**kwargs

Python TOTAL

Los argumentos con palabras clave (*keyword arguments*, abreviado ***kwargs***), sirven para identificar el argumento por su nombre, independientemente de la posición en la que se pasen a su función. Si no se conoce de antemano su cantidad, se utiliza el parámetro **kwargs que los agrupa en un diccionario.

*Al igual que para *args, el nombre **kwargs no es mandatorio pero si es sugerido como buena práctica. Cualquier nombre iniciado en ** referirá a estos argumentos de cantidad variable.*

```
def atributos_persona(**kwargs):
```

```
atributos_persona(ojos="azules", pelo="corto")
```

**kwargs = {'ojos': 'azules', 'pelo': 'rubio'}

*args

Python TOTAL

En aquellos casos en los que no se conoce de antemano el número exacto de argumentos que se deben pasar a una función, se debe utilizar la sintaxis *args para referirse a todos los argumentos adicionales luego de los obligatorios.

*El nombre *args no es mandatorio pero si es sugerido como buena práctica. Cualquier nombre iniciado en * referirá a estos argumentos de cantidad variable.*

La función recibirá los argumentos indefinidos *args en forma de tupla, a los que se podrá acceder o iterar de las formas habituales dentro del bloque de código de la función.

```
def mi_funcion(arg_1, arg_2, *args):
```

```
mi_funcion("ejemplo", 25, 40, 75, 10):
```

arg_1

arg_2

args = (40, 75, 10)

interacción entre funciones

Python TOTAL

Las salidas de una determinada función pueden convertirse en entradas de otras funciones. De esa manera, cada función realiza una tarea definida, y el programa se construye a partir de la interacción entre funciones.

```
def funcion_1():
    ...
    return a

def funcion_2(a):
    ...
    return b

def funcion_3(b):
    ...
    return c

def funcion_4(a,c):
    ...
    return d
```

funciones dinámicas

> La integración de diferentes herramientas de control de flujo, nos permite crear funciones dinámicas y flexibles. Si debemos utilizarlas varias veces, lograremos un programa más limpio y sencillo de mantener, evitando repeticiones de código.

- Funciones
- Loops (for/while)
- Estructuras condicionales
- Palabras clave (return, break, continue, pass)

```
def mi_funcion(argumento):
    for item in ...
        if a == b ...
                ...
        else:
                return ...
    return ...
```

Python TOTAL

Python TOTAL

return

Para que una función pueda devolver un valor (y que pueda ser almacenado en una variable, por ejemplo), utilizamos la declaración return.

```
def mi_funcion():
    return [algo]
```

La declaración return provoca la salida de la función
Cualquier código que se encuentre después en el bloque de la función, no se ejecutará

```
resultado = mi_funcion()
```

La variable **resultado** almacenará el valor devuelto por la función **mi_funcion()**

funciones

Python TOTAL

> Una función es un bloque de código que solamente se ejecuta cuando es llamada. Puede recibir información (en forma de *parámetros*), y devolver datos una vez procesados como resultado.

una función es definida
mediante la palabra clave **def**

```
def mi_funcion(argumento):
```

Los argumentos contienen información que la función utiliza o transforma para devolver un resultado

```
mi_funcion(mi_argumento)
```

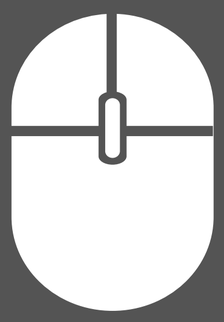
Para llamar a una función, basta utilizar su nombre, entregando los argumentos que la misma requiere entre paréntesis.

Python TOTAL

documentación

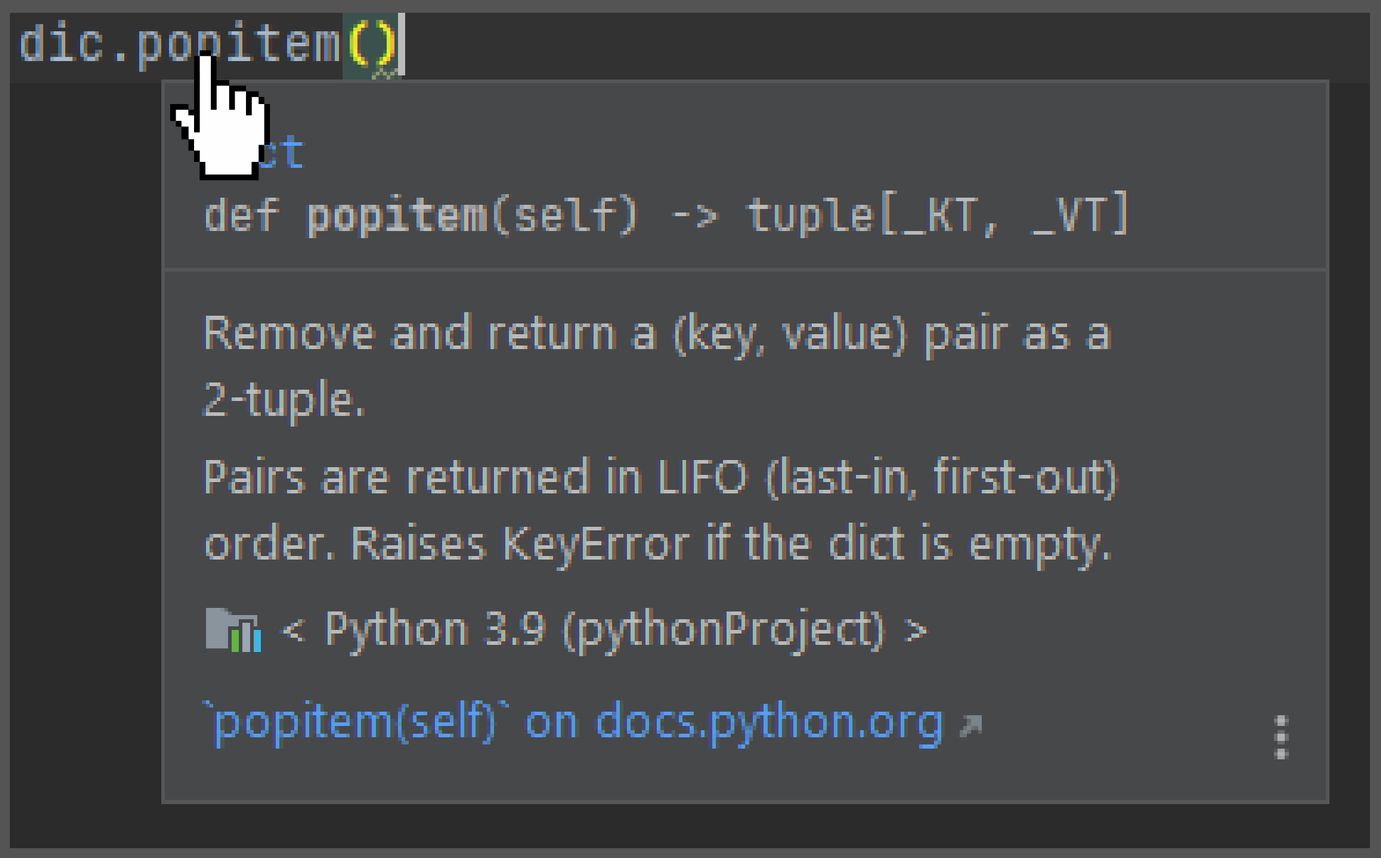
La documentación es nuestra biblioteca de consulta en Python. Al escribir código, es de uso permanentemente para solucionar dudas relacionadas al funcionamiento de métodos y los argumentos que reciben.

Si utilizas PyCharm:



Sostén el cursor sobre el nombre del método del que deseas obtener información. Se desplegará una ventana flotante.

Esto funcionará en otros IDEs del mismo modo.



También, debes mantener cerca la documentación oficial de Python (o Biblioteca Estándar de Python), que contiene toda la información necesaria:

https://docs.python.org/es/3.9/library/index.html

No dejes de buscar en Google tus dudas, para hallar una explicación que se ajuste a ti.