Project Development

Delivery of sprint-3

Date	09 November 2022
TeamID	PNT2022TMID49530
ProjectName	IOT Based Smart Crop Protection for Agriculture

Sprint-3 coding

Crop protection

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-


import time
import string # helps parse strings
#from datetime import datetime
from ph_read import Ezo # import class
for pH
from moist_read import Chirp # import
class for soil moist & temp
from weather_read import SHT31 # import
class for weather Temp & Humidity
# from store_data import Store_data #
import class to store data in a txt file

#----------------CLASSES DEFINITIONS----------
-----------------
device_ph = Ezo() #class to call pH sensor
device_moist_temp = Chirp(1, 0x20) #class
to call soil moist & temp sensor
device_weather = SHT31() # class to call
weather temp and humidity
# store = Store_data()

#---------pH_EZO COMMANDS----------
#cal,? TO CALIBRATE
#t,? TO SET TEMPERATURE FOR
CALIBRATION
```

```
#slope,? TO SEE DIFFERENCE FROM IDEAL
 pH CALIBRATION SLOPE IN ACID AND
ALKALINE
#status,? SHOWS STATUS FROM EZO
#r READS pH
#c CONTINOUSLY READ pH
 """
 print()
 print(" SOIL CONDITIONS")
 print(" pH: %s" %
device_ph.query('R'))
 print(" Moist: %s %%" %
(device_moist_temp.moist()))
 print(" Temp: %s °C\n" %
(device_moist_temp.temp())) # to obtain
soil moist

 print(" WEATHER CONDITIONS")  temp,
humi = device_weather.sht31()  print("
Temp: %s °C" % (temp))
  print(" Humidity: %s %%" % (humi))
 """




### ==================== FUNCTIONS
====================== ###

# agrimodule suggest the crops suitable
 based on the pH of soil
def
step2_agrimodule_suggesting():
print
 print (" -----------STEP 2-----------")  print ("
AgriModule suggest for a soil pH  of %s" %
(ph_soil))
 print (" to grow any crop in the list
 below:")
 print
 crops_suggested = []
 count_crop_suggested = 0
 for crop, data in crops.items():

 if(ph_soil >= data['ph-min'] and ph_soil  <=
data['ph-max']):
 count_crop_suggested += 1
crops_suggested.append(crop)  #print ("
%s: pH-range between (%s  - %s) with an
optimum of %s" %
```

```
(crop.upper(), data['ph-min'], data['ph
max'], data['ph-opt'] )

 while True:
 print (" Number of crops suggested
are: %s" % (count_crop_suggested))
 print
 for crop in crops_suggested:
 print (" %s: pH-range between (%s - %s)
with an optimum of %s" %
(crop.upper(), crops[crop]['ph-min'],
crops[crop]['ph-max'], crops[crop]['ph-opt']
))

 print
 print (" Choose a suggested crop:")
print
 crop_chosen = input(" I want to grow:  ")

 if crop_chosen in crops_suggested:
crop_chosen_index =
crops_suggested.index(crop_chosen)
else:
 print
 print (" ----------------------------------
------")
 print (" Your choosing does not
 match the suggestions")
 print (" ----------------------------------
------")
 continue
 break
 return crop_chosen



# user chooses their crop of chocie to grow
within the list of agrimodule database def
step2_user_choosing():

 while True:
 print
 print (" -----------STEP 2-----------")
print (" What are you currently
growing?")
 print (" Choose from the list below.")
print
 print (' CROPS LIST')
 print
 for crop, data in crops.items(): # to  print
all crops available in the agri database  in a
```

```
list for user to see which one can select
print (' %s' % (crop))
 print
 crop_currently_growing = input(' I am
growing: ') # ask user to select a crop  if
crop_currently_growing in
crops.keys(): # if the crop selected by the
user match one in our database then
print (' ASK HERE FOR HOW MANY  DAYS
AGO WAS PLANTED ETC')
 return crop_currently_growing #  return
the crop selected by the user  else: #if the
crop selected by the user  does not match
in our database
 print
 print (" ----------------------------------
------")
 print (" Your choosing does not
match the suggestions")
 print (" ----------------------------------
------")
 continue # continue inside this loop  until
user selects one from our database


### ==================== END
FUNCTIONS ======================
###



### ==================== AGRI
DATABASE ====================== ###

pump = {'energy':360, 'qmax-lph':440,
'qmax-lpm':7.33, 'qmax-lps':0.12, 'h
max':0.75, }

crops = {
"tomato" : {'moist':20, 'water':1.5,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':17, 'temp
max':28, 'temp-opt':25 },
"lettuce" : {'moist':25, 'water':2.5,
'yield':626, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.2, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':10, 'temp
max':19, 'temp-opt':16 },
"arugula" : {'moist':20, 'water':3.5,
'yield':671, 'dtm':180, 'dtg':12, 'ph-min':6,
```

'ph-max':6.8, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':10, 'temp
max':26, 'temp-opt':18 },
"radicchio" : {'moist':35, 'water':4.5,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':5.5,
'ph-max':6.8, 'ph-opt':6.2, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':7, 'temp-
max':24, 'temp-opt':17 },
"bell_pepper" : {'moist':35, 'water':5.5,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':16, 'temp
max':25, 'temp-opt':20 },
"cabbage" : {'moist':45, 'water':6.5,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':7.2, 'ph-opt':6.6, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':15, 'temp
max':20, 'temp-opt':18 },
"coriander" : {'moist':55, 'water':7.5,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':7.5, 'ph-opt':6.7, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':18, 'temp
max':25, 'temp-opt':23 },
"basil" : {'moist':65, 'water':8.5, 'yield':2.8,
'dtm':180, 'dtg':12, 'ph-min':5.5,
'ph-max':7, 'ph-opt':7, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':18, 'temp
max':30, 'temp-opt':22 },
"squash" : {'moist':75, 'water':9.5,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':7.5, 'ph-opt':6.5, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':15, 'temp
max':32, 'temp-opt':28 },
"chive" : {'moist':60, 'water':7.0,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':7, 'temp
max':35, 'temp-opt':26 },
"sweet_corn" : {'moist':25, 'water':2.0,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':5.5,
'ph-max':7, 'ph-opt':6.3, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':16, 'temp
max':35, 'temp-opt':27 },
"black_bean" : {'moist':25, 'water':3.0,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':5.8,
'ph-max':7, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':21, 'temp
max':27, 'temp-opt':25 },
"strawberry" : {'moist':25, 'water':4.0,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':7, 'ph-opt':7, 'rh-min':60, 'rh

```
max':70, 'rh-opt':65, 'temp-min':12, 'temp
max':25, 'temp-opt':23 },

"tomato2" : {'moist':25, 'water':1.9,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6.5,
'ph-max':7.5, 'ph-opt':7, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':17, 'temp
max':28, 'temp-opt':25 },
"rice" : {'moist':25, 'water':2.9,  'yield':2.8,
'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':17, 'temp
max':28, 'temp-opt':25 },
"chilli" : {'moist':25, 'water':3.9,  'yield':2.8,
'dtm':180, 'dtg':12, 'ph-min':6.5,
'ph-max':7.5, 'ph-opt':7, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':17, 'temp
max':28, 'temp-opt':25 },
"cottom" : {'moist':25, 'water':4.9,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':17, 'temp
max':28, 'temp-opt':25 }

}

### ==================== END AGRI
DATABASE ====================== ###


### ==================== MAIN
PROGRAMM ======================
###
### ====================
INTRODUCTION
====================== ###


print
print ("
---------------------------------------------- ")
print ("
---------------------------------------------- ")
print
print (" WELCOME TO AGRIMODULE; YOUR
FARMING ASSISTANT.")
print
print ("
---------------------------------------------- ")
print ("
---------------------------------------------- ")
```

```
print
input()
print (' AgriModule will walk you through
to')
print (' succesfully set up your
farm') print
input()
print (' BEFORE WE BEGIN:')
print (' 1: Make sure your pump is
connected to AgriModule')
print (' 2: Make sure the sensors are
connected to AgriModule')
print
input()

### ==================== END
INTRODUCTION
====================== ###


### ==================== STEP 0
====================== ###
#ask for personal information: farm name,
location (country, city)

print
print (" ----------STEP 0----------")
print (" Please name your Farm")
farm_name = input(' ') + '\'s farm' print ('
In which city is %s' % (farm_name))
farm_city = input(' ')
print
print (' THANKS!')
print (' Now we will proceed to Set Up %s
in %s' % (farm_name, farm_city))
#time.sleep(3)
print (' Just follow next 3 steps to begin')
input()
### ==================== END STEP 0
====================== ###




### ==================== STEP 1
====================== ###
#insert the sensors in the soil

print
```

```
print (" ----------STEP 1----------")
print (" Please insert the sensors into the
soil and")
print (" Wait 5 minutes for sensors to
adapt to soil conditions")
print
input()
#ph_soil = device_ph.query('R')
ph_soil = 5.6 # comment line later when
sensor is reading the real ph value to
avaluate
print (" The pH of your soil is: %s" %
(ph_soil))

### ==================== END STEP 1
====================== ###



### ==================== STEP 2
====================== ###
# to check either if user wants to grow
their own or get suhggestions from
agrimodule

while True:
 print
 print (" ----------STEP 2----------")
print (" Are you currently growing
something in this soil?")
 print
 step2_input = input(' Yes or No? ')

 if (step2_input.upper().startswith('Y')):
crop_chosen = step2_user_choosing()  elif
(step2_input.upper().startswith('N')):
crop_chosen =
step2_agrimodule_suggesting()
 else:
 print (' ATTENTION: please, choose  Yes
or NO')
 continue
 break

### ==================== END STEP 2
====================== ###


### ==================== STEP 3
====================== ###
```

```
#check how many plants or how big is the
space for then to grow
print ('add to database the space per plant
needed')


### ==================== END STEP 3
====================== ###

input()

#print (" The crop chosen is %s for a soil pH
of %s:" % (ph_soil, crop_chosen.upper())




### ==================== CROP
CHOSEN DATA ======================
###

#Calculate Water & Energy Consumption
water_daily = crops[crop_chosen]['water']
water_cycle = ( crops[crop_chosen]['dtg'] +
crops[crop_chosen]['dtm'] ) * ( water_daily
)
energy_daily = ( water_daily /
pump['qmax-lps'] ) * ( pump['energy'] /
3600 )
energy_daily1 = ( water_daily /
pump['qmax-lps'] )
energy_daily2 = ( float(pump['energy']) /
3600 )
#print pump['energy']
#print energy_daily
#print energy_daily1
#print energy_daily2
energy_cycle = ( crops[crop_chosen]['dtg']
+ crops[crop_chosen]['dtm'] ) * (
energy_daily )
#print energy_cycle


#
==================================
= ================


print
print (" You have chosen to grow %s" %
```

```
(crop_chosen.upper()))
print (" in a soil with a pH of %s" %
(ph_soil))
print
print (" ----------------------------")
print (" PRODUCTION INFORMATION")
print (" Yield per plant: %s kg" %
(crops[crop_chosen]['yield']))
print (" Days for germination: %s days" %
(crops[crop_chosen]['dtg']))
print (" Days for harvesting: %s days" %
(crops[crop_chosen]['dtm']))

print
print (" ----------------------------")
print (" PUMP INFORMATION")
print (" Water per day: %s l" %
(crops[crop_chosen]['water']))
print (" Water per cycle: %s l" %
(water_cycle))
print (" Energy per day: %s kWh" %
(energy_daily))
print (" Energy per cycle: %s kWh" %
(energy_cycle))

print
print (" ----------------------------")
print (" CROP INFORMATION")
print (" pH minimum: %s" %
(crops[crop_chosen]['ph-min']))
print (" pH optimum: %s" %
(crops[crop_chosen]['ph-opt']))
print (" pH maximun: %s" %
(crops[crop_chosen]['ph-max']))
print
print (" Temp minimum: %s" %
(crops[crop_chosen]['temp-min']))
print (" Temp optimum: %s" %
(crops[crop_chosen]['temp-opt']))
print (" Temp maximun: %s" %
(crops[crop_chosen]['temp-max']))
print
print (" RH minimum: %s " %
(crops[crop_chosen]['rh-min']))
print (" RH optimum: %s " %
(crops[crop_chosen]['rh-opt']))
print (" RH maximun: %s " %
(crops[crop_chosen]['rh-max']))
print
```

```
### ==================== END CROP
CHOSEN DATA ======================
###
#--------------------START THE SYSTEM

start_command = input('Start y/n:

')

 print
 print
 ('-----------------------------------------------')
 print
 ('-----------------------------------------------')
 print (' SYSTEM STARTED') print
 ('-----------------------------------------------')
 print
 ('-----------------------------------------------')
 print

while True:
 print()
 print(" SOIL CONDITIONS")
 print(" pH: %s" %
device_ph.query('R'))
 print(" Moist: %s %%" %
(device_moist_temp.moist()))
 print(" Temp: %s °C\n" %
(device_moist_temp.temp())) # to obtain
soil moist

 print(" WEATHER CONDITIONS")  temp,
humi = device_weather.sht31()  print("
Temp: %s °C" % (temp))  print("
Humidity: %s %%" % (humi))  continue


 n = 0
 if(start_command.upper().startswith('Y'))
 :  while True:
 #check moisture levels
 # initialization
 n += 1
 print ('Store Times: %s' % (n))  save_data
= device_ph.query('R') +' '+  date_time
 print (save_data)


 '''
```

```
 #date_time = time.ctime() # gets
current time
 #print (date_time)

 #store soil ph
 save_data = date_time
#device_ph.query('R') #+' '+
date_time)  filename =
'soil_ph_data.txt'
store.store_data(filename,save_data,'a')



 #store soil moist
 store =
str(round(device_moist_temp.moist(),2) +'
'+ date_time)
 filename = 'soil_moist_data.txt'
store.store_data(filename,store,'a')
print store

 #store soil temp
 store =
str(round(device_moist_temp.temp(),2) +'
'+ date_time)
 filename = 'soil_temp_data.txt'
store.store_data(filename,store,'a')
print store

 #gets separetly temp and humidity  from
class and stores in 2 variables  temp, humi
= device_weather.sht31()   #store weather
temp
 store = str(round(temp,2) +' '+
date_time)
 filename = 'weather_temp_data.txt'
store.store_data(filename,store,'a')
print store

 #store weather humidity
 store = str(round(humi,2) +' '+
date_time)
   filename   =   'weather_humi_data.txt'
store.store_data(filename,store,'a')
print store
 '''
 #time of sleep
 time.sleep(0.5)

 continue
```

```
"""
print
print (" SOIL CONDITIONS")
print (" pH: %s" %
(device_ph.query('R')) # to obtain pH value
from sensor
print (" Moist: %s %%" %
(round(device_moist_temp.moist(),2)) # to
obtain soil temp value from sensor
print (" Temp: %s °C\n" %
(round(device_moist_temp.temp(),2)) # to
obtain soil moist value from sensor
CALIBRATED
print (" WEATHER CONDITIONS"
temp, humi = device_weather.sht31()
print (" Temp: %s °C" %
(round(temp,2))
print (" Humidity: %s %%\n" %
(round(humi,2))
"""
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-


import time
import string # helps parse strings #from
datetime import datetime from ph_read
import Ezo # import class  for pH
from moist_read import Chirp # import
class for soil moist & temp
from weather_read import SHT31 # import
class for weather Temp & Humidity # from
store_data import Store_data #  import
class to store data in a txt file
```

```
#----------------CLASSES
DEFINITIONS---------- ----------
device_ph = Ezo() #class to call pH sensor
device_moist_temp = Chirp(1, 0x20) #class
to call soil moist & temp sensor
device_weather = SHT31() # class to call
weather temp and humidity
# store = Store_data()

#---------pH_EZO
COMMANDS---------- #cal,? TO
CALIBRATE
#t,? TO SET TEMPERATURE FOR
CALIBRATION
#slope,? TO SEE DIFFERENCE FROM IDEAL
pH CALIBRATION SLOPE IN ACID AND
ALKALINE
#status,? SHOWS STATUS FROM EZO
#r READS pH
#c CONTINOUSLY READ pH
"""
 print()
 print(" SOIL CONDITIONS")
 print(" pH: %s" %
device_ph.query('R'))
 print(" Moist: %s %%" %
(device_moist_temp.moist()))
 print(" Temp: %s °C\n" %
(device_moist_temp.temp())) # to obtain
soil moist

 print(" WEATHER CONDITIONS")  temp,
humi = device_weather.sht31()  print("
Temp: %s °C" % (temp))  print("
Humidity: %s %%" % (humi))
"""




### ==================== FUNCTIONS
====================== ###

# agrimodule suggest the crops suitable
 based on the pH of soil
def
step2_agrimodule_suggesting():
print
 print (" -----------STEP 2-----------")  print ("
AgriModule suggest for a soil pH  of %s" %
```

```
(ph_soil))
 print (" to grow any crop in the list
 below:")
 print
 crops_suggested = []
 count_crop_suggested = 0
 for crop, data in crops.items():

 if(ph_soil >= data['ph-min'] and ph_soil  <=
data['ph-max']):
 count_crop_suggested += 1
crops_suggested.append(crop)  #print ("
%s: pH-range between (%s  - %s) with an
optimum of %s" %
(crop.upper(), data['ph-min'], data['ph
 max'], data['ph-opt'] )

 while True:
 print (" Number of crops suggested
are: %s" % (count_crop_suggested))
print
 for crop in crops_suggested:
 print (" %s: pH-range between (%s - %s)
with an optimum of %s" %
(crop.upper(), crops[crop]['ph-min'],
crops[crop]['ph-max'], crops[crop]['ph-opt']
))

 print
 print (" Choose a suggested crop:")
print
 crop_chosen = input(" I want to grow:  ")

 if crop_chosen in crops_suggested:
crop_chosen_index =
crops_suggested.index(crop_chosen)
else:
 print
 print (" ----------------------------------
------")
 print (" Your choosing does not
 match the suggestions")
 print (" ----------------------------------
------")
 continue
 break
 return crop_chosen
```

```
# user chooses their crop of chocie to grow
within the list of agrimodule database def
step2_user_choosing():

 while True:
 print
 print (" -----------STEP 2-----------")
print (" What are you currently
growing?")
 print (" Choose from the list below.")
print
 print (' CROPS LIST')
 print
 for crop, data in crops.items(): # to  print
 all crops available in the agri database
 in a list for user to see which one can
select  print (' %s' % (crop))
 print
 crop_currently_growing = input(' I am
growing: ') # ask user to select a crop  if
crop_currently_growing in  crops.keys(): #
if the crop selected by the  user match one
in our database then  print (' ASK HERE
FOR HOW MANY  DAYS AGO WAS PLANTED
ETC')
 return crop_currently_growing #  return
the crop selected by the user  else: #if the
crop selected by the user  does not match
in our database
 print
 print (" ----------------------------------
------")
 print (" Your choosing does not
 match the suggestions")
 print (" ----------------------------------
------")
 continue # continue inside this loop  until
 user selects one from our database


### ==================== END
FUNCTIONS ======================
###



### ==================== AGRI
 DATABASE ====================== ###
```

```
pump = {'energy':360, 'qmax-lph':440,
'qmax-lpm':7.33, 'qmax-lps':0.12, 'h
max':0.75, }

crops = {
"tomato" : {'moist':20, 'water':1.5,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':17, 'temp
max':28, 'temp-opt':25 },
"lettuce" : {'moist':25, 'water':2.5,
'yield':626, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.2, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':10, 'temp
max':19, 'temp-opt':16 },
"arugula" : {'moist':20, 'water':3.5,
'yield':671, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.4, 'rh-min':60,
'rh-
max':70, 'rh-opt':65, 'temp-min':10, 'temp
max':26, 'temp-opt':18 },
"radicchio" : {'moist':35, 'water':4.5,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':5.5,
'ph-max':6.8, 'ph-opt':6.2, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':7, 'temp
max':24, 'temp-opt':17 },
"bell_pepper" : {'moist':35, 'water':5.5,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':16, 'temp
max':25, 'temp-opt':20 },
"cabbage" : {'moist':45, 'water':6.5,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':7.2, 'ph-opt':6.6, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':15, 'temp
max':20, 'temp-opt':18 },
"coriander" : {'moist':55, 'water':7.5,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':7.5, 'ph-opt':6.7, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':18, 'temp
max':25, 'temp-opt':23 },
"basil" : {'moist':65, 'water':8.5,  'yield':2.8,
'dtm':180, 'dtg':12, 'ph-min':5.5,
'ph-max':7, 'ph-opt':7, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':18, 'temp
max':30, 'temp-opt':22 },
"squash" : {'moist':75, 'water':9.5,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':7.5, 'ph-opt':6.5, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':15, 'temp
```

```
max':32, 'temp-opt':28 },
"chive" : {'moist':60, 'water':7.0,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':7, 'temp
max':35, 'temp-opt':26 },
"sweet_corn" : {'moist':25, 'water':2.0,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':5.5,
'ph-max':7, 'ph-opt':6.3, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':16, 'temp
max':35, 'temp-opt':27 },
"black_bean" : {'moist':25, 'water':3.0,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':5.8,
'ph-max':7, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':21, 'temp
max':27, 'temp-opt':25 },
"strawberry" : {'moist':25, 'water':4.0,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':7, 'ph-opt':7, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':12, 'temp
max':25, 'temp-opt':23 },
"tomato2" : {'moist':25, 'water':1.9,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6.5,
'ph-max':7.5, 'ph-opt':7, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':17, 'temp
max':28, 'temp-opt':25 },
"rice" : {'moist':25, 'water':2.9,  'yield':2.8,
'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':17, 'temp
max':28, 'temp-opt':25 },
"chilli" : {'moist':25, 'water':3.9,  'yield':2.8,
'dtm':180, 'dtg':12, 'ph-min':6.5,
'ph-max':7.5, 'ph-opt':7, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':17, 'temp
max':28, 'temp-opt':25 },
"cottom" : {'moist':25, 'water':4.9,
'yield':2.8, 'dtm':180, 'dtg':12, 'ph-min':6,
'ph-max':6.8, 'ph-opt':6.4, 'rh-min':60, 'rh
max':70, 'rh-opt':65, 'temp-min':17, 'temp
max':28, 'temp-opt':25 }

}

### ==================== END AGRI
DATABASE ====================== ###


### ==================== MAIN
PROGRAMM ======================
###
```

```
### ====================
INTRODUCTION
====================== ###


print
print ("
---------------------------------------------- ")
print ("
---------------------------------------------- ")
print
print (" WELCOME TO AGRIMODULE; YOUR
FARMING ASSISTANT.")
print
print ("
---------------------------------------------- ")
print ("
---------------------------------------------- ")
print
input()
print (' AgriModule will walk you through
to')
print (' succesfully set up your
farm') print
input()
print (' BEFORE WE BEGIN:')
print (' 1: Make sure your pump is
connected to AgriModule')
print (' 2: Make sure the sensors are
connected to AgriModule')
print
input()

### ==================== END
INTRODUCTION
====================== ###


### ==================== STEP 0
====================== ###
#ask for personal information: farm name,
location (country, city)

print
print (" ----------STEP 0----------")
print (" Please name your Farm")
farm_name = input(' ') + '\'s farm' print ('
In which city is %s' % (farm_name))
farm_city = input(' ')
print
```

```
print (' THANKS!')
print (' Now we will proceed to Set Up %s
in %s' % (farm_name, farm_city))
#time.sleep(3)
print (' Just follow next 3 steps to begin')
input()
### ==================== END STEP 0
====================== ###




### ==================== STEP 1
====================== ###
#insert the sensors in the soil

print
print (" ----------STEP 1----------")
print (" Please insert the sensors into the
soil and")
print (" Wait 5 minutes for sensors to
adapt to soil conditions")
print
input()
#ph_soil = device_ph.query('R')
ph_soil = 5.6 # comment line later when
sensor is reading the real ph value to
avaluate
print (" The pH of your soil is: %s" %
(ph_soil))

### ==================== END STEP 1
====================== ###



### ==================== STEP 2
====================== ###
# to check either if user wants to grow
their own or get suhggestions from
agrimodule

while True:
 print
 print (" ----------STEP 2----------")
print (" Are you currently growing
something in this soil?")
 print
 step2_input = input(' Yes or No? ')
```

```
 if (step2_input.upper().startswith('Y')):
crop_chosen = step2_user_choosing()  elif
(step2_input.upper().startswith('N')):
crop_chosen =
step2_agrimodule_suggesting()
 else:
 print (' ATTENTION: please, choose  Yes
or NO')
 continue
 break

### ==================== END STEP 2
====================== ###


### ==================== STEP 3
====================== ###
#check how many plants or how big is the
space for then to grow
 print ('add to database the space per plant
 needed')
### ==================== END STEP 3
====================== ###

 input()




#print (" The crop chosen is %s for a soil pH
of %s:" % (ph_soil, crop_chosen.upper())




### ==================== CROP
CHOSEN DATA ======================
###

#Calculate Water & Energy Consumption
water_daily = crops[crop_chosen]['water']
water_cycle = ( crops[crop_chosen]['dtg'] +
crops[crop_chosen]['dtm'] ) * ( water_daily
)
energy_daily = ( water_daily /
 pump['qmax-lps'] ) * ( pump['energy'] /
```

```
3600 )
energy_daily1 = ( water_daily /
pump['qmax-lps'] )
energy_daily2 = ( float(pump['energy']) /
3600 )
#print pump['energy']
#print energy_daily
#print energy_daily1
#print energy_daily2
energy_cycle = ( crops[crop_chosen]['dtg']
+ crops[crop_chosen]['dtm'] ) * (
energy_daily )
#print energy_cycle


#
==================================
= ================


print
print (" You have chosen to grow %s" %
(crop_chosen.upper()))
print (" in a soil with a pH of %s" %
(ph_soil))
print



print (" ----------------------------")
print (" PRODUCTION INFORMATION")
print (" Yield per plant: %s kg" %
(crops[crop_chosen]['yield']))
print (" Days for germination: %s days" %
(crops[crop_chosen]['dtg']))
print (" Days for harvesting: %s days" %
(crops[crop_chosen]['dtm']))

print
print (" ----------------------------")
print (" PUMP INFORMATION")
print (" Water per day: %s l" %
(crops[crop_chosen]['water']))
print (" Water per cycle: %s l" %
(water_cycle))
print (" Energy per day: %s kWh" %
(energy_daily))
print (" Energy per cycle: %s kWh" %
(energy_cycle))

print
```

```
print (" ----------------------------")
print (" CROP INFORMATION")
print (" pH minimum: %s" %
(crops[crop_chosen]['ph-min']))
print (" pH optimum: %s" %
(crops[crop_chosen]['ph-opt']))
print (" pH maximun: %s" %
(crops[crop_chosen]['ph-max']))
print
print (" Temp minimum: %s" %
(crops[crop_chosen]['temp-min']))
print (" Temp optimum: %s" %
(crops[crop_chosen]['temp-opt']))
print (" Temp maximun: %s" %
(crops[crop_chosen]['temp-max']))
print
print (" RH minimum: %s " %
(crops[crop_chosen]['rh-min']))
print (" RH optimum: %s " %
(crops[crop_chosen]['rh-opt']))
print (" RH maximun: %s " %
(crops[crop_chosen]['rh-max']))
print
### ==================== END CROP
CHOSEN DATA ======================
###


#--------------------START THE SYSTEM

start_command = input('Start y/n:

')

print
print
('---------------------------------------------')
print
('---------------------------------------------')
print (' SYSTEM STARTED') print
('---------------------------------------------')
print
('---------------------------------------------')
print

while True:
 print()
 print(" SOIL CONDITIONS")
 print(" pH: %s" %
device_ph.query('R'))
 print(" Moist: %s %%" %
```

```
(device_moist_temp.moist()))
 print(" Temp: %s °C\n" %
(device_moist_temp.temp())) # to obtain
soil moist

 print(" WEATHER CONDITIONS")  temp,
humi = device_weather.sht31()  print("
Temp: %s °C" % (temp))  print("
Humidity: %s %%" % (humi))  continue


n = 0
if(start_command.upper().startswith('Y'))
:  while True:
 #check moisture levels
 # initialization
 n += 1
 print ('Store Times: %s' % (n))  save_data
= device_ph.query('R') +' '+  date_time
 print (save_data)


 '''
 #date_time = time.ctime() # gets
current time
 #print (date_time)

 #store soil ph
 save_data = date_time
#device_ph.query('R') #+' '+
date_time)  filename =
'soil_ph_data.txt'
store.store_data(filename,save_data,'a')



 #store soil moist
 store =
str(round(device_moist_temp.moist(),2) +'
'+ date_time)
 filename = 'soil_moist_data.txt'
store.store_data(filename,store,'a')
print store

 #store soil temp
 store =
str(round(device_moist_temp.temp(),2) +'
'+ date_time)
 filename = 'soil_temp_data.txt'
store.store_data(filename,store,'a')
```

```
print store

 #gets separetly temp and humidity  from
class and stores in 2 variables  temp, humi
= device_weather.sht31()   #store weather
temp
 store = str(round(temp,2) +' '+
date_time)
 filename = 'weather_temp_data.txt'
store.store_data(filename,store,'a')
print store

 #store weather humidity
 store = str(round(humi,2) +' '+
date_time)
   filename   =   'weather_humi_data.txt'
store.store_data(filename,store,'a')
print store
 '''
 #time of sleep
 time.sleep(0.5)

 continue


"""
print
print (" SOIL CONDITIONS")
print (" pH: %s" %
(device_ph.query('R')) # to obtain pH value
from sensor
print (" Moist: %s %%" %
(round(device_moist_temp.moist(),2)) # to
obtain soil temp value from sensor print ("
Temp: %s °C\n" %
(round(device_moist_temp.temp(),2)) # to
obtain soil moist value from sensor
CALIBRATED
print (" WEATHER CONDITIONS"
temp, humi = device_weather.sht31()
print (" Temp: %s °C" %
(round(temp,2))
print (" Humidity: %s %%\n" %
(round(humi,2))
"""
```