Advenced calculator – Ofek Merhav

מהלך בניית הפרוייקט:

תכנון:

תחילה היה עלי לתכנן את צורת הפתרון בה ארצה לעבוד,

* התוכנית עובדת בשיטה המקובלת במתמטיקה הקלאסית והיא,

פתירת סוגריים בעדיפות ראשונה ופתירת אופרטורים כעדיפות שנייה על פי סדר פעולות החשבון.

* בחרתי לעבוד עם String של פייתון מכיוון שהוא אובייקט עם המון דינאמיות אשר מאפשר איטרציה ושינויים בצורה נוחה וקלה.

כך , התרגיל מיוצג על ידי מחרוזת ולאחר כל פתירה ושינוי המחרוזת מתעדכנת בהתאם.

* קבעתי פונקציות יסוד אשר יהיו אבן הבניין בפרוייקט ביניהן פותר סוגריים ופותר ביטוי מסביב לאופרטור אותן תפעיל פונקצית הפותר הראשית.

כתיבת הקוד:

פעלתי על פי מספר שלבים במהלך כתיבת הקוד:

* בניית המודולים
* קביעת הפונקציות הבסיסיות לכל מודול
* מימוש פונקציות העזר לפונקציות הפתירה
* מימוש פונקציות הפתירה (פותר תרגיל,פותר סוגרים,פותר אופרטור)

בעיות/קשיים במהלך פיתוח הפרויקט:

בשלב מתקדם של פיתוח הפרוייקט התברר כי חלה אי הבנה

ושהמינוס לא קודם לכל כמו שחשבנו , כך ש 2^5- אינו שווה 25 אלה 25-

דרך הפתרון שלי הושפעה בצורה דרסטית מכיוון שכתרגיל כמו : 2^2-+5

נפתר בצורה הבאה באלגוריתם הישן:

2^2-+5 -> 4+5 -> 9

ועל פי הנדרש לאחר השינוי:

2^2-+5 -> 4-+5 -> 1

כך לאחר שביצעתי מספר שינויים והאלגוריתם שונה נוצרה בעיה חדשה והיא:

כאשר סוגריים נפתרות האלגוריתם מציב את התוצאה מבלי הסוגריים,דבר שהוביל לתוצאות לא רצויות :

2^(1-3)+5 -> 2^2-+5 -> 4-+5 -> 1

בעוד שהנכונה:

2^(1-3)+5 -> 2^2-+5 -> 5+4 -> 9

לעטוף בסוגריים עגולים את תוצאת פתירת הסוגריים לא הייתה אופציה טובה מכיוון שהיא תיצור ריקוריסיה אין סופית של פתירת סוגריים ועטיפתן.

לאחר ניסיונות רבים למניפולציות בשלבי פתירת התרגיל נוצר קוד מסורבל ולא גנרי לפתירת הבעיה (שינוי יכל להביא לאי עמידת התוכנית בנדרש).

החלטתי שיש לשמור על פשטות האלגוריתם ושיטת הפתרון (סוגריים כעדיפות ראשונה ולאחר מכון פתירת האופרטור החזק ביותר) ולכן החלטתי שיש לשנות את צורת פתירת הבעיה עד שרעיון פשוט למדי התאים לפתרון הבעיה והוא עטיפת תשובה שלילית מסוגריים או אופרטור בסוגריים **ריבועיים** כך שריקורסיה אין סופית לא תתבצע לאחר פתרון סוגריים ולאחר התאמת הקוד – לא היה צורך בקוד המסורבל שנוצר עם פתרון הבעיה בדרך הישנה.

דרך הפתרון לאחר השינוי -

אם האופרטור החזק ביותר בתרגיל נמוך או שווה במשקלו למינוס ימחקו הסוגריים הריבועיים ורצפי מינוסים יעודכנו בהתאם. דוגמא:

*אך כאשר האופרטור החזק גדול במשקלו ממינוס לא ימחקו הסוגריים הריבועיים בתרגיל והפתרון יתבצע על פי הערכים שבתוכם:*

1- =

*מבנה הקוד –*

*על מנת שהקוד יהיה גנרי חילקתי את המודולים ואת תפקידי המערכת לחלקים ובכך יצרתי קוד מודולרי בו אפשר לערוך שינויים (גריעה/הוספה) במינימום עבודה.*

*לדוגמא, על מנת להוסיף אופרטור* על המתכנת לכתוב את מימוש האופרטור ולהוסיפו אל מילון האופרטורים בהתאם למבנה הנדרש –

מפתח – תו בודד המייצג את האופרטור

ערך – tuple עם שם מימוש הפעולה, משקל הפעולה והצד בו האופרטור מופיע ביחס לאופרנד/ים עליו הוא פועל.

נוסף על כך הסוגריים במחשבון הן דינאמיות כך שבמקום הסוגריים העגולים הבסיסיים תווים אחרים יכולים גם הם להיקבע לייצוג של סוגריים כמוהם - מלבד קבועי התוכנית על מנת לא ליצור כפילות!

בונוס:

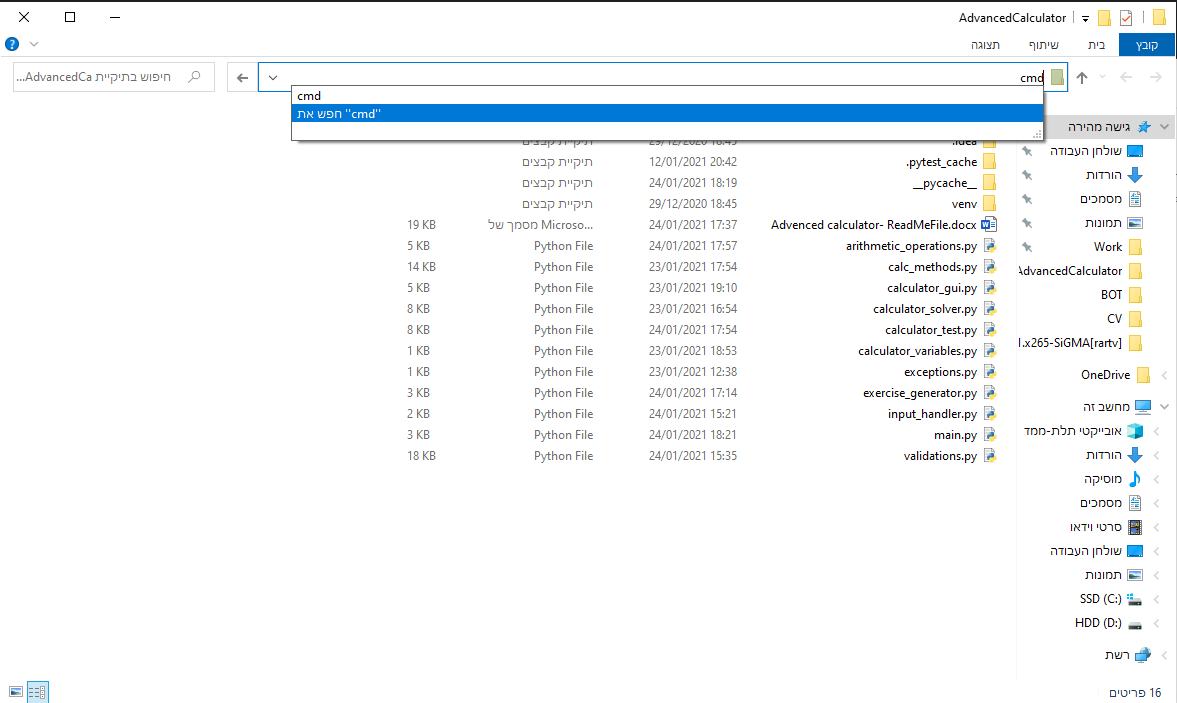
בעת כתיבת הטסטים לתוכנה הבנתי שדרך יותר יעילה וטובה לבדיקת המערכת הייתה ליצור תרגילים רנדומאליים ואיתם לבדוק את המערכת מכיוון שבדרך הרנדומאלית מצבים שלא חשבתי עליהם יכולים להתרחש.

לכן, יצרתי מודול שאחראי על יצירת תרגילים באופן רנדומאלי(תוך שימוש בספריית הrandom) באורך הרצוי ועל ידי כך לאחר הרצה מרובה של האלגוריתם "תפסתי" שני מקרי קצה שבהם התוכנית קרסה אותם לא הצלחתי למצוא לפני כן.

לאחר השימוש במודול הנ"ל לבדיקת התוכנית הטמעתי אותו אל תוך התפריט כאופציה לשימוש המשתמש.

הרצת התוכנית –

התוכנית כתובה בעזרת פייתון 3.6 לכן הורד במידה והגרסא אינה מותקנת במחשבך.

על מנת להריץ את התוכנית יש לחלץ את קובץ הrar של הפרוייקט, לאחר מכן יש לפתוח את התייקיה של הפרוייקט לכתוב "cmd" בשורה של כתובת הפרוייקט במחשב וללחוץ Enter 

לאחר מכן כתוב את הפקודה ""python main.py בתוך שורת הcmd.