Translations:Dimensionner une installation photovoltaïque autonome/265/en

1. function check surface volume

```
def iter(data,consoelecday,v0,p0,joursnoelec):
```

```
    #elec is energy in the battery : initial elec=v0 (battery volume at t0)
    elec=v0
    #current_streak is the number of days of consecutive blackout
    current_streak = 0
    #listjnoelec is the list in which we register the number of days of blackout
    listjnoelec=[]
    #loop on the input date (argument of the function)
    for i in range(len(data)):
        #recupday is electricity produced at day i (power_of_1kWc*p0)
        #p0 is the peak power entered as argument of the function
        recupday=data.iloc[i]*p0
        #consoday is the consumption of day i entered as argument of the function
        consoday=consoelecday
        #energy updated with consoday and recupday
        elec=elec+recupday-consoday
        #if updated energy is higher than battery volume
        if elec>v0:
            #print("battery full")
            #number of consecutive blackout days is reinitialized
            current_streak=0
            #energy is equal to the volume of the battery
            elec=v0 #hypothese gestion du trop plein ok
            continue
        #if updated energy is lower than the max percentage of discharge and the number of consecutive days of blackout is lower than the tolerated number
        elif elec<d*v0 and current_streak<joursnoelec:
            #if the number of consecutive days of blackout is equal to zero
            if current_streak==0:
                #add current day's date and a counter to the listjnoelec list
                listjnoelec.append([data.index[i].strftime('%d-%m-%Y'),1])
            #otherwise
            else:
                #increment the counter of the last entry of listjnoelec
                listjnoelec[-1][-1]+=1
            #reinitialization of the negative value of elec
            elec=0
            #incrementation of the consecutive days of blackout
            current_streak+=1
            continue
        #if energy is lower to the max percentage of discharge and the number of consecutive days of blackout is higher than the number of tolerated days
        elif elec<d*v0 and current_streak>=joursnoelec:
            #the function returns a null tuple
            return (0,0)
        #otherwise
        else:
            #reinitialization of the number of consecutive days of blackout
            current_streak=0
    #nb of episode of blackout  = lenght of list listjnoelec
    nb_episode_no_elec=len(listjnoelec)
    #if listjnoelect is not empty
    if len(listjnoelec)!=0:
        #list of durations of blackout episodes
        list_duree_episode_no_elec=[length[1] for length in listjnoelec]
        #mean of this list
        duree_moy_episode_no_elec=sum(list_duree_episode_no_elec)/len(list_duree_episode_no_elec)
    #otherwise
    else:
        #mean duration equal to zero
        duree_moy_episode_no_elec=0
    #print("powers and batteries permit to meet the electricity consumption needs on the dataset")
    #return variable necessary to process results
    return (v0,p0,joursnoelec,nb_episode_no_elec,duree_moy_episode_no_elec,listjnoelec)
```

1. hypothesis full battery at t0

elec=batterie0 resultpuissancevolume=(batterie0,puissance0,jnoelec,0)