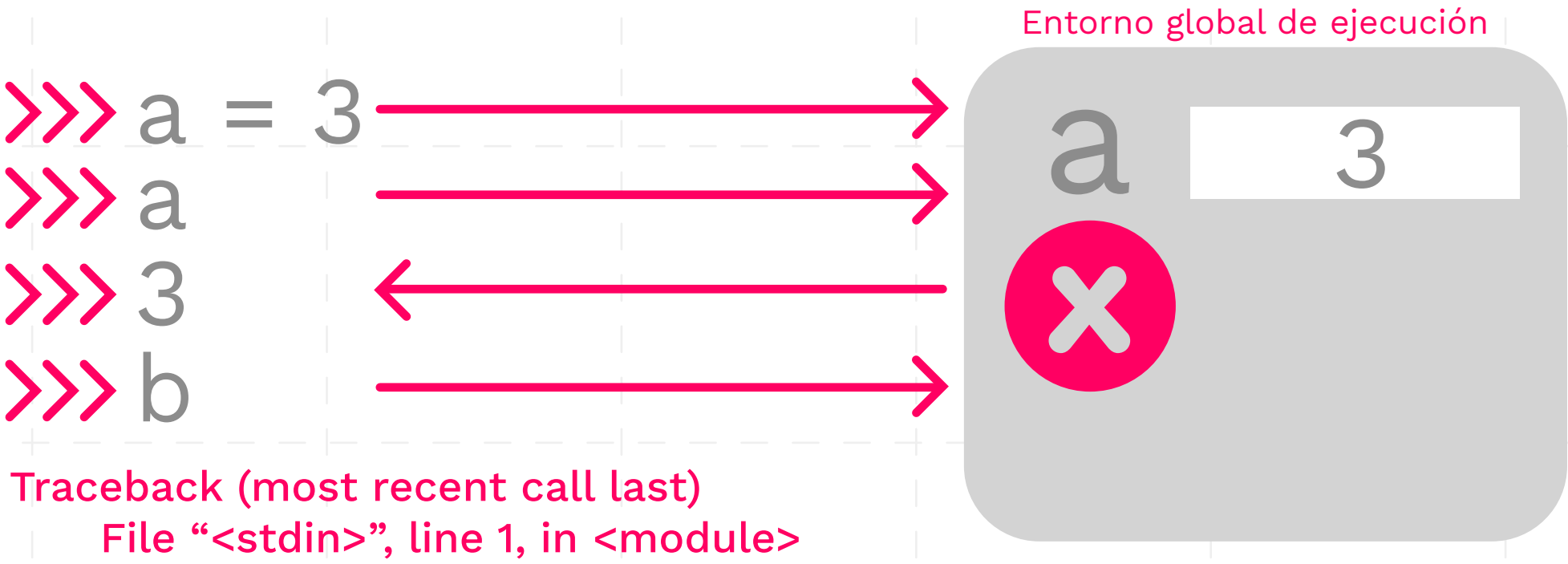
En la memoria del computador o memoria principal RAM (*Random Access Memory*) se encuentran almacenadas las variables y constantes que conviven con las instrucciones de los programas que se están ejecutando. Este espacio se llama Entornos.

Una instrucción >>> a=3 en un programa que indica que se está almacenando en la variable a el valor 3. Esto demuestra que dicha instrucción siempre se ejecuta en un entorno y, a menos que no tenga muy claro en qué entorno se encuentra en la variable, será muy difícil definir o identificar su resultado, los efectos laterales que puedan tener o cambiar un valor.

En este caso se está ejecutando esta instrucción en la consola de Python (Replit) que indica que está en un entorno global, en el cual está ejecutándose el intérprete de Python y al hacer esto se crea un espacio en la memoria que va a estar identificado con a y que tendrá un valor de 3, por eso es que cuando se escribe a lo que hace el intérprete de Python es verificar si en el entorno global de ejecución se encuadra una a y, efectivamente es cierto este aspecto, por lo cual esto permite retornar su valor.



Si se escribe b, al instante que se escribe lo que hace el intérprete es ir al entorno de ejecución y verificar si hay algo que se llama b y, efectivamente, retorna que no se encuentra b y solo identifica a a. Esto va a generar un error en la línea uno (1)



Traceback (most recent call last)
File “<stdin>”, line 1, in <module>
NameError: name ‘b’ is not defined

En el editor tenemos una función X que tiene una variable llamada variable local, luego se imprime y después, en el punto de entrada en la aplicación, simplemente se está llamando a esta función que seguramente lo que hará es imprimir: “juego de local”

```
def funcion_x():
    Local_var= ‘Juego de local’
    print ( local_var)

#============================
#Punto de entrada a ala aplicación
Funcion_x()
```

Cuando se ejecuta el programa, el intérprete lo primero que hace es ir al punto inicial de la aplicación y en ese momento el intérprete genera un entorno global de ejecución con la instrucción de interpretar un programa que le crea su entorno global de ejecución y allí, cuando se dirige a la función y se llama la función x, lo que hace es que crear un entorno local solo para esa función, es decir, que se crea un espacio de memoria pequeño dentro del entorno global y particularmente, para el entorno local de ejecución.

```
def funcion_x():
      Local_var= 'Juego de local'
      print ( local_var)

#============================
#Punto de entrada a ala aplicación
Funcion_x()
```

Entorno global de ejecución

Entorno local funcion_x

Cuando se define esta variable con el identificador local_var con el valor "juego de local" y, que este identifique que es un tipo *String*, inmediatamente genera esa variable internamente en la en la función_x , luego la imprime y muestra como resultado: "juego de local". Luego que termine la ejecución, simplemente el entorno global de ejecución de esta función_x termina y regresa a donde ya no tiene nada más que hacer, lo que deja la memoria libre y así terminan los dos entornos, tanto el entorno local; como el entorno global.

```
def funcion_x():
      Local_var= 'Juego de local'
      print ( local_var)

#============================
#Punto de entrada a ala aplicación
Funcion_x()
```

Entorno global de ejecución

Juego de local

Si se adiciona una instrucción luego de hacer el llamado a la función_x, se mostrará que el entorno global de ejecución en este momento está sin ninguna variable que se haya ejecutado y escribimos local_var, es decir, está tratando de imprimir una variable que tuvo su entorno en la funcion_x, pero que ya no está presente, entonces tratará de buscar esa variable que se llama local_var y no la va a encontrar, así que mostrará como respuesta en el módulo principal en línea diez (10) que dicha función no cuenta con un valor o existe un error.

```
def funcion_x():
      Local_var= 'Juego de local'
      print ( local_var)

#============================
#Punto de entrada a ala aplicación
Funcion_x()
print (local_var)
```

```
Traceback (most recent call last)
      File "main.py", line 10, in <module>
            print (local_var)
NameError: name 'local_var' is not defined
```

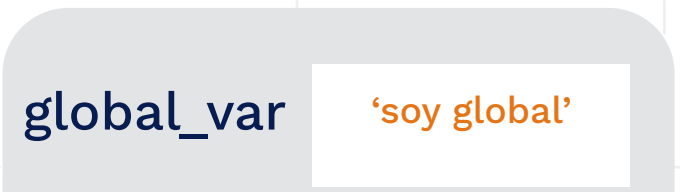
Entorno global de ejecución

En el siguiente ejemplo, la variable que se encuentra en la parte superior llamada global_var está en el entorno global de ejecución.

```
global_var = ' soy global'
#============================
def funcion_x():
      Local_var= 'Juego de local'
      print ( local_var)

#============================
#Punto de entrada a ala aplicación
Funcion_x()
```

Entorno global de ejecución



El programa recoge la variable global_var: “soy global” y la reconoce como de tipo *string*. Al llamar la funcion_x, imprime la variable local_var, la cual reconoce su valor por encontrarse en el entorno local, que es como si hiciera parte de su “vecindario”. Por lo tanto no encuentra problema alguno e imprime su valor. Al pasar por la variable, global_var, “soy global”, identifica que se encuentra en un entorno global y esto hace que sea reconocida en toda la ejecución del programa y se pueda obtener su valor en cualquier momento sin dificultad alguna.

```
global_var = ‘ soy global’
#============================
def funcion_x():
        Local_var= ‘Juego de local’
        print ( local_var)
        print (global_var)

#============================
#Punto de entrada a ala aplicación
Funcion_x()
```

A pesar que la variable global ha sido identificada en cualquier parte del programa, se recomienda no hacer uso de esta técnica con frecuencia, ya que no es una buena práctica de programación. Una vez termina la función, las variables locales pierden su valor pero la variable global no lo pierde hasta que se cierre todo el programa.

Se ha tratado el tema de los entornos de ejecución, específicamente de su importancia con las variables. Cuando una variable está definida dentro de una función, esto indica que se trata de una variable local y se encuentra en el entorno local de la función, por lo tanto se puede utilizar sin problemas en la función definida, pero no en otras funciones. También es posible contar con variables globales que pueden ser invocadas dentro de cualquier función y, si bien, pueden ser accedidas desde cualquier función, esto no es una buena práctica de programación, debido a que se pierde la trazabilidad de saber en qué momento y por quien ha sido alterada la variable.

Universidad de Caldas