

תקציר פרויקט: מערכת תכנון משימות עם אופטימיזציה של מזג אוויר, מטוסים וטייסים

סקירת הפרויקט

הפרויקט מתמקד ביצירת מערכת תכנון משימות עבור תקיפות אוויריות צבאיות. המערכת תבצע אופטימיזציה של הקצאת טייסים, מטוסים ומטרות תקיפה על בסיס גורמים כמו מרחק, תנאי מזג אוויר, יכולות המטוס ורמת המיומנות של הטייסים. המטרה היא להבטיח שמשימות בעלות עדיפות גבוהה יתבצעו בתנאים מיטביים תוך שימוש במשאבים הטובים ביותר הקיימים.

מרכיבי הפרויקט המרכזיים

1. תעדוף מטרות:
 - רשימה של 15 ערי מטרה במדינות עוינות, לכל אחת מהן מוקצה רמת עדיפות (1-5), כאשר 5 היא העדיפות הגבוהה ביותר.
2. יכולות מטוסים:
 - 7 סוגי מטוסים זמינים, לכל אחד מהם מהירות וקיבולת דלק שונים, המשפיעים על יכולתם לבצע את המשימות בהצלחה על פי המרחק מהיעד.
3. מיומנויות הטייסים:
 - 5 טייסים עם רמות מיומנות שונות (1-10). ככל שרמת המיומנות גבוהה יותר, כך הטייס מתאים יותר למשימות בתנאים קשים כמו מזג אוויר רע.
4. תנאי מזג אוויר:
 - המערכת תשקלל את תנאי מזג האוויר, כולל מהירות הרוח, עננות וגשם, ותשפיע על ניקוד המשימה בהתאם ליכולת לבצע את המשימה בתנאים אלו.
5. זמני ביצוע:
 - המערכת תחשב את הזמן האופטימלי לביצוע המשימה על בסיס תחזית מזג האוויר לחמשת הימים הקרובים, ותמליץ על הזמן הטוב ביותר לתקיפה.

מטרות הפרויקט:

1. יצירת רשימה ממוינת בצורה אופטימאלית של מטרות לתקיפה עם הקצאה (ציוות של מטוס, טייס וזמן ביצוע)
2. הדרך לייצר את הרשימה זה באמצעות כתיבת אלגוריתם (הסבר על זה בהמשך)
3. עליכם להשתמש בצורה נכונה במבני הנתונים שלמדתם במהלך השבוע
4. הערות חשובות
 - a. יש יותר מטרות ממטוסים וטייסים ולכן המיון לפי חשיבות הוא קריטי

5. **בונוס:** הקצאת טייסים, מטוסים וזמנים לביצוע למשימות אחרי הקצאה ראשונית (כלומר, אחרי שכבר ציוותנו טייסים ומטוסים למשימות הכי חשובות, נוכל לנסות להקצות אותם למשימות הבאות, אבל בזמנים מאוחרים יותר כי כבר יצאו למשימות בהתקפה הראשונה

מטרות למידה:

בנוסף ללמידת מושגי ליבה ב Python-OOP, הסטודנטים ילמדו:

- איך לעבוד עם APIs לשליפת נתונים בזמן אמת (מזג אוויר).
- ליישם עקרונות OOP לעיצוב קוד מודולרי ומדרגי.
- להשתמש במבני נתונים כמו **רשימות** ו-**מילונים** לניהול ואחסון נתוני משימה.
- ליישם **הבנת רשימות** (List Comprehensions) לעיבוד נתונים בצורה יעילה ופייתונית.
- לעבוד עם קבצים לצורך **ייבוא JSON** ו-**שמירת CSV**.
- להוסיף **עיכובים בזמן אמת** לסימולציית התקיפה.

מושגים חדשים שנבדקים:

1. אינטגרציה עם API:

- שימוש בספריית requests לשליפת נתוני מזג אוויר בזמן אמת מ API-ציבורי.

2. תכנות מונחה עצמים (OOP):

- שימוש ב Classes-לניהול מטוסים, טייסים ומטרות.

3. מבני נתונים:

- שימוש ברשימות ומילונים לניהול ואחסון נתוני משימות.

4. הבנת רשימות:

- שימוש בהבנת רשימות לסינון והצגת תוצאות של תקיפות אוויריות.

5. ניהול קבצים:

- קריאה מתוך קובצי JSON ושמירת תוצאות לקבצי CSV.

מאפיינים חדשים בפרויקט:

1. אינטגרציה עם API למזג אוויר:

- שליפת **מזג אוויר עכשווי** למיקום מסוים באמצעות API כמו [OpenWeatherMap API](#).
- שימוש בתנאי מזג האוויר שנשלפו (כגון מזג אוויר בהיר, רוח, סערה) כפרמטר שמשפיע על הסיכוי להצלחת המשימה.

2. עיצוב: OOP

- שימוש במחלקות לשמור נתונים עבור כל יישות במערכת, לדוגמא: טייס, מטוס וכו'
- מזג האוויר ישולב כפרמטר דינמי שמשפיע על הצלחת המשימה, ומבוסס על נתונים בזמן אמת שנשלפים מ-API-

3. מערכת תפריט:

- המערכת תציג תפריט שבו המשתמש יוכל לבחור פעולות שונות:

1. טעינת קבצים (מטרות וכל המידע הנוסף) מ-JSON

2. הצגת טבלת המלצה לתקיפות

3. שמירת כל התקיפות לקובץ CSV

4. יציאה

4. ניהול קבצים:

- ייבוא משימות מתוך קובץ JSON ושמירת תוצאות ל-קובץ CSV.

תצוגה רצויה להדפסה ב-CSV בהתאמה

| Target City | Priority | Assigned Pilot | Assigned Aircraft | Distance (km) | Weather Conditions | Pilot Skill | Aircraft Speed (km/h) | Fuel Capacity (km) | Mission Fit Score |
|-------------|----------|----------------|-------------------|---------------|--------------------|-------------|-----------------------|--------------------|-------------------|
| Damascus | 5 | Robert White | Fighter Jet | 216 | Clear | 10 | 1500 | 3000 | 0.92 |
| Beirut | 4 | John Doe | Bomber | 210 | Cloudy | 8 | 900 | 5000 | 0.85 |
| Baghdad | 5 | Michael Clark | Heavy Bomber | 876 | Clear | 9 | 850 | 6000 | 0.88 |
| Tehran | 4 | Alice Johnson | Stealth Fighter | 1580 | Windy | 7 | 1800 | 3500 | 0.80 |
| Gaza City | 5 | Jane Smith | Drone | 70 | Clear | 6 | 500 | 1000 | 0.90 |
| Riyadh | 3 | Michael Clark | Heavy Bomber | 1415 | Clear | 9 | 850 | 6000 | 0.87 |
| Cairo | 3 | Robert White | Fighter Jet | 404 | Cloudy | 10 | 1500 | 3000 | 0.83 |

הסבר על האלגוריתם:

חישוב ציונים עבור כל גורם:

- מרחק: ככל שהמטרה קרובה יותר, כן ייטב. זה יפחית את צריכת הדלק ויקל על המטוס להגיע ליעד.
- סוג מטוס: למטוסים מסוימים יהיה טווח ומהירות טובים יותר, מה שהופך אותם למתאימים יותר למשימות ארוכות יותר.
- מיומנות טייס: טייס ברמת מיומנות גבוהה יותר מתאים יותר למשימות קשות (למשל, רחוק או מזג אוויר גרוע).
- תנאי מזג האוויר: מזג אוויר נוח (שמיים בהירים, רוח חלשה, ללא גשם) יקבל ציון גבוה יותר.
- זמן ביצוע: המטרה היא לבצע את המשימה בהקדם האפשרי. העדיפו זמני התקפה אפשריים מוקדמים יותר בהתבסס על תנאי מזג האוויר.

משקל כל גורם:

הקצה משקל לכל גורם. המשקולות יגדירו את חשיבותו של כל גורם בקביעת ההתאמה הטובה ביותר למשימה.

שקלול לדוגמה:

מרחק: 20%

סוג מטוס: 25%

מיומנות טייס: 25%

תנאי מזג אוויר: 20%

זמן ביצוע: 10%

□ חישוב ציון משוקלל: עבור כל שילוב מטרה ומשימה, יש לחשב ניקוד באמצעות המשקולות. השילוב עם הציון הגבוה ביותר יהיה המתאים ביותר.

תהליך יצירת הפרויקט:

1. רישום לאתר
2. קריאת נתוני מטרת מ-JSON ושמירה במבנה נתונים
3. חילוץ מיקום גיאוגרפי עבור כל מטרה (lon ו-lat) באמצעות ה-API:
4. חישוב מרחק בין נקודות גיאוגרפיות באמצעות נוסחא שקיבלתם למטה
5. בניית האלגוריתם
6. הרצת האלגוריתם על המטרות
7. יצירת תצוגה ממוינת ב-CSV ועל המסך

רשימות

משקולות

```
weights = {  
    "distance": 0.20,      # 20%  
    "aircraft_type": 0.25, # 25%  
    "pilot_skill": 0.25,   # 25%  
    "weather_conditions": 0.20, # 20%  
    "execution_time": 0.10 # 10%  
}
```

```
def weather_score(weather):  
    if weather['condition'] == "Clear":  
        return 1.0 # Best condition  
    elif weather['condition'] == "Clouds":  
        return 0.7 # Clouds are moderate  
    elif weather['condition'] == "Rain":  
        return 0.4 # Rainy weather  
    elif weather['condition'] == "Stormy":  
        return 0.2 # Stormy weather is least favorable  
    else:  
        return 0 # Unfavorable condition
```

מטרות

| | Target City | Priority |
|----|-------------|----------|
| 1 | Damascus | 5 |
| 2 | Beirut | 4 |
| 3 | Amman | 3 |
| 4 | Cairo | 3 |
| 5 | Baghdad | 5 |
| 6 | Tehran | 4 |
| 7 | Riyadh | 3 |
| 8 | Tripoli | 2 |
| 9 | Ankara | 4 |
| 10 | Khartoum | 3 |
| 11 | Gaza City | 5 |
| 12 | Sanaa | 2 |
| 13 | Manama | 2 |
| 14 | Kuwait City | 3 |
| 15 | Doha | 4 |

מטוסים

```
aircrafts = [  
    {"type": "Fighter Jet", "speed": 1500, "fuel_capacity": 3000},  
    {"type": "Bomber", "speed": 900, "fuel_capacity": 5000},  
    {"type": "Drone", "speed": 500, "fuel_capacity": 1000},  
    {"type": "Helicopter", "speed": 400, "fuel_capacity": 800},  
    {"type": "Stealth Fighter", "speed": 1800, "fuel_capacity": 3500},  
    {"type": "Recon Drone", "speed": 600, "fuel_capacity": 1200},  
    {"type": "Heavy Bomber", "speed": 850, "fuel_capacity": 6000}  
]
```

טייסים

```
pilots = [  
    {"name": "John Doe", "skill": 8},  
    {"name": "Jane Smith", "skill": 6},  
    {"name": "Michael Clark", "skill": 9},  
    {"name": "Alice Johnson", "skill": 7},  
    {"name": "Robert White", "skill": 10},  
    {"name": "Tom Cruise", "skill": 9},  
    {"name": "Chris Pratt", "skill": 6},  
    {"name": "David Miller", "skill": 5},  
    {"name": "Sarah Connor", "skill": 8}  
]
```

רישום ועבודה עם API של מזג אוויר

רישום

1. רישום ע"י יצירת חשבון חינאמי – 1,000 בקשות API חינם ביום
2. הערה: ייתכן ותצטרכו לייצר נקודת אינטרנט משלכם באמצעות מכשיר פלאפון שלכם או של מרצה/עוזר הוראה, כי יש מגבלה של קריאות מאותה כתובת IP (כתובת מחשב). תעזרו במרצה במידה ונתקלתם בבעיה זו
3. קישור לרישום [Members \(openweathermap.org\)](https://openweathermap.org/membership)
4. המטרה היא לקבל API KEY, מחרוזת ארוכה שנשתמש בה לצורך זיהוי מול ה-API


Create New Account

We will use information you provided for management and administration purposes, and for keeping you informed by mail, telephone, email and SMS of other products and services from us and our partners. You can proactively manage your preferences or opt-out of communications with us at any time using Privacy Centre. You have the right to access your data held by us or to request your data to be deleted. For full details please see the OpenWeather [Privacy Policy](#).

- ☐ I am 16 years old and over
- ☐ I agree with [Privacy Policy](#), [Terms and conditions of sale](#) and [Websites terms and conditions of use](#)

I consent to receive communications from OpenWeather Group of Companies and their partners:

- ☐ System news (API usage alert, system update, temporary

Weather in your city  Guide API Dashboard Marketplace Pricing Maps Our Initiatives Partners Blog **For Business** to

How and where will you use our API?

Hi! We are doing some housekeeping around thousands of our customers. Your impact will be much appreciated. All you need to do is to choose in which exact area you use our services.

Company

*** Purpose**

... have sent the confirmat

ew Products Service

Ask a question

ny location

the data in the [Historical Weather Collection](#).

We have sent the confirmation link to tomer_sagi@webiks.com. Please check your email.

[New Products](#) [Services](#) [API keys](#) [Billing plans](#) [Payments](#) [Block logs](#) [My orders](#) [My profile](#) [Ask a question](#)



Historical weather for any location

Our new technology, Time Machine, has allowed us to enhance the data in the [Historical Weather Collection](#).

- Historical weather data available for **ANY** coordinate
- The depth of historical data have been extended to **40 YEARS**

You can download data from [Personal account](#) or [contact us](#) to order it.

[Learn more](#)

[Go to purchase](#)



Weather Dashboard

The [OpenWeather Dashboard](#) is a lightweight and flexible visual tool for our customers who would like to be notified weather events to make informed decisions and plan actions based on the

← | | 1 of 274 < >

OpenWeatherMap Account confirmation External Inbox x



OpenWeather Team <robot@openweathermap.org>
to me ▾

11:03 AM (1 minute ago) ☆ ↶ ⋮

Dear Customer!

Thank you for choosing [OpenWeatherMap](#)!

Please confirm your email address to help us ensure your account is always protected.




[Verify your email](#)

For further technical questions and support, please contact us at info@openweathermap.org

We are looking forward to cooperating with you!

New Products Services **API keys** Billing plans Payments Block logs My orders My profile Ask a question

You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

| Key | Name | Status | Actions | Create key |
|---|---------|--------|---|---|
|  | Default | Active |   | <input type="text" value="API key name"/> <input type="button" value="Generate"/> |

- את ה-KEY אתם מעתיקים ושמים בקוד כדי לבצע קריאת API. מומלץ לבדוק ב-Postman שאתם מצליחים לבצע קריאות

דרכים להשתמש ב-API

- תיעוד על ה-API (כללי): [Current weather data - OpenWeatherMap](#)
- חילוץ מיקום גיאוגרפי של עיר (כדי לבצע חישוב מרחק בין נקודות):
 - שימו לב ל-endpoint שכתוב וגם לשימוש ב-appid שהוא בעצם ה-KEY שקיבלתם באתר.
 - שימו לב לשם העיר שמופיע אחרי הפרמטר 'q'

<http://api.openweathermap.org/geo/1.0/direct?q=teheran&appid=XXXXXXXXXXXXXXX>

- דוגמא ל-JSON שמקבלים

```

1  [
2    {
3      "name": "Tehran",
4      "local_names": {
109    },
110    "lat": 35.6892523,
111    "lon": 51.3896004,
112    "country": "IR"
113  }
114 ]
    
```

- חילוץ תחזית מזג אויר עבור עיר:

<https://api.openweathermap.org/data/2.5/forecast?q=yemen&appid=XXXXXXXXXXXXXXX>

○ דוגמא ל-JSON שמקבלים

שימו לב לשדות חשובים שנרצה להשתמש בהם בתוכנה שלנו (מסומן באדום)

שימו לב שאנחנו מגדיר תחזית של ימים שלמים (לא בקבוצות של 3 שעות כמו שה-API מחזיר). אנחנו נחפש את התחזית לימים שלמים ע"י הסתכלות על חצות מהיום שללמחרת. כלומר מה-13.9.24 בחצות (כמו שמופיע למטה)

```
222 {
223   "dt": 1726185600,
224   "main": {
225     "temp": 300.84,
226     "feels_like": 299.98,
227     "temp_min": 300.84,
228     "temp_max": 300.84,
229     "pressure": 1006,
230     "sea_level": 1006,
231     "grnd_level": 892,
232     "humidity": 29,
233     "temp_kf": 0
234   },
235   "weather": [
236     {
237       "id": 804,
238       "main": "Clouds",
239       "description": "overcast clouds",
240       "icon": "04n"
241     }
242   ],
243   "clouds": {
244     "all": 88
245   },
246   "wind": {
247     "speed": 1.82,
```

```
242 ],
243 "clouds": {
244   "all": 88
245 },
246 "wind": {
247   "speed": 1.82,
248   "deg": 184,
249   "gust": 2.18
250 },
251 "visibility": 10000,
252 "pop": 0,
253 "sys": {
254   "pod": "n"
255 },
256 "dt_txt": "2024-09-13 00:00:00"
257 }
```

פונקציות חישוב מרחק:

שימו לב שהפונקציה מקבלת 2 קבוצות של lat ו-lon, כדי לחשב מרחק ביניהם ומחזירה מרחק בקילומטרים

```
# Function to calculate the distance between two coordinates using the Haversine formula
def haversine_distance(lat1, lon1, lat2, lon2):
    r = 6371.0 # Radius of the Earth in kilometers

    # Convert degrees to radians
    lat1_rad = math.radians(lat1)
    lon1_rad = math.radians(lon1)
    lat2_rad = math.radians(lat2)
    lon2_rad = math.radians(lon2)

    # Calculate differences between the coordinates
    dlat = lat2_rad - lat1_rad
    dlon = lon2_rad - lon1_rad

    # Apply Haversine formula
    a = math.sin(dlat / 2)**2 + math.cos(lat1_rad) * math.cos(lat2_rad) * math.sin(dlon / 2)**2
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))

    # Calculate the distance
    distance = r * c
    return distance
```

קריטריונים להערכה:

1. דיוק:

- התוכנית צריכה לדמות תקיפות אוויריות בצורה נכונה על בסיס תנאי מזג האוויר שנשלפים מה API-לטעון משימות מ JSON-להציג תוצאות, ולשמור את התוצאות ל-CSV.

2. יישום: OOP

- הפרויקט צריך להיות מאורגן היטב באמצעות עקרונות OOP (הכמסת מידע, עיצוב מודולרי).

3. אינטגרציה עם: API

- נתוני מזג האוויר צריכים להישלף בזמן אמת באמצעות ה API-וישפיעו על תוצאות המשימה.

4. שימוש במבני נתונים:

- שימוש יעיל ברשימות ומילונים לניהול נתוני משימות.

5. הבנת רשימות:

- שימוש בהבנת רשימות (List Comprehensions) לסינון תוצאות המשימה להצגה.