## 5.- PROCESSING.PY: FUNCIÓNS

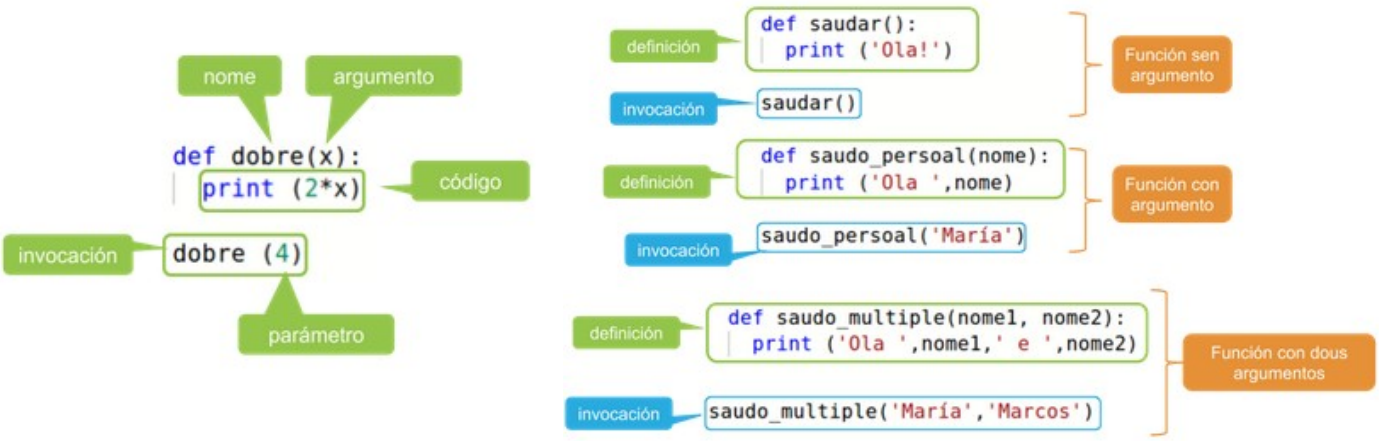
**Unha función é un subalgoritmo** que creamos e logo utilizamos (decimos **invocar** ou **chamar**) escribindo o seu nome para aforrar tempo (non temos que volver a escribir o código da función).

### Tipos de funcións

- **Funcións predeterminadas**; en Scratch, calquera dos bloques que usamos; en Python son **métodos**. Por exemplo, a función print(). Sempre levan dous parénteses e dous puntos.
- **Funcións creadas** por nós para un programa determinado. Usamos a palabra def e logo o nome da función
- **Bibliotecas** formadas por **métodos**, funcións que poden levarse dun programa a outro para aforrar tempo. As bibliotecas son chamadas en Python **módulos** ou nunha mala tradución do inglés " librerías". En Processing podemos instalar librerías.

### Sintaxe da función

Unha función adoita coller datos de entrada e da unha saída. Na creación da función usamos unha ou varías variables, chamadas **argumento(s)**. Cando invocamos á función usamos valores concretos, chamados **parámetros**. Podemos facer funcións sen argumentos.




Exemplo: funcións sen argumentos para facer un dado de número de caras variables e indicalo usando print()

```
1  def setup():
2      carasDado = int(random (3,21))
3      dado()
4      print(carasDado)
5
6  def dado(): #función sen argumento
7      print ('O número de caras do dado é...')
```

```
1  def setup(): #o programa é máis sinxelo usando unha función
2      carasDado()
3
4  def carasDado(): #función sen argumento
5      print ('O número de caras do dado é...')
6      caras = 1+int(random(20))
7      print(caras)
```

Exemplo de funcións con argumentos e logo invocadas con parámetros. Queremos facer un dado de caras variables decidido por nós e logo unha tirada aleatoria.

```
def setup():
    tiradaDado(20) #invocamos a función con un parámetro

def tiradaDado(carasDado): #función con argumento
    print ('O número de caras do dado e o número que saiu son... ')
    tirada = 1+int(random(carasDado))
    print carasDado, tirada
```

A venrtaxa da función é que podemos invocala varias veces:

```
def setup():
    tiradaDado(20) #invocamos a función con un parámetro
    tiradaDado(6)
    tiradaDado(12)

def tiradaDado(carasDado): #función con argumento
    print ('O número de caras do dado e o número que saiu son... ')
    tirada = 1+int(random(carasDado))
    print carasDado, tirada
```

**Exercicio: fai en processing o típico programa dun dado de 6 caras. Ten que sair unha tirada aleatoria ao darlle ao rato, moverse as 6 caras e remtar con unha.**

Nos programas adoitamos usar a palabra **return** para gardar o resultado da función nunha variable que usamos no programa.

```
def setup():
    pesoTerraqueo = 70 #parámetro para a función
    pesoMarciano = pesoEnmarte(pesoTerraqueo) #gardamos o resultado nunha variable
    print ' o teu peso en Marte sería...', pesoMarciano

def pesoEnmarte(peso): #peso é un argumento, non un parámetro
    pesoM = peso*0.38
    return pesoM #cálculo que devolve a función, o resultado depende dun parámetro
```