מבוא למדעי המחשב 67101

תרגיל 11 -2nd order programming

להגשה בתאריך 6/1/2016 בשעה 22:00

יעדים

התנסות בעבודה עם פונקציות מדרגה שנייה - פונקציות אשר פועלות על פונקציות (לדוגמא פונקציות המקבלות פונקציה כפרמטר או מחזירות פונקציה).
בתרגיל זה תצרו פונקציות מתמטיות שונות היוצגו על המסך באופן גרפי.

משימות

ציור פונקציה למסך

1. יישמו את הפונקציה

```
def plot_func(graph, f, x0, x1, num_of_segments=100, c='black'):
```

פונקציה זו מקבלת ארבעה עד שישה ארגומנטים ומציירת את הפונקציה המתמטית f לגרף בתחום בין הנקודות x0 ו-x1.
ארגומנטים:
graph - אוביקט GUI שיישמנו עבורכם ויוגדר מיד. הAPI של האובייקט מאפשר ציור של קווים ישרים למסך. דוגמאת שימוש מופיעה בהמשך העמוד.
f - פונקציה בפייתון המגדירה פונקציה מתמטית (ומחזירה ערך מספרי), למשל:

```
def f(x):
        return 3*x
```

x0 ו-x1 - תחום הפונקציה f. כלומר x מוגדר עבור $x0 <= x <= x1$. בתחום זה עליכם לצייר את הפונקציה.
num_of_segments - כאמור, graph יאפשר ציור של קווים ישרים בלבד. כיצד בכל זאת נצייר פונקציות (שאינן מורכבות בהכרח מקווים ישרים)? נקרב את ציור הפונקציה על ידי ציור של מספר מקטעים (num_of_segements) ישרים, כך שגודלו של כל אחד מהמקטעים הוא $(x1 - x0)/num\_of\_segments$ ויחדיו הם ייצגו את כל תחום הפונקציה.
למשל הישר הראשון בתחום אותו תציירו למסך יהיה הישר העובר בין שתי הנקודות הנ"ל:

1. (x0,f(x0))
2. (x0+(x1-x0)/number_of_segments), f(x0+(x1-x0)/number_of_segments)

הישר השני יצא מנקודה 2 לנקודה 3 שתחושב באופן דומה.
c - הצבע בו תצוייר הפונקציה למסך. בחירה של צבע תעשה מתוך מרחב הצבעים של tkinter: (http://wiki.tcl.tk/37701).

אתם רשאים להניח שהפונקציה מוגדרת בכל נקודה שלה

כאמור הפונקציה תעשה שימוש באובייקט השייך למחלה Graph. מחלקה זו מגדירה אובייקט GUI. בתרגיל זה הגדרנו עבורכם מחלקה פשוטה המציירת קווים ישרים למסך. המחלקה מיושמת בקובץ עזר.

הבנאי של Graph צריך לקבל כארגומנט ראשון את משתנה הGUI (מכונה master). ואת גבולות המסך (min_x, min_y, max_x, max_y).

שימוש באובייקט הגרף:

הAPI של האובייקט מאפשר אתחול שלו ושימוש בפונקציה בודדת - פונקצית plot_line. פונקציה זו של המחלקה מקבלת כאינפוט שלושה ארגומנטים (שתי נקודות במרחב הדו מימדי שתיוצגנה על ידי שני צמדים, וצבע) ומציירת את הקו הנ"ל למסך בצבע שהוגדר לו. למשל קריאה ל

graph.plot_line((2,4), (3,5), 'green')

תצייר קו ישר בין הנקודה x=2 וy=4 לנקודה x=3 וy=5 בצבע ירוק.

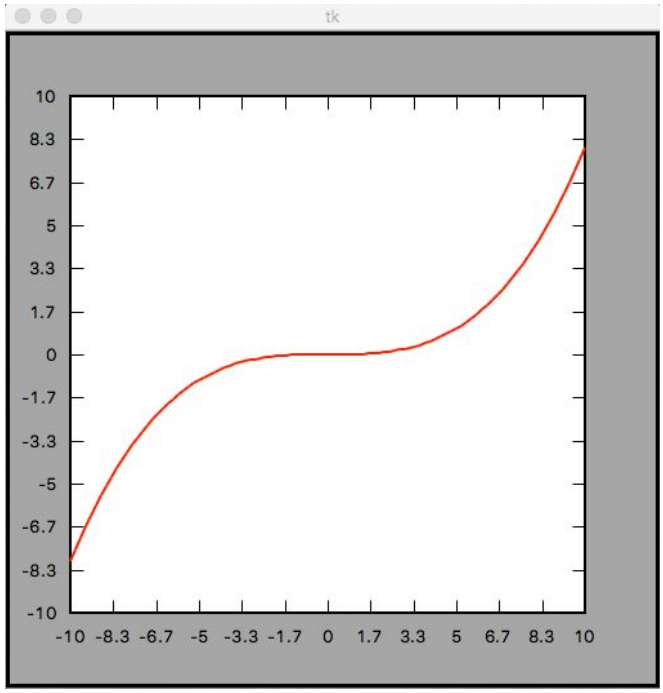
אתם לא צריכים לחשוש לגבי גבולות הגרף, הטיפול בכך הוא פנימי לאובייקט. במידה ותעבירו ישר שאחת מנקודותיו מחוץ למסך הוא לא יוצג. כן עליכם לוודא שאינכם מעבירים None לפונקציה plot_line.

לאחר יישום הפונקציה `plot_func`

הרצה של הקוד הבא בקובץ ex11.py תצייר למסך את התמונה המופיעה מתחת לקוד:

```
def example_func(x):
        return (x/5)**3

if __name__=="__main__":
        import tkinter as tk
        from ex11helper import Graph
        master = tk.Tk()
        graph = Graph(master,-10,-10,10,10)
                                                #יצירת אובייקט הGUI
        plot_func(graph, example_func, -10, 10, 100, 'red')
                                # ציור למסך של f בין הנקודות 10- ו10 באמצעות 100 מקטעים ישרים
        master.mainloop()
```



פונקציות מדרגה שניה:

במשימות הבאות עליכם להגדיר **פונקציות שערך ההחזרה שלהן הוא פונקציה**.
השבוע צורף לתרגיל שלכם קובץ בדיקות חלקי. קובץ זה בודק את האופן שבו יישמתם את הפונקציות בחלק זה. הסברים נוספים בתחתית התרגיל.

2. יישמו את הפונקציה:

def const_function(c):

פונקציה זו מקבלת כארגומנט מספר קבוע ו**מחזירה פונקציה** מתמטית הממפה כל x לc, כלומר const_function מחזירה את הפונקציה:

f(x) = c

דוגמא לשימוש בפונקציה:

```
my_f = const_function(17)
print(my_f(4))
# prints 17
```

3. יישמו את הפונקציה

def identity():

פונקציה זו **מחזירה פונקציה** מתמטית כזו שלכל x מחזירה את x
כלומר identity מחזירה את הפונקציה:

f(x) = x

דוגמא לשימוש בפונקציה:

```
my_f = identity()
print(my_f(7))
#prints 7
```

4. יישמו את הפונקציה:

def sin_function():

פונקציה זו אינה מקבלת ארגומנטים ו**מחזירה את הפונקציה** המחשבת עבור כל x (ברדיאנים) את ערך הסינוס שלו, כלומר sin_function מחזירה את הפונקציה:

f(x) = sin(x)

5. א. יישמו את הפונקציה:

def sum_functions(g, h):

פונקציה זו מקבלת כארגומנט שתי פונקציות מתמטיות ו**מחזירה את הפונקציה** שהיא סכומם:

f(x) = g(x)+h(x)

ב. יישמו את הפונקציה:

def sub_functions(g, h):

פונקציה זו מקבלת כארגומנט שתי פונקציות מתמטיות ו**מחזירה את הפונקציה** שהיא הפרשם:

f(x) = g(x)-h(x)

ג. יישמו את הפונקציה:

def mul_functions(g, h):

פונקציה זו מקבלת כארגומנט שתי פונקציות מתמטיות ו**מחזירה את הפונקציה** שהיא מכפלתם:

f(x) = g(x)*h(x)

ד. יישמו את הפונקציה:

def div_functions(g, h):

פונקציה זו מקבלת כארגומנט שתי פונקציות מתמטיות ו**מחזירה את הפונקציה** שהיא מנתם:

f(x) = g(x)/h(x)

במידה ועליכם לחלק ב0, בצעו פעולה זו בכל זאת (פייתון יזרוק שגיאת חלוקת ב0 - תקין מבחינתכם עבור תרגיל זה, אין צורך לבדוק את הקלט)

6. יישמו את הפונקציה

def solve(f, x0=-10000.0, x1=10000.0, epsilon=1e-5):

המקבלת כארגומנט פונקציה, בתחום x0<=x<=x1 ואפסילון

ומוצאת x בתחום כך ש

|f(x)|<epsilon

שימו לב, בניגוד לפונקציות הקודמות, פונקציה זו **אינה מחזירה פונקציה אלא מספר** (int or float) שהוא פתרון לפונקציה בתחום הנ"ל. במידה ולא קיים פתרון לפונקציה בתחום החזירו None.

אנו מצפים לפתרון עם זמן ריצה לוגריתמי. אתם יכולים להניח רציפות בתחום המוגדר.

החזירו None במידה ולא מתקיים כי f(x0)*f(x1)<0, שכן אז לא מובטח כי קיים פתרון וניתן למצואו בזמן לוגריתמי (כמו בפונקציה x^2+1 למשל).

שימו לב, משימה זו לא נבדקת על ידי קובץ הבדיקות שספקנו לכם. על כן באחריותכם לבדוק כי פונקציה זו פועלת בצורה שמצופה ממנה.

7. יישמו את הפונקציה:

def inverse(g, epsilon=1e-5):

המקבלת כארגומנט פונקציה g ו**מחזירה את הפונקציה ההופכית** שלה באופן מקורב.

כלומר עליכם להחזיר f כך ש

f(g(x)) = x

או מכיוון שנסתפק גם בפתרון מקורב , עליכם להחזיר f כך ש:

|f(g(x)) - x| < epsilon

פונקציה זו צריכה לעבוד רק עבור פונקציות מונטוניות ורציפות.

לצורך מימוש פונקציה זו עליכם להשתמש בפונקציה solve. אך כיצד תדעו באיזו תחומים להגדירה? נסו לחשוב כיצד להתחיל את החיפוש בטווח קטן ולהגדילו באופן יעיל במידה ולא מצאתם פתרון.

8. יישמו את הפונקציה:

def compose(g, h):

המקבלת כארגומנט שתי פונקציות מתמטיות ו**מחזירה את הפונקציה** שהיא הרכבה של h על g.

f(x) = (g ∘ h)(x)

למשל במקרה שבו

g(x) = sin(x)

h(x) = x+3

עליכם להחזיר את הפונקציה

sin(x+3)

9. יישמו את הפונקציה:

def derivative(g, delta = 0.001):

המקבלת כארגומנט פונקציה מתמטית ודלתא ומחזירה את הנגזרת המקורבת של הפונקציה.
באינפי אנו מגדירים קירוב של נגזרת הפונקציה באמצעות:

$$f(x) = \frac{g(x+\Delta)-g(x)}{\Delta} \sim g'(x)$$

השתמשו בדלתא בכדי להגדיר את הדיוק של הנגזרת באזור של x

10. יישמו את הפונקציה:

def definite_integral(f, x0, x1, num_of_segments):

המקבלת כארגומנט פונקציה מתמטית, תחום (בין x0 וx1) ומספר סגמנטים, ומחשבת את קירובו של האינטגרל המסויים בין x0 וx1 מתחת לפונקציה f באמצעות שיטת סכומי רימן (סכומי המרובעים):

$$S = \sum_{i=1}^{n} f\left(\frac{x_{i-1}+x_i}{2}\right) * (x_i - x_{i-1})$$

https://en.wikipedia.org/wiki/Riemann_sum

את המקטע בין x0 לx1 עליכם לחלק לnum_of_segements מקטעים שווים באורכם (כל אחד מהם באורך (x1-x0)/num_of_segements). כל מקטע כזה יהיה צלע במלבן שמוגדר באמצעות שיטת סכומי רימן.

שימו לב, פונקציה זו מחזירה משתנה בודד מסוג float או int.

11. יישמו את הפונקציה

def integral_function(f, delta = 0.01):

המקבלת כארגומנט פונקציה ודלתא ו**מחזירה את הפונקציה** המתמטית שהיא האינטגרל הלא מסוים של f.

כלומר עליכם להחזיר את F כך ש

F'(x) = f(x)

נשים לב שניתן להגדיר את האינטגרל של f בנקודה x באופן הבא:

עבור x חיובי:

$$\int_0^x f(t)dt$$

עבור x שלילי:

$$\int_x^0 -f(t)dt$$

מכיוון שאיננו יכולים לחשב אינטגרל מדויק, נחזיר קירוב של האינטגרל. הקירוב יוגדר באמצעות delta. לכל x נמצא את האינטגרל המסוים שלו באמצעות הפונקציה שהגדרתם בסעיף הקודם, כאשר מספר הסגמנטים יוגדר להיות:

num_of_segments = math.ceil(|x|/delta)

מימוש פונקציות.

בחלק האחרון של התרגיל תדרשו לממש פונקציות מתמטיות אמיתיות. חלק זה ייבדק בפרוטרוט על ידי הבודקים:

12. יישמו פונקציה התקרא

def ex11_func_list()

בה עליכם להחזיר רשימה של פונקציות. כל אחד מאיברי הרשימה שתחזירו הוא יישום של אחת הפונקציות ברשימה הבאה. אתם יכולים להשתמש בערכים הדיפולטיבים שהוגדרו לקירובים הנדרשים בקובץ הטמפלייט.

דוגמא (שאותה יש לכלול בפתרון) :

0. f(x) = 4 ---> const_function(4)

ממשו את פונקציות 1-7:

1. f(x) = 3-sin(x)
2. f(x) = sin(x-2)
3. f(x) = 10/[2+sin(x)+(x^2)]
4. f(x) = cos(x) /(sin(x)-2)
5. f(x) = $-0.1 * \int(0.3x^2 + 0.7x - 1)dx$
6. f(x) = $[cos(sin(x)) - 0.3 * cos(x)] * 2$
7. f(x) = #the inverse function of (2-x^3)

וכאמור ערך ההחזרה הוא רשימה של פונקציות כך שלמשל:

```
func_list = []
func_list.append(const_function(4))
```

הוא הפתרון לשאלה 0 .

בסיום הפונקציה החזירו את func_list

שימו לב עבור המשימות הללו אסור לכם להשתמש בפונקציות של math (מעבר לשימוש שכבר עשיתם במשימות 1-11) או באופרטורים מתמטים כלשהם. מותר וכדאי לשמור את הפונקציות למשתנים.

נהלי הגשה

עליכם לממש את הפונקציות הדורשות מימוש בקובץ ex11.py.

בתחתית הקובץ מופיע מקטע main, כולל מקטעים מסויימים שאותם הפכנו להערות. לאחר יישום הפונקציות אתם יכולים להסיר את סימן הסולמית בכדי לצייר למסך את הפונקציות שיצרתם. מקטע זה לא יבדק בבדיקות האוטמטיות.

שימו לב. , בתרגיל זה ספקנו לכם טסטר הנקרא ex11utests.py. הטסטר בוחן את הפונקציות אותם יישמתם. עליכם להריץ את הטסטר על הקוד שלכם ולבדוק תקינות הקוד לפני הגשתו. בנוסף לטסטר זה התרגיל שלכם יעבור בדיקות נוספות לאחר הגשתו (כבכל שבוע).

Doctest - לחלק מהפונקציות צורפו דוקטסטס - טסטרים קצרים שנועדו לעזור בהבנת הפונקציה וממומשים כבר ברמת הדוקומנטציה. בכדי להבין כיצד להריצם ולפרטים נוספים: https://docs.python.org/3.4/library/doctest.html

הקבצים ex11.py וex11helper.py וex11utests.py ניתנים להורדה מאתר הקורס.

הגשת התרגיל:

הגישו קובץ zip הנקרא ex11.zip, המכיל את הקובץ ex11.py את קובץ הREADME ולא כולל אף קובץ נוסף, בפרט לא אתex11helper וex11teusts.

בהצלחה