دانشگاه آزاد اسلامی واحد علوم و تحقیقات
گروه علوم ومهندسی آب

# دوره کارورزی کارشناسی

## عنوان پروژه کارورزی : بهینه سازی آب آبیاری گیاه پسته با پایتون

نام دانشجو: محمدهادی پی‌سپار

نام استاد : دکتر مهدی سرائی تبریزی

تاریخ انجام دوره کارورزی : از ۲۱ فروردین ۱۴۰۲ الی ۱۰ مرداد ۱۴۰۲

فهرست عناوین :

جدول زمانبندی اعضای کارورزی (دکتر مهدی سرائی تبریزی)

نام و نام خانوادگی: محمدهادی پی‌سپار

| روز و تاریخ | مدت زمان | ملاحظات(فعالیت روزانه) | امضا |
|---|---|---|---|
| ۱۴۰۱/۰۲/۱۳ | ۴ ساعت | انجام پروژه | |
| ۱۴۰۱/۰۲/۲۰ | ۴ ساعت | انجام پروژه | |
| ۱۴۰۱/۰۲/۲۷ | ۴ ساعت | انجام پروژه | |
| ۱۴۰۱/۰۳/۱۳ | ۴ ساعت | انجام پروژه | |
| ۱۴۰۱/۰۳/۱۰ | ۴ ساعت | انجام پروژه | |
| ۱۴۰۱/۰۳/۱۷ | ۴ ساعت | انجام پروژه | |
| ۱۴۰۱/۰۴/۱۴ | ۴ ساعت | انجام پروژه | |
| ۱۴۰۱/۰۴/۲۱ | ۴ ساعت | انجام پروژه | |
| ۱۴۰۱/۰۴/۲۸ | ۴ ساعت | تهیه گزارش | |
| ۱۴۰۱/۰۵/۰۴ | ۴ ساعت | تهیه گزارش | |

محل امضا استاد :

## چکیده

این سند، مستندی در مورد کد Python برای بهینه‌سازی مصرف آبیاری در کشت پسته است. این کد اطلاعات مربوط به گیاه و داده‌های آبیاری را از یک فایل اکسل می‌خواند و با محاسبات مختلف، برنامه زمان‌بندی آبیاری بهینه را تعیین می‌کند. هدف از بهینه‌سازی، حداکثر کردن عملکرد محصول مورد انتظار و کاهش مصرف آب است .

کتابخانه‌های مورد نیاز شامل پانداس برای تیمار داده‌ها، math.prodبرای محاسبه حاصل‌ضرب، tabulateبرای نمایش نتایج به صورت جدولی و scipy.optimize.minimize برای بهینه‌سازی عددی هستند .

سند شامل توابعی مانند خواندن مقادیر سلول‌ها، محاسبه عملکرد مورد انتظار محصول و حداکثر کردن بهره‌وری از داده‌های تاریخی است. همچنین، کار با DataFrame برای محاسبات و برنامه‌ریزی آبیاری نیز انجام می‌شود .

با استفاده از این کد، می‌توان مصرف آبیاری را برای کشت پسته بهینه‌سازی کرده و با در نظر گرفتن عوامل مختلف مانند عملکرد مورد انتظار محصول، نسبت آب و خاک و داده‌های تاریخی محصول، تصمیم‌گیری درباره مدیریت مناسب آبیاری انجام داد. نتایج به صورت جدولی نمایش داده می‌شوند تا تفسیر و تصمیم‌گیری آسان‌تر باشد .

**مقدمه :**

هدف از کد Python ارائه شده بهینه‌سازی استفاده از آبیاری برای کشت پسته است. این کد اطلاعات گیاه و داده‌های آبیاری را از یک فایل اکسل می‌خواند و محاسبات مختلفی را انجام می‌دهد تا برنامه زمان‌بندی آبیاری بهینه را تعیین کند. این بهینه‌سازی بر اساس حداکثر کردن عملکرد مورد انتظار محصول و حداقل کردن مصرف آب انجام می‌شود .

## کتابخانه های مورد نیاز :

**پانداس:** پانداس یک کتابخانه محبوب برای تیمار داده‌ها در Python است. این کتابخانه ساختارهای داده‌ای مانند DataFrame و Series و توابع برای خواندن، نوشتن و تیمار داده‌ها را به صورت کارآمد ارائه می‌دهد .

**پروداکت:** تابع prod() از ماژول math حاصلضرب تمام عناصر یک داده‌ی قابل تکرار را محاسبه می‌کند .

**تبولیت:** کتابخانه tabulate برای نمایش نتایج نهایی به صورت جدولی برای بهبود خوانایی استفاده می‌شود .

**مینیمایز:** تابع scipy.optimize.minimize برای بهینه‌سازی عددی استفاده می‌شود. سعی می‌کند حداقل یک تابع اسکالر را با تنظیم مقادیر ورودی به صورت تکراری پیدا کند .

## توابع :

۱- read_input(cellName) :

این تابع مقدار مرتبط با نام سلول داده‌شده را از برگه "ورودی" در فایل اکسل می‌خواند .

پارامترها - cellName (str) : نام سلول برای خواندن مقدار از آن .

خروجی: مقدار مرتبط با نام سلول داده‌شده یا None اگر سلول پیدا نشد .

۲-get_expected_productivity():

عملکرد مورد انتظار محصول پسته را بر اساس پارامترهای ورودی مانند عملکرد مورد انتظار محصول، استحکام آب خاک و سن گیاه محاسبه می‌کند.
خروجی: عملکرد مورد انتظار محصول پسته.

۳- get_max_productivity() :
بهره‌وری حداکثر محصول را از داده‌های تاریخی در برگه "تاریخچه محصول" فایل اکسل تعیین می‌کند.
خروجی: بهره‌وری حداکثر محصول.

۴- create_dataframe():
شیت "Irrigation" را از فایل اکسل می‌خواند و یک DataFrame با ستون‌های لازم ایجاد می‌کند.
خروجی DataFrame: حاوی داده‌های آبیاری.

۵- calculate_columns(df):
توضیحات: محاسبات مختلفی را برای ایجاد ستون‌های جدید در DataFrame انجام می‌دهد. این تابع مقادیر مربوط به محاسبات رئس و برنامه زمان‌بندی آبیاری برای هر ماه را محاسبه می‌کند.
پارامترها df (DataFrame) - DataFrame: حاوی داده‌های آبیاری.
تغییرات DataFrame: ورودی df را با افزودن ستون‌های محاسبه‌شده جدید تغییر می‌دهد.

۶- calculate_objective(df):
توضیحات: تابع هدف برای بهینه‌سازی را محاسبه می‌کند. از حاصل‌ضرب محاسبات رئس و ترکیب آن با بهره‌وری مورد انتظار و حداکثر برای تعیین مقدار هدف استفاده می‌کند.
پارامترها: دیتافریم حاوی داده‌های آبیاری.
خروجی: مقدار هدف محاسبه‌شده

۷- objective_function(x, df) :

توضیحات: تابع هدف برای کمینه‌سازی در بهینه‌سازی است. برنامه زمان‌بندی آبیاری را در DataFrame بر اساس پارامترهای ورودی x به‌روزرسانی می‌کند و با استفاده از تابع calculate_objective()مقدار هدف جدید را محاسبه می‌کند .

پارامترها :

- x (array)پارامترهای ورودی برای بهینه‌سازی که برنامه زمان‌بندی آبیاری را نمایند.
- df (DataFrame) - DataFrameحاوی داده‌های آبیاری.

خروجی: مقدار مطلق مقدار هدف جدید.

۸- constraint(x) :

توضیحات: این تابع محدودیت بهینه‌سازی را تعریف می‌کند. اطمینان حاصل می‌کند که برنامه زمان‌بندی آبیاری بهینه‌سازی شده باید کمتر یا مساوی مقادیر آبیاری اصلی باشد.

پارامترها :

- پارامترهای ورودی برای بهینه‌سازی که برنامه زمان‌بندی آبیاری را نمایند.
- x_valuesمقادیر آبیاری اصلی ازDataFrame .
- y_values (array)مقادیر آبیاری بهینه‌سازی شده (زمان‌بندی).

خروجی: تفاوت بین مقادیر آبیاری اصلی و مقادیر بهینه‌سازی شده .

**خلاصه‌ی کتابخانه‌ها :**

الف : pandas

پانداس یک کتابخانه قدرتمند برای تیمار داده‌ها و تجزیه و تحلیل است. این کتابخانه ساختارهای داده‌ای مانند DataFrameو Series را ارائه می‌دهد که کار با داده‌های جدولی و سری‌های زمانی را آسان می‌کند .

پانداس از توابع خواندن، نوشتن، فیلتر کردن، ادغام و تجمیع داده‌ها پشتیبانی می‌کند .

پانداس از مدیریت داده‌های ناقص، داده‌های زمانی و انواع فرمت‌های داده‌ها از جملهCSV ، Excel، پایگاه‌های داده SQL و غیره پشتیبانی می‌کند .

ج: tabulate .

کتابخانه tabulate برای ایجاد جداول زیبا و فشرده از داده‌ها استفاده می‌شود .
این کتابخانه داده‌های ورودی، مانند لیستی از دیکشنری‌ها یا DataFrame را در یک جدول ساختاریافته قرار می‌دهد .
تابع tabulate چندین سبک خروجی) مانند"plain" ، "grid"، "pipe"، "html"و غیره (را برای سفارشی‌سازی ظاهر جدول ارائه می‌دهد .

د: scipy.optimize.minimize .

این تابع جزء کتابخانه SciPy است که برای محاسبات علمی و فنی در Python استفاده می‌شود .
تابع minimize برای بهینه‌سازی عددی، به ویژه برای پیدا کردن حداقل یک تابع اسکالر از یک یا چند متغیر استفاده می‌شود .
این تابع از روش‌های مختلف بهینه‌سازی مانندNelder-Mead ، Powell، CG، BFGSو غیره پشتیبانی می‌کند تا به صورت کارآمد حداقل را پیدا کند .

## ژوپیتر نوتبوک:

ژوپیتر نوت بوک یک برنامه (یا کتابخانه) است که با آن می توانیم به صورت تعاملی در مرورگرمان یک فایل حاوی کد، عکس و ... بسازیم و آن را در مرورگر ویرایش و اجرا کنیم.

در این پژوهش برای فهم بهتر کد و مستند سازی برای آن از این ابزار استفاده شده است.

در ادامه مشروح پروژه در خروجی ژوپیتر بررسی خواهد شد.

# Pistachio Irrigation Water Optimization

This Jupyter Notebook contains Python code that optimizes the irrigation water for pistachio cultivation. The code reads plant information and irrigation data from an Excel file, performs various calculations, and then optimizes the irrigation schedule to maximize expected crop yield while minimizing water consumption.

## Required Packages

We use following packages to optimize irrigation:

a. pandas: Pandas is a popular data manipulation library in Python. It provides data structures like DataFrame and Series, and functions for reading, writing, and manipulating data efficiently. b. math.prod: The prod() function from the math module calculates the product of all elements in an iterable.

c. tabulate: The tabulate package is used to display the final results in a tabulated format for better readability.

d. scipy.optimize.minimize: This function is used for numerical optimization. It attempts to find the minimum of a scalar function by iteratively adjusting the input parameters.

```
import pandas as pd from math
import prod from tabulate import
tabulate from scipy.optimize import
minimize
```

## Function Descriptions

The code consists of several functions that perform specific tasks. Below are the descriptions of each function:

`read_input(cellName)`

This function reads the value associated with a given cell name from the 'Input' sheet in the Excel file. It takes the cell name as input and returns the corresponding value or None if the cell is not found.

```
inputSheet = pd.read_excel('data.xlsx', sheet_name='Input',
skiprows=1, nrows=5, usecols="A:B") print(inputSheet) def
read_input(cellName):
    global inputSheet
df = inputSheet
    df.columns = ['key', 'value']
key_row = df[df['key'] == cellName]
```

```
        if not key_row.empty:
value = key_row.iloc[0]['value']
return value     else:         return
None
```

```
    Soil Water Solidity       5
0  Irrigation Frequency    40.0
1   Expected Crop Yeild  5000.0
2        Field Capacity   157.5
3                   Age    10.0
```

`get_expected_productivity()`

This function calculates the expected productivity of the pistachio plant based on input parameters such as expected crop yield, soil water solidity, and plant age. It adjusts the expected crop yield based on the soil water solidity and plant age, and then returns the expected productivity.

```
def get_expected_productivity():     expected =
read_input('Expected Crop Yeild')     soilWaterSolidity =
read_input('Soil Water Solidity')
    age = read_input('Age')     if
soilWaterSolidity and soilWaterSolidity > 7:
expected = 100 - 3.6 * (expected - 7)     if(age):
if(age < 3):
            expected   *=   .4
elif age < 6:
            expected *= .7
elif age < 9:
expected *= .9     return
expected
```

`get_max_productivity()`

This function extracts the maximum crop efficiency from the 'Crop History' sheet in the Excel file. It returns the maximum crop efficiency value.

```
cropHistory = pd.read_excel('data.xlsx', sheet_name='Crop History',
skiprows=1, nrows=9, usecols="B:B")

def get_max_productivity():
    global cropHistory
df = cropHistory
    df.columns = ['Efficiency']
    maxCropEfficiency = df['Efficiency'].max()
return maxCropEfficiency
```

`create_dataframe()`

This function reads the 'Irrigation' sheet from the Excel file and creates a DataFrame with the required columns for irrigation data.

```
def create_dataframe():     df = pd.read_excel('data.xlsx',
sheet_name='Irrigation', skiprows=1, nrows=23,
usecols="A:H")     pd.set_option('expand_frame_repr', False)
    df.columns = ['Month', 'Decade', 'Progress Stage', 'Kc', 'ETo',
'ETc', 'Effective Precipitation', 'Irrigation']
return df df = create_dataframe() print(df)
```

| | Month | Decade | Progress Stage | Kc | ETo | ETc | Effective Precipitation | Irrigation |
|---|---|---|---|---|---|---|---|---|
| 0 | Farvardin | 2 | 0.50 | 0.38 | 42.93 | 12.85 | 4.61 | 8.24 |
| 1 | Farvardin | 3 | 0.50 | 0.38 | 55.07 | 16.48 | 2.89 | 13.59 |
| 2 | Ordibehesht | 1 | 0.85 | 0.43 | 53.85 | 18.23 | 0.00 | 18.23 |
| 3 | Ordibehesht | 2 | 0.85 | 0.49 | 53.46 | 20.63 | 1.89 | 18.74 |
| 4 | Ordibehesht | 3 | 0.85 | 0.54 | 64.76 | 27.54 | 0.57 | 26.97 |
| 5 | Khordad | 1 | 0.85 | 0.60 | 64.18 | 30.33 | 0.00 | 30.33 |
| 6 | Khordad | 2 | 0.85 | 0.58 | 67.12 | 30.66 | 0.00 | 30.66 |
| 7 | Khordad | 3 | 0.85 | 0.58 | 77.65 | 35.47 | 0.00 | 35.47 |
| 8 | Tir | 1 | 0.85 | 0.58 | 72.93 | 33.31 | 0.00 | 33.31 |
| 9 | Tir | 2 | 0.85 | 0.58 | 71.82 | 32.80 | 0.00 | 32.80 |
| 10 | Tir | 3 | 0.85 | 0.58 | 77.14 | 35.23 | 0.00 | 35.23 |
| 11 | Mordad | 1 | 0.85 | 0.58 | 68.66 | 31.36 | 0.00 | 31.36 |
| 12 | Mordad | 2 | 0.60 | 0.58 | 67.66 | 30.90 | 0.45 | 30.45 |
| 13 | Mordad | 3 | 0.60 | 0.55 | 70.81 | 30.67 | 0.00 | 30.67 |
| 14 | Shahrivar | 1 | 0.60 | 0.53 | 61.15 | 25.52 | 0.00 | 25.52 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 15 | Shahrivar | 2 | 0.60 | 0.51 | 57.12 | 22.94 |
| 0.00 | 22.94 | | | | | |
| 16 | Shahrivar | 3 | 0.60 | 0.48 | 58.87 | 22.25 |
| 0.00 | 22.25 | | | | | |
| 17 | Mehr | 1 | 0.60 | 0.46 | 48.04 | 17.40 |
| 0.00 | 17.40 | | | | | |
| 18 | Mehr | 2 | 0.60 | 0.44 | 42.67 | 14.79 |

```
0.00        14.79
19          Mehr        3             0.60   0.42   37.93   12.55
0.00        12.55
20          Aban        1             0.60   0.40   31.53    9.93
1.27         8.66
21          Aban        2             0.60   0.38   20.31    6.08
0.59         5.49
```

`calculate_columns(df)`

This function performs various calculations to create new columns in the DataFrame. It calculates Raes Method1, Raes Method2, Raes Method3, Raes Method4, and irrigation scheduling for each month based on the irrigation frequency.

```
def calculate_columns(df):     df['Raes Method1'] =
df['Progress Stage'] * (df['Irrigation Scheduling'] /
df['Irrigation'])
    df['Raes Method2'] = 1 - df['Raes Method1']
df['Raes Method3'] = pow(df['Raes Method2'], 10/220)
df['Raes Method4'] = 1 - df['Raes Method3']     cycle =
read_input('Irrigation Frequency')     cycle_key =
f'Irrigation Scheduling F={cycle}'     df[cycle_key] =
None     jump = cycle / 10     for i in range(0,
len(df)):         if (i % jump) == 0:             sum =
df.loc[i : i + jump - 1, 'Irrigation Scheduling'].sum()
            capacity = read_input('Field Capacity')
df.loc[i, cycle_key] = sum if capacity > sum else capacity
```

```
          Month  Decade  Progress Stage    Kc    ETo    ETc  Effective
Precipitation  Irrigation  Irrigation Scheduling  Raes Method1  Raes
Method2  Raes Method3  Raes Method4 Irrigation Scheduling F=40.0
0     Farvardin       2            0.50  0.38  42.93  12.85
4.61        8.24                  8.24          0.50          0.50
0.968984      0.031016                          58.8
1     Farvardin       3            0.50  0.38  55.07  16.48
2.89       13.59                 13.59          0.50          0.50
0.968984      0.031016                          None
2     Ordibehesht       1            0.85  0.43  53.85  18.23
0.00       18.23                 18.23          0.85          0.15
0.917381      0.082619                          None
```

```
3           Ordibehesht        2              0.85   0.49   53.46   20.63
            1.89        18.74                    18.74           0.85
            0.15
0.917381      0.082619                          None
```

```
4           Ordibehesht         3               0.85  0.54  64.76  27.54
            0.57        26.97                     26.97           0.85
            0.15
0.917381      0.082619                        123.43
5           Khordad         1               0.85  0.60  64.18  30.33
            0.00        30.33                     30.33           0.85
            0.15
0.917381      0.082619                          None
6           Khordad         2               0.85  0.58  67.12  30.66
            0.00        30.66                     30.66           0.85
            0.15
0.917381      0.082619                          None
7           Khordad         3               0.85  0.58  77.65  35.47
            0.00        35.47                     35.47           0.85
            0.15
0.917381      0.082619                          None
8           Tir       1           0.85  0.58  72.93  33.31
0.00       33.31                   33.31          0.85          0.15
0.917381      0.082619                         132.7
9           Tir       2           0.85  0.58  71.82  32.80
0.00       32.80                   32.80          0.85          0.15
0.917381      0.082619                          None
10          Tir       3           0.85  0.58  77.14  35.23
0.00       35.23                   35.23          0.85          0.15
0.917381      0.082619                          None
11          Mordad          1               0.85  0.58  68.66  31.36
            0.00        31.36                     31.36           0.85
            0.15
0.917381      0.082619                          None
12          Mordad          2               0.60  0.58  67.66  30.90
            0.45        30.45                     30.45           0.60
            0.40
0.959206      0.040794                        109.58
13          Mordad          3               0.60  0.55  70.81  30.67
            0.00        30.67                     30.67           0.60
            0.40
0.959206      0.040794                          None
14          Shahrivar         1             0.60  0.53  61.15  25.52
            0.00        25.52                     25.52           0.60
            0.40
0.959206      0.040794                          None
15          Shahrivar         2             0.60  0.51  57.12  22.94
            0.00        22.94                     22.94           0.60
            0.40
0.959206      0.040794                          None
16          Shahrivar         3             0.60  0.48  58.87  22.25
            0.00        22.25                     22.25           0.60
            0.40
```

```
0.959206      0.040794                          66.99
17          Mehr       1            0.60  0.46  48.04  17.40
0.00       17.40                  17.40          0.60          0.40
0.959206      0.040794                          None
18          Mehr       2            0.60  0.44  42.67  14.79
0.00       14.79                  14.79          0.60          0.40
0.959206      0.040794                          None
19          Mehr       3            0.60  0.42  37.93  12.55


0.00       12.55                  12.55          0.60          0.40
0.959206      0.040794                          None

20         Aban       1            0.60  0.40  31.53   9.93

1.27        8.66                   8.66          0.60          0.40
0.959206      0.040794                         14.15

21         Aban       2            0.60  0.38  20.31   6.08

0.59        5.49                   5.49          0.60          0.40
0.959206      0.040794                          None
```

`calculate_objective(df)`

This function calculates the objective value for optimization. It uses the product of Raes Method3, the maximum crop efficiency, and the expected productivity to determine the objective value.

```
def calculate_objective(df):     product =
df['Raes Method3'].product()     max =
get_max_productivity()     expected =
get_expected_productivity()     objective =
expected - ((1 - product) * max)     return
objective
```

`objective_function(x, df)`

This function defines the objective function for optimization. It updates the irrigation scheduling in the DataFrame based on the input parameters x, calculates the objective value using calculate_objective(), and returns the absolute value of the objective.

```
def objective_function(x, df):
    df['Irrigation Scheduling'] = x
calculate_columns(df)     objective =
calculate_objective(df)     return
abs(objective)
```

`constraint(x, x_values, y_values)`

This function defines the constraint for optimization. It ensures that the optimized irrigation scheduling is less than or equal to the original irrigation values.

```
def constraint(x, x_values, y_values):
return y_values - x_values

# Perform optimization
result = minimize(objective_function, df['Irrigation
Scheduling'].values, args=(df,), constraints={'type': 'ineq', 'fun':
constraint, 'args': (df['Irrigation'].values, df['Irrigation
Scheduling'].values), }, bounds=[(0, None)]) optimized_y_values
= result.x
df['Irrigation Scheduling'] = optimized_y_values
```

```
calculate_columns(df)
optimized_objective = calculate_objective(df)

# Display results
print("Optimized Objective Value:", optimized_objective)
print("Total Irrigation Water: ", df['Irrigation Scheduling'].sum())

table = tabulate(df, headers=df.columns, tablefmt='outline')
print(table)
```

```
Optimized Objective Value: 2.7939677238464355e-09
Total Irrigation Water:  430.1229769879437
+----+-------------+----------+------------------+------+-------
+-------+----------------------------+-------------
+-------------------------+-----------------+----------------
+----------------+----------------+--------------------------------+
|    | Month       |   Decade |   Progress Stage |   Kc |   ETo |
ETc |   Effective Precipitation |   Irrigation |   Irrigation
Scheduling |   Raes Method1 |   Raes Method2 |   Raes Method3 |   Raes
Method4 |   Irrigation Scheduling F=40.0 |
+====+=============+==========+==================+======+=======+=====
==+============================+==============+=======================
=+================+================+================+================+
================================+
|  0 | Farvardin   |        2 |              0.5 | 0.38 | 42.93 |
12.85 |                       4.61 |          8.24 |
7.23254 |       0.438868 |       0.561132 |       0.974078 |
0.0259217 |                         39.9817 |
|  1 | Farvardin   |        3 |              0.5 | 0.38 | 55.07 | 16.48
|                       2.89 |         13.59 |
12.5003  |       0.459909 |       0.540091 |       0.972388 |
0.0276124 |                                |
|  2 | Ordibehesht |        1 |             0.85 | 0.43 | 53.85 |
18.23 |                        0    |         18.23 |
9.90113 |       0.461654 |       0.538346 |       0.972245 |
0.0277555 |                                |
|  3 | Ordibehesht |        2 |             0.85 | 0.49 | 53.46 | 20.63
|                       1.89 |         18.74 |
```

10.3477 | 0.469346 | 0.530654 | 0.971609 | 0.0283912 | |
| 4 | Ordibehesht | 3 | 0.85 | 0.54 | 64.76 | 27.54 | 0.57 | 26.97 | 19.4234 | 0.612157 | 0.387843 | 0.957861 | 0.0421389 | 98.9764 |
| 5 | Khordad | 1 | 0.85 | 0.6 | 64.18 | 30.33 | 0 | 30.33 | 24.148 | 0.67675 | 0.32325 | 0.949962 | 0.0500379 | |
| 6 | Khordad | 2 | 0.85 | 0.58 | 67.12 | 30.66 | 0 | 30.66 | 24.594 | 0.681831 | 0.318169 | 0.949278 | 0.0507217 | |
| 7 | Khordad | 3 | 0.85 | 0.58 | 77.65 | 35.47 | 0 | 35.47 | 30.811 | 0.738352 | 0.261648 | 0.940877 | 0.0591235 | |
| 8 | Tir | 1 | 0.85 | 0.58 | 72.93 | 33.31 | 0 | 33.31 | 28.0799 | 0.716538 | 0.283462 | 0.944307 | 0.0556926 | 111.544 |
| 9 | Tir | 2 | 0.85 | 0.58 | 71.82 | 32.8 | 0 | 32.8 | 27.4214 | 0.710616 | 0.289384 | 0.945195 | 0.0548046 | |
| 10 | Tir | 3 | 0.85 | 0.58 | 77.14 | 35.23 | 0 | 35.23 | 30.5119 | 0.736166 | 0.263834 | 0.941232 | 0.0587676 | |
| 11 | Mordad | 1 | 0.85 | 0.58 | 68.66 | 31.36 | 0 | 31.36 | 25.531 | 0.692009 | 0.307991 | 0.947876 | 0.0521235 | |
| 12 | Mordad | 2 | 0.6 | 0.58 | 67.66 | 30.9 | 0.45 | 30.45 | 29.8619 | 0.588412 | 0.411588 | 0.960452 | 0.0395482 | 106.752 |
| 13 | Mordad | 3 | 0.6 | 0.55 | 70.81 | 30.67 | 0 | 30.67 | 30.0878 | 0.58861 | 0.41139 | 0.960431 | 0.0395692 | |
| 14 | Shahrivar | 1 | 0.6 | 0.53 | 61.15 | 25.52 | 0 | 25.52 | 24.7597 | 0.582125 | 0.417875 | 0.961114 | 0.0388861 | |
| 15 | Shahrivar | 2 | 0.6 | 0.51 | 57.12 | 22.94 | 0 | 22.94 |

```
22.0426  |       0.576529 |       0.423471 |       0.961695 |
0.0383048 |                                 |
| 16 | Shahrivar   |        3 |             0.6  | 0.48 | 58.87 | 22.25
|                      0    |        22.25 |
21.3073  |       0.57458  |       0.42542  |       0.961896 |
0.0381041 |                          59.9964 |
| 17 | Mehr        |        1 |              0.6  | 0.46 | 48.04 | 17.4
|                      0    |        17.4  |                15.9625  |
0.55043  |       0.44957  |       0.964313 |      0.035687  |
|
| 18 | Mehr        |        2 |             0.6  | 0.44 | 42.67 | 14.79
|                      0    |        14.79 |
12.8456  |       0.521119 |       0.478881 |       0.967085 |
0.0329145 |                                 |
| 19 | Mehr        |        3 |              0.6  | 0.42 | 37.93 |
12.55 |                       0    |        12.55 |
9.88098 |       0.472397 |       0.527603 |       0.971354 |
0.0286459 |                                 |
| 20 | Aban        |        1 |              0.6  | 0.4  | 31.53 |
9.93 |                       1.27 |         8.66 |
7.63698 |       0.529121 |       0.470879 |       0.966345 |
0.0336549 |                          12.8722 |
| 21 | Aban        |        2 |             0.6  | 0.38 | 20.31 |
6.08 |                       0.59 |         5.49 |
5.23521 |       0.572154 |       0.427846 |       0.962145 |
0.0378554 |                                 |
+----+-------------+----------+-----------------+------+-------
+-------+---------------------------+--------------
+-------------------------+----------------+----------------
+----------------+----------------+---------------------------------+
```

## نتیجه‌گیری :

کد Python ارائه شده به طور کارآمد آبیاری پسته را بهینه‌سازی می‌کند با در نظر گرفتن عوامل مختلفی مانند عملکرد مورد انتظار محصول، استحکام آب خاک، سن گیاه و داده‌های تاریخی محصول. از تابع بهینه‌سازی کتابخانه SciPy برای پیدا کردن برنامه زمان‌بندی آبیاری بهینه استفاده می‌شود که محاسبات رئس را با بهره‌وری مورد انتظار و ماکسیمم ترکیب می‌کند. نتایج به صورت جدولی ارائه می‌شوند تا به راحتی تفسیر و تصمیم‌گیری درباره مدیریت آبیاری برای محصولات پسته امکان‌پذیر باشد.

## منابع


- https://www.scipy.org/
- https://pandas.pydata.org/docs/
- https://docs.python.org/
- https://www.python.org/
- https://www.kaggle.com/
- https://www.datacamp.com/
- https://towardsdatascience.com/
- https://realpython.com/
- Automate the Boring Stuff with Python: Practical Programming for Total BeginnersAl Sweigart, 2015
- Fluent Python
  Luciano Ramalho, 2015