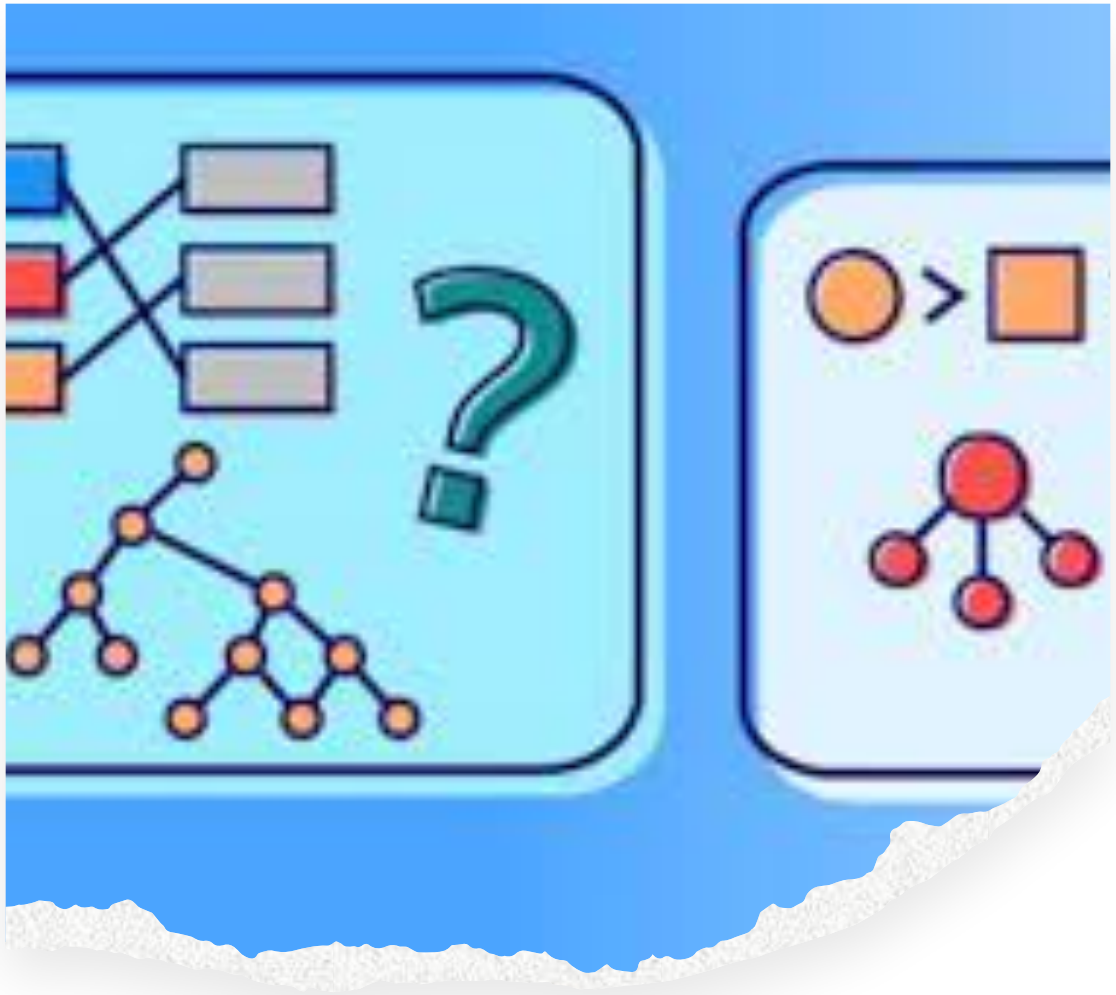
COLOMBIA POTENCIA DE LA VIDA

SENA

Diccionarios y funciones en Python



Declaración de un diccionario

```
diccionario = dict()
# Diccionario vacío inicializado con llaves
diccionario= {}
# Diccionario inicializado con valores
Diccionario = {'nombre':'Sandra' , 'edad': 44}
```

Accediendo a los elementos de un diccionario

```
print(Diccionario ['nombre’])
print(Diccionario.get('nombre','No existe’))
```

Agregar datos al diccionario después de creado

```
calificaciones.update({"Rosa": 3.7, "German": 4.8})
```

Técnicas de iteración

```
calificaciones = {
 'nombre': 'Sandra',
 'notafinal': 5.0
 }
calificaciones = {
 'Sandra': 5.0,
 'Adriana':5.0,
 'Mauricio':4.5,
 'Jose':2.5
 }

for i, j in calificaciones.items():

    print(i,j)
```

Técnicas de iterar los diccionarios

```
print("Técnicas por clave")
for i in calificaciones.keys():
    print(i)

print("Iterar por valor")
for j in calificaciones.values():
    print(j)

nombres = ['Maria', 'Sebastian', 'Ana']
edades = ['18', '25', '30']
for n, e in zip(nombres, edades):
    print('Tú nombre es {0}  y tu edad {1}.'.format(n, e))
```

Operaciones sobre los diccionarios

```
dicaleatorio={x: x**2 for x in (2, 4, 6)}
print(dicaleatorio)
```

IMPRIMIR NÚMEROS EN REVERSA

```
print("Números en reversa")
for i in reversed(range(1, 10, 2)):
        print(i)
```

BORRAR UN ELEMENTO DEL DICCIONARIO

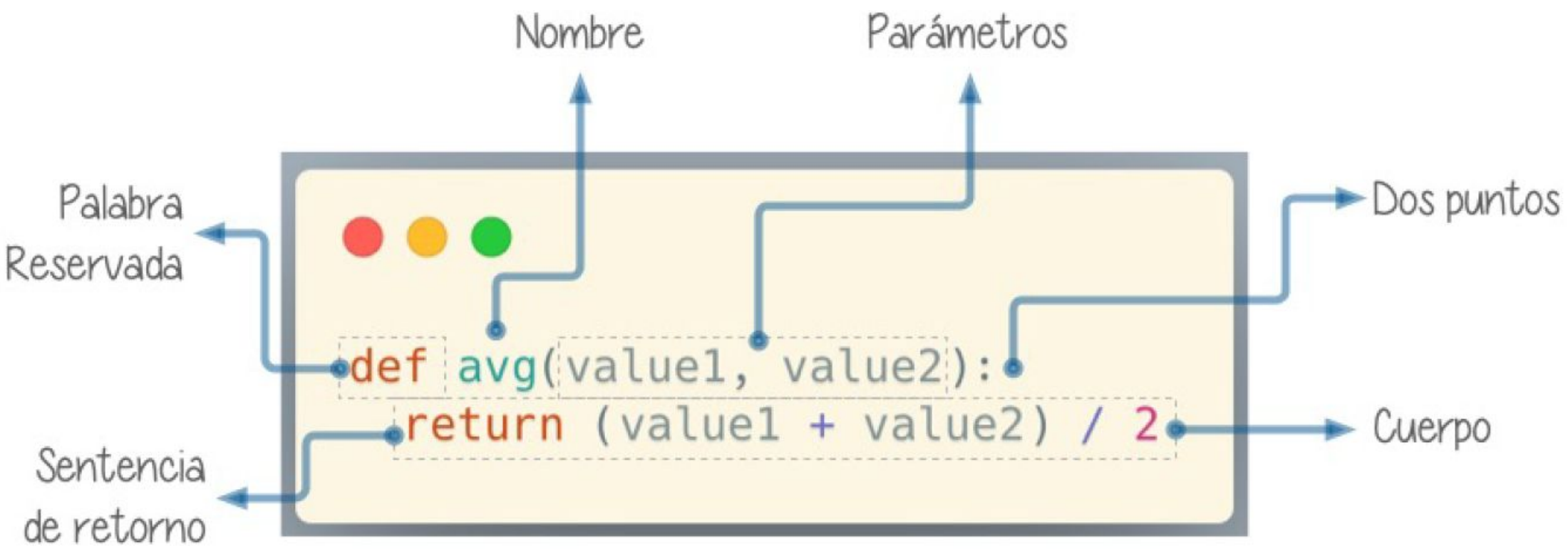
```
del(calificaciones['Rosa'])
for i, j in calificaciones.items():

    print(i,j)
```

Funciones

Es una estructura que nos permite agrupar código. Persigue dos objetivos claros:

1. No repetir segmentos de código durante nuestro programa.
2. Reutilizar el código para distintas situaciones.

Estructura de una función



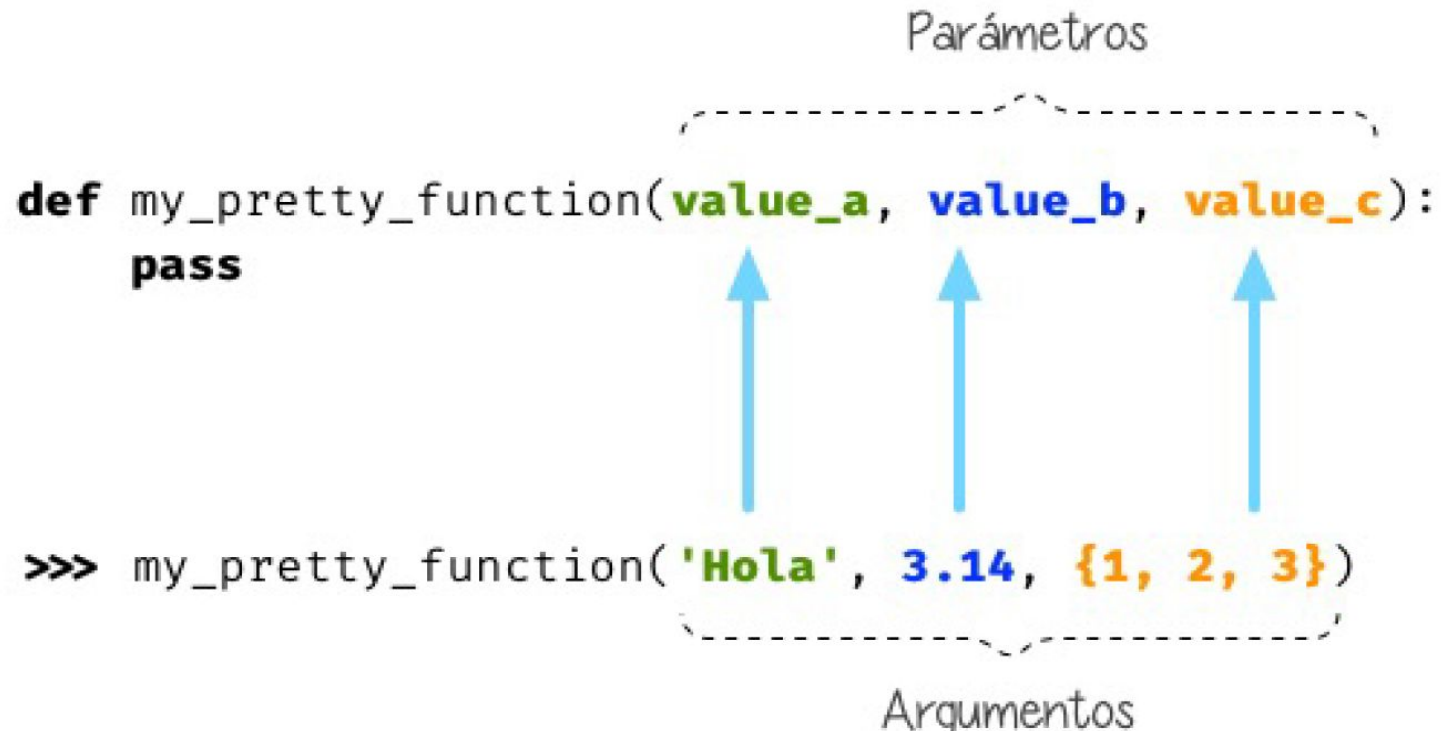
Definir una función

```
def saludar():
    print("saludo")

def retornarnumero():
    return 1

Saludar()
retornarnumero()

if retornarnumero()==1:
    print("devolvió un uno")
else:
    print("No devolvió un uno")
```

Funciones con Parámetros

```
def raiz(value):
    return value ** (1/2)

print(f'La raiz cuadrada es: {raiz(4)}')

def validarsiexiste(obj):
    if obj:
        print(f"{obj} is True")
    else:

        print(f"{obj} is False")

validarsiexiste(1)
```

Funciones con Parámetros



Ejercicio

Escriba una función en Python que reproduzca lo siguiente:

$f(x, y) = x2 + y2$ Valor para X=3 y valor para Y=5

Funciones con Parámetros Posicionales

```
def compra(marca,cantidad,valor):
    return dict(
        marca=marca,
        cantidad=cantidad,
        valor=valor*cantidad
    )

print(f' lo comprado fue:{compra("AMD",3,2500000)}')
```

Funciones con Parámetros Nominales

```
def compra(marca,cantidad,valor):
    return dict(
        marca=marca,
        cantidad=cantidad,
        valor=valor*cantidad
    )

print(f' lo comprado fue:{compra(marca=“AMD",cantidad=3,valor=2500000)}')
```

Parámetros por defecto

```
def compra(marca,cantidad,valor=2500000):
    return dict(
        marca=marca,
        cantidad=cantidad,
        valor=valor*cantidad
    )

print(f' lo comprado fue:{compra("AMD",3)}')
```

Modificando parámetros mutables

```
def lista(arg, result=[]):
    result.append(arg)
    print(result)

lista(‘domingo', [])
```

Ejercicio: Tomé el presente ejercicio, y pasé a la función la lista con los días de la semana restantes

Modificando parámetros mutables

```
def listalimpia(arg, result=None):
    if result is None:
        result = []
        result.append(arg)
        print(result)
listalimpia("a")

listalimpia("b")
```

Funciones anónimas «lambda»

Una función lambda tiene las siguientes propiedades:

1. Se escribe con una única sentencia.
2. No tiene nombre (anónima).
3. Su cuerpo tiene implícito un return.
4. Puede recibir cualquier número de parámetros.

Ejemplo:

```
numero_palabras = lambda t: len(t.strip().split())

print(numero_palabras("hola, esto es una prueba con lambda"))
```

Ejemplo 2 función Lambda

```
for i in range(2):

    for j in range(2):
        print(f"{i} & {j} = {operadorand(i, j)}")



operadorand = lambda x, y: x & y
```

Ejercicio propuesto

Se debe desarrollar una aplicación en Python que permita llevar el control de novedades de los equipos que se encuentran en los diferentes ambientes. La aplicación debe permitir:

-Agregar los equipos de cómputo con su ID y dispositivos (cargador y mouse) utilizando un diccionario para almacenar la información de cada equipo.

-Agregar novedades sobre los equipos de cómputo, fecha y descripción.

-Buscar un equipo de cómputo utilizando su ID y mostrar su información.

-Mostrar un reporte de equipos que presentan novedades utilizando una función que recorra la lista de novedades y muestre solo las que correspondan a equipos que presentan novedades.

-Se deben crear funciones para agregar, modificar y eliminar equipos, así como para mostrar la lista completa de equipos y su estado actual.

GRACIAS

Línea de atención al ciudadano: 01 8000 910270
Línea de atención al empresario: 01 8000 910682

@SENAComunica

www.sena.edu.co