קובץ הסבר SmartCalculator – תום פלג

לפניך תוכנית המחשבון המשוכלל שלי בפייתון. בעת הרצת התוכנית תופיע הודעה שתבקש שתכניס משוואה לפתרון או מילת המפתח "stop" על מנת לעצור את התוכנית.

***אנא הורד את הגרסה העדכנית ביותר של SmartCalculator מGitHub***

נתקלתי בהרבה קשיים במהלך כתיבת המחשבון.  
ראשית כללי המחשבון נוגדים בחלקם את חוקי המתמטיקה דבר שאינו אינטואיטיבי לי.  
שנית, העדר הוראות ברורות – מכיוון וכל הזמן חשבנו על מקרי קיצון חדשים היינו צריכים להחליט הרבה פעמים על פתרון אחיד לכולנו, מה שהביא לכך שפתרון המטלה שנתתם לנו היה מאוד קטוע וכמעט בלתי אפשרי לתכנון מראש.  
שלישית, קושי תקשורת - מכיוון והעברנו הודעות דרך אמיר ואתה העברת דרכו גם, אני חושב שנוצר צוואר בקבוק לא נחוץ שהוביל לקושי בתקשורת, אם הייתה קבוצה של שאלות כמו זו שהייתה בסודוקו לדעתי היינו יכולים להימנע מכמה כשלים בתקשורת ולבצע עבודה טובה יותר ומלאה יותר.

החלטתי להשתמש ברשימה כמבנה נתונים להחזקת המשוואה משום שפיתון בנוי בצורה נוחה לשימוש וטיפול ברשימה ובאמצעות רשימה ניתן לשלב סוגים שונים של איברים במשוואה ללא המרת ביניים מיותרת – מספרים ואותיות

התוכנית מחולקת לכמה מודלים מרכזיים:

1. **Calculation** – המודל הלוגי של התוכנית, מכיל את הפונקציות בהם השתמשתי על מנת לפתור את התרגיל שאקבל מהמשתמש, הפונקציה המרכזית ששולחת את האיברים מהמשוואה לפתרון היא רקורסיבית כך שבתחילת כל איטרציה שלה היא ראשית מוצאת את הסוגריים הפנימיים ביותר (אם קיימים), לאחר מכן מוצאת את מיקום האופרטור החזק ביותר בתוך תחום הסוגריים שנמצאו ובהתאם לסוג האופרטור שולחת את כמות האיברים המתאימים לפעולת האופרציה.  
   החלטתי ליצור אובייקט בשם calculator על מנת שאוכל לשמור את המשוואה, לשמור את סוג התוצאה שאני רוצה לקבל. רציתי גם שאוכל בקלות לשלב הרצת כמה מחשבונים במקביל והיכולות להוסיף בעתיד עוד תכונות ופיצ'רים למחשבון כמו ציור גרפים ושינוי סוג התרגילים (מספרים לא ממשיים).   
   בפעולת solve נתפסים רוב שגיאות זמן הריצה, זאת באמצעות try except, שכן רק בעת פתירה אוכל לדעת אם הערכים שהוכנסו למשוואה עומדים בחוקי המתמטיקה, ובהגבלות המערכת (גדלים של משתנים בעלי הגבלה – מספר גדול מדי או מספר לא ממשי מה שהוגדר מראש כערך לא חוקי במחשבון)  
   חשוב לציין שמכיוון ומינוס ('-') מהווה גם אופרטור בעל קדימות 1 וגם אופרטור של סימן, אם נמצא בפעולה אופרטור שאחריו מופיע מינוס ראשית נשלח המינוס על מנת להפוך סימן לאופרנד.  
   כמו כן, הדואליות של מינוס הביאה איתה הרבה קשיים בשבילי ולכן:
2. בפונקציה הרקורסיבית אני ראשית מטפל במקרה בו מינוס הוא סימן ולאחר מכן אופרטור, אם האופרטור הנמצא הוא מינוס אני בודק אם לפניו יש אופרטור אחר וכך אני מסוגל להבדיל בין השניים.
3. שרשור מינוסים מטופל לפני תחילת הפתרון על ידי פעולה שסופרת אותם ומוסיפה/מורידה מינוסים בהתאם לכמותם ומיקומם (תחילת משוואה צריכה להיות ללא סימן אך אמצע משוואה צריכה להישאר עם אופרטור + או -)
4. **Validation** – המודל הבודק את שלמות האינפוט שהמשתמש הכניס, ראשית הפונקציה ממירה את המשוואה לרשימה, לאחר מכן בודקת אם קיימים איברים לא ולידיים (ג'יבריש) ורק לאחר כך ממירה ספרות למספרים.

ישנם מספר רב של פונקציות הפועלות על מנת לבדוק את תקינות המשוואה, ושחרור הודעות מתאימות לכל שגיאה.   
על מנת לשמור על מודולריות של התוכנית תכננתי את הפעולות של בדיקת תקינות האופרטורים כך שיתייחסו למשקל שהוחלט מראש לכל אחד מהם ולמיקום של האופרנדים שהם דורשים, כעת קיימים ארבעה מצבים אך ניתן להוסיף עוד, לדוגמא אופרטור שדורש אופרנד מימינו, שמאלו, משני צדדיו או לפחות מצד ימין שלו (מינוס)

1. **Utility** – מודל בו שמרתי את כל הפונקציות הגנריות שאצטרך להשתמש בהם בכל המודלים השונים לדוגמא, פונקציות המחזירות האם תו מסוים הוא מספר, אופרטור, סוגריים, סימן.

שאר המודלים פחות משמעותיים ונועדו על מנת לארגן את הקוד בצורה נוחה וקריאה ועל מנת לאפשר הוספה עתידית של פונקציות שונות, להלן הסבר קצר על כל אחד מהם:

1. **new\_math:** מודל המממש מחדש את פעולות המתמטיקה לשימוש המחשבון, חשוב לציין שבתוך פעולת factorial ישנו exception מיוחד שאני יוצר על מנת לעמוד בחוקי המתמטיקה – לא ניתן לעשות עצרת למספר שלילי או לא שלם

2. **operator\_data:** מודל המחזיק את כל ההגדרות שהוחלטו מראש לגבי האופרטורים במחשבון, כמו כיוון אופרטורים, סוגי אופרטורים, המשקל שלהם, והאופרציות שלהם.

3. **formatting:** מודל האחראי לפירמוט האינפוט של המשתמש, כרגע מטפל ברווחים למיניהם במשוואות ואיחוד ספרות לכדי מספר שלם.

4. **errors**: מודל המחזיק את כל הודעות השגיאה השונות שהגדרתי למערכת מחולקים לפי שגיאות ולידציה או פתירה, מודל זה נועד על מנת שיהיה קל בעתיד להוסיף שגיאות למחלקות הקיימות או אף ליצור סוג שגיאה חדשה.

הוספת אופרטורים חדשים

על מנת להוסיף אופרטור חדש, גש למודל operator\_data והוסף למילונים המתאימים את התו שמייצג את האופרטור, הכוח שלו, אילו אופרנדים הוא לוקח, (אחד מימין, אחד משמאל, שניהם או שילוב של שניים), והוסף פונקציה מתאימה המתארת את הפעולה החשבונית שהאופרטור מקיים, אני איגדתי את כל הפונקציות הקיימות במודל new\_math, ניתן להוסיף לשם או לפתוח מודל חדש.