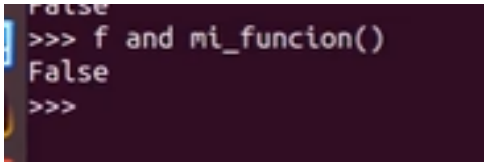
Semana 04

AND produce corto circuito:

Si el primer elemento del operador es falso entonces no hay que evaluarlo



no dara error incluso si mi_funcion() no ha sido declarada, porque no se evalua el segundo elemento.

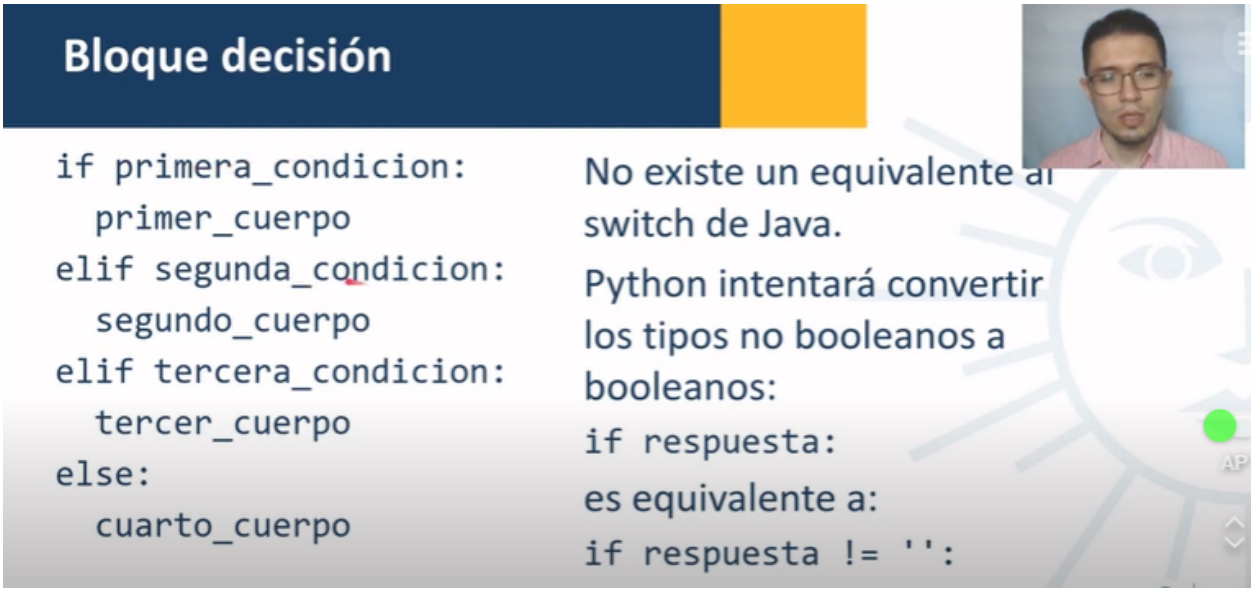
OR produce corto circuito, en caso de true


```
>>> f or mi_funcion()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'mi_funcion' is not defined
```

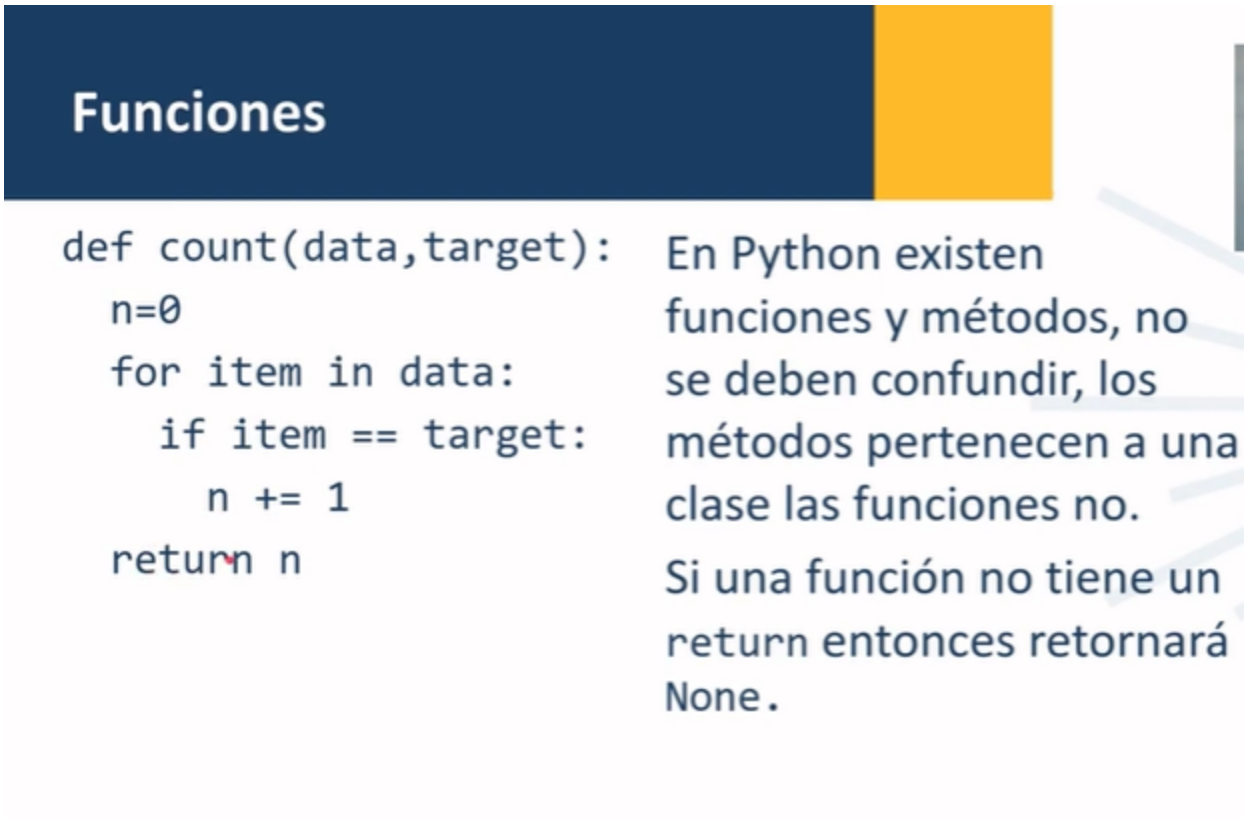
Operadores de igualdad

1. Operadores de Equivalencia
    - Verifica si 2 valores son equivalentes
2. Operadores de Identidad
    - Verifica si los identificadores son Aliases, es decir, verifica si dos nombres hacen referecncia al mismo objeto

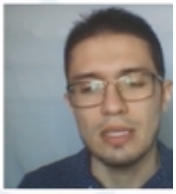
Control de Flujo



en python no existe el switch, se puede emular con un diccionario cuyos valores sean funciones.



Paso de Parámetros



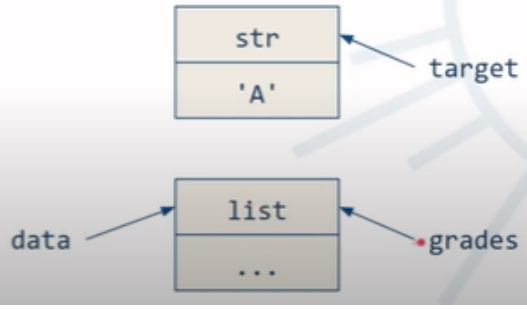
El paso de parámetros sigue la semántica del enunciado de asignación.

Al hacer esto:

```
prizes = count(grades, 'A')
```

Implícitamente ocurre:

```
data = grades
target = 'A'
```



Parámetros por Defecto

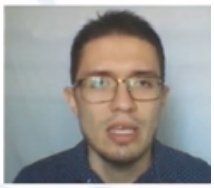
Python permite la definición de parámetros por defecto de la siguiente forma:

```
def foo(a, b=15, c=27):
    ...

foo(a)
```

Hay que considerar que si un parámetro se define con un valor por defecto, todos los siguientes también tiene que serlo.

Parámetros por Palabra Clave



Para el envío de parámetros por palabra clave, lo único que se necesita es un asignación de forma explícita de los parámetros actuales a los parámetros formales utilizando su nombre.

```
def set_alarm(day, month, year, hour, minutes)
    ...
set_alarm(year=2019, month=1, day=31, minutes=15, hour=10)
```

Lanzamiento de Excepciones

```
import collections

def sum(values):
  if not isinstance(values, collections.Iterable):
    raise TypeError('values must be Iterable')
  total = 0
  for v in values:
    if not isinstance(v, (int, float)):
      raise TypeError('elements must be numeric')
    total = total+ v
  return total
```

Definición de una Clase

```
def get_customer(self):
   return self._customer

def get_bank(self):
   return self._bank

def get_account(self):
   return self._account

def get_limit(self):
   return self._limit

def get_balance(self):
   return self._balance
```

Definición de una Clase

```
def charge(self, price):
   """Realiza un cargo a la tarjeta.

  Retorna True si el cargo fue procesado y Falso sino.
   """
   if price + self._balance > self._limit:
     return False
   else:
     self._balance += price
     return True

def make_payment(self, amount):
   """Recibe un pago del cliente."""
   self._balance -= amount
```

Uso de la Herencia

```
import CreditCard

class AbusiveCreditCard(CreditCard):
  """Una tarjeta de crédito abusiva."""
  def __init__(self, customer, bank, acnt, limit, apr):
     """El balance inicial es cero.
     customer  nombre del cliente (ej.:, 'John Bowman')
     bank      nombre del banco (ej.:, 'California Savings')
     acnt      número de la tarjeta (ej.:, '5391 0375 9387 5309')
     limit     límite de crédito
     apr       tasa de porcentaje anual (ej.:, 0.0825 for 8.25% APR)
     """
     super().__init__(customer, bank, acnt, limit)
     self._apr = apr
```