In [2]:

```
#Code2: This part of the code estimated the Peak Scheme Water Demand (PSWD) and t

#import the required Python packages

import pandas as pd
import datetime   #check this
#import pyeto
import numpy as np
import ast        #check this
from ast import literal_eval  #check this
from pandas import DataFrame

#from pyeto import fao
from datetime import datetime
%matplotlib inline
#check this
```

In [3]:

```
data=pd.read_excel('Pilot20190124_Part1.xlsx') #This is the output file from code
#data
```

In [4]:

```
#To add a new coloumn for date palms irrigated area (1ha=10000m2  and *assuming t

data['harv_i_ha']= data['area_m2']/(10000)
```

In [5]:

```
%%time

#Setting the default value for these parameters



for i in range (1,13):
    data['ACWR_{}'.format(i)]=0       #ACWR: Average Crop Water Requirement in (m
    data['PCWR_{}'.format(i)]=0       #PCWR: Peak Crop Water Requirement (l/s/ha)
    data['harvested_{}'.format(i)]=0  #This repsents the actual area harvested in
    data['PWD_{}'.format(i)]=0        #PWD: Peak Water Demand in (l/s)
    data['SSWD_{}'.format(i)]=0       #SSWD: Seasonal Scheme Water Demand in (m3)
```

```
CPU times: user 61.9 ms, sys: 15.5 ms, total: 77.4 ms
Wall time: 76.1 ms
```

In [6]:

```
%%time
#STEP 1: Compute the ACWR from ETc - check FAO1992- page 43-

#acwr=row['ETo_{}'.format(i)]*30*row['kc_{}'.format(i)] - row['eff_{}'.format(i)]
#once the available water content layer is obtained, the last past should be adde

for i in range(1, 13):
    eto = f'ETo_{i}'
    kc = f'kc_{i}'
    eff = f'eff_{i}'
    acwr = f'ACWR_{i}'
    data[acwr] = data[eto]*30*data[kc] - data[eff]*30
    data.loc[data[acwr]<0,acwr] = 0
```

```
CPU times: user 350 ms, sys: 286 ms, total: 636 ms
Wall time: 629 ms
```

In [7]:

```
%%time
# STEP 2: Computing the PCWR (l/s/ha)

# The following notes expatins the equaitons:
# 1. Crop Water Requirment, CWR (m3/ha) = CWR (mm) * 10
# 2. Average Crop Water Requirement (m3/ha/d) = (CWR (m3/ha) / Length of crop dur
# 3. Assuming here that the lenght of the season is 30 (1 month)
# 4. Assuming that the Peak Crop Water Requirement = 2 * average crop water requi
# 5. conversion factor from (m3/d/ha) to (l/sec/ha) is 0.012
# Naming changed from (crop water need,CWN) in the initial code to (crop water re




for i in range(1,13):
    data['PCWR_{}'.format(i)]=((data['ACWR_{}'.format(i)]*10)/30)*2*0.012
```

```
CPU times: user 70.5 ms, sys: 36.6 ms, total: 107 ms
Wall time: 89 ms
```

In [8]:

```
%%time

for index,row in data.iterrows():
    len_init1= (len(range(row['init1_start_month'],row['init1_end_month']))+1)
    len_init2= (len(range(row['init2_start_month'],row['init2_end_month']))+1)
    len_init= (len_init1)+(len_init2)
#len_init2= len(range(row['init2_start_month'],row['init2_end_month']))+1
#len_init=len_init1 + len_init2
#len_init1=(row['init1_start_month'] - row['init2_end_month'])
print(len_init)
```

```
5
CPU times: user 2.76 s, sys: 121 ms, total: 2.88 s
Wall time: 2.9 s
```

In [9]:

```
%%time

#STEPs 3 and 4: Estimating Peak Water Demand (PWD) in (l/s) and Seasonal Scheme W
# In order to estimate PWD and SSWS we need first to compute the irrigated area u



#PWD = PCWR / Irrigation efficiency(IrrEff) ; IrrEff = Field Application Efficien
#deff (distribution efficieny %): 0.95 (all scenarios)
#aeff (field application efficiency %) : 0.6 (SU), 0.75 (SP), 0.9 (DR)

##Scenario option might be: Surface irrigation aeff = 0.6, Sprinkler aeff = 0.75



count_p=0 #To adjust the count of months in the loop below
count_h=0 #To adjust the count of months in the loop below
pumping_hours_per_day=10 #is this an assumption??
deff= 1
aeff= 0.6

init1_count = np.zeros(len(data))
init2_count = np.zeros(len(data))
late_count = np.zeros(len(data))

for i in [11,12,1,2,3,4,5,6,7,8,9,10]:

    init1 = [(i >= j) & (i <= k) for j, k in zip(data['init1_start_month'],data['
    init2 = [(i >= j) & (i <= k) for j, k in zip(data['init2_start_month'],data['

    init1_count += init1 * 1
    init2_count += init2 * 1
    init_count = init1_count + init2_count

    data.loc[np.array(init1) | np.array(init2),'harvested_{}'.format(i)] =(data['
    data.loc[np.array(init1) | np.array(init2),'PWD_{}'.format(i)]= (data['PCWR_{
    data.loc[np.array(init1) | np.array(init2),'SSWD_{}'.format(i)]= (data['ACWR_

    dev = [(i >= j) & (i <= k) for j, k in zip(data['dev_start_month'],data['dev_

    data.loc[dev,'harvested_{}'.format(i)]=data['harv_i_ha']
    data.loc[dev,'PWD_{}'.format(i)]=(data['PCWR_{}'.format(i)]*data['harv_i_ha']
    data.loc[dev,'SSWD_{}'.format(i)]= (data['ACWR_{}'.format(i)]*10*data['harv_i


    mid = [(i >= j) & (i <= k) for j, k in zip(data['mid_start_month'],data['mid_

    data.loc[mid,'harvested_{}'.format(i)]=data['harv_i_ha']
    data.loc[mid,'PWD_{}'.format(i)]=(data['PCWR_{}'.format(i)]*data['harv_i_ha']
    data.loc[mid,'SSWD_{}'.format(i)]= (data['ACWR_{}'.format(i)]*10*data['harv_i

    late = [(i >= j) & (i <= k) for j, k in zip(data['late_start_month'],data['la

    late_count += late * 1

    data.loc[late,'harvested_{}'.format(i)]=(data['harv_i_ha']/([len(range(i,j+1)
    data.loc[late,'PWD_{}'.format(i)]= (data['PCWR_{}'.format(i)]*(data['harveste
    data.loc[late,'SSWD_{}'.format(i)]= (data['ACWR_{}'.format(i)]*10*(data['harv
```

```
CPU times: user 8.72 s, sys: 1.18 s, total: 9.91 s
Wall time: 9.85 s
```

In [10]:

```
#Create a Pandas Excel writer using XlsxWriter as the engine.
writer = pd.ExcelWriter('Pilot20190124_Part2C.xlsx', engine='xlsxwriter')

# Convert the dataframe to an XlsxWriter Excel object.
data.to_excel(writer, sheet_name='test_all')


# Close the Pandas Excel writer and output the Excel file.
writer.save()
```

In [ ]:

In [2]:

In [3]:

In [4]:

In [5]:

```
CPU times: user 61.9 ms, sys: 15.5 ms, total: 77.4 ms
Wall time: 76.1 ms
```

In [6]:

```
CPU times: user 350 ms, sys: 286 ms, total: 636 ms
Wall time: 629 ms
```

In [7]:

```
CPU times: user 70.5 ms, sys: 36.6 ms, total: 107 ms
Wall time: 89 ms
```

In [8]:

```
5
CPU times: user 2.76 s, sys: 121 ms, total: 2.88 s
Wall time: 2.9 s
```

In [9]:

```
CPU times: user 8.72 s, sys: 1.18 s, total: 9.91 s
Wall time: 9.85 s
```

In [10]:

In [ ]: