In [1]:
```
def circuito(tension,
        estado="cerrado",
        accion="puerta bloqueada",
        tipo="JA-U7") :
#
#--------Imprime datos del circuito
    print("-"*30)
    print(">> Tipo circuito :", tipo)
    print(">> Este circuito tiene ", accion)
    print(">> si le aplicas", tension , "voltios")
    print(">> queda en estado", estado)
#fin funcion circuito
#---------------------proceso principal
#
circuito(110, estado="abierto", tipo="X23")
circuito(500, estado="reiniciando")
```

```
------------------------------
>> Tipo circuito : X23
>> Este circuito tiene  puerta bloqueada
>> si le aplicas 110 voltios
>> queda en estado abierto
------------------------------
>> Tipo circuito : JA-U7
>> Este circuito tiene  puerta bloqueada
>> si le aplicas 500 voltios
>> queda en estado reiniciando
------------------------------
>> Tipo circuito : X23
>> Este circuito tiene  puerta bloqueada
>> si le aplicas 110 voltios
>> queda en estado cerrado
```

In [3]:
```
import io
def guarda_usuarios(*args) :
    f = open ("./dat/usuarios.dat", "a", encoding='utf-8')
    for arg in args:
        f.write(arg+ '\n')
    f.close()
#-------------------------Inici
guarda_usuarios("Rosa", "Marta", "Joan", "Pere")
```

In [5]:
```
# Función lambda para calcular el cuadrado de un número
square = lambda x: x ** 2
# Dado un texto y una palabra devuelve el número de veces
contar = lambda texto, palabra : texto.count(palabra)
# Pasa un carácter numérico a su letra [a-z]
numletra = lambda cnum : chr(ord(cnum)+48)
#---------------Probando las funciones
print(square(3))
print (contar ("es aplicable a multitud de situaciones", "es"))
```

```
9
2
b
```

Practica S02 Escribe las funciones lambda que se especifica más abajo, y pruébalas

x = lamda con parámetros a, b , que devuelve el producto de a * b -y = lamda con parámetros a,b,c que devuelve la suma de los 3 parámetros -z = lamda con parámetros a, b , que devuelve True si a**b > b**a -y False en otro caso.

In [14]:
```
x = lambda a, b : a * b
y = lambda a,b,c : a+b+c
z = lambda a,b : a**b > b**a
zz = lambda a,b :  a**b < b**a

print (x(2,2))
print (y(2,2,2))
print (z(3,2))
```

```
4
6
True
False
```

Crea funciones lambdaequivalentes a estas definiciones. defpor4_mas_b(a, b) : return(a*4 + b) defraicesk(a,b,c) : return(-b + 4 * a * c / 2 * a) defquita_puntos(cadena) : return(cadena.replace(".",""))

In [18]:
```
#def por4_mas_b(a, b) : return(a*4 + b)

por4 = lambda a,b : a*4 + b

#def raicesk(a,b,c) : return(-b + 4 * a * c / 2 * a)

raiz = lambda a, b, c: -b + 4 * a * c / 2 *a

#def quita_puntos(cadena) : return(cadena.replace(".",""))

quita = lambda cadena : cadena.replace(".","")


print(por4(3,4))

print(raiz(1,2,3))
```

```
16
4.0
HolaMundo
```

In [23]:
```
scores = [70, 60, 80, 90, 50]
filtered = filter(lambda score: score < 60, scores)
```

```
[50]
```

In [35]:
```
def addition (n):
    return n * 5
numbers = (1,2,3,4)
result = map(addition,numbers)
print(list(result))


mult = lambda a : a * 5

numeros =(1,2,3,4)
resultado = map(mult, numeros)
```

```
[5, 10, 15, 20]
[5, 10, 15, 20]
```

In [40]:
```
def repetir (n):
    if n>0:
        repetir(n-1)
    print(n)
    return
```

```
0
1
2
3
4
5
```

In [42]:
```
def cuenta_atras(num):
#------------ejemplo de recursiva
    num -= 1
    if num > 0:
        print(num)
        cuenta_atras(num)
    else:
        print("BOOOOOOOOM !!!!!!")
    print(">>Fin función ", num)
#----------------------inicio
#
```

```
4
3
2
1
BOOOOOOOOM !!!!!!
>>Fin función  0
>>Fin función  1
>>Fin función  2
>>Fin función  3
>>Fin función  4
```