בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

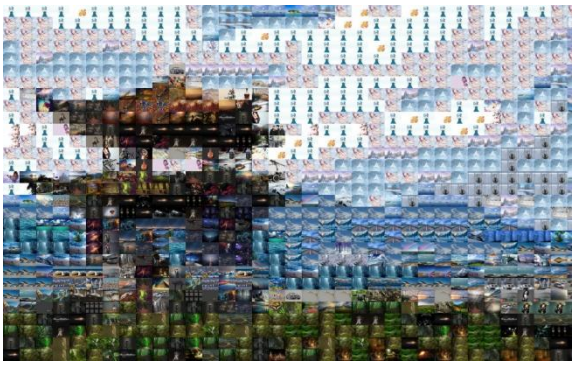
מבוא למדעי המחשב 67101

תרגיל 6 - עיבוד תמונה Photomosaic

להגשה בתאריך 02/12/2015 בשעה 22:00

הקדמה

תמונת פסיפס (Photomasaic) היא תמונה המורכבת ממספר רב של תמונות קטנות יותר (אריחים) היוצרות אותה. כפי שניתן להתרשם בדוגמא הבאה:




תמונות כאלה נוצרות על ידי לקיחת תמונה מקורית ואוסף של "אריחים" אפשריים, חלוקת תמונת המקור למקטעים, ובחירת האריח המתאים ביותר להחלפת כל מקטע (האריחים בדרך כלל שונים מקטעי התמונה שהם מחליפים). ישנם אלגוריתמים שונים ליצירת תמונות כאלה: חלקם עושים שימוש בגדלי אריחים שונים לקבלת התאמה טובה יותר (במקרים כאלה, החלוקה למקטעים עשויה להתבצע בצורה דינמית עם ריצת האלגוריתם), חלקם משנים מעט את גוון האריחים על מנת לקבל התאמה טובה יותר, וחלקם אף ממזגים מעט מן התמונה המקורית לתוך תמונת הפסיפס לקבלת תוצאה קרובה יותר לתמונה המקורית.

הבסיס ליצירת תמונת הפסיפס הינו ההשוואה בין מקטעים מהתמונה המקורית לאריחים השונים ה"מועמדים" להחליף אותם בתמונת הפסיפס הסופית. הדרך הפשוטה ביותר לביצוע השוואה זו היא על ידי מיצוע כלל הצבעים עבור כל אחד מהאריחים, והשוואת הצבע הממוצע לצבע הממוצע למקטע אותו רוצים להחליף.

דרך מורכבת יותר, היא על ידי השוואה פיקסל-פיקסל (כפי שיוסבר בהמשך) בין האריחים למקטעים. דרך זו מגיעה לתוצאות טובות יותר, אך היא גוררת עלות חישובית גבוהה – שכן עבור כל מקטע, יש לעבור על כל הפיקסלים של כל אחד מהאריחים.

בתרגיל זה נכתוב תוכנית הממשת אלגוריתם המשלב בין שתי הדרכים. כך, זמן הריצה של האלגוריתם נשאר נמוך יחסית, בעוד שהתוצאות טובות מהתוצאות המתקבלות על ידי השוואת הצבע הממוצע בלבד.

עליכם להוריד מהאתר את הקבצים הדרושים לתרגיל:

1. mosaic.py – מודול זה כבר מומש בשבילכם, והוא מכיל מספר פונקציות הדרושות לתרגיל. אל תעשו שום שינוי בקובץ mosaic.py!
2. ex6.py – זה הקובץ שאתם תצטרכו לממש בו את הפונקציות הדרושות, ע"פ ההוראות המפורטות בהמשך. (יש למחוק את המילה pass בפונקציות אלה, ובמקומה להכניס את המימוש שלכם לפונקציות ואת התיעוד עבורן).
3. im1.jpg – התמונה ההתחלתית (תמונת המקור):

4. images.zip – מקבץ תמונות לשימוש כאריחים.

שימו לב, התרגיל מורכב בצורה מובנית ממספר משימות, שבסופו של דבר יתחברו ביחד וירכיבו את המוצר הסופי הממשמש Photomosaic. לכן עקבו אחר ההוראות והשלבים של התרגיל, והקפידו לכתוב את הקוד שלכם במדויק על פי הנחיות התרגיל. כמו כן, מומלץ בחום לקרוא את כלל התרגיל (ובפרט את סעיף ה"טיפים וההנחיות" בסופו) לפני תחילת הפתרון.

אתם יכולים לכתוב בקובץ ex6.py פונקציות עזר נוספות מלבד אלה הדרושות בתרגיל, ולהשתמש בהן בקוד שלכם. אבל הפונקציות הדרושות בתרגיל חייבות להיכתב בדיוק על פי הדרישות המפורטות להלן.

הקפידו לכתוב תיעוד לקוד שלכם ובפרט לכל פונקציה שאתם כותבים.

הספריה PIL

הקובץ – mosaic.py הנתון לכם בתרגיל זה עושה שימוש בספריה PIL של פייתון, ולכן צריך אותה על מנת להריץ את התרגיל.

במעבדת המחשבים של האוניברסיטה (האקוריום) כבר מותקנת ספריה זו, ואין צורך להתקין שום דבר.

כמו כן, הספרייה כלולה ב- WinPython (זמין להורדה ב- http://winpython.github.io).

האלגוריתם

האלגוריתם שנממש בתרגיל זה מבוסס על בחירת מקטע מתמונת המקור, ומציאת האריח המתאים לו ביותר, וחוזר חלילה. תמונת הפסיפס נבנית על ידי קביעת האריח הנבחר בכל שלב במקום בו היה המקטע לו האריח מתאים.

האלגוריתם מתואר באופן הבא:

קלט: תמונת מקור (im1), ואוסף תמונות (אריחים) שכולן באותו הגודל.

פלט: תמונת Photomosaic המורכבת מאוסף האריחים.

בחירת האריח המתאים לכל מקטע באלגוריתם מתבצעת בשני שלבים: בראשון נבחרת קבוצת מועמדים ראשונית מתוך אוסף האריחים (על ידי השוואת ממוצע צבעי האריחים לממוצע צבע המקטע). בשני, נבחר האריח המתאים ביותר מתוך תת קבוצה זו – על ידי השוואה פיקסל-פיקסל.

שלבי האלגוריתם:

1. עיבוד מקדים על כלל האריחים: חישוב ושמירת צבעו הממוצע של כל אריח.
2. חזרה עד לכיסוי כל התמונה:
   - א. בחירת מקטע התמונה הבא (נקבע לפי פינה שמאלית וגודל).
   - ב. חישוב ממוצע צבע המקטע.
   - ג. בחירת אוסף של num_cadidates אריחים מועמדים שממוצעם דומה ביותר לממוצע המקטע.
   - ד. בחירת האריח המועמד הדומה ביותר למקטע התמונה (ע"י השוואה פיקסל-פיקסל כפי שיוסבר בהמשך)
   - ה. קביעת האריח הנבחר מסעיף ד' במקום המתאים בתמונת הפסיפס.

למעשה, על ידי בחירת אוסף האריחים המועמדים, אנו חוסכים חישוב יקר של השוואת אריחים שאינם רלבנטיים למקטע התמונה – ובכך משפרים משמעותית את זמן הריצה של האלגוריתם עבור רשימות אריחים ארוכות.

עבודה עם תמונות בתרגיל

הקובץ mosaic.py ממסך את העבודה עם אובייקטי תמונה של פיית'ון. במקום, "תמונה" בתרגיל מיוצגת ע"י רשימה של רשימות. כאשר תמונה בגובה 100 פיקסלים וברוחב 200 פיקסלים היא רשימה של 100 רשימות – של 200 פיקסלים כל אחת. קריאה ל- image[row][column] תתן את הפיקסל שבשורה row ובעמודה column, כאשר הפיקסל השמאלי העליון הוא הפיקסל בשורה 0 ועמודה 0.

גישה כזו לפיקסל נותנת tuple בגודל 3 של ערכי ה-RGB (Red, Green, Blue) של הפיקסל. כלומר image[row][column] ייתן tuple של 3 מספרים שהם int בין 0 ל-255, המייצגים את מידת האדום, הירוק והכחול שיש בפיקסל שבשורה row והעמודה column בתמונה, שביחד יוצרים את צבע הפיקסל כפי שאנו רואים אותו בתמונה.

הקובץ mosaic.py

לצורך התרגיל מסופק לכם הקובץ mosaic.py, הממש פונקציות מסוימות להן אתם נדרשים בתרגיל:

build_tile_base(tiles_dir, tile_height):

פונקצייה זו מקבלת כפרמטרים את שם התיקייה בה נמצאות התמונות המשמשות כ"אריחים" (String), ואת הגובה של כל אריח בתמונת הפסיפס שתתקבל (int). הפונקציה מחזירה רשימה של תמונות - שהן כל האריחים האפשריים (מרשימת אריחים זו יבחרו האריחים שירכיבו את תמונת הפסיפס). הפונקציה קובעת את גודלו של כל אריח על ידי שינוי גודלו (מתיחה או כיווץ) לקבלת הגובה (tile_height) הרצוי, ולאחר מכן חיתוך כל האריחים לקבלת אריחים ברוחב אחיד. כך, כלל האריחים ברשימה המתקבלת זהים בגודלם.

load_image(image_filename):

פונקצייה זו מקבלת את שם הקובץ של תמונת המקור (String), ומחזירה את תמונת המקור.

save(image, filename):

פונקצייה זו מקבלת תמונה (רשימה של רשימות) ושומרת אותה בקובץ ששמו (String) הוא filename. **שימו לב**: על מנת שלא לדרוס בטעות קובץ קיים, הפונקציה **לא תשמור את הקובץ במידה וכבר קיים קובץ בשם זה**.

מעבר לכך, לנוחותכם גם הפונקציה:

show(image):

המציגה תמונה (רשימה של רשימות).

<u>חלק ראשון: פונקציות מרחק</u>

כפי שאמרנו קודם, הבסיס לאלגוריתם הוא השוואה בין אריחים למקטעים של התמונה המקורית, למציאת האריח הדומה ביותר לכל מקטע תמונה. לשם כך, עלינו להגדיר פונקציית מרחק בין תמונות – כך נוכל להעריך את ההבדל בין כל מקטע תמונה לכל אחד מהאריחים, ולבחור את האריחים שמרחקם מינימלי.

בתרגיל זה, נגדיר את המרחק בין שתי תמונות להיות סכום המרחקים בין כל הפיקסלים (במקומות המתאימים) בין שתי התמונות. במילים אחרות, אם נתונות שתי תמונות image1, image2 בגודל Height x Width, ובהנתן פונקציית מרחק בין פיקסלים dist(pixel1, pixel2) – המרחק בין התמונות יהיה:

$$distance(image1, image2) = \sum_{\substack{0 \le row < Height \\ 0 \le column < Width}} \text{dist}(\text{image1}[\text{row}][\text{column}], \text{image2}[\text{row}], [\text{column}])$$

אם כן, ראשית יש לממש פונקציית מרחק בין שני פיקסלים בודדים.

1. **הפונקציה** compare_pixel

עליכם לממש את הפונקציה compare_pixel, המחשבת את המרחק בין 2 פיקסלים שונים. חתימת הפונקציה צריכה להיות:

def compare_pixel(pixel1, pixel2):

הפונקציה מקבלת 2 פרמטרים שהם ערכי הצבעים של שני הפיקסלים (כל פיקסל נתון כ tuple של 3 מספרים (r,g,b) כפי שהוסבר קודם). הפונקציה צריכה להחזיר מספר המייצג את סכום המרחקים בין ערכי- red, green ו- blue של הפיקסלים לפי הנוסחא הבאה: $|r_1 - r_2| + |g_1 - g_2| + |b_1 - b_2|$.
נקרא לערך זה "המרחק בין הפיקסלים".

2. הפונקציה compare

עליכם לממש את הפונקציה compare, המחשבת את המרחק בין 2 תמונות שונות. חתימת הפונקציה צריכה להיות:

def compare (image1, image2):

הפונקציה מקבלת 2 פרמטרים: image1, image2 שהן שתי התמונות להשוואה, ומחזירה את סכום המרחקים בין כל הפיקסלים של התמונות (נקרא לערך זה 'המרחק בין התמונות').

אם אחת התמונות גדולה מהשניה, **החישוב יתבצע רק עבור הפיקסלים המשותפים לשתי התמונות (כאשר הפינה השמאלית העליונה של שניהם חופפת)**.

לדוגמא, אם image1 תמונה בגודל 3x2, ו-image2 תמונה בגודל 1x3, הקריאה:

compare (image1, image2)

תחזיר את הערך:

compare_pixel_rgb(image1[0][0], image2[0][0]) + compare_pixel_rgb(image1[0][1], image2[0][1])

חלק שני: פונקציות כלליות

3. הפונקציה get_piece

עליכם לממש את הפונקציה get_piece, המחזירה מקטע מתמונה. חתימת הפונקציה צריכה להיות:

def get_piece(image, upper_left, size):

הפונקציה מקבלת 3 פרמטרים:

image – התמונה המקורית ממנה נלקח המקטע

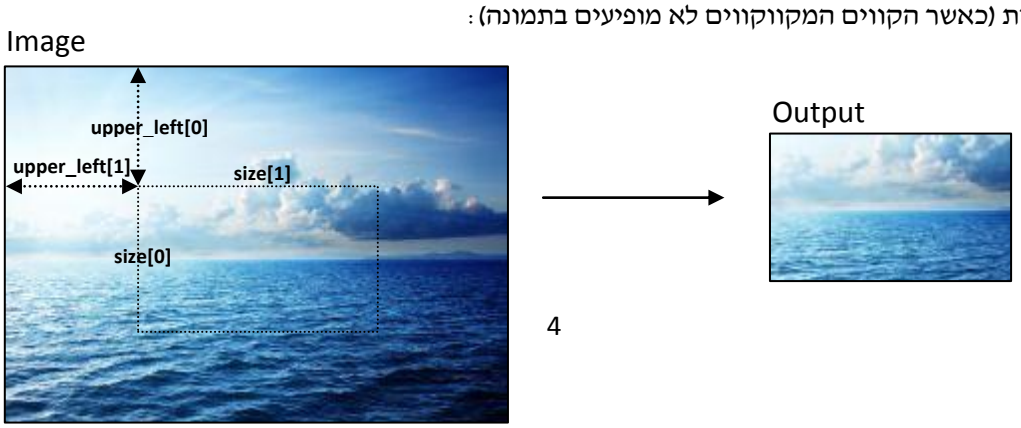
upper_left – tuple של שני מספרים (int) המייצג את מיקום הפיקסל (בתמונה המקורית) השמאלי העליון של המקטע. מיקום הפיקסל נתון לפי (row, column). ניתן להניח כי מיקום פיקסל זה הוא בתוך גבולות התמונה המקורית.

size – tuple של שני מספרים (int) המתאר את גודל המקטע הנלקח. הגודל נתון לפי (height, width).

הפונקציה צריכה להחזיר תמונה (רשימה של רשימות של פיקסלים) בה כל שורה היא רשימה באורך width, ובה height שורות, כאשר הפיקסל במקום ה- [0][0] בתמונה הנוצרת הוא הפיקסל במקום ה- [column] [row] בתמונה המקורית, והפיקסל במקום ה- [width-1][height-1] בתמונה הנוצרת הוא הפיקסל במקום ה-

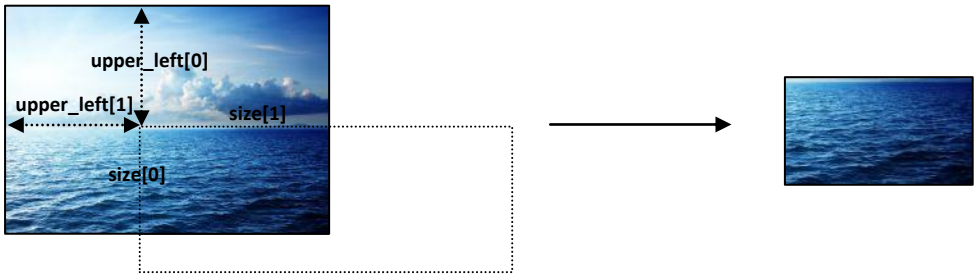
[column+width-1] [row+height-1] בתמונה המקורית.

או בצורה גרפית (כאשר הקווים המקווקווים לא מופיעים בתמונה):



שימו לב: אם מקטע התמונה הנבחר חורג מגבולות התמונה המקורית, **הפונקציה צריכה להחזיר את החלק מהמקטע שנמצא בתמונה!**

כלומר:



4. **הפונקציה** set_piece

עליכם לממש את הפונקציה set_piece, המחליפה מקטע מסוים בתמונה הנתונה בפיקסלים מתמונה אחרת. חתימת הפונקציה צריכה להיות:

```
def set_piece(image, upper_left, piece):
```

הפונקציה מקבלת 3 פרמטרים:

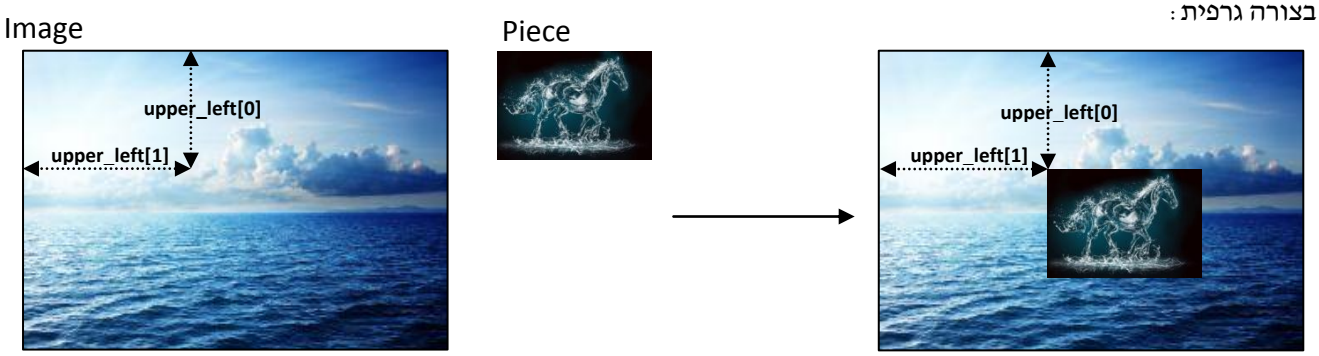
image – תמונת המקור.

upper_left – הפיקסל השמאלי העליון של מקטע התמונה המוחלף, בדומה לפונקציה הקודמת.

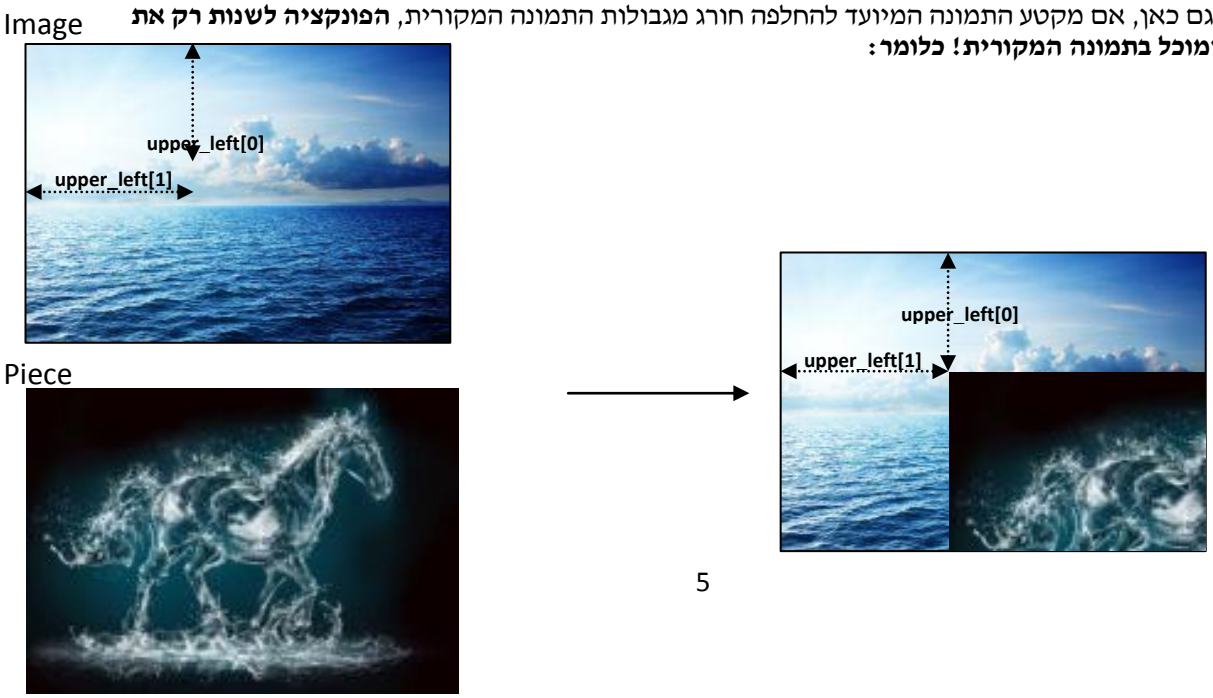
piece – התמונה המחליפה את מקטע התמונה

לפונקציה **אין ערך החזרה** אך היא **משנה** את תמונת המקור כך שבמקטע התמונה שגודלו כגודל התמונה piece, והפיקסל השמאלי העליון שלו הוא הפיקסל upper_left תופיע התמונה piece.

בצורה גרפית:



שימו לב: גם כאן, אם מקטע התמונה המיועד להחלפה חורג מגבולות התמונה המקורית, **הפונקציה לשנות רק את המקטע שמוכל בתמונה המקורית! כלומר:**



5. **הפונקציה** average

עליכם לממש את הפונקציה average, המחזירה את הצבע הממוצע של הפיקסלים בתמונה. חתימת הפונקציה צריכה להיות:

def average(image):

הפונקציה מקבלת תמונה (כרשימה של רשימות) וצריכה להחזיר tuple, שכל אחד משלושת ערכיו (red, green, blue) הוא הממוצע של ערך הצבע המתאים על פני כלל הפיקסלים בתמונה. שימו לב שערך ההחזרה עשוי להיות לא שלם (float).

חלק שלישי: האלגוריתם

6. **הפונקציה** preprocess_tiles

עליכם לממש את הפונקציה preprocess_tiles, המחזירה רשימה של האריחים, עם הצבעים הממוצעים לכל אריח.

חתימת הפונקציה צריכה להיות:

def preprocess_tiles(tiles):

הפונקציה מקבלת פרמטר אחד שהוא רשימה של תמונות (האריחים), והיא צריכה להחזיר את רשימה של tuples, כך שה- tuple ה- [i] ברשימה הוא הצבע הממוצע של tiles[i] (כפי שמוסבר בסעיף הקודם)

7. **הפונקציה** get_best_tiles

עליכם לממש את הפונקציה get_best_tiles, המחזירה רשימה של האריחים שפיקסל הצבע הממוצע שלהם דומה ביותר לפיקסל הצבע הממוצע של תמונת יעד.

חתימת הפונקציה צריכה להיות:

def get_best_tiles(objective, tiles, averages , num_candidates):

הפונקציה מקבלת 4 פרמטרים:

objective – תמונת היעד.

tiles – רשימת האריחים

averages – רשימה המכילה את ממוצעי הצבעים לכל אריח (כפי שיוצרו בסעיף הקודם).

num_candidates – מספר (int) התמונות ברשימת המועמדים המוחזרת.

הפונקציה צריכה להחזיר רשימה באורך num_candidates של האריחים (מתוך רשימת האריחים הנתונה) שהצבע הממוצע שלהם הוא הדומה ביותר לצבע הממוצע של תמונת המטרה.

שימו לב – כיוון שאנו מחפשים מספר קבוע של האריחים הדומים ביותר – אין צורך לבצע מיון. די *למצוא את האריח המתאים ביותר מבין האריחים שלא נמצאו עד כה* num_candidates פעמים.

8. **הפונקציה** choose_tile

עליכם לממש את הפונקציה choose_tile, הבוחרת את האריח המתאים ביותר מתוך רשימת אריחים. חתימת הפונקציה צריכה להיות:

def choose_tile(piece, tiles):

הפונקציה מקבלת 2 פרמטרים: תמונה (שהיא למעשה המקטע מתמונת המקור אותו רוצים להחליף) ורשימת האריחים האפשריים לבחירה שכולם בגודל זהה.

הפונקציה צריכה להחזיר אריח (תמונה) מתוך רשימת האריחים tiles שמרחקו מהמקטע מתמונת המקור מינימאלי (כמתואר בפונקציה compare).

9. הפונקציה make_mosaic

עליכם לממש את הפונקציה make_mosaic, היוצרת תמונת Photomosaic לפי האלגוריתם. חתימת הפונקציה צריכה להיות:

```
def make_mosaic(image, tiles, num_candidates):
```

הפונקציה מקבלת 3 פרמטרים:

image – תמונת המקור.

tiles – רשימת תמונות (האריחים). ניתן להניח כי כל התמונות ברשימה בגודל זהה.

num_candidates – מספר (int) האריחים ברשימה ממנה נבחר האריח המתאים ביותר.

הפונקציה צריכה להחזיר תמונה, שהיא תמונת פסיפס של תמונת המקור בצורה הבאה:

הפונקצייה בוחרת את האריח המתאים למקם כך שהפיקסל השמאלי-עליון שלו ב- [0][0], נניח שגודלו Height x Width. לאחר מכן, הפונקציה בוחרת את האריח המתאים ביותר למקם מימין לאריח הראשון – כלומר שהפיקסל השמאלי העליון שלו ב- [0][Width], וכן הלאה עד כיסוי השורה הראשונה של תמונת המקור.

לאחר מכן, נבחר האריח המתאים ביותר למקם מתחת לאריח הראשון – כלומר כך שהפיקסל השמאלי-עליון שלו ב- [Height]x[0]. לאחר מכן נבחר האריח שלימינו – וכן הלאה שורה שורה עד שמכוסה כלל תמונת המקור.

בחירת האריח המתאים ביותר בכל מקטע מתבצעת בהתאם לאלגוריתם:

ראשית נבחרים num_candidates האריחים שצבע הפיקסל הממוצע שלהם דומה ביותר לצבע הפיקסל הממוצע של המקטע, ומתוך אריחים אלה נבחר האריח שמרחקו למקטע קטן ביותר.

10. הרצת התרגיל

בנוסף לפונקציות שעליכם למלא בקובץ ex6.py , עליכם לכתוב בו main (כפי שנלמד בתרגול 5) שמריץ את התוכנית ושומר את התוצאה בקובץ (לצורך כך, תיאלצו להשתמש בפקודות מהקובץ mosaic.py). הרצת התוכנית מתבצעת על ידי הפקודה:

```
python3 ex6.py <image_source> <images_dir> <output_name>  <tile_height> <num_candidates>
```

כאשר:

image_source - string שהוא השם של תמונת המקור.

images_dir - string שהוא שם התיקייה בה נמצאות התמונות המשמשות כ"אריחים".

output_name – string שהוא השם של תמונת ה Photomosaic הנשמרת.

tile_height - מספר int חיובי שהוא הגובה של כל אריח בתמונת הפסיפס שתתקבל.

num_cadidates – מספר int חיובי המייצג את מספר האריחים בקבוצת המועמדים באלגוריתם.

דוגמא להרצת ex6.py עם פרמטרים שונים:

```
python3 ex6.py "im1.jpg" "images" "mosaic.jpg" 40 10
```

אם נניח כי הקובץ ex6.py נמצא בסיפריית הבית "./" , אזי על ההרצה הנ"ל ליצור תמונת Photomosaic לתמונת המקור "./im1.jpg" ולשמור אותה בקובץ "./mosaic.jpg" (ללא הצגה למסך), כאשר האריחים הם כלל התמונות שנמצאות בסיפרייה "./images/". כל אחד מהאריחים בתמונה הסופית יהיה בגובה 40 פיקסלים, ומספר האריחים

האפשריים אותם יבחר האלגוריתם הוא 10 (להזכירכם, הפרמטרים בהרצת קובץ פייתון שמורים ב- sys.argv). הרצה עם פרמטרים אלה יביאו ליצירת התמונה שבהקדמה.

במידה והקובץ ex6.py מורץ עם **מספר פרמטרים שונה מהדרוש**, עליכם להדפיס למסך הודעה אינפורמטיבית המפרטת את הדרך הנכונה להקרוא לקובץ. לדוגמא, תוכלו להדפיס את המחרוזת:

”Wrong number of parameters. The correct usage is:\nex6.py <image_source> <images_dir> <output_name> <tile_height> <num_candidates>”

שימו לב: כאשר מייבאים את הקובץ ex6.py (על ידי import) הרצת התוכנית והשמירה לקובץ לא אמורים להתבצע!

אתם יכולים להניח תקינות של הקלט (כלומר שכל הנתונים בשורת הפקודה תקינים).

טיפים והנחיות

- על מנת שהחישובים השונים יתבצעו במהירות – מומלץ לעבוד עם תמונה קטנה כתמונת יעד בשלבי כתיבת התרגיל, ועם סט אריחים קטן יחסית, ועם ערך num_candidates נמוך.
- כמו כן – בתהליך הכתיבה, מומלץ לקבל חיווי (הדפסת הודעה, לדוגמא) בשלבים שונים של ריצת אלגוריתם א'. למשל אחרי בחירת כל אריח. כך תוכלו לדעת שהאלגוריתם "מתקדם". **אך שימו לב לא להשאיר חיוויים כאלה כ"זבל" בתרגיל הסופי!**
- ניתן להניח תקינות הקלטים לכל אחד מהסעיפים (בהתאם להגדרת הפרטנית של כל סעיף). בפרט, ניתן להניח כי כל התמונות ניתנות בפורמט תקין, כי מספר האריחים הנתונים אינו קטן מ- num_candidates, כי כל הרשימות הן אכן רשימות לא ריקות, וכל ה-tuples בפורמט הנתון.
- למעט במקומות **המוגדרים**, **אין לשנות את הפרמטרים המתקבלים באף אחת מהפונקציות.**
- הקפידו לכתוב את הקוד שלכם בצורה מדויקת וברורה, ולשים לב לאינדקסים וגבולות של לולאות וקונטיינרים.
- לצורך פיתרון התרגיל תוכלו גם להשתמש במודולים sys ו-copy כפי שראיתם בתרגולים.

נהלי הגשה

הלינק להגשה של התרגיל הוא תחת השם: ex6

בתרגיל זה עליכם להגיש את הקבצים הבאים:

1. ex6.py – עם המימושים שלכם לפונקציות.
2. README (על פי פורמט ה-README לדוגמא שיש באתר הקורס, ועל פי ההנחיות לכתיבת README המפורטות בקובץ נהלי הקורס).

יש להגיש קובץ zip הנקרא ex6.zip המכיל בדיוק את שני הקבצים הנ"ל.

בהצלחה! ☺