

אוניברסיטת תל-אביב
הפקולטה להנדסה, המחלקה להנדסת תעשייה
המעבדה ליצור בעזרת מחשב

רובוטיקה - מבוא

שימוש ב-Verifier:

[סרטון הסבר](#).

נושאי המעבדה:

מבנה הרובוט

מערכת קואורדינטות: עולם הרובוט ופרקי הנעה.

מטרות המעבדה:

לימוד מבנה כללי של הרובוט

תכנות פשוט בשפת python, תוך שימוש במערכת קואורדינטות שונות.

מהלך המעבדה:

במהלך מעבדה זו עליכם לבצע שתי מטלות:

מטלה ראשונה - תכנית מיון בסיסית אשר תמייין 3 סוגי חלקים לפי גודלם.

מטלה שנייה - בקרת תנועה - הרובוט (אוחז בעפרון) יצייר ציור, שיינתן על ידי המדריכים במעבדה.

שימו לב: עיקר מעבדה זו היא בתכנות הרובוט. לכן חשוב מאוד לעבור על פקודות ה-python שבמהלך התדריך לשליטה ברובוט ולהבין אותם לפני תחילת העבודה.

שימו לב: במעבדה נכתוב קוד python בסביבת העבודה PyCharm.

את PyCharm יש להפעיל רק אחרי איפוס הרובוט!

סביבת עבודה זו מאפשרת לכתוב קוד בקלות, באמצעות השלמת פקודות אוטומטית ותיקון שגיאות. למי שלא התנסה בסביבת עבודה זו בעבר, יש לקרוא הסבר קצר באתר:

<https://www.jetbrains.com/help/pycharm/step-1-creating-and-running-your-first-python-project.html>

להלן סדר הפעולות כפי שתידרשו לעשות במהלך המעבדה. לכל פעולה ישנה הפניה לפירוט והסבר בהמשך התדריך.

1. התחלת עבודה - ממשק גרפי למשתמש (עמ' 2)
2. איפוס הרובוט (עמ' 3)
3. הוראות לכתיבת תכנית (עמ' 3)
4. מטלה ראשונה - תכנית מיון (עמ' 4)
 - 4.1 הגדרת נקודות במרחב (עמ' 4)
 - 4.2 הגדרת סוגי החלקים למיון ע"י הרובוט (עמ' 5)
 - 4.3 כתיבת תכנית מיון והרצה (עמ' 6)
5. מטלה שנייה - בקרת תנועה (עמ' 7)

שימו לב: בעמ' 9 ישנן הנחיות לכתיבת דו"ח מסכם.

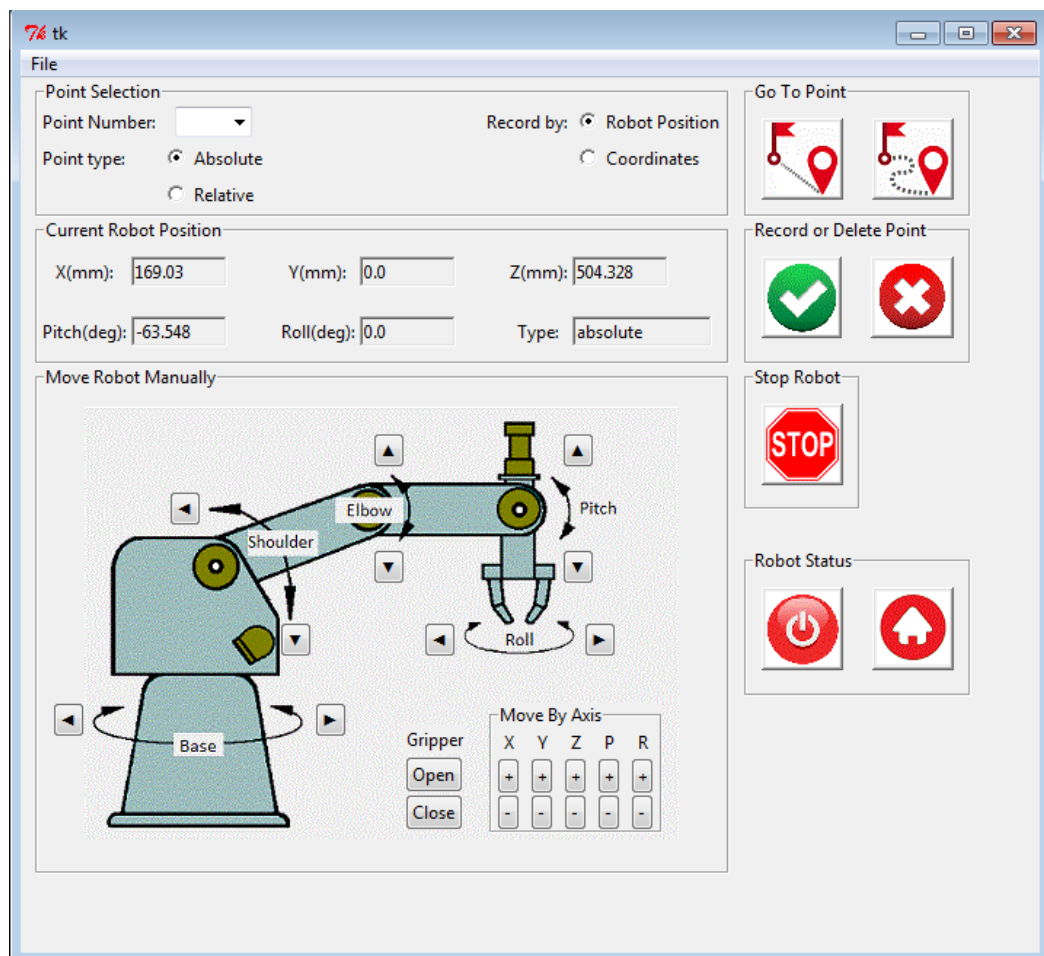
התחלת עבודה – ממשק גרפי למשתמש




על מנת להפעיל את הרובוט באמצעות המחשב יש להפעיל את ממשק המשתמש ע"י לחיצה על הקיצור שנמצא על שולחן העבודה. **המתניו מספר שניות לפתיחת הממשק הגרפי.**

הממשק הגרפי משמש הן לשליטה על הרובוט בזמן אמת באמצעות שלט וירטואלי והן להקלטת נקודות במרחב. **במידה ותבקשו מהרובוט לבצע פעולה לא חוקית, יופיע בממשק זה חלון המסביר את השגיאה.**

שימו לב: לכל אורך העבודה עם הרובוט, הממשק הגרפי מוכרח להיות פתוח. סגירת הממשק הגרפי תנתק את התקשורת עם הרובוט ותגרום לאובדן כל המידע שלא נשמר.



לאחר פתיחת הממשק הגרפי יש לוודא כי כפתור ההפעלה  המעיד על פעילות תקינה, ירוק. במידה והוא אדום לחץ עליו להפעלת הרובוט.

שימו לב: במידה ובמהלך העבודה, תבקשו מהרובוט לצאת מגבולות הגזרה שלו, או שהוא יתנגש בחפץ כלשהו, הרובוט יזרוק שגיאה וכפתור זה יהפוך לאדום. לחצו עליו שוב כדי להחזיר את השליטה על הרובוט.

1. איפוס הרובוט

1. את הרובוט יש לאפס במצבים הבאים:

- לפני תחילת העבודה עם הרובוט
- כאשר הרובוט נתקל דבר מה תוך כדי תנועה (לדוגמא בשולחן) וכפתור האיפוס חוזר להיות אדום.

חשוב! מומלץ מאוד לאפס את הרובוט עם הכניסה למעבדה, מאחר ואיפוס הרובוט דורש מספר

דקות.



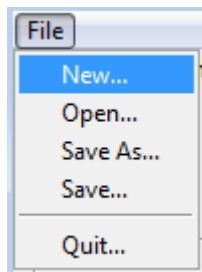
איפוס הרובוט נעשה על ידי לחיצה על כפתור האיפוס:

בזמן האיפוס הלחצנים בממשק הגרפי מושבתים.

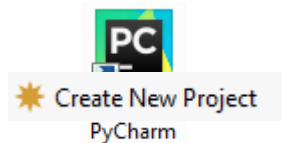
כאשר האיפוס מסתיים בהצלחה כפתור האיפוס יהפוך לירוק.

2. כתיבת תכנית

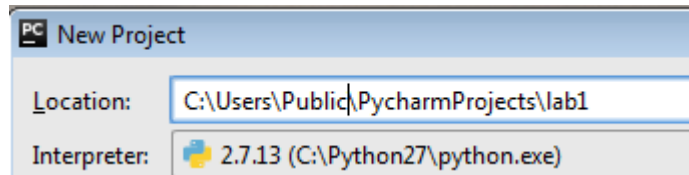
- צרו תיקיה חדשה בה תשמרו את קבצי התכנית שלכם ואת קובץ הנקודות שתקליטו.
- הנקודות נשמרות אוטומטית בקובץ לאחר הגדרתם בממשק המשתמש.



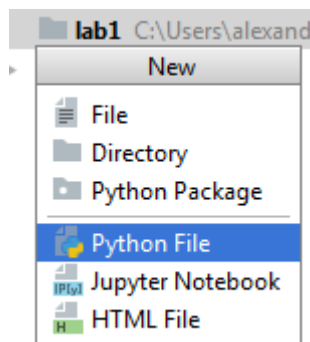
- פיתחו את סביבת העבודה PyCharm בה תכתבו את הסקריפטים בשפת פייתון. קובץ ההפעלה נמצא על שולחן העבודה.



- צרו פרויקט חדש בתיקיה אשר יצרתם קודם:



- לבסוף, צרו קובץ python חדש בתוך הפרויקט, ובו תכתבו את התכנית שלכם: PyCharm שומר אוטומטית את הקוד אותו אתם כותבים בקובץ.



3. תכנית מיון בסיסית

תיאור התכנית: עם הרצת התוכנית, ייגש הרובוט לנקודת איסוף חלקים שנקבעה מראש, ויתפוס חלק הממתין לו שם. לאחר תפיסת החלק הרובוט ישאל את המשתמש מה מספר החלק על מנת לזהותו (סה"כ 3 סוגי חלקים) ואז יניח אותו במקום ייעודי למספר החלק כך שכל חלק יונח (ולא ייזרק!) במקום המיועד לו מראש. המיון יתבצע 3 פעמים.

4.1 הגדרת נקודות במרחב

חשוב: לרשותכם 100 נקודות אפשריות להקליט. ניתן להשתמש בכל מספר נקודה בין 1-99.

בשלב הראשון יש להגדיר נקודות במרחב אותן הרובוט יכיר. לשם כך נדרשות 4 נקודות בסיסיות על השולחן: נקודת איסוף, ושלוש נקודות פיזור – אחת עבור כל סוג חלק.

בנוסף, כדי למנוע היתקלויות בשולחן העבודה ולאפשר פתיחה של מלקחי הרובוט, יש להגדיר מעל כל נקודה שנמצאת בגובה השולחן נקודה נוספת הגבוהה ממנה בכ-10 ס"מ (בציר z). סה"כ יש להגדיר 8 נקודות.

המלצה: הגדירו את הנקודות שמעל עם קשר מתמטי לנקודות שבגובה השולחן. למשל, הנקודה 1 בגובה השולחן ונקודה 10 גבוהה ממנה בכ-10 ס"מ, נקודה 2 ונקודה 20 מעליה, וכן הלאה...

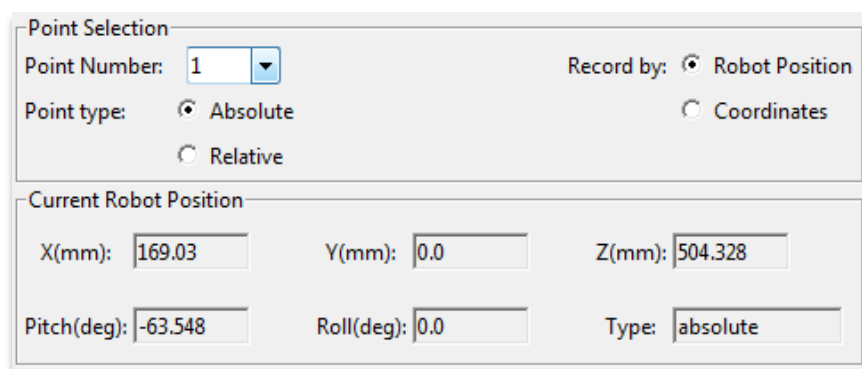
שימו לב: על הרובוט לנוע במערכת צירים קרטזית, כלומר במאונך למקביל למשטח ולא בתנועת אלכסון!

הגדרת הנקודות:

נקודה אבסולוטית (נמדדת ביחס לראשית הצירים של הרובוט):

הבא את הרובוט ע"י הזזה שלו לנקודה המדויקת שברצונך להקליט (ע"י השלט הווירטואלי – ראו נספח ראשון עבודה עם הרובוט)

בחלון ה- Point Selection יש להגדיר את מספר הנקודה בשדה Point Number. ודאו כי סוג הנקודה אותה אתם מקליטים הוא אבסולוטי, וכן כי ההקלטה מוגדרת לפי מיקום הרובוט. (נשים לב כי הקואורדינטות בחלון Current Robot Position מתעדכנות עם תנועת הרובוט).



בכדי להקליט את הנקודה.

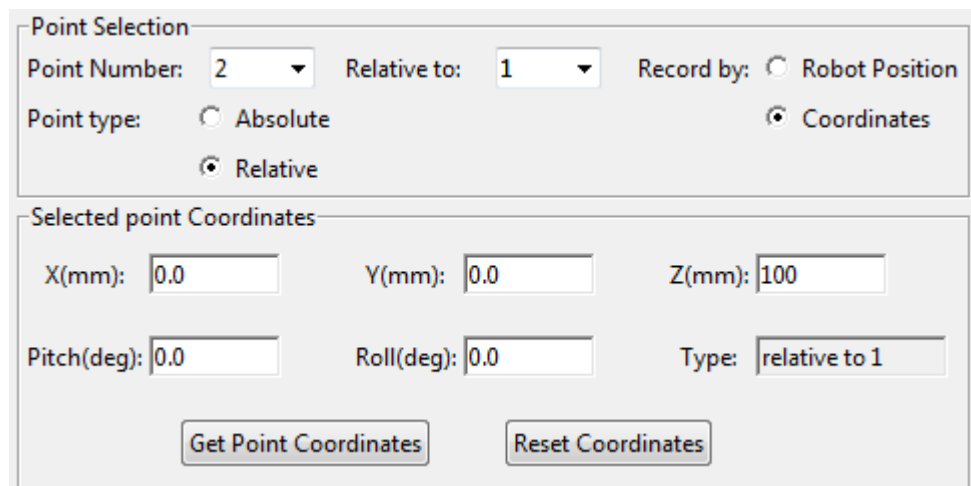


לחצו על כפתור ה-record

הגדרת נקודה יחסית לנקודה אחרת:

- בחלון ה- Point Selection יש להגדיר מספר נקודה חדש.
- יש ללחוץ על הקלטה לפי קואורדינטות: ☒ Coordinates
- נגדיר את סוג הנקודה להיות יחסי: ☒ Relative
- בשדה Relative to נבחר את מספר הנקודה אשר ביחס אליה נגדיר את הנקודה החדשה.
- כעת יש להזין את ההזזה היחסית של הנקודה החדשה עבור כל ציר בנפרד בחלון הקואורדינטות.
- לאחר שמגדירים את הנקודה יש ללחוץ על כפתור ה-record בכדי להקליט את הנקודה.

לדוגמא בתמונה מצ"ב הוגדרה נקודה 2 במרחק של 100 מ"מ בכיוון ציר Z יחסית לנקודה 1.



חשוב: בעת שימוש בהגדרת נקודה יחסית – במידה ומגדירים את נקודה 2 ביחס לנקודה 1, בעת הזזת נקודה 1, נקודה 2 תזוז באופן יחסי.

4.2 הגדרת סוגי החלקים למיון ע"י הרובוט

בשלב זה יש להגדיר לרובוט מהו סוג החלק אותו הוא מחזיק אשר עליו למיין.

במהלך ריצת התוכנית, לאחר שהרובוט יתפוס חלק, התוכנית תבקש לקבל קלט מהמשתמש אודות סוג החלק אשר נמצא בתפסן (סוג 1/2/3). יש לכתוב פונקציה אשר תדרוש מהמשתמש הזנה של מספר חלק, ותחזיר את מספרי שתי הנקודות המתאימות לחלק זה: הנקודה על השולחן, והנקודה היחסית הגבוהה ממנה ב-10 ס"מ. (אם אתם משתמשים בקשר מתמטי בין שתי הנקודות ניתן להחזיר רק נקודה יחידה).

בכדי לקבל קלט מהמשתמש נשתמש בפקודה `input()`.

תכנית לדוגמא: הפונקציה הבאה מבקשת מהשתמש קלט. אם המספר אותו המשתמש הזין הוא 1, הפונקציה תחזיר את המספרים 2 ו-3:

```
def getAnumber():
    number = input('enter a number:')
    if number == 1:
        return 2, 3

# אם נריץ את התכנית הבאה המשתמש בפונקציה זו:
first, second = getAnumber()
print 'first number returned is: ', first
print 'second number returned is: ', second
```

נקבל את הפלט:

```
enter a number:1
first number returned is: 2
second number returned is: 3
```

4.3 כתיבת תכנית מיון והרצה

חשוב: במידה והתכנית שכתבתם תבקש מהרובוט לבצע פעולה לא חוקית, התכנית תעוף, והודעת שגיאה תופיע בממשק הגרפי. היעזרו בהודעה זו בכדי להבין איפה שגיתם

כעת, כשהרובוט מכיר את כל הנקודות החשובות לו במרחב, ניעזר בפונקציה שכתבנו בסעיף הקודם ונכתוב תכנית לפיה הרובוט ייגש לנקודת האיסוף, יאסוף חלק הממתין לו שם, יבקש מהמשתמש להזין את מספר החלק, ועל פי מספר החלק שיוזן ע"י המשתמש יניח את החלק במקום המתאים.

המיון יתבצע 3 פעמים (באמצעות לולאת while או for)

חשוב: בכדי לתקשר עם הרובוט יש לייבא אל התכנית שתכתבו את המחלקה **Client** מתוך המודול **scorbotAPI**.

בשורה הראשונה בקובץ של התכנית נכתוב:

```
from scorpy.scorbotAPI import Client
```

כעת נאתחל מופע של המחלקה ונשמור מצביע אליו במשתנה:

```
()robot = Client
```

תכנית לדוגמא: התכנית הבאה תפתח ותסגור את התפסן של הרובוט:

```
from scorpy.scorbotAPI import Client

robot = Client()
robot.open_gripper()
robot.close_gripper()
```

פקודה רלבנטית לתנועת הרובוט

robot.move(position, speed=30)	<p>Move the robot to a given point</p> <p>Parameters:</p> <p>Position(int)</p> <p>Speed(int between 1-100): this parameter is optional and defaults to 30 if not specified</p> <p>Returns:</p> <p>1 on success and 0 on failure</p>
--------------------------------	---

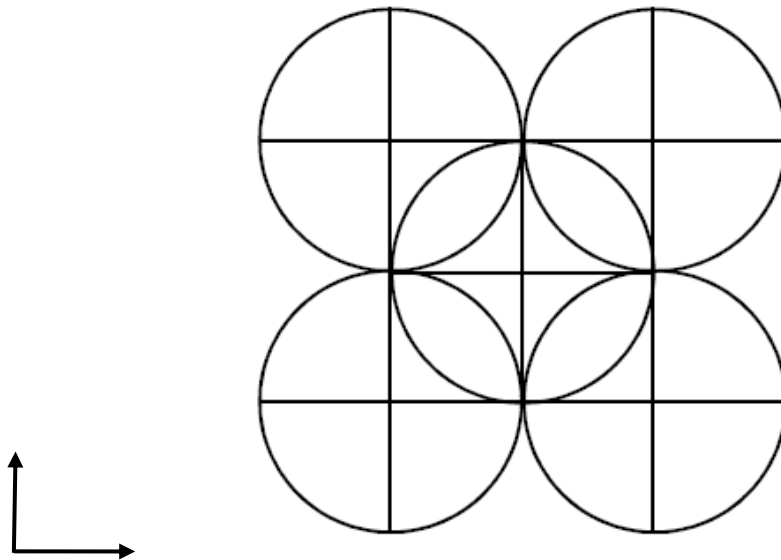
חשוב:

יש ליצור מופע מהמחלקה ולהריצו באמצעות שימוש בתנאי שלהלן, כך שהתוכנית תרוץ כתוכנית עצמית.

```
if __name__ == '__main__':
    main()
```

מטלה שנייה – בקרת תנועה

במטלה זו, נכתוב תכנית אשר באמצעותה הרובוט האוחז בעפרון יצייר את הציור הבא:



שימו לב כי הציור מורכב מחמישה מעגלים, כשבכל מעגל משורטטים שני קווים מאונכים זה לזה. אפשר להניח שהם בעלי רדיוס 10 מ"מ.

לצורך הציור, יש להקליט נקודה אחת בלבד במרחב בצורה אבסולוטית (באמצעות הממשק הגרפי). את שאר הנקודות הנחוצות לכם תלמדו את הרובוט באמצעות פקודות בתכנית שתכתבו.

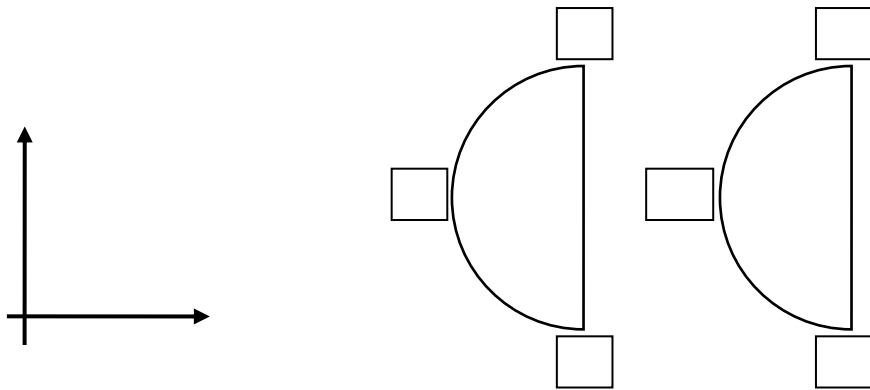
- יש לשים לב שלרובוט נשאר טווח תנועה בכל אחד מהצירים, כלומר שהציור לא חורג ממעטפת העבודה שלו.
- השתמשו בפקודות המופיעות בתוכנית לדוגמא על מנת לשנות את הקואורדינטות של הנקודה שהגדרתם, ולהגדיר נקודות חדשות

הדרכה במקרה ונתקעתם: שימו לב כי הציור מורכב מאותה צורה (מעגל שבו שני קווים מאונכים) החוזרת על עצמה מספר פעמים. חלקו את עבודתכם לשני שלבים:

1. כתבו פונקציה באמצעותה יצייר הרובוט את הצורה פעם אחת מהנקודה האבסולוטית אותה הקלטתם. תוך כדי שרטוט הצורה, מותר להרים את העיפרון מהדף ולכן, מעל כל נקודה שעל דף השרטוט, יש להגדיר נקודה הגבוהה ממנה ב- 10 מ"מ (אפשר להגדיר בקלות את הנקודות שמעל באמצעות לולאת for, שקשורות מתמטית לנקודות בגובה השולחן שכבר הוגדרו, כבמטלה הראשונה).
2. הגדירו רשימה שמכילה את כל הנקודות של מרכזי המעגלים ביחס לנקודת ההתחלה שהקלטתם.
3. כעת, בלולאה (כמספר הצעדים הדרוש), הגדירו מחדש את הנקודה האבסולוטית (באמצעות הרשימה מהסעיף הקודם) וקראו שוב ושוב לפונקציה אותה כתבתם, עד אשר הציור יושלם.

טרם כתיבת התוכנית, יש לעיין היטב בתוכנית לדוגמא המובאת להלן ולהבין היטב את הפקודות הרשומות בה.

התכנית לדוגמא מציירת את הציור הבא (בהנחה שרדיוס המעגל הוא 50 מ"מ):



בממשק הגרפי: נביא את הרובוט לנקודה 1 ונקליט אותה באופן אבסולוטי.

כעת בתכנית: באמצעות הפקודות הבאות נקליט את נקודות מספר 2 ו-3 באופן יחסי לנקודה 1:

```
from scorpy.scorbotAPI import Client
robot = Client()

robot.teach_relative_xyz_position(2, -50, 50, 0, 0, 0, 1)
robot.teach_relative_xyz_position(3, 0, 100, 0, 0, 0, 1)
```

קריאה לפונקציה הבאה תצייר את חצי המעגל השמאלי:

```
def drawSomething():
    robot.move(1)
    robot.move_circular(2,3)
    robot.move_linear(1)
```

באמצעות הפקודה הבאה נקבל את מערך הקואורדינטות של הנקודה 1:

```
crds = robot.get_position_coordinates(1)
```

נקליט מחדש את הנקודה 1 באופן אבסולוטי במרחק 100 מ"מ בציר ה-x מהנקודה המקורית (מסומנת 1' בציר):

```
robot.teach_absolute_xyz_position(1, crds[0]+100, crds[1], crds[2],
crds[3], crds[4])
```

נשים לב, כי הנקודות 2 ו-3 הוגדרו באופן יחסי לנקודה 1, ולכן כעת הן מוגדרות ביחס לנקודה 1'.

כלומר, קריאה מחדש לפונקציה `drawSomething()` תצייר את חצי המעגל הימני.
התכנית לדוגמא במלואה


```
from scorpy.scorbotAPI import Client
robot = Client()

def drawSomething():
    robot.move(1)
    robot.move_circular(2,3)
    robot.move_linear(1)

robot.teach_relative_xyz_position(2, -50, 50, 0, 0, 0, 1)
robot.teach_relative_xyz_position(3, 0, 100, 0, 0, 0, 1)

drawSomething()

crds = robot.get_position_coordinates(1)
robot.teach_absolute_xyz_position(1, crds[0]+100, crds[1], crds[2],
crds[3], crds[4])

drawSomething()
```

:Verifier

1. יש להגיש ב-verifier סרטים של:
 - a. תכנית המיון הבסיסית.
 - b. תכנית בקרת התנועה.
2. לא לשכוח לצלם את מספר הקבוצה בתחילת הסרט.

בהצלחה !