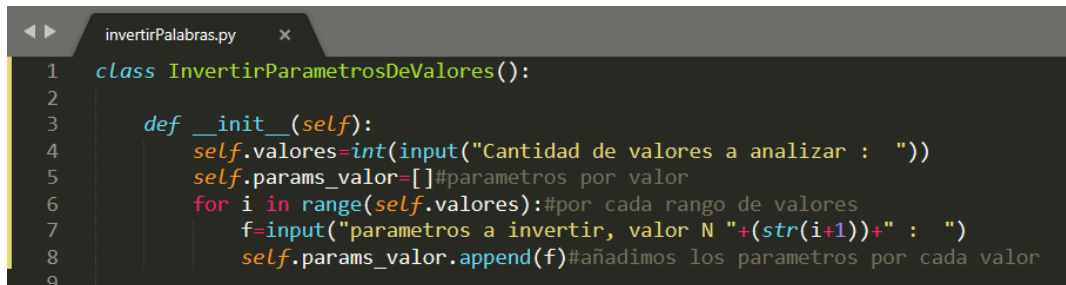


```
class InvertirParametrosDeValores():
```

```
    def __init__(self):
```



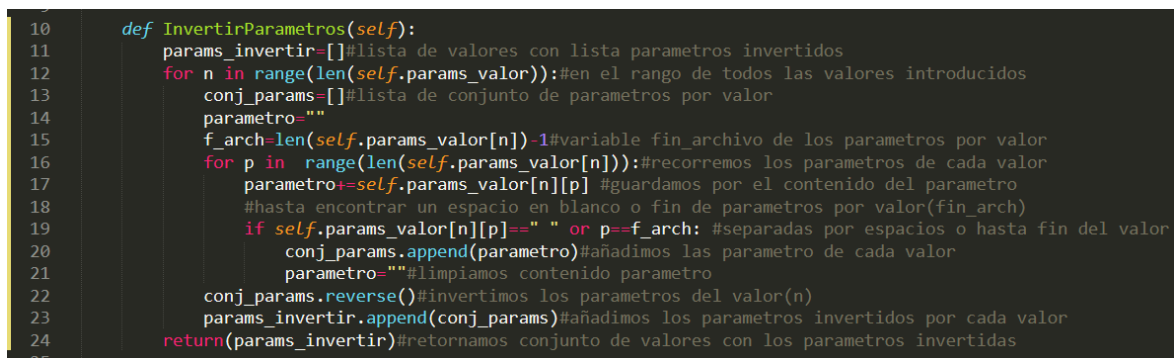
```
1 class InvertirParametrosDeValores():
2
3     def __init__(self):
4         self.valores=int(input("Cantidad de valores a analizar : "))
5         self.params_valor=[]#parametros por valor
6         for i in range(self.valores):#por cada rango de valores
7             f=input("parametros a invertir, valor N "+str(i+1)+" : ")
8             self.params_valor.append(f)#añadimos los parametros por cada valor
9
```

Inicializo la clase con la cantidad de valores que voy a pedir por pantalla

Creo una lista x (valores) con cada cadena que pido

por pantalla

```
def InvertirParametros(self):
```



```
10 def InvertirParametros(self):
11     params_invertir=[]#lista de valores con lista parametros invertidos
12     for n in range(len(self.params_valor)):#en el rango de todos los valores introducidos
13         conj_params=[]#lista de conjunto de parametros por valor
14         parametro=""
15         f_arch=len(self.params_valor[n])-1#variable fin_archivo de los parametros por valor
16         for p in range(len(self.params_valor[n])):#recorremos los parametros de cada valor
17             parametro+=self.params_valor[n][p] #guardamos por el contenido del parametro
18             #hasta encontrar un espacio en blanco o fin de parametros por valor(fin_arch)
19             if self.params_valor[n][p]==" " or p==f_arch: #separadas por espacios o hasta fin del valor
20                 conj_params.append(parametro)#añadimos las parametro de cada valor
21                 parametro=""#limpiamos contenido parametro
22         conj_params.reverse()#invertimos los parametros del valor(n)
23         params_invertir.append(conj_params)#añadimos los parametros invertidos por cada valor
24     return(params_invertir)#retornamos conjunto de valores con los parametros invertidas
25
```

Para invertir los parámetros de cada cadena de cada valor creo el método:

```
def InvertirParametros(self):
```

En este método recorro el contenido de la cadena de cada valor buscando

el separador " " espacio en blanco.

Creo la lista llamada conjunto de parámetros con los campos siguientes:

parámetros encontrados en la cadena pedida por consola que estaban separados por el espacio en blanco.

Y poder invertir cada campo del conjunto de parámetros.

Guardamos el conjunto de campos invertidos en una lista llamada params\_invertid

por cada valor.

Devolvemos un lista valores con cada parámetro (conjunto de campos invertidos)

def Mostrar(self,f):

```
26 def Mostrar(self,f):
27     #en este metodo recorremos de lista de parametros invertidos por cada lista de un valor f
28     self.f=f
29     parametro=""
30     #recorremos la lista de parametros(f) para crear la cadena a mostrar separados por espacio en blanco
31     for i in range(len(self.f)):#recorremos i parametros por valor long de cada valor
32         parametro+=self.f[i]+" "#guardamos cada parametro en una cadena separada por espacio en blanco
33     return(parametro)#devolvemos la cadena de parametros de ese valor f
34
35
```

En este método muestro por cada parámetro pasado f(lista de campos invertidos)

Muestro una cadena = al conjunto de los campos invertidos o parámetros separados por espacio en blanco.

MOSTRAR POR PANTALLA:

```
36
37 arr_frases=InvertirParametrosDeValores()
38 invertir=arr_frases.InvertirParametros()
39 for i in range(len(invertir)):#en el rango de la lista valores invertidos
40     f=invertir[i]#por cada valor invertido
41     pantalla=arr_frases.Mostrar(f)#la llamada Mostrar() nos devuelve por cada valor f
42     #una cadena de parametros separados por un espacio en blanco
43     print("CASE #", i+1,"": " ", pantalla)
44
```

Para mostrar por pantalla recorro la lista con los valores y le paso al método:

Mostrar (el parámetro o conjunto de campos invertidos)

que hay en cada valor.

De esta manera muestro en el rango de cada valor:

el case k corresponda a ese valor:      + el resultado de la llamada Mostrar():  
(la cadena de cada valor invertida )