

סיפור כללי:

אנחנו בונים מערכת של ניהול טיסות מכירת כרטיסים לנוסעים.
למערכת יש שתי פנים - הפן של מנהל חברת התעופה והפן של הלקוח
מנהל חברת התעופה יכול ליצור קווי תעופה חדשים - בתנאי שיש מספיק תקציב.
נוסע יכול לקנות כרטיס לקו מסוים, בתנאי שיש קו כזה.

התרגיל מבוסס על **persistence** - שמירת מידע מחוץ לאפליקציית פייתון כך שהוא מתקיים גם אחרי
שהאפליקציה מסיימת את ריצתה.
שמירת המידע תיעשה על 3 סוגי קבצים שונים:

1. קובץ טקסט רגיל
2. קובץ CSV
3. קובץ JSON

הפעולות שאתם נדרשים לתמוך בהן:

כמנהל:

1. הזדהות כמנהל על ידי סיפוק שם משתמש וסיסמה
2. קניית כניסה לנמל תעופה חדש (אם התקציב מאפשר את זה)
3. יצירת נתיב תעופה חדש (אפשרי רק אם נקנתה כניסה לנקודת המוצא ולנקודת הקצה)

כנוסע:

1. קניית כרטיס טיסה לפי נקודת מוצא ונקודת קצה

חלק מהקבצים יסופקו לכם, ואתם צריכים רק להכניס אותם לפרוייקט ולוודא את קיומם.

הנחיות כלליות:

1. הפרוייקט כולו מוגש ב-REPO בגיטהאב
2. העבודה תיעשה עם ענף לכל שלב. יש לפתוח ענפים שיוצאים מהענף של אותו שלב עבור המשימות הקטנות ועוברות MERGE עם PULL REQUEST לתוך הענף של השלב.
3. הענף של השלב עובר MERGE עם PR לתוך MAIN כשכל העבודה על אותו שלב הסתיימה.
4. יש לשמור על קוד נקי ומסודר
5. אין למרג'ג' לתוך MAIN קוד שנמצא בתוך הערות, רווחי שורות מיותרים, וPRINT מיותר.

בכל שלב תתוארנה המשימות שאותו שלב דורש. אתם צריכים לפרק את המשימות, להבין מה בדיוק נדרש, וליצור ענף חדש לכל תת משימה (ולמרג'ג' אותו חזרה לתוך הענף של אותו שלב כשהמשימה הסתיימה).
לפעמים תתוארנה חלק מהפונקציות שאתם נדרשים לממש - אלו לא כל הפונקציות הנדרשות. אתם נדרשים לקרוא היטב את הדרישות בכל פיצ'ר ולממש כל פונקציה שצריך כדי שכל תכונה וכל פעולה תתקיים.

שלב א - תשתיות בסיסיות

פיצ'ר יצירת קובץ תקציב ובדיקת קיום הקובץ בתחילת הריצה:

1. יצירת קובץ טקסט שישמור את התקציב של חברת התעופה. אין חובה ליצור את הקובץ ולאכלס אותו במספר דרך פייתון - אפשר לעשות זאת דרך הטרמינל/UI. שם הקובץ: budget.txt
2. כשמתחילים את האפליקציה יש לוודא בעזרת מודול OS שקיים קובץ התקציב. אם לא - יש לזרוק שגיאה ולעצור את ריצת התוכנה.

חלק מהפונקציות ספציפיות לפיצ'ר הזה:

1. פונקציה שטוענת את קובץ budget ומחזירה את המספר ששמור בקובץ כמספר
2. פונקציה שבודקת את קיומו של קובץ בשם budget.txt - מחזירה True או False בהתאם לתוצאה.

פיצ'ר הזדהות כמנהל

1. יש ליצור קובץ בשם credentials.csv. יצירת הקובץ תיעשה דרך UI/טרמינל, אין צורך לייצר ולאכלס את הקובץ דרך פייתון.
2. הקובץ הוא CSV שמדמה טבלה של שמות משתמש וסיסמאות. השורה הראשונה היא: username,password
3. הכניסו ידנית בשורה השניה ערכים של שם משתמש וסיסמה. הערכים האלו יישמשו להזדהות כמנהל לטובת פעולות המנהל. (אקסטרה - אם אין ערכים בהרצה ראשונה, בקשו מהמנהל להכניס שם משתמש וסיסמה, ושימרו את הסיסמה כ-HASH)
4. כשמתחילים את האפליקציה יש לוודא בעזרת מודול OS שקיים קובץ CREDENTIALS. אם לא - יש לזרוק שגיאה ולעצור את ריצת התוכנה.

חלק מהפונקציות נדרשות לפיצ'ר הזה:

1. פונקציה שבודקת את קיומו של קובץ בשם credentials.csv - מחזירה True או False בהתאם לתוצאה.
2. פונקציה שטוענת את הקובץ, ועוברת עליו שורה שורה. אפשר להשתמש במודול CSV או בכל כלי אחר. הפונקציה מחזירה את תוכן הקובץ כמטריצה - כל שורה היא רשימה נפרדת.
3. פונקציה שמקבלת כפרמטר שם משתמש וסיסמה, ואת המטריצה של תוכן הקובץ CREDENTIALS. הפונקציה עוברת על תוכן הקובץ ומחפשת את השם משתמש והסיסמה הנתונים. מחזירה TRUE או FALSE בהתאם לתוצאה.

שלב ב - הרצה בסיסית, תפריט בסיסי, הזדהות

פיצ'ר תשתיות:

1. יש ליצור קובץ `main.py`. הקובץ הזה יהיה הקובץ היחיד שמריצים.
2. הפעולה הראשונה שתבצע היא בדיקת קיומם של הקבצים `BUDGET` ו `CREDENTIALS`. יש להשתמש בפונקציות שמומשו בשלב הקודם.
3. הקובץ יציג תפריט בסיסי. השאלה הראשונה של התפריט היא האם המשתמש הוא מנהל או נוסע.
4. אם הוא מנהל - התפריט יבקש מהמשתמש שם משתמש וסיסמה.
5. יש לבדוק את תקינות שם המשתמש והסיסמה תוך שימוש בפונקציה שהוגדרה בשלב הקודם.

שלב ג - יצירת קווי תעופה חדשים כמנהל

פיצ'ר טעינת בנק שדות תעופה:

1. נתון לכם קובץ בשם airports.csv. בקובץ זה מפורטים בפורמט CSV כלל שדות התעופה הזמינים, יחד עם המחיר שיש לשם כדי להשתמש בשדה התעופה הזה כחלק מקו טיסה. בנוסף, כתוב שם שדה התעופה כקוד בן 3 אותיות, והיבשת שבה שדה התעופה נמצא (רלוונטי לחישוב עלות טיסה, ראו בהמשך). עברו על הקובץ וודאו שאתם מבינים את המבנה שלו.
2. יש לוודא את קיומו של הקובץ של שדות התעופה בדומה לקבצי התקציב והסיסמאות - באותו שלב של האפליקציה, ולזרוק שגיאה באופן דומה אם הקובץ לא נמצא.

פיצ'ר יצירת קו חדש:

1. יש להוסיף קובץ טקסט חדש שישמור את הקווים הקיימים של חברת התעופה שלנו

- a. שם הקובץ: available_flights.json
- b. יש לוודא את קיומו באותו שלב ובאותו אופן כמו שאר קבצי הטקסט.
- c. מבנה הJSON:

```
d. {  
  "available_lines": [  
    {  
      "origin_airport": <airport code>,  
      "destination_airport": <airport code>  
    },  
    { ... }  
  ]  
}
```

2. אחרי הזדהות מוצלחת כמנהל, יש להציג בתפריט את האפשרות להוסיף קו תעופה חדש.
 - a. אם המשתמש בוחר באפשרות זו, יש לבקש מהמשתמש לתת 3 אותיות שהן הקוד של שדה התעופה - אחד עבור נקודת המוצא של הקו ואחד עבור נקודת הקצה של הקו.
 - b. יש לטעון את קובץ שדות התעופה ולוודא ששני צמיד 3 האותיות קיימים במאגר. אם אחד מהם לא קיים במאגר - יש לבקש קוד אחר מהמשתמש במקום זה שאינו קיים.
3. אחרי שיש לנו 2 קודים של שדות תעופה קיימים, התוכנה צריכה לוודא שיש למנהל מספיק כסף בתקציב כדי לקנות כניסה לשני שדות התעופה האלה.
 - a. יש לטעון את קובץ התקציב כמספר. יש להשתמש בפונקציה משלב א'.
 - b. יש לטעון את סכום של דמי הכניסה של שני שדות התעופה מתוך קובץ שדות התעופה.
 - c. יש להשוות בין התקציב לבין הסכום - אם הסכום גדול מהתקציב, יש לכתוב הודעה שמסבירה את הבעיה, לזרוק שגיאה ולחזור לתפריט המנהל.
 - d. אם התקציב גדול מהסכום - יש לבקש אימות סופי לקניית קו התעופה.
 - e. אחרי קבלת אימות סופי:
 - i. יש לטעון את קובץ available_lines.json
 - ii. יש להוסיף את הקו החדש לתוך רשימת הקווים
 - iii. יש לשמור את הקובץ מחדש - הפעם עם הקו החדש בתוכו.

בנוסף: מימוש מחיקת קווים כמנהל, עריכת קווים כמנהל - שינוי תחנת מוצא או תחנת יעד.

שלב ד - קניית כרטיס טיסה כנוסע.

פיצ'ר הצגת תפריט בסיסי כנוסע

1. יש לעדכן את התפריט הראשי בבחירה הראשונה - ולאפשר בחירה כנוסע, ולא רק כמנהל.
2. אם המשתמש בוחר להמשיך כנוסע, על התפריט להציג לו את כלל הטיסות הזמינות.
 - a. יש לטעון את קובץ `available_lines.json`
 - b. יש להציג בצורה מסודרת את כל הקווים - קו מוצא וקו יעד.
 - c. המשתמש צריך לבחור טיסה.
 - d. ברגע שהוא בוחר בחירה תקינה, מתבצע חישוב מחיר הכרטיס.

פיצ'ר חישוב מחיר הכרטיס

1. כחברת תעופה חשוב לנו להישאר רווחיים. לכן נדרש חישוב מדויק של מחיר הכרטיס.
 2. חישוב מחיר הכרטיס מחושב באופן הבא:
 - a. יש לקחת את מחיר הכניסה לשדות התעופה של שדה המוצא ושדה היעד.
 - b. יש לחלק את הסכום ל-400 - כמות הנוסעים בטיסה ממוצעת. זהו **מחיר הבסיס**.
 - c. יש להוסיף לחשבון את אורך הטיסה. לצורך כך, ניתן לכם כנתון הקובץ `continents_pricing.csv` - שמציג כמה מחיר כללי עולה טיסה בין יבשות שונות.
 - i. יש לוודא את קיום הקובץ הזה יחד עם שאר קבצי הטקסט
 - ii. יש להתנהג אם קובץ זה חסר בצורה זהה להתנהגות של חוסרים עם שאר קבצי הטקסט.
 - d. יש לזהות את היבשת של שדה המוצא:
 - i. יש לטעון את קובץ `airports.csv`
 - ii. יש לחפש לפי קוד שדה תעופה
 - iii. יש לשלוף מהשורה את היבשת
 - e. לחזור על השלב הקודם עבור שדה היעד.
 - f. יש לטעון את קובץ `continents_pricing.csv` ולבדוק מה המחיר לפי שתי היבשות לפי יבשת מוצא -> יבשת יעד
 - g. יש לקחת את הסכום הזה, לחלק אותו ל-400 ולהוסיף אותו למחיר הבסיס. זהו **מחיר הכרטיס**.
3. יש להציג למשתמש את המחיר של הכרטיס ולבקש אימות סופי שהוא מעוניין לקנות את הכרטיס.
4. אם המשתמש מאשר:
 - a. יש להוסיף לקובץ התקציב של החברה את הסכום שהמשתמש שילם על הכרטיס
 - b. יש ליצור ID אקראי בן 8 אותיות המורכב מאותיות ומספרים - זהו ID הכרטיס.
 - c. אחרי העדכון של התקציב יש להודיע למשתמש כי הכרטיס נקנה בהצלחה, יש להציג את הID ולחזור לתפריט המשתמש הראשי.

בנוס:

1. הגבלת כמות הכרטיסים פר טיסה - ברגע שנקנה כרטיס יש כרטיס אחד פחות
2. אפשרות ביטול של כרטיס (ועדכון התקציב בהתאם) עד 10 ימים מרגע הקנייה.
3. אפשרות של עריכת הכרטיס.

שלב ה - תוספות

פיצ'רים נוספים:

1. יצירת יישות של טיסות כמנהל חברת התעופה. כל טיסה:
 - a. קיימת על קו קיים של שדה יעד ושדה מוצא
 - b. בעלת ID ייחודי
 - c. בעלת מספר נוסעים מסוים, שרק עד אליו ניתן לקנות כרטיסים.
 - d. מחולקת למחלקות שמשנות את מחיר הכרטיס - אקונומי, עסקים, ראשונה.
2. לוגים:
 - a. כדי להבין מה קורה באפליקציה, יש לייצר לוגים מסודרים לפי הפורמט:
date | log level | message
 - b. יש להכניס את הלוגים האלו לפני כל פעולה מורכבת ואחריה