

```
mirror_mod = modifier_ob.  
#set mirror object to mirror_  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True
```

```
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_  
mirror_ob.select = 0  
= bpy.context.selected_obj  
data.objects[one.name].se  
print("please select exactly
```

```
-- OPERATOR CLASSES ----
```

```
types.Operator):  
# X mirror to the selected  
object.mirror_mirror_x"  
mirror X"
```

```
context):  
context.active_object is not
```

FUNCIONES

ING. CARLOS H. RUEDA C.



CONTENIDO

1. Funciones
 - 1.1. Ventajas
 - 1.2. ¿Cómo definir e invocar una función?
2. Ejemplos
3. Ejercicios

1. Funciones

Imaginemos que hay un conjunto de tareas que se necesitan realizar una y otra vez. En lugar de escribir el mismo código una y otra vez, se puede usar funciones para agrupar esas tareas en un solo lugar.

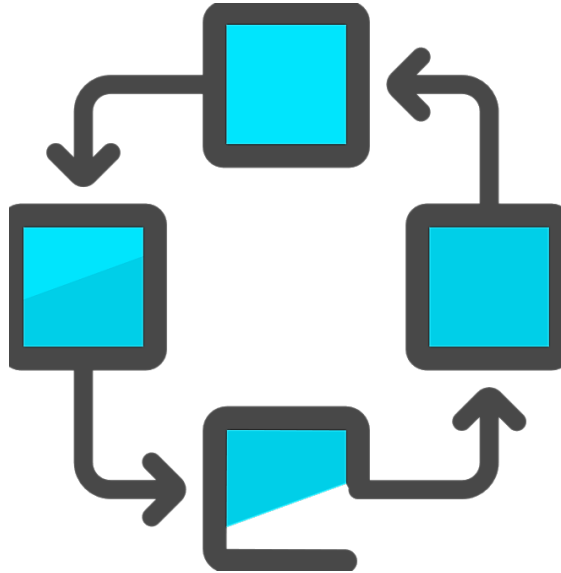
1. Funciones

Las funciones, son fragmentos de código a los que se les asigna un nombre específico.

1. Funciones

Dentro de una función, encontramos las mismas estructuras que en un algoritmo común, incluyendo sentencias secuenciales, condicionales y de repetición.

1. Funciones

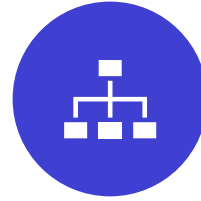


Al asignar un nombre a una función, podemos llamarla posteriormente tantas veces como sea necesario. Cuando se invoca la función, se ejecuta el conjunto de sentencias que contiene.

1.1. Ventajas de las funciones



REUTILIZACIÓN
DE CÓDIGO



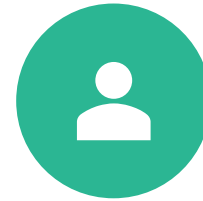
ORGANIZACIÓN



ABSTRACCIÓN

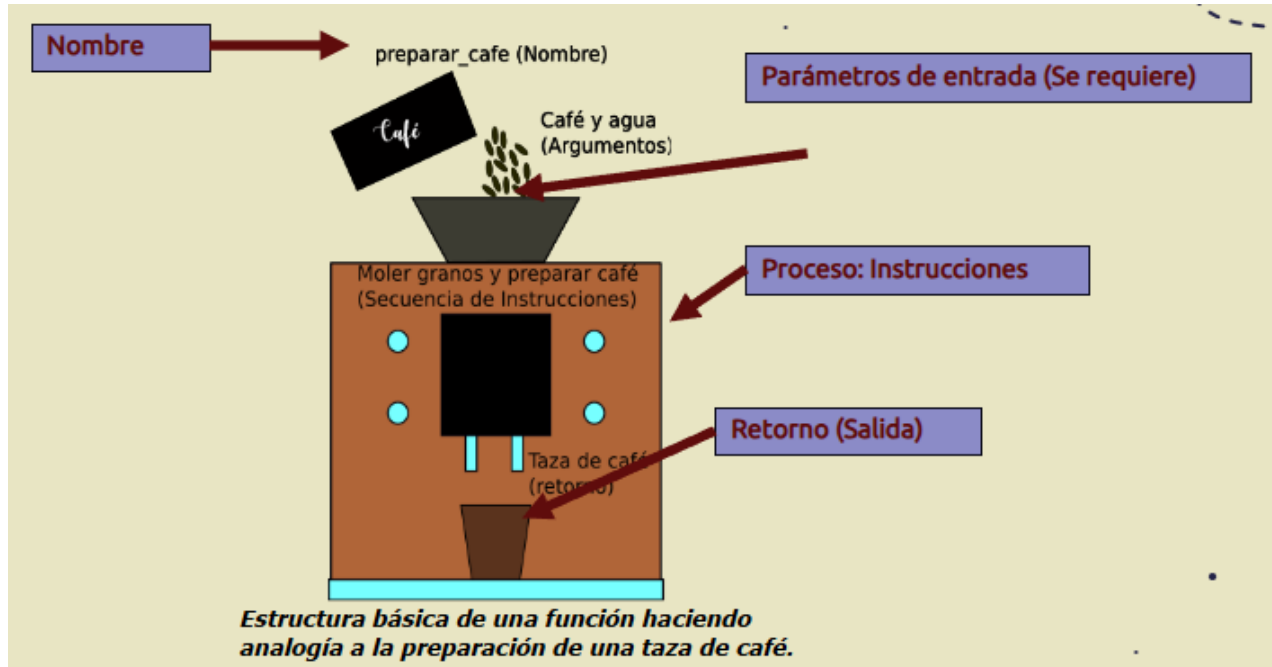


MODULARIDAD

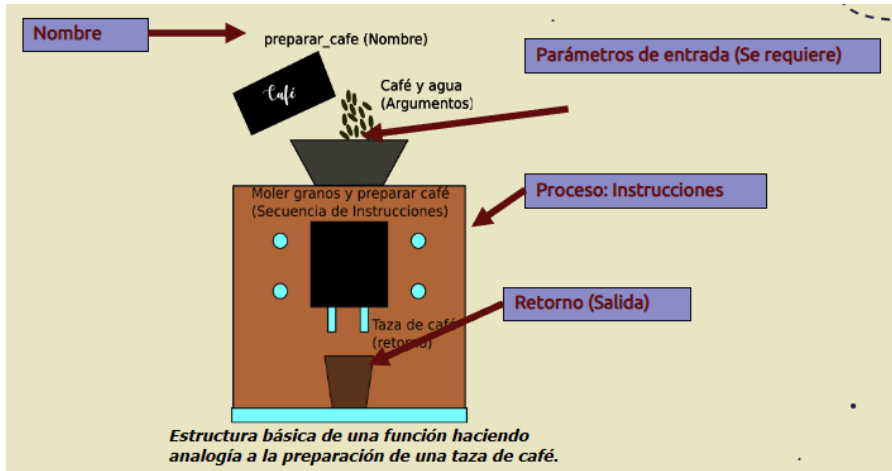


DEPURACIÓN
SIMPLIFICADA

1.2. ¿Cómo definir e invocar una función?



1.2. ¿Cómo definir e invocar una función?



Declarar una función

función Decir_mi_nombre (*nombre*)

frase = "Hola mi nombre es " + *nombre*
devolver frase

1.2. ¿Cómo definir e invocar una función?

Declarar una función

```
función Decir_mi_nombre (nombre)  
    frase = "Hola mi nombre es " + nombre  
    devolver frase
```

Invocar una función

```
Decir_mi_nombre('Brayan') // Hola mi nombre es Brayan  
Decir_mi_nombre('Stiven') // Hola mi nombre es Stiven
```

- se le denominan parámetros de una función
- se le denomina argumentos de una función
- se le denomina el retorno de una función

3. Ejemplos

Desarrollar una función que sume dos números

3. Ejemplos

Desarrollar funciones que realicen las operaciones básicas, suma, resta, multiplicación, división, módulo.

Hacer un llamado a cada uno de estas funciones.

3. Ejemplos

Desarrollar una función que genere saludos personalizados.

```
Función generar_saludo(nombre):
```

```
    Saludo = "¡Hola, " + nombre + "!"
```

```
    Devolver Saludo
```

```
Nombre_usuario = "María"
```

```
Saludo_personalizado = generar_saludo(Nombre_usuario)
```

3. Ejemplos

Desarrollar una función que calcule el factorial de un número. El factorial de un número entero positivo n se define como el producto de todos los enteros positivos desde 1 hasta n . Se denota como $n!$.

$$n! = n \times (n - 1) \times (n - 2) \times \dots \times 3 \times 2 \times 1$$

Casos especiales:

- $0!$ se define como 1 por convención.
- $1! = 1$.

3. Ejemplos

Desarrollar una función que calcule devuelva verdadero o falso si un número es primo.

Un número es primo, si solo es divisible entre 1 y el mismo.

Escriba un programa donde el usuario digite un número y el programa le muestre si el número es primo o no.

3. Ejemplos

Desarrollar una función que calcule la combinatoria

Donde:

$$C(n, k) = \frac{n!}{k! \times (n - k)!}$$

- n es el número total de elementos.
- k es el número de elementos seleccionados.
- $n!$ (factorial de n)
- $k!$ es el factorial de k
- $(n-k)!$ es el factorial de la diferencia entre n y k .

3. Ejemplos

Desarrolle una función que calcule la fila n del triángulo de pascal usando las combinatorias.

The diagram illustrates the relationship between binomial coefficients and the numerical values in Pascal's triangle. On the left, the first five rows of the triangle are shown using binomial coefficient notation $\binom{n}{k}$. On the right, the same triangle is shown with numerical values. A large gray arrow points from the left representation to the right representation.

$\binom{0}{0}$						1
$\binom{1}{0}$	$\binom{1}{1}$					1 1
$\binom{2}{0}$	$\binom{2}{1}$	$\binom{2}{2}$				1 2 1
$\binom{3}{0}$	$\binom{3}{1}$	$\binom{3}{2}$	$\binom{3}{3}$			1 3 3 1
$\binom{4}{0}$	$\binom{4}{1}$	$\binom{4}{2}$	$\binom{4}{3}$	$\binom{4}{4}$		1 4 6 4 1

4. Ejercicios
