

פעילות 5 - ייצוג מידע ויזואלי במחשב

בפעילות זו נתרגל כתיבת קוד בשפת python לעבודה עם תמונות, נלמד כיצד פותחים קובץ תמונה קיים, כיצד בנוי הקובץ וכיצד לעבד את הנתונים בקובץ. בפעילות זו נעשה שימוש ב- PIL שהם ראשי התיבות של Python Imaging Library שהיא ספרייה ב- Python המספקת יכולות עיבוד תמונות.

מקורות:

<https://matplotlib.org/>

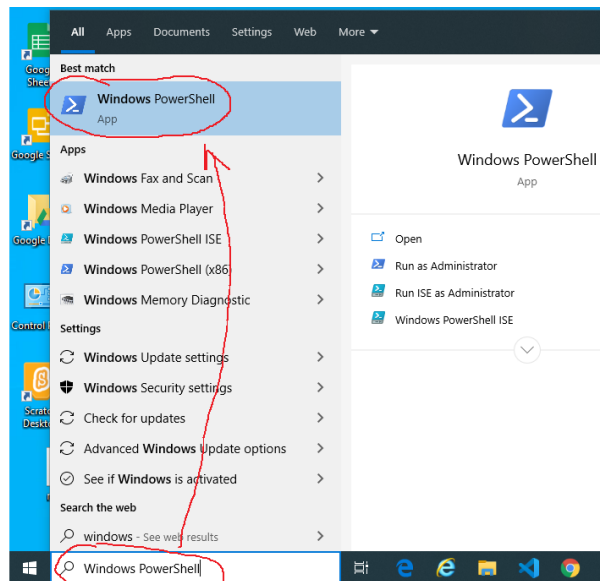
<https://stackoverflow.com/questions/46615554/how-to-display-multiple-images-in-one-figure-correctly/46616645>

https://matplotlib.org/users/pyplot_tutorial.html

התקנות

Matplotlib היא ספרייה ב- Python להצגת תמונות/גרפים ונתונים כקובץ תמונה.
Pillow היא ספרייה ב- Python לטיפול בתמונות (Python Imaging Library).
NumPy היא הספרייה המתמטית של - Python.

נפתח את חלון הפקודות Windows PowerShell על ידי ההוראה הבא:



נבדוק תחילה שהמפרש של Python מותקן במחשב, נעשה זאת על ידי ההוראה:

```
> python --version
```

```
Windows PowerShell
PS C:\Users\gadi> python --version
Python 3.7.4
PS C:\Users\gadi>
```

התקנת הספרייה תתבצע על ידי ההוראה הבאה:

```
> python -m pip install -U pip
> python -m pip install -U matplotlib
> python -m pip install -U Pillow
> python -m pip install -U numpy
```

נקבל כפלט את המסך הבא:

```
Windows PowerShell
PS C:\Users\gadi> python -m pip install -U pip
Requirement already up-to-date: pip in c:\users\gadi\appdata\local\
PS C:\Users\gadi> python -m pip install -U matplotlib
Collecting matplotlib
  Downloading https://files.pythonhosted.org/packages/71/13/0720e50
matplotlib-3.1.1-cp37-cp37m-win32.whl (8.9MB)
    | 8.9MB 344kB/s
Collecting numpy>=1.11
  Downloading https://files.pythonhosted.org/packages/ce/ad/2e88f36
umpy-1.17.4-cp37-cp37m-win32.whl (10.7MB)
    | 10.7MB 364kB/s
Collecting kiwisolver>=1.0.1
  Downloading https://files.pythonhosted.org/packages/20/6a/e5fff2e
kiwisolver-1.1.0-cp37-none-win32.whl (44kB)
    | 51kB 328kB/s
Requirement already satisfied, skipping upgrade: python-dateutil>=2
e-packages (from matplotlib) (2.8.0)
Collecting cyclor>=0.10
```

משימה 1: פתיחת קובץ תמונה

נכנס לאתר שיתוף התמונות <https://pixabay.com> ונוריד למחשב תמונה. שימו לב שמדובר באתר המאפשר להוריד תמונה ממאגר תמונות ענקי שנכון להיום התמונות שבו ניתנות בחינם ללא זכויות יוצרים.

לדוגמה הורדתי למחשב את התמונה הבא תחת השם train.jpg



Image by [David Mark](#) from [Pixabay](#)

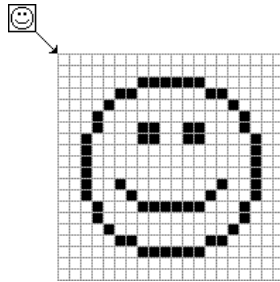
כדי לפתוח את התמונה תוך שימוש בספריה PIL נכתוב את הקוד הבא:

```
from PIL import Image

img = Image.open("data/train.jpg")
width, height = img.size
print(width, height)
img.show()
```

פלט התוכנית יהיה **1920 1392** כך שהערך 1920 מייצג את רוחב התמונה ו-1392 מייצג את גובה התמונה. כלומר מדובר על תמונה הכוללת 2,672,640 פיקסלים.

פיקסל היא יחידת מידע גרפית בסיסית במחשב, המתארת נקודה בתמונה דיגיטלית. החלק הקטן ביותר של הקווים, התווים, והצורות שעל צג המחשב או בתמונה המודפסת. כל תמונה בנויה ממאות אלפי עד עשרות מיליוני נקודות קטנות, ונקודה זו היא יחידת תמונה. (מקור: ויקיפדיה <https://he.wikipedia.org/wiki>)



מקור התמונה: https://commons.wikimedia.org/wiki/File:SMILE_FACE_16x16_PIXEL_EXAMPLE1.PNG (לשימוש חוזר עם אפשרות לשינויים)

הפעולה open של המחלקה Image מקבלת מחרוזת המייצגת את מיקום התמונה. הפעולה מחזירה עצם בשם img המייצג את התמונה.

הפעולה img.size מחזירה את האורך והרוחב של התמונה.

הפעולה img.show מציגה את התמונה.

משימה 2: פעולות על תמונה

המחלקה Image כוללת מספר פעולות המאפשרות לנו לבצע עיבוד תמונה. נדגים דרך הקוד הבא חלק מהם:

```
from PIL import Image

img = Image.open("data/train.jpg")
width, height = img.size
img45 = img.rotate(45)
area = (width/2-200, height/2-200, width/2+200, height/2+200)
img_crop = img.crop(area)
```

```

newsize = (width//4, height//4)
img_resize = img.resize(newsize)
img_transpose = img.transpose(Image.FLIP_LEFT_RIGHT)

img45.save("data/train45.jpg")
img_crop.save("data/train_cropped.jpg")
img_resize.save("data/train_resized.jpg")
print("Old image size:", img.size)
print("New image size:", img_resize.size)
img45.show()
img_crop.show()
img_resize.show()
img_transpose.show()

```

הפעולה `img.rotate` מקבלת זווית ומחזירה תמונה הכוללת הטייה בזווית הרצויה



הפעולה `img.crop` מבצעת חיתוך של קטע מתמונה על פי פרמטר שנקלט דרך המשתנה `area`

```

area = (width/2-200, height/2-200, width/2+200, height/2+200)

```



הפעולה `img.resize` מבצעת שינוי בגודל התמונה על פי פרמטר המתקבל דרך המשתנה `newsize`

```

newsize = (width//4, height//4)

```

```
img_resize = img.resize(newsize)
```

הפעולה `img.transpose` מבצעת היפוך לתמונה על פי הפרמטר שהיא מקבלת. בדוגמה שלנו ההפוך יהיה מימין לשמאל (`Image.FLIP_LEFT_RIGHT`)

```
img_transpose = img.transpose(Image.FLIP_LEFT_RIGHT)
```



הפעולה `save` שומרת את התמונה כקובץ בשם המסופק לפעולה כפרמטר

```
img45.save("data/train45.jpg")
```

משימה 3: המרת צבעים בתמונה

נדגים כיצד מעבירים תמונה צבעונית לשחור לבן:

```
from PIL import Image
img = Image.open("data/train.jpg")

img1 = img.convert("LA")
img2 = img.convert("1")
img1.show()
img2.show()
```

הפעולה `img.convert` מקבלת פרמטר המגדיר את אופי ההמרה. הפעולה מחזירה עצם תמונה לאחר ההמרה. נדגים את פלט התוכנית לאחר ההמרה:



פירוק התמונה לצבעים המרכיבים אותה

כל פיקסל בתמונה בנוי משלוש ערכים. הראשון עוצמת האדום, השני מידת הירק והשלישי הכחול. כדי להבין עיקרון זה נבחן תוכנית שמפרקת את התמונה לשלוש תמונות שונות כאשר בכל תמונה רואים צבע אחד בלבד. להלן קוד התוכנית:

```
from PIL import Image
img = Image.open("data/train.jpg")
width, height = img.size
data = img.getdata()
R = []
G = []
B = []
for i in data:
    R.append((i[0], 0, 0))
    G.append((0, i[1], 0))
    B.append((0, 0, i[2]))
imgR = Image.new(mode = "RGB", size = (width, height))
imgG = Image.new(mode = "RGB", size = (width, height))
imgB = Image.new(mode = "RGB", size = (width, height))
imgR.putdata(R)
imgG.putdata(G)
imgB.putdata(B)
imgR.show()
imgG.show()
imgB.show()
```

נקבל את הפלט הבא:



הפעולה `img.getdata` הופכת את הערכים של כל אחד מהפיקסלים המרכיבים את התמונה למערך של מספרים כאשר כל פיקסל מקבל 3 מספרים המייצגים את שלוש הצבעים.

מבנה הנתונים של תמונה השמור במחשב עם סיומת jpg אינו בנוי כמערך של פיקסלים כאשר לכל פיקסל שלוש צבעים. הסיבה לכך היא שקובץ jpg שומר את המידע על הפיקסלים במבנה דחוס וזאת במטרה לשמור במחשב קובץ קטן ככל הניתן תוך כדי שמירה על איכות התמונה. הפעולה getdata מבצעת הליך המשחזר על הערכים של כל פיקסל.

למידע נוסף על מבנה קובץ jpg ניתן לקבל בקישור הבא:

<https://he.wikipedia.org/wiki/JPEG>

לאחר פענוח מערך הפיקסלים על ידי הפעולה getdata נגדיר 3 מערכים בשמות R G ו-B ונשמור בכל מהם צבע בודד.

```
R = []
G = []
B = []
for i in data:
    R.append((i[0], 0, 0))
    G.append((0, i[1], 0))
    B.append((0, 0, i[2]))
```

נעזר בקוד הבא כדי להבין טוב יותר כיצד נראה כל פיקסל:

```
from PIL import Image
img = Image.open("data/train.jpg")
data = img.getdata()
for i in range(100):
    print("pixel", i, "=", data[i])
```

פלט תוכנית זו יציג לנו את 100 הפיקסלים הראשונים של הקובץ train.jpg מיתוך 2,672,640 הפיקסלים המרכיבים אותה.

```
pixel 1 = (0, 41, 117) pixel 26 = (0, 42, 118) pixel 51 = (0, 42, 118) pixel 76 = (0, 42, 118)
pixel 2 = (0, 41, 117) pixel 27 = (0, 42, 118) pixel 52 = (0, 42, 118) pixel 77 = (0, 42, 118)
pixel 3 = (0, 41, 117) pixel 28 = (0, 42, 118) pixel 53 = (0, 42, 118) pixel 78 = (0, 42, 118)
pixel 4 = (0, 41, 117) pixel 29 = (0, 42, 118) pixel 54 = (0, 42, 118) pixel 79 = (0, 42, 118)
pixel 5 = (0, 41, 117) pixel 30 = (0, 42, 118) pixel 55 = (0, 42, 118) pixel 80 = (0, 42, 118)
pixel 6 = (0, 41, 117) pixel 31 = (0, 42, 118) pixel 56 = (0, 42, 118) pixel 81 = (0, 42, 118)
pixel 7 = (0, 41, 117) pixel 32 = (0, 41, 117) pixel 57 = (0, 42, 118) pixel 82 = (0, 42, 118)
pixel 8 = (0, 41, 117) pixel 33 = (0, 41, 117) pixel 58 = (0, 42, 118) pixel 83 = (0, 42, 118)
pixel 9 = (0, 41, 117) pixel 34 = (0, 41, 117) pixel 59 = (0, 42, 118) pixel 84 = (0, 42, 118)
pixel 10 = (0, 41, 117) pixel 35 = (0, 41, 117) pixel 60 = (0, 42, 118) pixel 85 = (0, 42, 118)
pixel 11 = (0, 41, 117) pixel 36 = (0, 41, 117) pixel 61 = (0, 42, 118) pixel 86 = (0, 42, 118)
pixel 12 = (0, 41, 117) pixel 37 = (0, 41, 117) pixel 62 = (0, 42, 118) pixel 87 = (0, 42, 118)
pixel 13 = (0, 41, 117) pixel 38 = (0, 41, 117) pixel 63 = (0, 42, 118) pixel 88 = (0, 42, 118)
pixel 14 = (0, 41, 117) pixel 39 = (0, 41, 117) pixel 64 = (0, 42, 118) pixel 89 = (0, 42, 118)
pixel 15 = (0, 41, 117) pixel 40 = (0, 41, 117) pixel 65 = (0, 42, 118) pixel 90 = (0, 42, 118)
pixel 16 = (0, 41, 117) pixel 41 = (0, 41, 117) pixel 66 = (0, 42, 118) pixel 91 = (0, 42, 118)
pixel 17 = (0, 41, 117) pixel 42 = (0, 41, 117) pixel 67 = (0, 42, 118) pixel 92 = (0, 42, 118)
pixel 18 = (0, 41, 117) pixel 43 = (0, 41, 117) pixel 68 = (0, 42, 118) pixel 93 = (0, 42, 118)
pixel 19 = (0, 41, 117) pixel 44 = (0, 41, 117) pixel 69 = (0, 42, 118) pixel 94 = (0, 42, 118)
pixel 20 = (0, 41, 117) pixel 45 = (0, 41, 117) pixel 70 = (0, 42, 118) pixel 95 = (0, 42, 118)
pixel 21 = (0, 41, 117) pixel 46 = (0, 41, 117) pixel 71 = (0, 42, 118) pixel 96 = (0, 42, 118)
pixel 22 = (0, 41, 117) pixel 47 = (0, 41, 117) pixel 72 = (0, 42, 118) pixel 97 = (0, 42, 118)
pixel 23 = (0, 41, 117) pixel 48 = (0, 42, 118) pixel 73 = (0, 42, 118) pixel 98 = (0, 42, 118)
pixel 24 = (0, 42, 118) pixel 49 = (0, 42, 118) pixel 74 = (0, 42, 118) pixel 99 = (0, 42, 118)
pixel 25 = (0, 42, 118) pixel 50 = (0, 42, 118) pixel 75 = (0, 42, 118)
```

תרגיל

תמונה בינארית היא תמונה שבה כל פיקסל מקבל את הערך 1 או 0. כלמור כל פיקסל יכול להיות שחור או לבן בלבד. זאת בניגוד לתמונת שחור לבן שבה יש מגוון ערכים בין שחור ללבן.

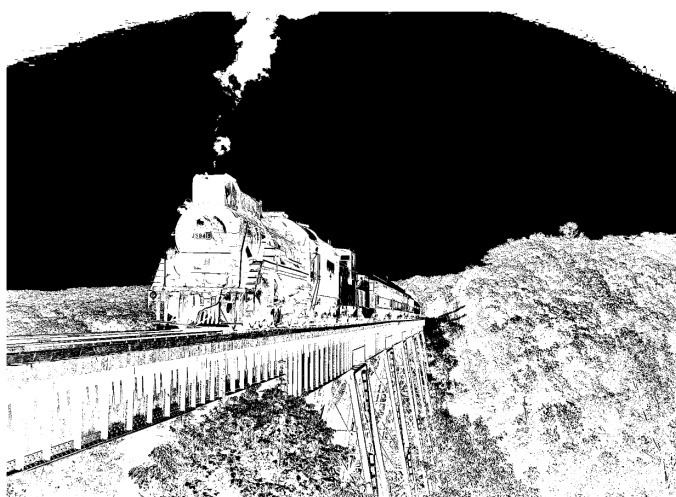
כתבו קוד בשפת python הממיר תמונה צבעונית לתמונה בינארית.

פתרון

להלן קוד התוכנית:

```
from PIL import Image
img = Image.open("data/train.jpg")
width, height = img.size
data = img.getdata()
BIN = []
for i in data:
    avg = (i[0]+i[1]+i[2])/3
    if avg<60:
        BIN.append((255, 255, 255))
    else:
        BIN.append((0, 0, 0))
imgBIN = Image.new(mode = "RGB", size = (width, height))
imgBIN.putdata(BIN)
imgBIN.show()
```

להלן דוגמה לפלט התוכנית:



תרגיל

שנה את קוד התרגיל הקודם כדי ליצור תמונה בשחור לבן במקום תמונה בינארית.

פתרון

להלן קוד התוכנית:

```
from PIL import Image
img = Image.open("data/train.jpg")
width, height = img.size
data = img.getdata()
BIN = []
for i in data:
    avg = int((i[0]+i[1]+i[2])/3)
    BIN.append((avg, avg, avg))
imgBIN = Image.new(mode = "RGB", size = (width, height))
imgBIN.putdata(BIN)
imgBIN.show()
```

להלן דוגמה לפלט התוכנית:



תרגיל

בחזית הרכבת יש לוחית זיהוי הכוללת מספר. כתבו תוכנית שתציג תמונה של לוחית הזיהוי כתמונה בינארית.

```

from PIL import Image
img = Image.open("data/train.jpg")
width, height = img.size
area = (500, 580, 640, 650)
img_crop = img.crop(area)

data = img_crop.getdata()
BIN = []
for i in data:
    avg = (i[0]+i[1]+i[2])/3
    if avg<60:
        BIN.append((255, 255, 255))
    else:
        BIN.append((0, 0, 0))
width, height = img_crop.size
imgBIN = Image.new(mode = "RGB", size = (width, height))
imgBIN.putdata(BIN)
newsize = (width*3, height*3)
imgBIN = imgBIN.resize(newsize)
imgBIN.show()

```

להלן דוגמה לפלט התוכנית:



משימה 4: יצירת תמונה תוך שימוש בכלי numpy

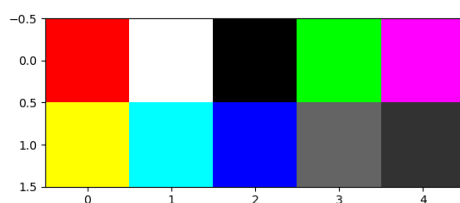
נדגים יצירת תמונה חדשה המופקת ממערך numpy array

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

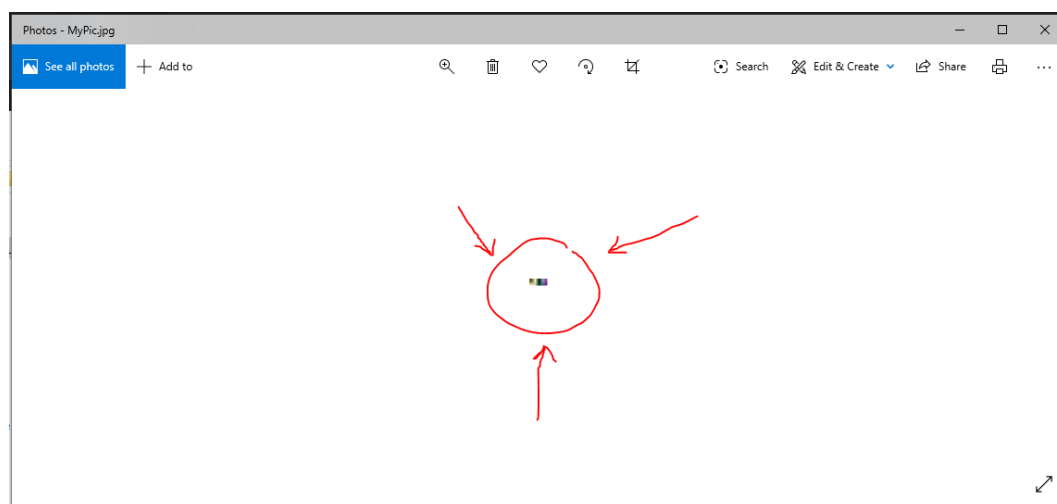
pixels = [
    [ [255,0,0],[255,255,255],[0,0,0],[0,255,0],[255,0,255] ],
    [ [255,255,0],[0,255,255],[0,0,255],[100,100,100],[50,50,50] ]
]

array = np.array(pixels, dtype=np.uint8)
new_image = Image.fromarray(array)
new_image.save("MyPic.jpg")
plt.imshow(new_image)
plt.show()
```

נקבל את הפלט הבא:



כמו כן נוצר לנו קובץ בשם MyPic.jpg הנראה כך בהגדלה הכי גדולה שניתן:



נעזר בידע שלמדנו בעבודה ב-numpy ונייצר מערך שלוש על שלוש מימדים באופן הבא:

```
pixels = np.zeros([400,400,3],dtype=np.uint8)
```

כלומר 400 שורות על 400 עמודות כאשר כל רשומה כוללת 3 תאים. בכל תא נשמור משתנה מטיפוס uint8 כולמר ערך בין 0 ל- 255 כפי שכבר ראינו באפיון רמת הצבע של כל פיקסל.

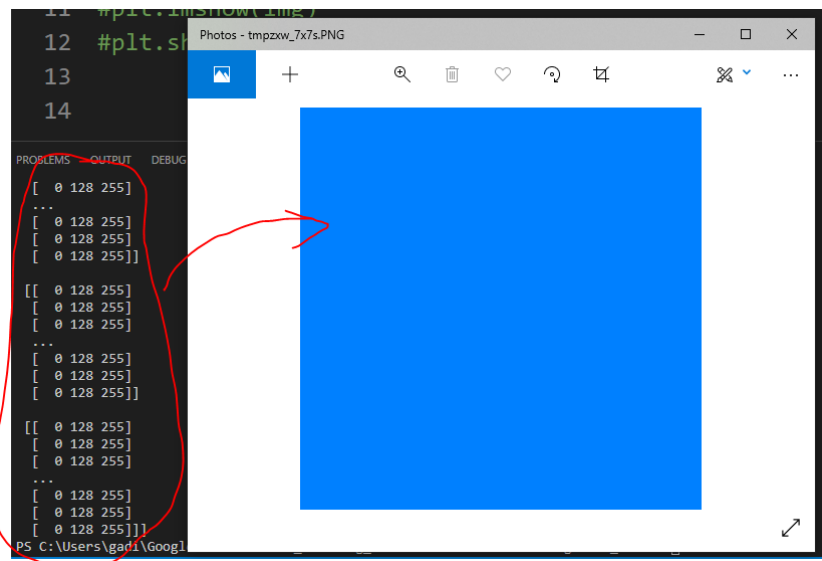
להלן קוד התוכנית:

```
from PIL import Image
import numpy as np

pixels = np.zeros([400,400,3],dtype=np.uint8)
pixels[:,:] = [0, 128, 255]
print(pixels)

img = Image.fromarray(pixels)
img.show()
```

נקבל את הפלט הבא:



ניתן לראות שקיבלנו מערך numpy הכולל 160,000 פיקסלים שכל אחד מהם מכיל את הערכים:

[0 128 255]

עד כה השתמשנו בפעולה show כדי להציג את התמונה. ניתן להיעזר בספריה matplotlib גם כן להציג תמונות. להלן דוגמה:

```

from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

pixels = np.zeros([400,400,3],dtype=np.uint8)
pixels[:,:] = [0, 128, 255]
print(pixels[:,:])

img = Image.fromarray(pixels)
plt.imshow(img)
plt.show()

```

ניתן ליצור תמונות הכוללת יותר מצבע אחד. להלן דוגמה:

```

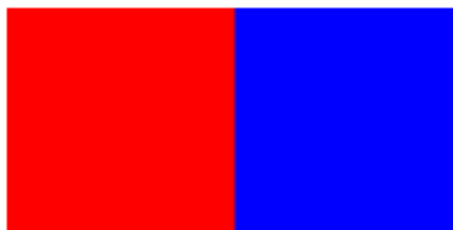
from PIL import Image
import numpy as np

pixels = np.zeros([100,200,3],dtype=np.uint8)
pixels[:, :100] = [255, 0, 0]
pixels[:, 100:] = [0, 0, 255]

img = Image.fromarray(pixels)
img.show()

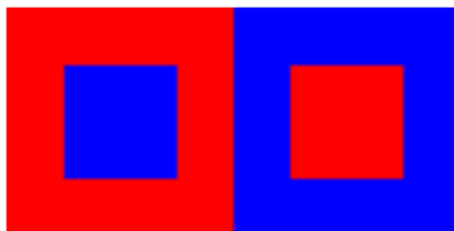
```

נקבל את הפלט הבא:



תרגיל

צור תמונה הכוללת את הצורה הזו:



פתרון

```
from PIL import Image
import numpy as np

pixels = np.zeros([100,200,3],dtype=np.uint8)
pixels[:,0:100] = [255, 0, 0]
pixels[:,100:] = [0, 0, 255]
pixels[25:75,25:75] = [0, 0, 255]
pixels[25:75,125:175] = [255, 0, 0]
print(pixels.shape,type(pixels))
img = Image.fromarray(pixels)
img.show()
```

משימה 5: קריאת תמונה לתוך מערך numpy

קריאת קובץ תמונה לתוך מערך תוך כדי עיבוד בסיסי של התמונה:

```
from PIL import Image
import numpy as np

img = Image.open("data/train.jpg")
img.load()
data = np.array(img, dtype=np.uint8)

data[0:100,0:100]=[255, 0, 0]
data[100:200,100:200]=[0, 255, 0]
```

```
data[200:300,200:300]=[0, 0, 255]
data[300:400,300:400]=[255, 0, 255]
data[400:500,400:500]=[0, 255, 255]
data[500:600,500:600]=[255, 255, 255]
```

```
img = Image.fromarray(data)
img.show()
```

נקבל את הפלט הבא:



נדגיש שההוראה שבקוד הבא:

```
data[0:100,0:100]=[255, 0, 0]
```

זוהי לשלוש ההוראות שבקוד הנ"ל:

```
data[0:100,0:100,0]=255
data[0:100,0:100,1]=0
data[0:100,0:100,2]=0
```

משימה 6: חיפש מאפיינים לזיהוי פריטים בתמונה (הכנות למשחק ים יבשה בגרסת מכונה לומדת)

לצורך זיהוי פריטים בתמונה או בקיצור זיהוי תמונות על ידי מכונה לומדת יש צורך לארגן מערך של תמונות מתאימות. נבחן את הסוגיה על ידי התבוננות על התמונה הבא:



Image by [maja7777](#) from [Pixabay](#)

כיצד אם כן נאפיין מי זה הכלב ומי החתול בתמונה? האם כל הלבנים חתולים? האם כל בעלי הפרווה כלבים? האם לכל מי שיש אף בצורת משולש הוא חתול? האם כל מי שיש לו אוזן ורודה הוא כלב?

נראה שיש צורך לחדד את הנושא ולעזור לנו לאפיין את הגורמים המבדילים בין פריטים. תהליך זה נקראה במכונות לומדות: Feature Selection in Machine Learning והוא בא לספק לנו כלי להגדרת המאפיינים שיכולים להבדיל בין דברים שאנו רוצים לסווג.

במקרה שלנו כאשר מדובר בתמונות, אנו מתייחסים ליכולת לאפיין מספר תכונות בתוך תמונה במטרה לסווג פריטים בתוך תמונה ברמת וודאות גבוהה.

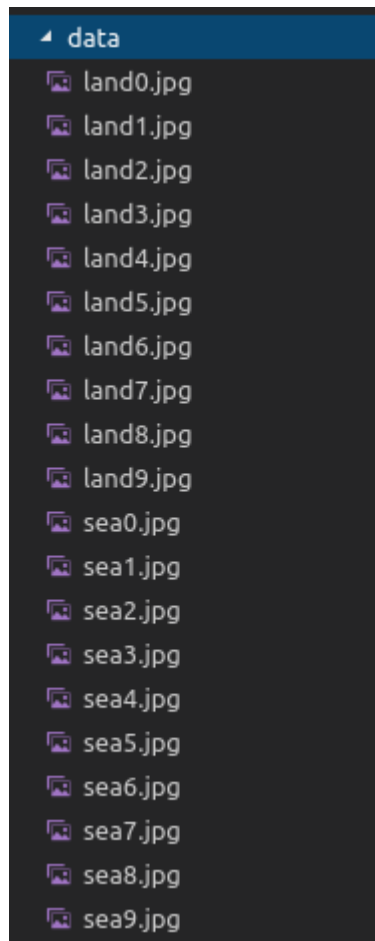
לצורך התנסות בנושא ובמטרה לתרגל עבודה עם תמונות נחפש תכונות Features שיכולים להבדיל בין תמונות ים לבין תמונות יבשה. בקיצור משחק ים יבשה בגרסת מכונה לומדת (נו טוב לא ממש דומה למשחק הילדים ששיחקנו פעם).

נוריד מהאינטרנט 10 תמונות של ים ו-10 תמונות של יבשה ונשמור אותם בתיקייה בשם data תחת השמות sea ומספר התמונה עבור כל התמונות המתארות ים ו- land ומספר התמונה עבור כל התמונות המתארות יבשה.

ניתן להוריד תמונות שאינם מוגנות מזכויות יוצרים מאתר:

<https://pixabay.com/>

להלן דוגמה למבנה הקבצים בתיקייה data הכולל את התמונות.



לאחר שהורדנו למחשב 20 תמונות נחפש מאפיינים שיכולים להבדיל בין תמונות נוף של ים ליבשה. האפשרות שנבדוק כמאפיין אפשרי יהיה היחס בין ממוצע הצבע הירוק בתמונות יבשה לבין יחס ממוצע הצבע הכחול בתמונות היבשה. לשם כך נכתוב קטע קוד המציג מתוך תמונה בודדת של ים ומתוך תמונה אחת של יבשה את ממוצע הצבע הירוק ואת ממוצע הצבע הכחול בכל אחת מהם:

```
from PIL import Image
import numpy as np

img = Image.open("data/sea0.jpg")
img.load()
sea0 = np.array(img, dtype=np.uint8)

print ("sea0 - Green =", sea0[:, :, 1].sum() / sea0[:, :, 1].size)
print ("sea0 - blue =", sea0[:, :, 2].sum() / sea0[:, :, 2].size)

img = Image.open("data/land0.jpg")
img.load()
```

```
land0 = np.array(img, dtype=np.uint8)

print ("land0 - Green =" , land0[:, :, 1].sum() / land0[:, :, 1].size)
print ("land0 - blue =" , land0[:, :, 2].sum() / land0[:, :, 2].size)
```

נקבל את הפלט הבא:

```
sea1 - Green = 123.1879816955684
sea1 - blue = 153.2496296965318
land1 - Green = 145.8581862745098
land1 - blue = 73.04027328431373
```

להלן 2 התמונות:



Image by [Sasin Tipchai](#) from [Pixabay](#)

Image by [Michelle Maria](#) from [Pixabay](#)

שאלה: האם זה הגיוני שהאלגוריתם שלנו מצא צבע ירוק בתמונה של הים?

תשובה: ברור שכן! כל נקודה על גבי המסך מורכבת מפיקסלים. כל פיקסל מורכב מאדום, ירוק וכחול. אז ברור שבחלק מהפיקסלים בייחוד בלבן יש מרכיב משמעותי של ירוק בתוך כל אחד מהם.

בשלב הבא נבנה 2 רשימות הכוללות את ממוצע הצבע ירוק בכל אחד מ-20 התמונות לפי הפרדה בין תמונות ים לתמונות יבשה:

```
from PIL import Image
import numpy as np

sea_list = list()
land_list = list()

for i in range(10):
    img = Image.open("data/sea" + str(i) + ".jpg")
    img.load()
    data = np.array(img, dtype=np.uint8)
```

```

sea_list.append(data[:, :, 1].sum() / data[:, :, 1].size)

img = Image.open("data/land" + str(i) + ".jpg")
img.load()
data = np.array(img, dtype=np.uint8)
land_list.append(data[:, :, 1].sum() / data[:, :, 1].size)

print("\nsea_list:\n", sea_list)
print("\nland_list:\n", land_list)

```

נקבל את הפלט הבא:

```

sea_list:
[130.4101615341768, 131.96297276468894, 154.34126688346825, 111.7069587628866, 115.01519280118855, 155.91324930555555, 129.21721614583333, 135.44244079172367, 137.0602974487363, 107.63884548611111]
land_list:
[121.23325928276738, 86.22154513888889, 147.22994865262908, 117.407721875, 122.98220265188997, 206.6399573089777, 154.72347739092003, 111.65416666666667, 135.62284365079364, 93.52566318926975]

```

בשלב הבא נבנה גרפים המציגים את הנתונים שקיבלנו מהשלב הקודם אך הפעם נציג את היחס בין ממוצע כל 2 צבעים בכל תמונה לפי חלוקה בין תמונות ים לתמונות יבשה:

```

from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

sea_colors = list()
for i in range(10):
    img = Image.open("data/sea" + str(i) + ".jpg")
    img.load()
    data = np.array(img, dtype=np.uint8)
    sea_colors.append([data[:, :, color].sum() / data[:, :, color].size for color in range(3)])

land_colors = list()
for i in range(10):
    img = Image.open("data/land" + str(i) + ".jpg")
    img.load()
    data = np.array(img, dtype=np.uint8)
    land_colors.append([data[:, :, color].sum() / data[:, :, color].size for color in range(3)])

```

```
sea_array = np.array(sea_colors)
land_array = np.array(land_colors)
```

```
plt.figure(1)
```

```
plt.subplot(131)
x1 = sea_array[:,0]
y1 = sea_array[:,1]
x2 = land_array[:,0]
y2 = land_array[:,1]
plt.plot(x1, y1, 'bo', x2, y2, 'r+')
plt.xlabel("red")
plt.ylabel("green")
plt.title("Sea or Land option 1")
```

```
plt.subplot(132)
x1 = sea_array[:,0]
y1 = sea_array[:,2]
x2 = land_array[:,0]
y2 = land_array[:,2]
plt.plot(x1, y1, 'bo', x2, y2, 'r+')
plt.xlabel("red")
plt.ylabel("blue")
plt.title("Sea or Land option 2")
```

```
plt.subplot(133)
x1 = sea_array[:,1]
y1 = sea_array[:,2]
x2 = land_array[:,1]
y2 = land_array[:,2]
plt.plot(x1, y1, 'bo', x2, y2, 'r+')
plt.xlabel("green")
plt.ylabel("blue")
```

```
plt.title("Sea or Land option 3")
plt.show()
```

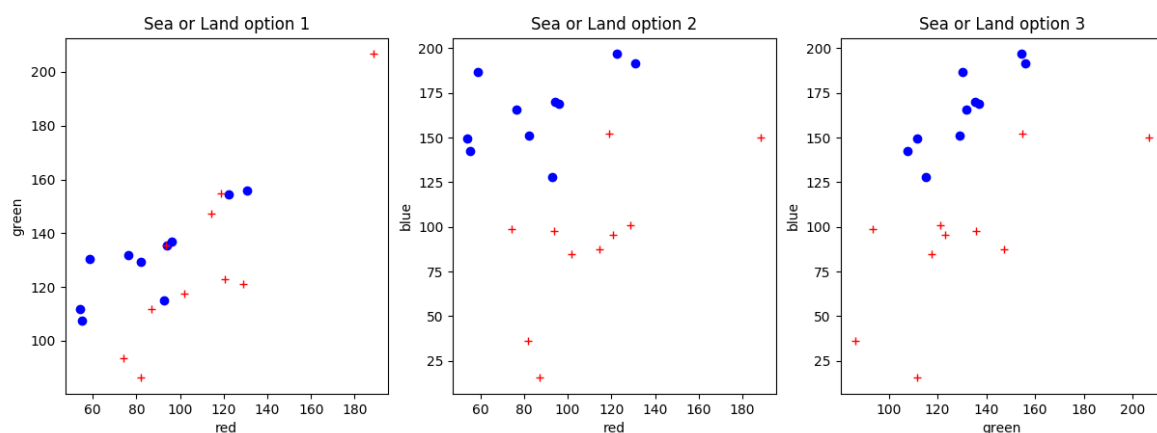
ההוראה:

```
land_colors.append([data[:, :, color].sum() / data[:, :, color].size for color in range(3)])
```

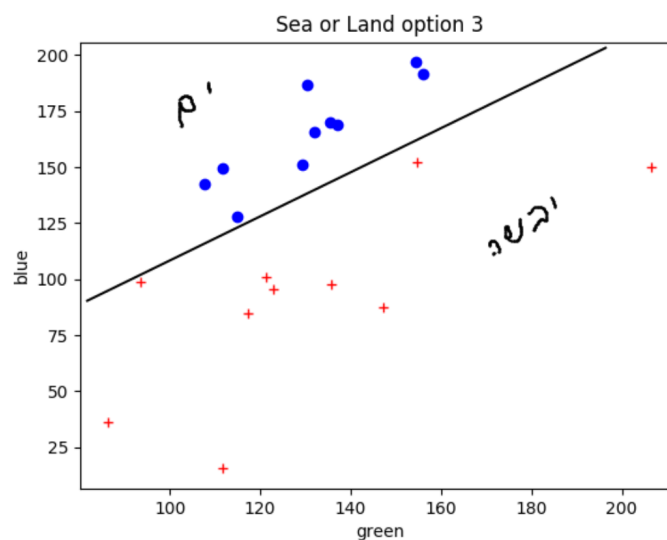
היא גרסה מקוצרת של כתיבת לולאת for בשפת Python. לפרטים ניתן לקבל בקישור הבא:

<https://stackoverflow.com/questions/52459681/for-loop-one-line-in-python/52460395>

נקבל את הפלט הבא:



ניתן ללמוד מניתוח מאפייני הצבעים שבתמונות מכר שמאפיין היחס בין צבע ירוק לצבע כחול יכול לספק לנו יכולת סיווג גבוה כיוון שעל פי מדגם 20 התמונות שלנו ניתן לסווג תמונות שלא מוכרות למוכנה שלנו על פי יחס זה. נדגים זאת על ידי הגרף הבא:



בפעילות זו למדנו כיצד מייצגים תמונות במחשב, תוך כדי מציאת גורם מבדיל בין תמונות נוף ים לבין תמונות נוף יבשתי.

ראינו שעל ידי בדיקת כמות הצבע הירוק וכמות הצבע הכחול בכל תמונות ניתן לזהות בסבירות טובה, יחסית למדגם שעשינו, מה הנוף בתמונה (ים או יבשה). בפעילויות הבאות נמשיך ונפתח את הרעיון כדי לבנות מכונה לומדת המסוגלת לזהות בין תמונות ים לבין תמונות יבשה.

בפעילות הבאה, נתמקד בעקרונות המתמטיים המאפשרים לנו למצוא את הקו הלינארי מייצג את ההפרדה בין 2 סוגי התמונות. נעשה זאת על ידי פעולה מתמטית בשם רגרסיה לינארית.

תנאי השימוש

תנאי השימוש במסמך זה הם לפי הסטנדרט הבא:

You are free:

to Share – to copy, distribute and transmit the material
to Remix – to adapt the material

Under the following conditions:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.