נדב דיין 209044445 דניאל ארבוב 207232760 אריאלה אפשטיין 328606686

1. Bnf
2. הסבר כללי
3. תשובות של חלק 2(בקובץ partTwo.py)

bnf

program = statement\_list

statement\_list = statement | statement statement\_list

statement = declaration\_statement

| if\_statement

| while\_statement

| comparison

declaration\_statement = 'new' variable '=' expression

if\_statement = 'if' comparison 'do' statement\_list

| 'if' comparison 'do' statement\_list 'other do' statement\_list

while\_statement = 'while' comparison 'do' statement\_list

expression = term

| expression '+' term

| expression '-' term

term = factor

| term '\*' factor

| term '/' factor

factor = integer

| variable

| '(' expression ')'

comparison = expression '>' expression

| expression

| expression '>=' expression

| expression '<' expression

| expression '<=' expression

| expression '?' '=' expression

integer = [0-9]+

float = [0-9]+.[0-9]

variable = lowercase | variable lowercase | variable uppercase

lowercase = [a-z]

uppercase = [A-Z]

operations = "+-/\*()="  
stopwords = [" "]  
declarations = "new"  
comparisons = [">", "<", ">=", "<=", "?="]  
reserved = ["if", "other", "do", "while"]

כללי שפה- בשפה שלנו יהיו כללי חיבור חיסור כפל וחילוק ושימוש בסוגריים וזה יתבצע לפי סדר פעולות החשבון. יהיו לנו משתנים לשפה שיתמכו בint או float כדי להצהיר על משתנה או כדי להשתמש בו נצטרך לכתוב new לפני לדוגמא: new a = 5. רק נוכל לתת שמות למשתנים שמתחילים באות קטן (אחרי זה אפשר לשים אותיות קטנות\גדולות). השפה שלנו רק מקבל 2 משתנים.

בנוסף פקודות התנאי שלנו יכתבו כך if (some comparisons) do (statement) other do (statement) לדוגמא: if 5>3 do new a=5 other do new b=3 ולולאת הwhile תהיה כך: while (condition) do (statement) לדוגמא : while a<5 do new a=a+1. נתמוך גם במשתנים שליליים

הסבר כללי- יש לנו 3 שלבים לבניית הקוד.

1. בשלב הראשון החלטנו על סוגי הtoken – שלנו (integer, float ועוד) לאחר מכן בנינו את המחלקה lexer . הלקסר יקבל קלט של string ויתייחס לזה בתור סדרת תווים הוא ימיר אותם לסדרת טוקנים שיצרנו וישים אותם בlist משלהם וישלח אותם לparser.
2. בשלב השני נבנה את מחלקת הparser שתקבל סדרה של טוקנים ומתאים אותו לפי הדקדוק של השפה כדי שהinterpreter יוכל לטפל בו. השתמשנו במבני נתונים של עץ בינארי כדי לתאר את התכנים והמבנה של הקוד.
3. בשלב השלישי בנינו את מחלקת הinterpreter הוא בודק את המשמעות של כל קטע שהparser חילק למבנה תכנותי ומבצע את הקוד הדרוש לקטע זה הוא עובר על כל תת עץ בצורה רקורסיבית בדרך לחישוב הפעולות. בנוסף בנינו מחלקה בשם data שתשמור את הערכים במקרה שנרצה לבצע השמה למשתנים בעזרת Dictionaries כך נשמור בkey את המשתנה ובvalue את הערך