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:
- כאשר 5 ערי מטרה במדינות עוינות, לכל אחת מהן מוקצה רמת עדיפות (1-5),
 כאשר 5 היא העדיפות הגבוהה ביותר.
 - 2. יכולות מטוסים:
 - 7סוגי 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.
 - :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
 - יש יותר מטרות ממטוסים וטייסים ולכן המיון לפי חשיבות הוא קריטי : .a

בונוס: הקצאת טייסים, מטוסים וזמנים לביצוע למשימות אחרי הקצאה ראשונית (כלומר, אחר	.5
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, ו-Python-הסטודנטים ילמדו:

- איך לעבוד עם APIs לשליפת נתונים בזמן אמת (מזג אוויר).
 - לעיצוב קוד מודולרי ומדרגי. • ליישם עקרונות OOP
- 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.
 - להוסיף עיכובים בזמן אמת לסימולציית התקיפה.

מושגים חדשים שנבדקים:

API: אינטגרציה עם.1

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

2. תכנות מונחה עצמים:(OOP)

o שימוש ב Classes-לניהול מטוסים, טייסים ומטרות. ⊙

3. מבני נתונים:

o שימוש ברשימות ומילונים לניהול ואחסון נתוני משימות.

4. הבנת רשימות:

שימוש בהבנת רשימות לסינון והצגת תוצאות של תקיפות אוויריות.

5. ניהול קבצים:

 CSV ושמירת תוצאות לקבצי JSON קריאה מתוך קובצי \circ

מאפיינים חדשים בפרויקט:

:אינטגרציה עם API אינטגרציה עם 1

- $\underline{ ext{OpenWeatherMap}}$ סמו באמצעות) API שליפת מזג אוויר עכשווי למיקום מסוים באמצעות \circ
- שימוש בתנאי מזג האוויר שנשלפו (כגון מזג אוויר בהיר, רוח, סערה) כפרמטר שמשפיע
 על הסיכוי להצלחת המשימה.

2. עיצוב:**00P**

- שימוש במחלקות לשמור נתונים עבור כל יישות במערכת, לדוגמא: טייס, מטוס וכו' 🧄
- ס מזג האוויר ישולב כפרמטר דינמי שמשפיע על הצלחת המשימה, ומבוסס על נתונים
 → API.

:3 מערכת תפריט

- ס המערכת תציג תפריט שבו המשתמש יוכל לבחור פעולות שונות:
 - JSON-טעינת קבצים (מטרות וכל המידע הנוסף) מ-1
 - 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.** ⊙

תצוגה רצויה להדפסה ב-קונסול וב-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:

. 17 -	 17 7	1711	DIMIN	חינשור י	

- 1. **מרחק**: ככל שהמטרה קרובה יותר, כן ייטב. זה יפחית את צריכת הדלק ויקל על המטוס להגיע ליעד.
- 2. **סוג מטוס**: למטוסים מסוימים יהיה טווח ומהירות טובים יותר, מה שהופך אותם למתאימים יותר למשימות ארוכות יותר.
- 3. **מיומנות טייס**: טייס ברמת מיומנות גבוהה יותר מתאים יותר למשימות קשות (למשל, רחוק או מזג אוויר גרוע).
 - 4. **תנאי מזג** האוויר: מזג אוויר נוח (שמיים בהירים, רוח חלשה, ללא גשם) יקבל ציון גבוה יותר.
- 5. **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 מטרות מ-JSON ושמירה במבנה נתונים
- 3. חילוץ מיקום גיאוגרפי עבור כל מטרה (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
```

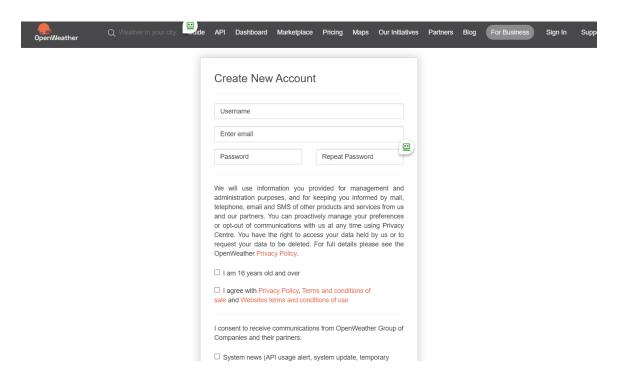
	Target City	Priority		
1	Damascus	5		
2	Beirut	4		
3	Amman	3		
4	Cairo	3		
5	Baghdad	5		
6	Tehran	4		
7	Riyadh	3		
8	Tripoli	2		
9	Ankara	4		
10	Khartoum	3		
11	Gaza City	5		
12	Sanaa	2		
13	Manama	2		
14	Kuwait City	3		
15	Doha	4		

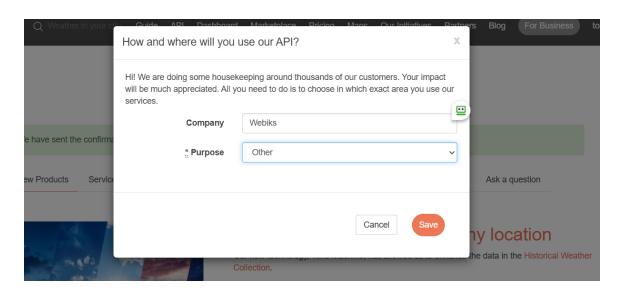
טייסים

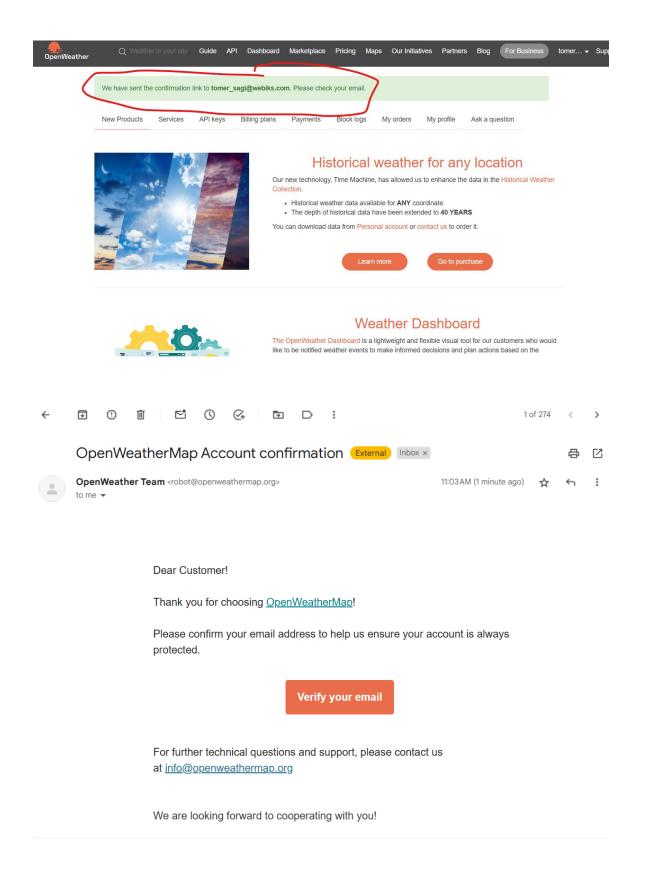
רישום ועבודה עם API של מזג אויר

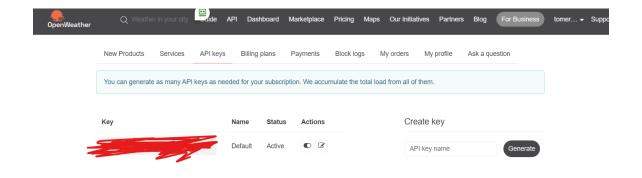
רישום

- 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
 - Members (openweathermap.org) קישור לרישום.3
 - 4. המטרה היא לקבל API KEY, מחרוזת ארוכה שנשתמש בה לצורך זיהוי מול ה-API.









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

דרכים להשתמש ב-API

- <u>Current weather data OpenWeatherMap</u> :(כללי) API- תיעוד על ה
 - חילוץ מיקום גיאוגרפי של עיר (כדי לבצע חישוב מרחק בין נקודות): •
- שקיבלתם KEY- שימו לב ל-endpoint שכתוב וגם לשימוש שכתוב וגם שכתוב שכתוב \circ באתר.
 - 'q' שימו לב לשם העיר שמופיע אחרי הפרמטר \circ

שמקבלים JSON - דוגמא ל ⊙

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

חילוץ תחזית מזג אויר עבור עיר:

שמקבלים JSON - דוגמא ל

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

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

פונקציות חישוב מרחק:

שימו לב שהפונקציה מקבלת 2 קבוצות של lat ו-lon, כדי לחשב מרחק בינהם 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 התוכנית צריכה לדמות תקיפות אוויריות מאוויר ל-BSON, שנשלפים מה API, לטעון משימות מCSV.

2. יישום:**00P**

ס הכמסת מידע, עיצוב 00P (הפרויקט צריך להיות מאורגן היטב באמצעות עקרונות α הפרויקט צריך להיות מאורגן היטב באמצעות עקרונות α מודולרי.

API:אינטגרציה עם.3

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

4. שימוש במבני נתונים:

שימוש יעיל ברשימות ומילונים לניהול נתוני משימות. ○

5. הבנת רשימות:

המשימה להצגה. (List Comprehensions) לסינון תוצאות המשימה להצגה. 🌣