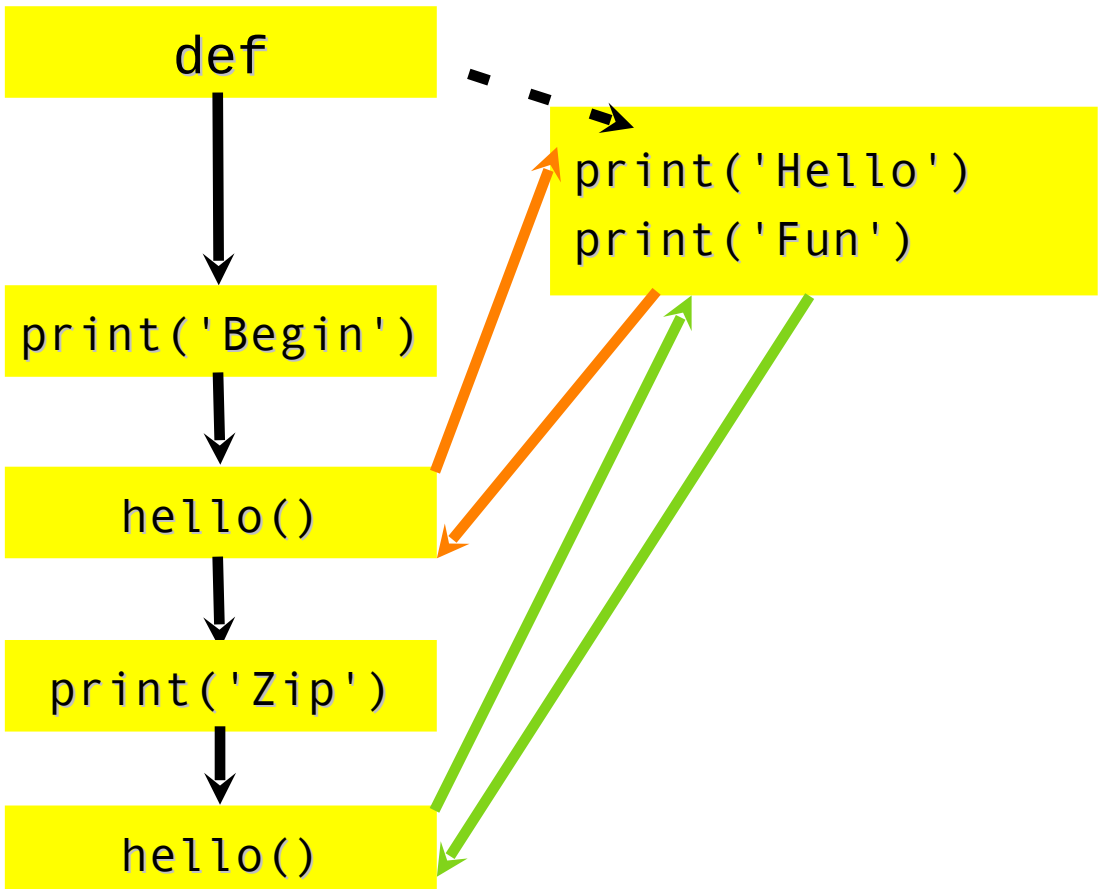
Unidad 4: Funciones

Declaración, parámetros y valores de retorno

Objetivos

- Utilizar adecuadamente el principio computacional de dividir un problema en módulos para su resolución.
- Crear funciones reconociendo adecuadamente los parámetros de entrada/salida y acciones a ejecutar para la resolución de problemas.
- Usar funciones adecuadamente para generar programas eficientes.
- Aplicar el envío de parámetros a funciones por referencia y valor y obtener el valor retornado para su uso

Instrucciones almacenadas y reutilizadas




```
# Programa
def hello():
    print('Hello')
    print('Fun')
##main
print('Begin')
hello()
print('Zip')
hello()
```

Output:

```
Begin
Hello
Fun
Zip
Hello
Fun
```

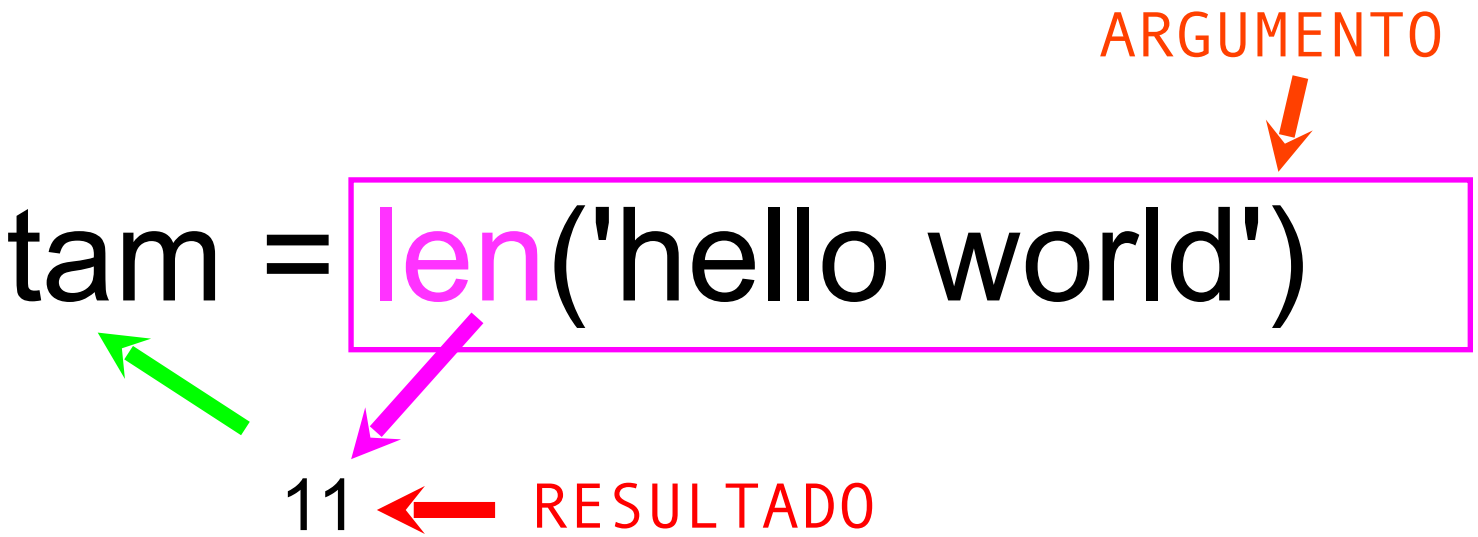
Podemos llamar a esos grupos de código reutilizables “funciones”.

Funciones en Python

- Hay dos tipos de funciones en Python.
  - Funciones internas (built-in) que vienen incluidas en Python - `input()`, `type()`, `float()`, `int()` ...
  - Funciones que definimos nosotros y luego usamos
- Tratamos los nombres de funciones como "nuevas" palabras reservadas y evitamos usarlas como nombres de variables

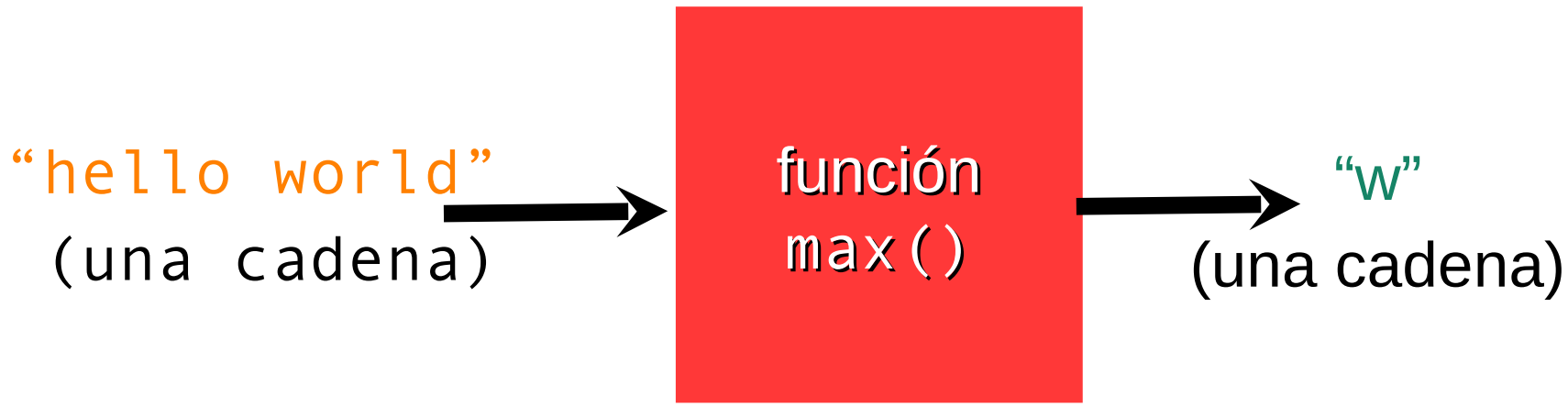
Definición de funciones

- Una función en Python es un código reusable que toma argumentos(s) como entrada, hace algún cómputo y retorna un resultado o resultados
- Una función se define usando la palabra reservada `def`
- Llamamos o invocamos a la función usando su nombre, paréntesis y argumentos en una expresión




```
>>> tam = len('hello world')
>>> print(tam)
>>> 11
>>> tiny = min('helloworld')
>>> print(tiny)
>>> d
```

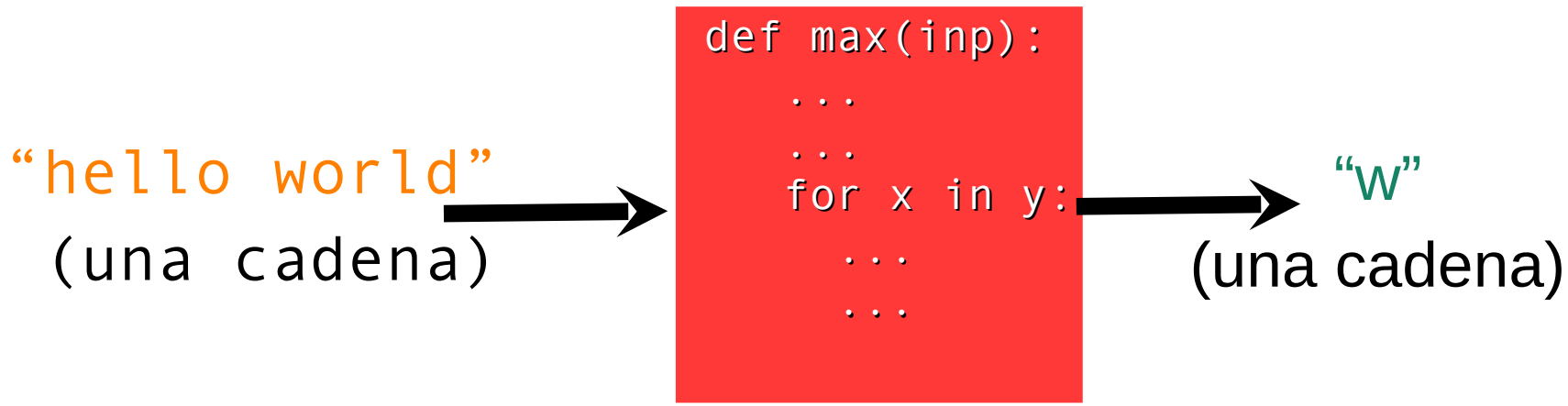
Función Max



Una función es un código almacenado que nosotros usamos. Una función toma alguna entrada y produce una salida.

```
>>> big = max('hello world')
>>> big
w
```

Función Max



Una función es un código almacenado que nosotros usamos. Una función toma alguna entrada y produce una salida.

```
>>> big = max('hello world')
>>> big
w
```

Argumentos

- Un argumento es un valor que pasamos (o enviamos) a la función como su entrada cuando llamamos a la función
- Usamos argumentos para poder llevar a la función a hacer distintos tipos de trabajo cuando la llamamos desde distintos sitios o veces
- Colocamos los argumentos en paréntesis después del nombre de la función

```
big = max('helloworld')
```

Creando nuestras propias funciones

- Creamos una nueva función usando la palabra clave `def` seguida por un nombre y por parámetros opcionales dentro de paréntesis.
- Identamos el cuerpo de la función
- Esto define la función pero no ejecuta el cuerpo de la función

```
# print_saludos() no recibe
# args
def print_saludos():
    print('ADN #ESPOL.')
    print('Felices Fiestas.')
```

Definición y usos

Una vez que hemos definido una función, podemos llamarla (o invocarla) tantas veces como queramos. Este es el modelo almacenar y reutilizar

```
# Definiciones de las funciones, previo al main
def printMensaje():
    print('ESPOL es A')
    print('ESPOL Universidad de Investigación.')
# Programa principal
printMensaje()
printMensaje()
print('End')
```

Parámetros

Un parámetro es una variable la cual usamos en la definición de la función, como "variable referencial" que será utilizada en el código de la función para accesar o referenciar los argumentos o valores a los cuales se aplicará la función en un llamado particular

```
# Definiciones de funciones
def factorial(n):
    fac = 1
    for i in range(1, n+1):
        fac = fac * i
    print(fac)
# Programa Principal
factorial(6)
factorial(8)
```

Valores de Retorno

Una función tomará sus argumentos, hará algún cómputo y devolverá (retornará) un valor para ser utilizado como el resultado o valor de la función invocada en la expression de llamada. La palabra clave return es utilizada para indicar el valor a devolver y finaliza la función.

```
# Definiciones de funciones
def factorial(n):
    fac = 1
    for i in range(1, n+1):
        fac = fac * i
    return(fac)
# Programa principal
x = factorial(6)
y = factorial(8)
print (x, y)
```

PARÁMETROS

RETORNO

ARGUMENTOS

Función void (vacía)

Cuando una función no retorna un valor, se denomina una función "void" o procedimiento

Las funciones void son vacías pero retornan un objeto None

```
# Definiciones:
def print_saludos():
    print('ADN #ESPOL.')
    print('Felices Fiestas.')
# Programa principal:
x = print_saludos()
print(x)
```

Output:

```
None
```

Parámetros opcionales

Un parámetro es opcional cuando tiene un valor ya establecido en la definición de la función. No se necesita especificar en la llamada a la función.

```
# Definiciones
def saludar(nombre, msj='Hola'):
    print(msj, nombre)
# Programa principal
saludar('Santiago')
saludar('Isabel', 'Buenos días')
```

sin especificar

Output:

```
Hola Santiago
Buenos días Isabel
```

Múltiples Valores de retorno

- En una función, se pueden retornar múltiples valores empaquetando estos en una tupla o lista. Se puede devolver varios valores simplemente retornándolos separados por comas

```
def operaciones(a, b):
    suma = a + b
    mult = a * b
    return suma, mult
# Programa principal
x , y = operaciones(10, 3)
print('suma:', x, 'mult:', y)
y , x = operaciones(5, 4)
print('suma:', y, 'mult:', x)
```

Output:

```
suma: 13 mult: 30
suma: 9 mult: 20
```