תרגיל בית 6

נושא: פונקציות מסדר גבוה, רקורסיה, שאלות דוגמא ממבחנים

1. (30%) בשאלה זו עליכם להשתמש בשיטת ניוטון רפסון למציאת נקודת ה 0 של פונקציה. מומלץ להסתכל על סוף המצגת של פונקציות מסדר גבוהה להסבר על השיטה הזו לפני שתיגשו לפתרון.

א. יש לכתוב פונקציה difference שמקבלת שתי פונקציות ומחזירה את פונקציית ההפרש בין הפונקציה הראשונה לפונקציה השניה. (אם f ו g הן פונקציות אז הפרש בין f ל g הוא הפונקציה h המוגדרת כך: h(x)=f(x)-g(x) לכל x.)

ב. כתבו פונקציה deriv המחזירה את קירוב פונקצית הנגזרת של פונקציה ע"י משפט הפרש המרכזי

$$\frac{f(x+h)-f(x-h)}{2h} \approx f'(x)$$

הפונקציה תקבע את h בתוכה ולא תקבלו כפרמטר. הריצו עם h=0.000001. היא תקבל את f ו g כפרמטרים.

ג. כתבו פונקציה NR שמממשת את שיטת ניוטון רפסון למציאת נקודת אפס של פונקציה. כדי למצוא נקודת חיתוך של שתי פונקציות נרצה למצוא x עבורו 0=f(x)-g(x) כפי שמימשנו בסעיף א. הנוסחה בגדול מחשבת את

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

תוכלו להגדיר למשל:
NR(f, a, b, epsilon=10**(-10), n=100) כאשר
a הגבול התחתון של הטווח שבו מחפשים את השורש.

- b הגבול העליון של הטווח שבו מחפשים את השורש.

epsilon רמת דיוק רצויה עבור השורש. ברגע שהערך המוחלט של הפונקציה f(x) קטן מערך זה, האלגוריתם עוצר ומחזיר את השורש שנמצא.
n המספר המקסימלי של איטרציות שהאלגוריתם ירוץ. אם האלגוריתם לא מצליח למצוא שורש בתוך מספר האליטרציות הזה, הוא מחזיר None.
אפשר להגריל נקודה ראשונה ע"י שימוש ב uniform בין a ו-b מספרית random.

ד. השתמשו בפונקציות מסעיפים א' ו-ג' כדי לכתוב תוכנית intersection_NR.py שקוראת מתוך קובץ input_functions.txt בכל פעם שתי פונקציות ושני מספרים a ו-b ומדפיסה לקובץ output_functions.txt את שתי הפונקציות, את הערכים של a ו b ואת נקודת חיתוך (x,y) של שתי הפונקציות הנתונות בטווח a<x<b. אם אין נקודת חיתוך בטווח הזה, יודפס no intersection. מומלץ להגדיר פונקציית עזר המחזירה את פונקצית הנגזרת של פונקציה.
מצורפים לדוגמה קובץ קלט וקובץ פלט. הדפיסו לפי הפורמט המצורף.
<u>הנחיות</u>:

- יש להניח שהקלט הוא בדיוק בפורמט שבדוגמה.
- הפלט צריך להיות באותו פורמט כמו בדוגמה.
- מומלץ להוסיף בתחילת התוכנית from math import * כדי שניתן יהיה להריץ את התוכנית על פונקציות כמו sin, cos, log בלי התחילית math.

- יש להשתמש בפונקציה eval כדי להפוך מחרוזת לפונקציה.
- כל יחידת קלט מורכבת מ 3 שורות: שורה לכל פונקציה ושורה שלישית שבה מופיעים שני המספרים עם רווח ביניהם. בין כל שתי יחידות קלט מפרידה שורה ריקה.
- ניתן להניח שהקלט תקין.
- יש להדפיס את הפלט בדיוק של 4 ספרות אחרי הנקודה העשרונית.

בהמשך שאלות ממבחנים:

2. (15%) תשמרו את הקוד בקובץ student.py

נתונה מחלקה Student המתארת סטודנט במכללה ובה 3 תכונות:

- __name – שם הסטודנט, מטיפוס str
- __id – ת"ז של הסטודנט מטיפוס str
- __grades – מילון ובו ציוני הסטודנט. המפתחות הם שמות הקורסים מטיפוס str והערכים הם הציונים בקורסים, מטיפוס int.

א. יש לכתוב בנאי שמקבל שם ומספר ת"ז. הבנאי יבדוק את הפרמטרים, יעשה השמה לתכונות בהתאמה ויצור מילון ריק עבור הציונים. הבנאי יזרוק חריגה במקרים הבאים:
- השם אינו מטיפוס str
- מס' הת"ז אינו מטיפוס str
- מס' הת"ז אינו מורכב מ 9 ספרות בדיוק.

ב. יש לכתוב שיטה update_grade שמקבלת שם של קורס וציון בקורס ומוסיפה את שם הקורס והציון למילון. אם שם הקורס כבר קיים אז השיטה תעדכן את הציון. אם הציון אינו מספר שלם בין 0 ל 100 השיטה תזרוק חריגה.

ג. יש לכתוב שיטה get_grade שמקבלת שם של קורס ומחזירה את הציון באותו קורס. אם שם הקורס לא קיים השיטה תזרוק חריגה.

ד. יש לכתוב שיטה better_grades שמקבלת כפרמטר סטודנט other_student. הפונקציה תחזיר את רשימת שמות הקורסים שבהם ל other_student יש ציון יותר גבוה מאשר ל self. אם אין קורסים כאלה יוחזר None. יש להשתמש בשיטה מסעיף ג'. אין להניח קיום של שיטות אחרות במחלקה (כולל getters ו setters).

ה. יש לכתוב פונקציה courses_not_taken שמקבלת רשימת קורסים (מחרוזות) courses_lst וסטודנט s ומחזירה את רשימת הקורסים מתוך courses_lst שלסטודנט s אין בהם ציון. יש להשתמש רק בשיטות שהופיעו בסעיפים הקודמים אין להניח קיום של שיטות אחרות במחלקה Student (כולל getters ו setters).

תוכלו לבדוק את הקוד עם התכנית המצורפת student_main.py. עם פלט מצופה בstudent_output.txt.

3. (15%)תשמרו את התכנית בקובץ dict.py

a. יש לכתוב פונקציה join_dict שמקבלת שני מילונים ומחזירה רשימה של זוגות (a,b) מטיפוס tuple, כך ש a ו b הם value עבור אותו המפתח בשני המילונים.
כלומר, קיים c כך ש c:a הוא זוג במילון הראשון ו c:b הוא זוג במילון השני.

דוגמה: אם המילונים הם

d1={10: "dog", 20: 157, 30: "cat", 40: 17}

d2={5:"apple", 10: 13, 15: "orange", 20: "banana"}

אז הפקודה join_dict(d1, d2) תחזיר את הרשימה:

[('dog', 13), (157, 'banana')]

b. יש לכתוב פונקציה is_mutual שמקבלת פרמטר tpl מטיפוס tuple ושני מילונים d1 ו d2. הפונקציה תחזיר True אם קיים c כך ש c:tpl[0] הוא זוג באחד משני המילונים וְ c:tpl[1] הוא זוג במילון האחר.

דוגמה: עבור המילונים d1 וְ d2 בדוגמה מסעיף 1.1 הפקודה

is_mutual(('dog', 13),d1,d2)

תחזיר True וגם הפקודה

is_mutual((13, 'dog'),d1,d2)

תחזיר True.

לעומת זאת הפקודה

is_mutual(('cat', 13),d1,d2)

תחזיר False.

תוכלו להריץ את dict_test.py לבדיקה. התוצאות הצפויות נמצאות בהערה בסוף קובץ הבדיקה.

4. (15%)כתבו בקובץ palindrome.py.

פלינדרום היא מחרוזת s שאם קוראים אותה מהסוף להתחלה היא נראית בדיוק כמו s. למשל, 'rotor', 'abba', 'neveroddoreven'.

a. כיתבו פונקציה רקורסיבית is_palindrome, שמקבלת מחרוזת s ומחזירה True אם s היא פלינדרום. אם לא, אז הפונקציה תחזיר False. אין להשתמש בלולאות, אין להשתמש בשיטות של str, אין להגדיר פרמטרים נוספים מלבד s, אין להשתמש בבדיקה s==s[::-1]. אין להשתמש בְּפונקציה מסוג helper. פתרון לא רקורסיבי לא יתקבל.

b. כיתבו פונקציה max_palindrome שמקבלת מחרוזת s ומחזירה את תת-המחרוזת הארוכה ביותר של s שהיא פלינדרום (תת-מחרוזת של s היא מחרוזת t הנמצאת ברצף בתוך s). אין להשתמש בלולאות, אין להשתמש בשיטות של str, אין להגדיר פרמטרים נוספים מלבד s, אין להשתמש בְּפונקציה helper. אם יש שני פלינדרומים בתוך s שהם באורך מקסימלי, יש להחזיר את אחד מהם. יש להשתמש בפונקציה מסעיף 2.1.

דוגמאות:

max_palindrome('abad') תחזיר 'aba'

max_palindrome('abbcc') תחזיר 'bb' או 'cc'.

max_palindrome('xrotoryz') תחזיר 'rotor'.

max_palindrome('abcd') תחזיר את 'a' או את 'b' או את 'c' או את 'd'.

תוכלו להריץ את palindrome_main.py לבדיקה. התוצאות הצפויות נמצאות בהערה בסוף קובץ הבדיקה.

(15%) יש לכתוב פונקציה רקורסיבית binsearch בקובץ binsearch.py שמקבלת רשימת מספרים ממויינת lst ומספר num. הפונקציה מבצעת חיפוש בינארי של המספר num ברשימה lst. אם המספר נמצא, היא תחזיר את

האינדקס שלו ברשימה lst. אם לא – היא תחזיר None. אפשר להניח שהרשימה lst ממויינת בסדר עולה.

הנחיות:

- ניתן לכתוב פונקציית עזר או להוסיף פרמטר עם ערך ברירת מחדל.
- אין להשתמש בלולאות ואין להתשמש במשתנים גלובליים.
- אין לשנות את הרשימה lst

למשל, עבור הרשימה

```
lst = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

הקריאה binsearch(lst, 17) תחזיר 6. הקריאה binsearch(lst, 27) תחזיר None.
תוכלו להריץ את binsearch_main.py לבדיקה. התוצאות הצפויות מופיעות בהערה בסוף.

0111401 מבוא למדעי המחשב
החוג למדעי המחשב, המכללה האקדמית תל-חי
סמסטר סתיו תשפ"ה

הנחיות הגשה:

1- יש להגיש תוכניות שרצות ללא שגיאות. תוכנית שתוגש עם שגיאות ריצה תקבל לכל היותר חצי מהנקודות.

2- בכל התרגילים יש להשתמש במנגנון exceptions כדי למנוע קריסת התוכנית במקרה של קובץ קלט לא קיים ובמקרה של קלט לא תקין.

3- יש לכתוב הערות לתוכנית: docstring בתחילת כל פונקציה, הסבר קצר בתחילת התוכנית, הסבר בתחילת לולאות.

4- אין להשתמש במודולים מלבד מודולים סטנדרטיים כמו math, random, sys,

5- יש לפתור כל שאלה בקובץ נפרד עם סיומת py. (אלא אם כן נאמר אחרת).

6- יש להגיש את כל הקבצים בקובץ אחד מכווץ עם סיומת zip. שם הקובץ המכווץ צריך להיות id_ex6.zip, כאשר id הוא מספר הת"ז שלכם. למשל 111112113_ex6.zip.
יש לכלול רק קבצים עם סיומת py. אין לכלול קובצי קלט ופלט.

7- כל קובץ יתחיל בהערה ובה המידע הבא:
   א. שם הסטודנט
   ב. מס' תעודת זהות
   ג. מספר התרגיל בית
   ד. שם התוכנית

למשל, עבור שאלה 2 תרגיל 6:

```
"""
Student: Lisa Marie Presley
ID: 001021968
Assignment no. 6
Program: minesweeper.py
"""
```

שימו לב: יש להקפיד על הנחיות ההגשה האלה. הגשה שלא בדיוק בפורמט הזה לא תקבל את מלוא הנקודות ואף עלולה להיפסל.