תכנות מונחה עצמים מטלה 1 - לומדים Python

מטלה זו מוקדשת ללימוד python כאשר היא למעשה מטלת המשך למטלה 0 (שבוצע ב java). כרגיל, מומלץ להתחיל בתכנון המטלה (ללא קוד) ורק לאחר שיש בידכם תכנון בסיסי כדאי לעבור לשלב מימוש הקוד.
בדומה למטלה 0, גם במטלה זו נפתח אלגוריתמיקה עבור מערכת מעליות חכמות של בניין רב קומות.
במטלה זו האתגר נתמקד בגרסת ה **offline** של הבעיה - משמע כל הקריאות ניתנות לנו מראש, ואנו רק נדרשים לשבץ כל קריאה באופן מיטיב (כך שכלל ההמתנה למעליות יצומצם למינימום). למעשה המטלה הנוכחית מתמקדת בעיקר באתגר של שיבוץ "קריאה" למעלית, אבל כדי לממש מטלה זו כמו שצריך תצטרכו גם להבין ולעשות סימולציה של תנועת המעליות.

מתחילים לעבוד "תכלס":
מטלה זו מתחלקת לשני חלקים חלק ראשון (סעיפים 1-3) היכרות תכנון, ומימוש בסיסי, חלק שני: בדיקות התאמה של המערכת + סעיף בונוס (סעיפים 4-5). מטרת החלוקה היא לחייב אתכם לחשוב קודם על מידול הבעיה, תכנון האלגוריתמים הנדרשים, ומימוש בסיסי - ורק לאחר מכן לבצע בדיקת ביצועים, השוואות וממשקים גרפיים (מטלת בונוס).
שלבי עבודה - חלק א' ללא קוד:

1. פתחו פרויקט Github, כתבו ב Readme (באנגלית כמובן) - ניתן ורצוי להביא את החומרים הרלוונטיים (למחזר) מהמטלה הקודמת שלכם, בפרט סקר הספרות שביצעתם, הסבר על מרחב הבעיה וכו'. כלל הפיתוח של המטלה צריך להיות ב Github, כל חברי הקבוצה חייבים לתרום לפרויקט.
2. נסחו אלגוריתם off-line עבור בעיית השיבוץ של קריאות למעליות - השתדלו לנסח את האלגוריתם באופן המדויק והפשוט ביותר להבנה (באנגלית כמובן).
3. כתבו תוכנה ב Python שמקבלת שלוש מחרוזת (שמייצגות שמות של קבצים)
   Ex1 <Building.json> <Calls.csv> <output.csv>
   - הקובץ שמייצג בניין (בפורמט JSON) כולל למעשה מערך של מעליות (ראו קבצים לדוגמא),
   - הקובץ שמייצג את הקריאות - הוא למעשה קובץ דומה למה שעבדתם איתו במטלה 0.
   - קובץ הפלט הוא זהה בצורתו לקובץ הקריאות - רק עם שיבוץ של האינדקס של המעלית לכל קריאה.

   Example for running the python program: python Ex1.py B1.json C2.csv out.csv
   התוכנה צריכה לחשב שיבוץ שיביא למינימום זמן המתנה מצרפי (של כלל הבקשות) - כל הקלט ניתן לכם מראש כך שהאלגוריתם שתכתבו מוגדר להיות offline.
   להלן קישור לקבצים לדוגמא של בניינים וקריאות, להלן קישור לקובץ פלט לדוגמא - עבור הרצה של בניין B2.json, על קלט של Calls_a.csv. שימו לב שמידע בעמודות אחרי (העמודה השישית - F) הוא לא רלוונטי ולמעשה הוא לא נקרא ע"י המערכת בדיקה שלנו, (שימו לב לסדר העמדות: עמודה 1: סתם מחרוזת, עמודה 2: זמן, עמודה 3: קומת מקור, עמודה 4: קומת יעד, עמודה 5 סטטוס מעלית (לא במימוש - אפשר להתעלם), עמודה 6: שיבוץ למעלית (אינדקס), אחרי עמודה 6 המידע לא נקרא - אין לו חשיבות.)

4. לאחר שתשלימו את כתיבת התוכנית, השתמשו במערכת הבדיקה המצורפת שמאפשרת הרצה של כל אחד מהפתרונות שלכם (ת"ז, בניין, קובץ שיבוץ מעליות, קובץ פלט). המערכת עושה שימוש בקובץ jar המצורף אותו ניתן להריץ הן משורת הפקודה, והן מתוכנית כגון זו.
   להלן דוגמא כיצד להריץ את מערכת הבדיקה משורת הפקודה:
   java -jar Ex1_checker_V1.2_obf.jar 1111,2222,3333 B2.json Ex1_Calls_case_2_b.csv out.log
   מערכת הבדיקה מדפיסה סטטיסטיקות של זמן המתנה ממוצע - ובדומה למטלה הקודמת ניתן לקבל קובץ log (ראו דוגמא), דוגמא להרצה של הקובץ ניתן למצוא בסוף המסמך. יסופק לכם קישור לפרסום התוצאות שלכם.
   דווחו על הפתרונות שלכם כפי שחושבו על כלי הבדיקה בטופס הבא, אנא בדקו שאכן הדיווחים שלכם התעדכנו בקישור הזה.
   יש להגיש קישור לפרויקט ה github שלכם בקישור הבא

5. סעיף בונוס: חשבו כיצד ניתן לייצר מערכת תצוגה (סימולציה גרפית שתציג את תנועת המעליות ביחס לבקשות) לפתרון שלכם - תכננו וממשו מערכת כזו (ב java או python לבחירתכם), צלמו קליפ של עד שתי דקות של התוכנה שלכם מסמלצת ב GUI את האלגוריתם שלכם, העלו את הסרטון לyoutube - והוסיפו קישור ל Readme, ולדיווח התוצאות.

```
EX1_checker — -zsh — 80×26
boazben-moshe@Boazs-MacBook-Air-2 EX1_checker % java -jar Ex1_checker_V1.2_obf.j
ar 111111,222222 B2.json Ex1_Calls_case_2_b.csv out.log
Elevator call,0.437472729039321,0,-1,3,0, Done, dt, 164.5625272709607
Elevator call,2.2094056301593525,0,-1,3,0, Done, dt, 162.79059436984065
Elevator call,18.77299317966884,-1,-2,3,0, Done, dt, 157.22700682033116
Elevator call,19.590869649656987,0,7,3,1, Done, dt, 193.40913035034302
Elevator call,29.579433566814906,0,1,3,0, Done, dt, 117.42056643318509
Elevator call,33.242673193933555,10,0,3,1, Done, dt, 120.75732680606644
Elevator call,52.96277472128405,3,-1,3,0, Done, dt, 112.03722527871595
Elevator call,64.38547514955395,2,-1,3,0, Done, dt, 100.61452485044605
Elevator call,72.26161212799506,10,0,3,1, Done, dt, 235.73838787200492
Elevator call,80.56754743307536,-1,-2,3,0, Done, dt, 95.43245256692464
Ex1 - OOP Simulator
Time: start: 0.0,  end: 200.56754743307536,  dt: 1.0
Building: Building [-2,10]
0) ex1.Elevator_A, Elev_, id,0,  speed, 1.0, open: 2.0,  start: 3.0, close: 2.0,
  stop: 3.0
1) ex1.Elevator_A, Elev_, id,1,  speed, 2.0, open: 2.0,  start: 3.0, close: 2.0,
  stop: 3.0

Call log: Elevator log, size: 10
************* Simulation Ended ************
Code Owners,111111,222222,  Building_file,B2.json,  Calls,Ex1_Calls_case_2_b.csv
Total waiting time: 239.0,  average waiting time per call: 23.9,  unCompleted ca
lls,0,  certificate, -261591202
boazben-moshe@Boazs-MacBook-Air-2 EX1_checker %
```

דוגמא להרצה של קובץ הבדיקה. שימו לב שהסימולציה רצה עד 120 שניות מעבר לקריאה האחרונה, אך אם השלמתם את ההרצות לפני היא תציג זאת בקובץ הפלט.

את המטלה יש להגיש עד ה 19.11.2021 כקישור יחיד ל Github דרך הפורום ההגשה, כל הקישורים נמצאים בקישור הבא:

בהצלחה!!

שאלות ותשובות -

Q & A

1. What is the Elevator algorithm (as implemented in the testing jar) - with respect to a given list of “calls”:
   A: The Elevator-Algorithm as implemented in the jar is defined as follows:
   1.1 Each elevator has a “flag” of UP/DOWN, and its start position is floor 0.
   1.2 At the beginning, once the first call arrives, the flag is defined to the value UP/DOWN according to the src value (is below 0 → DOWN, above 0 → UP).
   1.3 In case the elevator’s flag==True (UP), it will “go to” the closest floor above its current position. If none - it will change the flag = DOWN.
   1.4 In case the elevator’s flag==False (DOWN), it will “go to” the closest floor below its current position. If none - it will change the flag = UP.
   1.5 If the list of calls is Empty it will remain in LEVEL - state.

   - The expected time the elevator will travel from floor_a to floor_b is:
     CloseTime + StartTime + df/speed + StopTime + OpenTime (df == |floor_a-floor_b).

2. Do I need to implement a specific classes:
   A: No firm definition, yet you are expected to implement classes and method which will support the off-line algorithm as you have defined in #2. In particular you are expected to be able to simulate the position of each elevator at any given time (based on the allocated calls - and the above elevator-algorithm).

3. Can I use python packages (such as Numpy, Pandas, etc) for implementing my Ex1?
   A: But of course (YES - you can).

4. What is the potential extra credit for the bonus task:
   A: Up to 20%