

## ניתוח הקשר בין מקוריות בתכנות לבין איכות קוד

אמיר רובינשטיין  
אוניברסיטת תל אביב  
[amirr@tau.ac.il](mailto:amirr@tau.ac.il)

תמי ורטהימר  
אוניברסיטת תל אביב  
[tamarw1@mail.tau.ac.il](mailto:tamarw1@mail.tau.ac.il)

ארנון הרשקוביץ  
אוניברסיטת תל אביב  
[arnonhe@tauex.tau.ac.il](mailto:arnonhe@tauex.tau.ac.il)

## Analyzing the Associations between Originality and Code Quality in Computer Programming

Tammy Wertheimer  
Tel Aviv University  
[tamarw1@mail.tau.ac.il](mailto:tamarw1@mail.tau.ac.il)

Amir Rubinstein  
Tel Aviv University  
[amirr@tau.ac.il](mailto:amirr@tau.ac.il)

Arnon Hershkovitz  
Tel Aviv University  
[arnonhe@tauex.tau.ac.il](mailto:arnonhe@tauex.tau.ac.il)

### Abstract

Creativity has been recognized as an important skill in the modern world. The research on the relationship between creativity and computational thinking has recently increased. However, little work has been done in the field of creativity in computer programming. In this work, we use tools of distance between programs, that are based on tree representation of the code, to define code originality measure. We analyzed the correlation between our originality measures and code quality measures – correctness, efficiency, and style – using single-variable and multivariable models. We discovered a negative correlation between code originality score and coding style and additional interesting trends, that could be related to the commonly used assignment designs or faults in our measures. We suggest further investigating alternative assignment settings and focusing on creativity measurement by integrating various creativity and code quality measures.

**Keywords:** Originality, Creativity, Computational Thinking, Code Distance, Code Quality.

### תקציר

יצירתיות זוהתה כמיומנות חשובה בעולם המודרני. חקר מערכת היחסים בין יצירתיות לבין חשיבה חישובית התגבר לאחרונה. עם זאת, נעשתה מעט עבודה בתחום היצירתיות בתכנות. בעבודה זו, אנחנו משתמשים בכלים שמודדים מרחק בין קטעי קוד, אשר מבוססים על ייצוג עץ של קוד, כדי להגדיר מדד למקוריות. ניתחנו את הקורלציה בין מדדי המקוריות שלנו לבין מדדים לאיכות קוד – נכונות, יעילות, וסגנון – בעזרת מודלים חד-משתניים ורב-משתניים. ממצאינו מגלים קורלציה שלילית בין ציון מקוריות קוד לבין סגנון קוד ומגמות מעניינות נוספות, אשר עשויות להיות קשורות לסגנון הגדרת משימות תכנות מגביל המקובל כיום, או למגבלות בשיטות שלנו. אנחנו מציעים להמשיך ולחקור את הסוגייה תחת הגדרות מטלות

אלטרנטיביות וכן להעמיק במדידת יצירתיות בתכנות תוך שילוב מגוון מדדים ליצירתיות ולאיכות קוד.

**מילות מפתח:** מקוריות, יצירתיות, חשיבה חישובית, מרחק קוד, איכות קוד.

## מבוא

### יצירתיות וחשיבה חישובית

העניין בקשר בין יצירתיות לבין חשיבה חישובית (CT), וההשפעות של מיומנויות אלו זו על זו, עלה לאחרונה. שתי המיומנויות זוהו ככלים הכרחיים בעולם המודרני התורמים ללמידה ולתהליכי פתרון בעיות. יתר על כן, התגלה שטיפוח יצירתיות יכול להועיל לרכישת CT (Israel-Fishelson & Hershkovitz, 2022) על פי אחת ההגדרות, חשיבה חישובית היא הבסיס הרעיוני הנדרש עבור פתרון בעיות, בשימוש בשיטות אלגוריתמיות, כדי להגיע לפתרון בר-העברה (Shute, Sun, & Asbell-Clarke, 2017). מיומנויות CT יכולות להתבטא בתכנות. לכן, זיהוי יצירתיות בהקשר של תכנות וחקר היחסים בין מיומנויות יצירתיות לבין מיומנויות תכנות יכול להיות מעניין ואף פורה. נעשה שימוש בסביבות תכנות לצורך חקר יצירתיות ו-CT (Hershkovitz, 2022 & Israel-Fishelson), אך לא קיימת התעסקות רבה במדידת יצירתיות בקוד עצמו. מחקרים קודמים ניסו לנבא יצירתיות בעזרת כלים מלמידה חישובית (Kovalkov, et al., 2021). חוסר ההסכמה בין מומחים, בנוגע לניקוד יצירתיות של קוד מקשה על תהליך התיגום ומדגיש את הצורך במדד יצירתיות שלא תלוי בהערכת שופט אנושי.

### יצירתיות

על פי הטקסונומיה של Rhodes (1961), יצירתיות מוגדרת על ידי ארבעה רכיבים: Person (1), מתייחס למאפייני היוצר; Process (2), מתייחס לתהליך הקוגניטיבי המעורב ביצירה; Product (3), מתייחס למימוש הרעיון היצירתי; Press (4), מתייחס להשפעות סביבתיות על היוצר. המדדים בהם נתעסק בעבודה קשורים לרכיבי ה-"Process" וה-"Product" המתייחסים ליצירה עצמה או לתהליך היצירה ולא ליוצר. על אף שאנחנו לא מתכוונים להתייחס לרכיבי ה-"Person" וה-"Press" בעבודתנו, אנחנו מודעים לכך שהם עלולים להיות מסיחים שצריך להתחשב בהם.

הגדרות מוכרות ליצירתיות מתמקדות באחד מארבעת הרכיבים. בנוגע לממד ה-"Process" (Giulford, 1950) ו-Torrance (1965) תיארו יצירתיות כמיומנות שיכולה להיות מוגדרת ולהימדד על ידי ארבעה מאפיינים. נחשוב על מקרה בו אדם מתבקש לספק פתרונות שונים לבעיה מסוימת: (1) שטף (Fluency), מתייחס למספר הרעיונות השונים שניתנו; (2) גמישות (Flexibility) מתייחסת למספר הסוגים השונים של רעיונות שניתנו; (3) מקוריות (Originality) מתייחסת למידה בה הרעיונות אינם שכחים באוכלוסיית יחס מסוימת; (4) פירוט (Elaboration), עד כמה הרעיונות מובעים בצורה מפורטת. המאפיין בו נתמקד הוא המקוריות. הגדרת היצירתיות של Mackinnon (1962) מתייחסת לממד ה-"Product" ומתחשבת באיכות ותפקוד התוצר ביחס לבעיה שהוא מנסה לפתור. לפי ההגדרה של Mackinnon "יצירתיות אמיתית" חייבת לפתור את הבעיה, להתאים למצב, או להשיג מטרה בת-זיהוי. הגדרה זו נראית רלוונטית בהקשר של CT ותכנות.

### ייצוג קוד

שיעורנו שמדידת ההבדלים בין תוכניות מחשב (קוד) יכולה לעזור לנו למדוד את הייחודיות של התוכניות. כדי למדוד את המרחק בין קטעי קוד בחרנו לייצג את הקוד בעזרת עץ תחביר מופשט (AST – Abstract Syntax Tree). ניתן לנצל ייצוג זה למשימות מדידת מרחק.

AST הוא מבנה נתונים מופשט המקובל במדעי המחשב לייצוג קטעי קוד. הוא מופשט במובן שהוא לא מתייחס לכל פרט ופרט בקוד, אלא למבנה הכללי שלו הייצוג באמצעות AST מאפשר לנו להגדיר מרחקים בין קטעי קוד באמצעות מדידת המרחק בין העצים המייצגים אותם; מדידת מרחק בין עצים היא ענף מפותח בספרות, ולפיכך ייצוג זה שימושי.

כלים לזיהוי העתקת קוד משתמשים ב-AST לעתים קרובות לצורך מדידת מרחקים בין תוכניות, במטרה לזהות תוכניות שונות שמיוצגות על ידי AST במבנה זהה. למשל, Kikuchi וחבריו (2014) השתמשו ברכיבים ומאפיינים שנלקחו מ-AST כדי לחשב דמיון יחסי בין תוכניות. באופן דומה, השתמשנו בייצוג AST כדי ליישם שני מדדי מרחק: *tree edit distance* ו-*alignment of trees*.

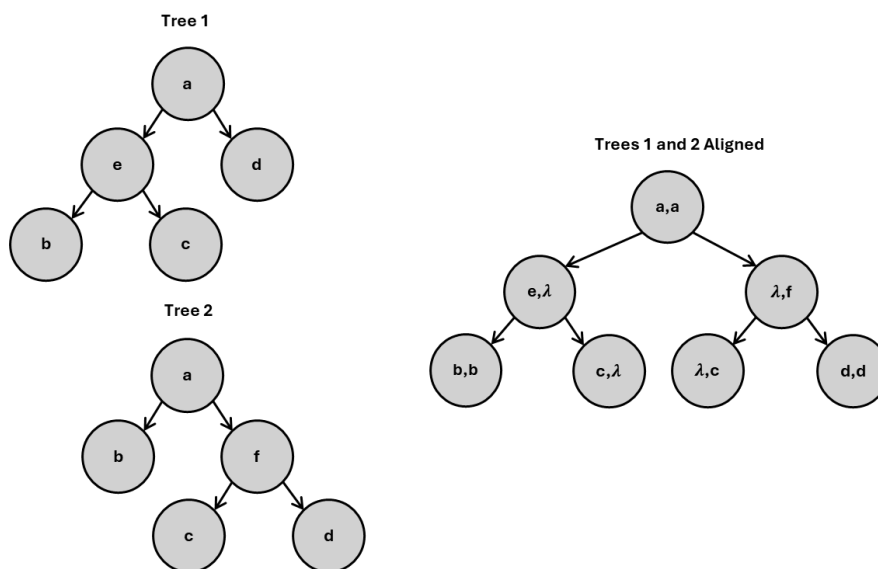
## מדדי מרחק

### Tree Edit Distance

Tree edit distance מגדיר מרחק בין עצים לפי מספר פעולות העריכה המינימלי הנדרש לשינוי עץ אחד לעץ אחר. פעולות העריכה כוללות הוספת צומת, מחיקת צומת, ושינוי שם של צומת. למדד מרחק זה קיימים שימושים רבים, כמו מדידת מרחק בין מבנים שניוניים של RNA (Le, Nussinov, & Maizel, 1989) וחישוב דמיון בטקסט לחישוב מרחק עריכת עץ (Zhang & Shasha, 1989).

### Alignment of Trees

Alignment of trees הוא מדד חלופי ל-tree edit distance, שהוצע על ידי Jiang וחבריו (1995). מדד זה מגדיר מרחק בין עצים לפי מספר פעולות העריכה המינימלי הנדרש לצורך מיפוי מלא בין שני העצים כך שנשמר המבנה של שניהם. גם כאן, פעולות העריכה המותרות הן הכנסת צומת, מחיקת צומת, ושינוי שם של צומת. בעוד שמרחקי עריכה ועימוד של סדרות או רצפים הם מושגים שקולים, במקרה של עצים מדובר במדדים שונים. Jiang וחבריו (1995) מדגימים בעבודתם מקרים בהם מרחק עריכה ומרחק עימוד שונים זה מזה במדידת מרחק בין שני עצים, כמתואר באיור 1. בפרט הם טוענים שציון העימוד נוטה להיות גבוה יותר ממרחק העריכה.



**איור 1.** דוגמה להבדל בין מרחק עריכה ומרחק עימוד. פעולות העריכה הדרושות לצורך מעבר מ-"Tree 1" ל-"Tree 2" הן מחיקת צומת "e" והכנסת צומת "f" – מרחק עריכה 2. בצד ימין עימוד (alignment) של שני העצים, כאשר "λ" מייצג עימוד של צומת מול צומת ריק (מתאר הכנסה/מחיקה) – מרחק עימוד 4.

## מתודולוגיה

### תיאור נתונים

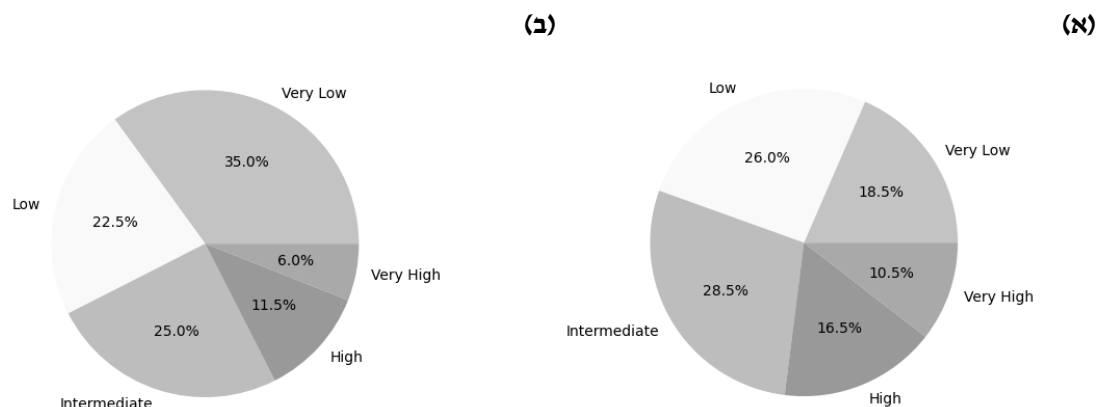
הדאטה עבור הניתוחים שלנו נאסף מהגשות שעורי בית בקורס מבוא מורחב למדעי המחשב (<http://tau-cs1001.py.wikidot.com/>). זהו קורס מבוא לתלמידי מדעי המחשב באוניברסיטת תל-אביב בו מודגמים רעיונות יסודיים במדעי המחשב בעזרת תכנות בפייתון. לכן, הדאטה שלנו מורכב כולו מתוכניות הכתובות בפייתון. קטעי הקוד בדאטה שלנו הם פתרונות לבעיית  $\max\_even\_seq(n)$ . בהינתן מספר שלם חיובי  $n$ , פונקציית  $\max\_even\_seq$  מחשבת את האורך של סדרה ארוכה ביותר המורכבת מספרות זוגיות בלבד במספר  $n$ . קטעי הקוד נאספו מהגשות בפעילות משוב עמיתים שהתקיימה במהלך הקורס. הפעילות הורכבה משלושה חלקים: (1) הסטודנטים הגישו ראשונה של המימושים של  $\max\_even\_seq(n)$ ; (2) סטודנטים נתנו ציוני נכונות, יעילות, וסגנון לתוכניות שהוגשו; (3) ניתנה הזדמנות להגשה חוזרת של קוד

מתוקן לפי המשוב שהתקבל. לכל הגשה ניתן ציון על ידי שלושה סטודנטים. ציוני משוב העמיתים שימושיים למטרותינו כמדדים לאיכות הקוד העשויים להיות קורלטיביים למקוריות קוד. הסטודנטים התבקשו לתת משוב בשלושה צירים: נכונות (האם התוכנית מחשבת מה שנדרש), יעילות (האם התוכנית עושה זאת ביעילות), וסגנון (האם התוכנית ברורה לקורא וכתובה על פי כללי סגנון מקובלים). נכונות נמדדה על ידי הרצת הקוד עם מספר קלטים מסוגים שונים; יעילות נמדדה בעזרת השוואה בין זמן הריצה של הקוד של הבודק לבין זמן הריצה של הקוד הנבדק עם מספר קלטים שונים; סגנון נמדד על ידי מספר מדדים שהוגדרו מראש בשיעור, כמו קריאות הקוד, אסטרטגית טיפול במקרי קצה, שימוש יעיל בזיכרון, שמות משמעותיים למשתנים, בחירת מבנה לולאה מתאים, ועוד. ניתן לסטודנטים חופש בבחירת הקלטים להרצת הקוד לצורך בדיקתו, תחת הנחיות כלליות בנוגע לסוגי הקלט שמצופה מהם לבדוק.

### סקירת המידע ועיבודו

קבצי הקוד בהם השתמשנו עבור הניתוחים שביצענו הם בעיקר מההגשה הראשונה בפעילות משוב העמיתים כיוון שהם קיבלו ציונים במהלך הפעילות כמתואר לעיל. מתוך 255 הגשות כללנו תוכניות עובדות של סטודנטים שהסכימו לשימוש בקוד שלהם למחקר ושהגישו שתי הגשות בפעילות. בסך הכל נכללו הגשות של 200 סטודנטים.

מידע נוסף שכללנו הוא רמת ניסיון קודם בתכנות, וזאת כדי לפלח את התוצאות בממד זה. לשם איסוף נתונים אלה העברנו סקר בתחילת הקורס בו סטודנטים דיווחו על ניסיון קודם בתכנות. התפלגות נתוני הניסיון הקודם, לפי תוצאות הסקר, מתוארת באיור 2.



איור 2. התפלגות רמות ניסיון קודם בתכנות על פי שאלון בדיווח עצמי. (א) ניסיון קודם בתכנות, (ב) ניסיון קודם בתכנות בפיתוח.

הערכנו את האמינות של ציוני משוב העמיתים בעזרת (Bartko, 1966) ICC intraclass correlation coefficient. ICC הוא כלי סטטיסטי מוכר המומלץ למדידת אמינות של שיטה ניסויית. כלי זה מתאר כמה חברים או יחידות באותה הקבוצה דומים זה לזה. במקרה שלנו, הקבוצות הם הציונים שניתנו להגשות על ידי שלושה מדרגים.

### ייצוג עץ

בחרנו את מבנה הAST לייצוג הקוד לצורך מדידת מרחקים בין קטעי קוד. לצורך פשטות חישוב המרחקים, המרנו את מבנה הAST הסטנדרטי של פייתון למבנה Tree Node של מודול zss (Henderson, 2014) בעזרת פונקציה המתוארת בנספח א.

**Tree Edit Distance**. השתמשנו במודול `zss` של Henderson (2014) לצורך חישוב מרחק עריכה. המודול מספק פונקציה המחשבת מרחק עריכה בין שני עצים ויישומים מועילים נוספים. המימוש של האלגוריתם במודול `zss` נלקח מהמאמר המקורי של Zhang ו-Shasha (1989) שהוזכר קודם. **Alignment of Trees**. השתמשנו במימוש שלנו על בסיס אלגוריתם התכנות הדינאמי המוצע על ידי Jiang וחבריו (1995).

## ניתוח מקוריות

**ציון מקוריות**. לכל תוכנית, חישבנו את המרחקים בינה לבין כל שאר התוכניות. ציון מקוריות של קטע קוד הוא המרחק הממוצע בינו לבין כל שאר קטעי הקוד. ציוני המקוריות חושבו בנפרד עבור קבוצת ההגשות הראשונות (לפני המשוב בפעילות משוב העמיתים) וקבוצת ההגשות החוזרות.

**קורלציה בין איכות קוד לבין מקוריות**. לכל רכיב ציון משוב העמיתים, חושבה הקורלציה בינו לבין ציון מקוריות. רכיבי ציון איכות הקוד הוגדרו להיות החציון בין שלושת הציונים שניתנו בפעילות משוב העמיתים. חזרנו על התהליך עם תתי-קבוצות לפי רמות ניסיון קודם. חישוב הקורלציות ורגרסיה לינארית בוצעו בעזרת `scipy` (Virtanen, et al., 2020) והגרפים בעזרת `matplotlib` (Hunter, 2007).

**מודל רב-משתנים**. ניתחנו את הקורלציה בין איכות קוד ומאפיינים אישיים לבין ציון מקוריות בעזרת מודל Random Forest (Breiman, 2001). השתמשנו ברמות ניסיון בתכנות ובפייתון ובערכי מינימום, מקסימום וחציון של ציוני משוב העמיתים כמשתנים במודל (22 משתנים). על מנת להשתמש בצורה נכונה במשתנים קטגוריאליים (רמות ניסיון) המרנו אותם למשתנים בינאריים (one hot encoding). חילקנו את הדאטה לקבוצת אימון (80%) וקבוצת מבחן (20%) על מנת להעריך את ביצועי המודלים. השתמשנו במימוש של `sklearn` למודל Random Forest (Pedregosa, et al., 2011).

**Partial Dependence Plots ו-Shapley Values**. בנוסף לבחינת ביצועי המודל ( $R^2$ ) וחשיבות הפיצורים, ניתחנו את השפעת המשתנים על הניבוי בעזרת `Shapley values` ו-`partial dependence plots (PDP)`.

`PDP` משמשים ללמידת היחסים הפונקציונליים בין משתנים במודלים ניבוי לבין התוצאה שמנבא המודל. הם מתארים בצורה גרפית את ההשפעה השולית של משתנה על התוצאה המנובאת אחרי התחשבות בהשפעה הממוצעת של כל שאר המשתנים.

`Shapley values` נלקחים מגישה נפוצה בתורת המשחקים הקואופרטיבית. ערכי `Shapley` מוחלטים מתארים את היקף התרומה של כל משתנה לתוצאה. חישוב `Shapley values` נחשב קשה חישובית, אך ישנן שיטות היוריסטיות בהן נהוג להשתמש. לצורך החישוב השתמשנו ב-`explainer` (Lundberg & Lee, 2017) ו-`partial dependence plots` (Friedman, 2001) של `SHAP`.

## תוצאות

### סקירת מדדי מרחק

ראשית, וידאנו שמדדי המרחק בהם השתמשנו אכן מודדים מרחק בין קטעי קוד. לפי התוצאות שלנו, המרחק בין שתי תוכניות זהות הוא אפס. בנוסף, המרחק הממוצע בין שתי תוכניות שהוגשו על ידי אותו סטודנט (כאשר ההגשה השנייה לא זהה להגשה הראשונה) נמוך מציון המקוריות הממוצע שחושב עבור קבוצת ההגשות הראשונות בפעילות משוב העמיתים (**טבלה 1**). בהנחה שקטעי הקוד בהגשה החוזרת כוללים בעיקר שינויים נקודתיים ביחס להגשה הראשונה (כלומר, חלק משמעותי מהמימוש נותר ללא שינוי), אפשר להסיק שהמדדים שלנו אכן נותנים ציון גבוה יותר לתוכניות שיותר שונות זו מזו. בקבוצת ההגשות החוזרות ניתן לראות ציוני מקוריות נמוכים יותר מאשר בהגשות הראשונות (**טבלה 1**). הירידה בשונות בין קטעי הקוד בהגשה החוזרת צפויה במשוב עמיתים (בו המשוב עשוי לכוון לשינוי מסוים בקוד בהתאם למימוש של כותב המשוב).

טבלה 1. סיכום מדידות מרחקי קוד. M (SD) – ממוצע (סטיית תקן); t – t-statistic; p – p-value.

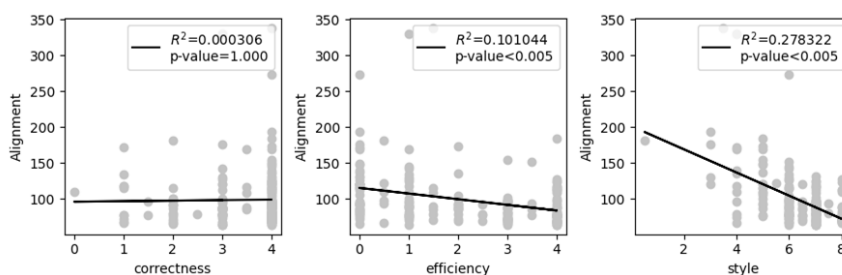
Alignment of Trees			Tree Edit Distance			N	סיכום ציוני מרחק
p <sup>3</sup>	t <sup>3</sup>	M (SD)	p <sup>3</sup>	t <sup>3</sup>	M (SD)		
<b>ציון מקוריות<sup>1</sup></b>							
		98.40 (38.06)			78.50 (32.46)	200	הגשה ראשונה
<0.001	8.77	79.60 (20.43)	<0.001	8.88	55.97 (15.19)	200	הגשה חוזרת
<b>מרחק בין הגשה ראשונה לחוזרת<sup>2</sup></b>							
<0.001	6.65	65.98 (54.72)	<0.001	5.27	56.15 (48.18)	165	

1. מרחק קוד ממוצע בין תוכנית לבין שאר התוכניות.
2. מרחק קוד בין הגשה ראשונה להגשה חוזרת של סטודנט (כאשר ההגשה החוזרת שונה מההגשה הראשונה).
3. p-value, t-statistic של מבחן t ביחס לציוני המקוריות שחושבו עבור קבוצת ההגשות הראשונות בפעילות משוב העמיתים.

### קורלציה בין ציונים לבין מקוריות

חיפשנו אחר קורלציה בין כל רכיב ציון משוב העמיתים לבין ציון מקוריות. בעקבות ערכי ICC1 (ICC המתאים למקרה בו כל מטרה מדורגת על ידי מדרגים שונים והמדרגים נבחרים באופן אקראי) נמוכים של ציוני יעילות (0.183) וסגנון (0.236) בחרנו בחציון של שלושת הדירוגים שניתנו לכל ציון, על פני הממוצע, כדי לאמוד את הציונים. באיור 3 ניתן לראות קורלציה שלילית בין מרחק קוד ממוצע וציון סגנון וקורלציה שלילית נמוכה בין מרחק קוד ממוצע לציון יעילות. הקורלציה השלילית בין מרחק קוד, שהוגדר כציון מקוריות, לבין ציון סגנון אינה מפתיעה כי מרחק קוד וסגנון קוד נופלים תחת קטגוריות שונות של הגדרת יצירתיות: אנחנו הערכנו את רכיב המקוריות ("Originality"), וציון הסגנון נותן מידע נוסף בנוגע לרכיב ה"Elaboration" של הגדרת היצירתיות. תוצאות אלו מדגישות את החשיבות של הכללת יותר ממרכיב אחד של יצירתיות כאשר מעריכים אותה. דרך אלטרנטיבית לכלול את רכיב ה"elaboration" בציון יותר מקיף ומייצג של יצירתיות הוא שקלול השפעת סגנון הקוד בציון בצורה של קנס או בונוס.

באופן מעניין, לא נראה שקיימת קורלציה בין ציון מקוריות לבין ציון נכונות הקוד. זה מרמז על כך שהמטלה אינה מוגבלת לפתרון ספציפי או לקבוצת פתרונות ספציפיים, ופתרונות יצירתיים מתקבלים. פרשנות נוספת לכך היא שאנחנו מודדים ייחודיות של קוד, מדד אשר יכול לקבל ערכים גבוהים גם במקרים של מימוש טוב וגם במקרים של מימוש גרוע, ואינו בהכרח מייצג יצירתיות. איור 4 מדגים מקרה בו שתי תוכניות מקבלות ציון מקוריות גבוה אך ציון הנכונות של המימושים שונה משמעותית.



איור 3. קורלציה בין ציון מקוריות (ציר X) לבין חציון ציוני משוב עמיתים (ציר Y). ערכי p-value מתוקנים על ידי תיקון בונפרוני. המגמות דומות בשימוש במרחק עריכה.

**(א) ציון נכונות גבוה**

```
def max_even_seq(n):
    assert isinstance(n, int)
    count = 0
    final_count = 1
    num = str(n)
    length = len(num) - 1
    i = 0
    while i <= length:
        if int(num[i])%2==0:
            count+=1
            break
        i+=1
    if count==0:
        return 0
    else:
        while i+1<=length:
            if int(num[i+1])%2==0:
                if int(num[i])%2==0:
                    count+=1
                else:
                    count=1
            else:
                count=0
        if count>final_count:
            final_count=count
        i+=1
    return final_count
```

**(ב) ציון נכונות נמוך**

```
def digit_in_index(n, i):
    return int(str(n)[i])

def num_length(n):
    return len(str(n))

def max_even_seq(n):
    counter = 0
    i = 0
    while i < num_length(n):
        if digit_in_index(n, i) % 2 == 0:
            c = 1
            j = i + 1
            stop = False
            while stop == False:
                if digit_in_index(n, j) % 2 == 0:
                    c += 1
                    j += 1
                else:
                    if c > counter:
                        counter = c
                    stop = True
                    i = j + 1
            else:
                i += 1
    return counter
```

**איור 4.** שני מימושים ל-`max_even_seq` בעלי ציוני מקוריות דומים **(א)** – עריכה: 155.93, עימוד: 183.70; **ב** – עריכה: 154.12, עימוד: 193.48) וציוני נכונות רחוקים מאוד **(א)** – 4.0; **ב** – 1.0, כלומר תוכנית א מחזירה פלט נכון עבור 4 מתוך 4 קלטים, ואילו תוכנית ב רק עבור 1 מתוך 4.

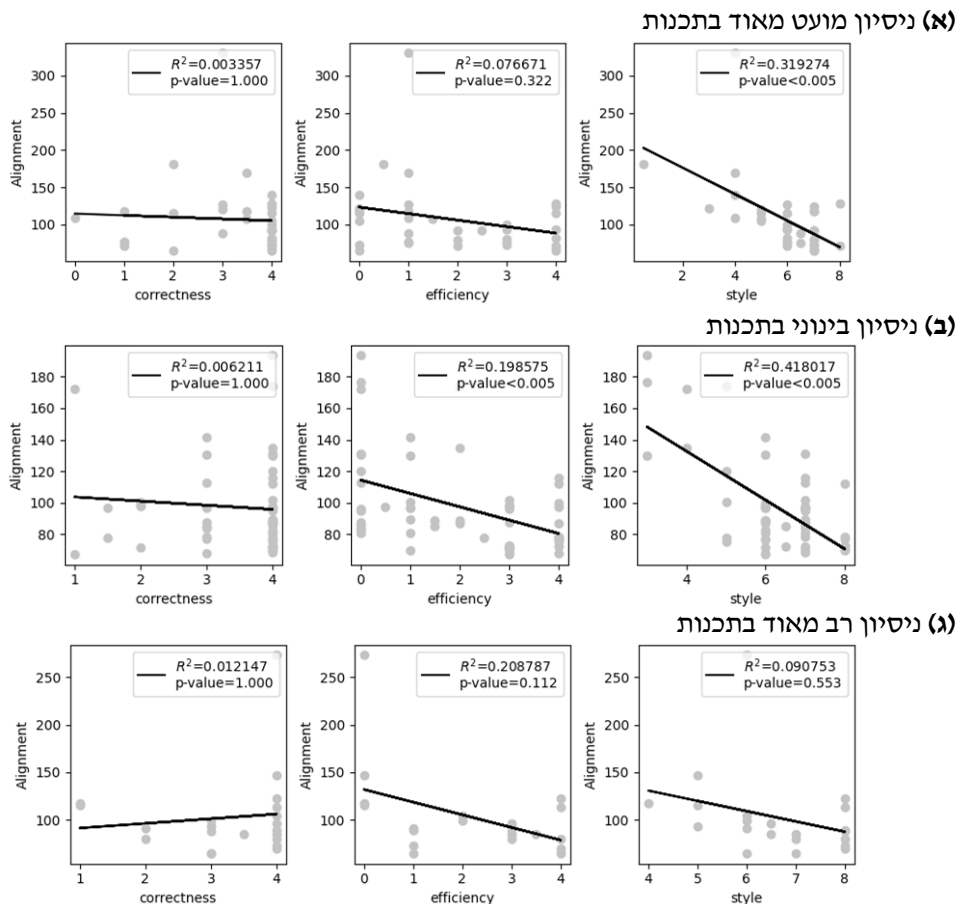
התוצאות במקרים של חלוקת הדאטה לתתי-קבוצות לפי רמות ניסיון **(איור 5)** מראות מגמות דומות של קורלציה בין ציון מקוריות לבין ציון סגנון בקבוצות המכילות סטודנטים בעלי ניסיון קודם נמוך מאוד וניסיון קודם בינוני (בתכנות), אבל אין מגמות משמעותיות בקבוצות בהן יש סטודנטים מנוסים יותר. הקורלציה בין ציון מקוריות לבין ציון יעילות חזקה יותר בקבוצה של סטודנטים עם ניסיון קודם ברמה בינונית.

**מודל רב-משתנים**

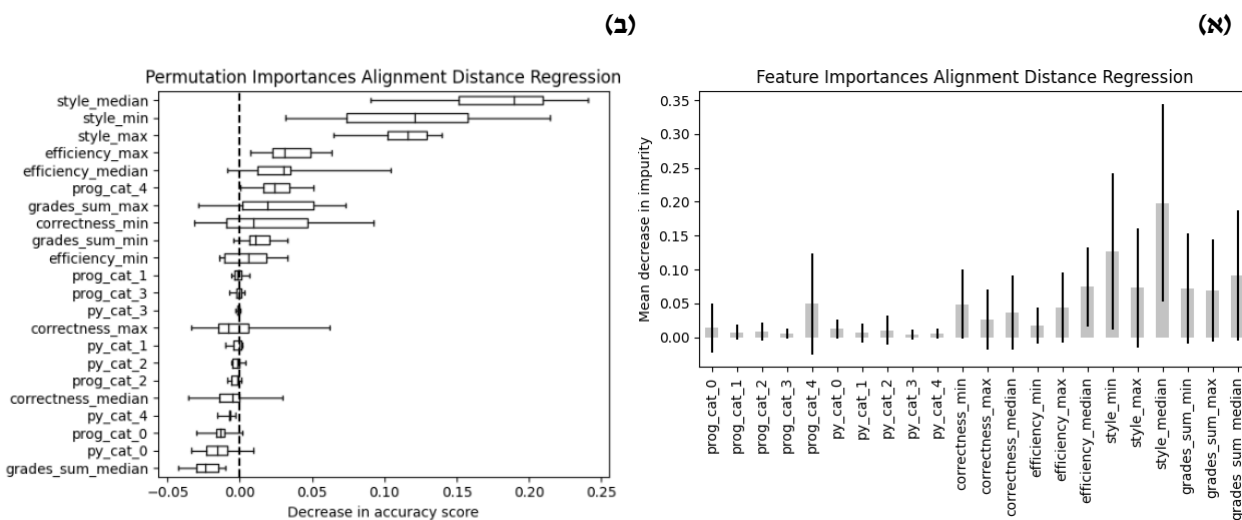
**קורלציה ו-Feature Importance.** הקורלציה בין ציוני משוב העמיתים לבין ציון מקוריות עשויה להיות מורכבת יותר משניתן לתאר על ידי מודל חד-משתני. על מנת לנתח את הקורלציה בתנאים מורכבים יותר השתמשנו במודל רב-משתנים. המשתנים של המודל כוללים את ציוני משוב העמיתים ומשתנים אישיים נוספים. ציוני ה- $R^2$  של מודלי הרגרסיה המנבאים את ציון המרחק הממוצע (מקוריות) הם 0.365 (edit distance) ו-0.406 (alignment).

**באיור 6** אנו רואים את תרומת המשתנים לניבוי על ידי feature importance של (MDI) Random Forest ו-`permutation importance`. תוצאות אלו עקביות ביחס לתוצאות של המודלים החד-משתניים – חציון ציוני הסגנון הוא בעל הקורלציה המשמעותית ביותר עם מדדי מקוריות קוד.

**Partial Dependence Plots and Shapley Values.** לצורך ניתוח השפעת המשתנים על תוצאת המודל השתמשנו ב-`partial dependence plots`. הגרפים **(איור 7א)** מראים את ההשפעה השלילית שיש למשתנה `style_median` על ביצועי המודל. **באיור 7ב** מוצגת תמונה מלאה של השפעת המשתנים על תוצאת המודל כפי שהיא מפורשת על ידי `Shapley values`.

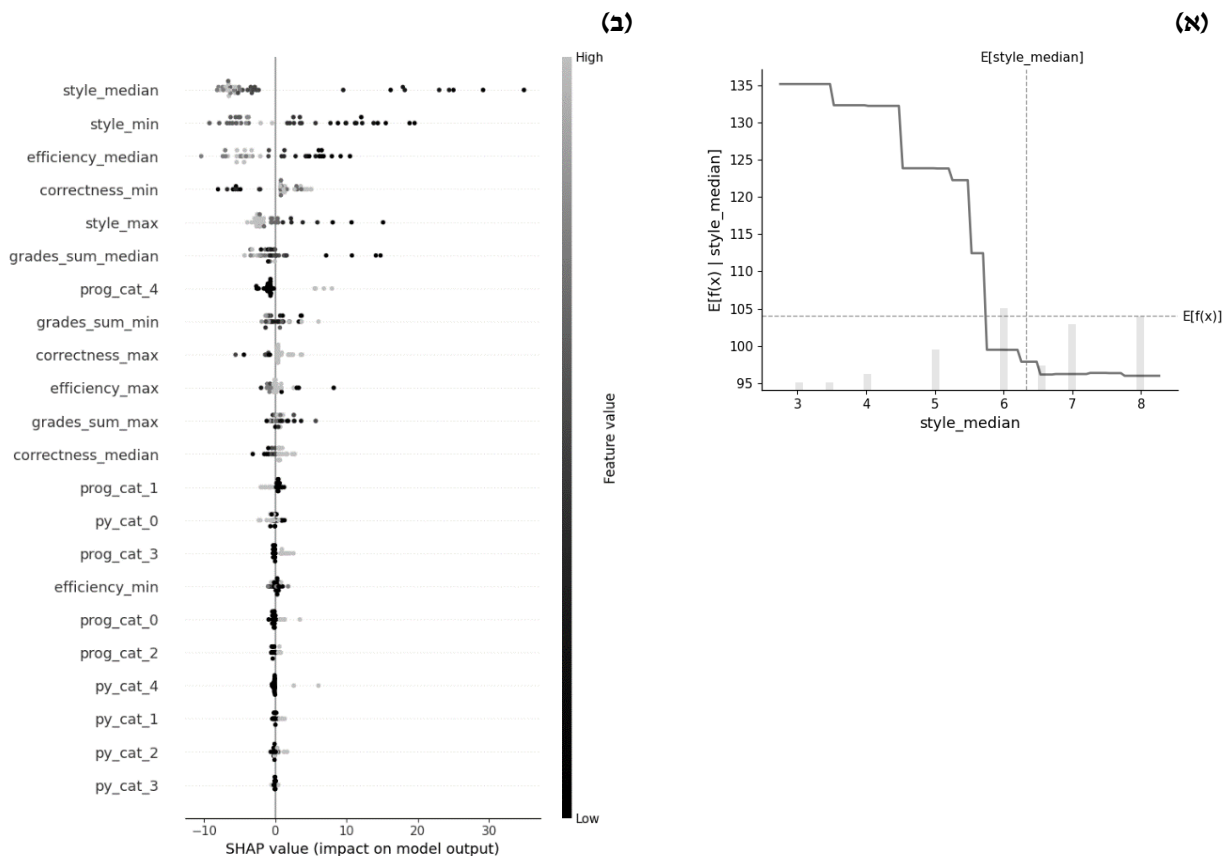


**איור 5.** דוגמאות לקורלציה בין חציוני ציוני משוב העמיתים (נכונות, יעילות, וסגנון) לבין מרחק עימוד ממוצע בתתי-קבוצות של סטודנטים בעלי רמות שונות של ניסיון קודם בתכנות. ערכי p-value מתוקנים על ידי תיקון בונפרוני. המגמות דומות בשימוש במרחק עריכה.



**איור 6.** MDI feature importance (א) ו-MDI feature importance (ב) לפי מודל רגרסיה Random Forest. המגמות דומות בשימוש במרחק עריכה.





**איור 7. (א) Partial Dependence Plot (PDP) המתאר את השפעת חציון ציון סגנון על תוצאת מודל רגרסיה המנבא מרחק עימוד ממוצע (ציון מקוריות). (ב) Beeswarm plot המסכם את השפעת כל המשתנים של מודל רגרסיה המנבא ציון מקוריות על תוצאת המודל. המגמות דומות בשימות במרחק עריכה.**

## דיון

בעבודה זו, ניסינו להגדיר מדד למקוריות בתכנות. השתמשנו בייצוג עץ של תוכניות מחשב כדי למדוד מרחקים בין תוכניות. השונו שני מדדי מרחק – מרחק עריכה ועימוד. חשוב לציין שבמקרה של מבנה עץ מרחק עריכה ועימוד אינם מושגים שקולים, בניגוד למקרה של מבנה של סדרה. השתמשנו בשלושה רכיבי ציון שניתנו במסגרת פעילות משוב העמיתים בקורס מבוא למדעי המחשב: נכונות, יעילות וסגנון. מדדים אלו מייצגים היבטים של איכות קוד, והשתמשנו בהם לצורך ניתוח הקורלציה בין איכות קוד לבין מרחק קוד, המייצג שונות בין קטעי קוד. בחנו את הקורלציה בעזרת מודלים חד-משתניים ומודלים רב-משתניים.

התוצאות שלנו הראו שקיימת קורלציה שלילית בין מרחק קוד ממוצע לבין ציון סגנון. הקורלציה השלילית הזאת רומזת על כך ששני המדדים שייכים למימדים שונים של יצירתיות. פרשנות אפשרית היא שככל שהקוד מקורי יותר, כך הוא חורג יותר ממוסכמות המקובלות בתחום התכנות, ולפיכך סגנונו נראה פחות מקובל ולכן מקבל ציון משוב נמוך יותר. השערה זו נתמכת במחקר מן העת האחרונה שהדגים קשר ישיר בין יצירתיות של קטעי קוד לאיכותם הסגנונית הנתפסת (Groeneveld, et al., 2022). עם זאת, אין בכך לומר על איכותו או שימושיותו של הקוד. שילוב של מאפיינים המתארים את איכות ושימושיות הקוד יכול להתבצע על ידי חילוץ נתונים ממדדי קוד סטטיים ודינאמיים כמו שמתארים (Manske & Hoppe, 2014), ולפיכך אנו ממליצים לבחון מדדים אלו עם מדדי יצירתיות. מאידך, מחקרים אחרים מדגישים את הקשרים החיוביים בין יצירתיות לבין התקדמות בתכנות (Israel-Fishelson, et al., 2021; Israel-Fishelson & Hershkovitz, 2022). לפיכך, מומלץ להדגיש את שני ההיבטים האלו: יצירתיות לצד סגנון.

מעניין לציין כי לא מצאנו קשרים בין יצירתיות הקוד לבין נכונותו. במחקר דומה שנערך לאחרונה, נמצא קשר שלישי בין יצירתיות קוד – שאף היא נמדדה על ידי ייצוג באמצעות AST – לבין הישגים בקורס מבוא לתכנות (Chou, Fossati, & Hershkovitz, 2024). ייתכן וביטוי היצירתיות אינו בעל השלכה ישירה על נכונות הקוד, אלא בעל השלכה מצטברת על הלמידה במהלך הקורס.

בנוסף, אנו ממליצים להעמיק במדידת יצירתיות בתכנות, ולכלול מימדים נוספים של יצירתיות. אלו יכולים לכלול בחינה של תוצרים (כפי שעשינו במחקר זה), או של תהליך התכנות; למשל, AST טמפורמלי (Moore, et al., 2022) יכול לספק מידע נוסף בנוגע לתהליך העבודה של הסטודנט במהלך התקדמות תהליך התכנות.

בעבודתנו קיימות מספר מגבלות. ראשית, ניתחנו מידע שנלקח ממטלה יחידה, עליה נערכה פעילות משוב העמיתים. כדי לקבל מסקנות יותר אמינות, יש לחזור על התהליך עם מידע נוסף. שנית, קטעי הקוד בהם השתמשנו לאנליזות שלנו נלקחו ממטלה בקורס אוניברסיטאי מבואי במדעי המחשב, ולכן חשוף להטיות הקשורות ללימודי תכנות קודמים (למשל בבית הספר), רמאות, שימוש במקורות מידע משותפים, או ניסוח מטלה שעשוי להוביל לבחירה בפתרונות מסוימים על פני אחרים. שלישית, ציוני משוב העמיתים ניתנו על ידי סטודנטים בשנתם הראשונה ללימודים – אינם בהכרח אמינים ואיכותם מוטלת בספק. למרות מגבלות אלו, אנו מאמינים כי המחקר הנוכחי שופך אור חדש וחשוב על הקשרים שבין יצירתיות ותכנות, אשר להם השלכות תיאורטיות ומעשיות חשובות.

## מקורות

- Bartko, J' J. (1966). 'The intraclass correlation coefficient as a measure of reliability'. *Psychological reports*, 11–3, 19.
- Breiman, L. (2001). Random forests. *Machine learning*, 45, 5–32.
- Chou, E., Fossati, D., & Hershkovitz, A. (2024). A code distance approach to measure originality in computer programming. 16th International Conference on Computer Supported Education. Angres, France.
- Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.
- Groeneveld, W., Martin, D., Poncelet, T., Aerts, K. (2022). Are Undergraduate Creative Coders Clean Coders? A Correlation Study. *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1* (pp. 314–320). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3478431.3499345
- Guilford, J. P. (1950). Creativity. *American psychologist*, 5, 444.
- Henderson, T. (2014). zss 1.1. 2-Tree edit distance using the Zhang Shasha algorithm. zss 1.1. 2-Tree edit distance using the Zhang Shasha algorithm.
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9, 90–95. doi:10.1109/MCSE.2007.55
- Israel-Fishelson, R & Hershkovitz, A. (2022). 'Studying interrelations of computational thinking and creativity: A scoping review (2020–2011)'. *Computers & Education*. 104353, 176, doi:https://doi.org/10.1016/j.compedu.2021.104353
- Israel-Fishelson, R., & Hershkovitz, A. (2022). Cultivating creativity improves middle school students' computational thinking skills. *Interactive Learning Environments*, 0, 1-16. doi:10.1080/10494820.2022.2088562
- Israel-Fishelson, R., Hershkovitz, A., Eguíluz, A., Garaizar, P., Guenaga, M. (2021). A log-based analysis of the associations between creativity and computational thinking. *Journal of Educational Computing Research*, 59, 926–959.
- Jiang, T., Wang, L., & Zhang, K. (1995). Alignment of trees—an alternative to tree edit. *Theoretical computer science*, 143, 137–148.
- Kikuchi, H., Goto, T., Wakatsuki, M., & Nishino, T. (2014). A source code plagiarism detecting method using alignment with abstract syntax tree elements. 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), (pp. 1-6). doi:10.1109/SNPD.2014.6888733
- Kovalkov, A., Paaßen, B., Segal, A., Pinkwart, N., Gal, K. (2021). Automatic Creativity Measurement in Scratch Programs Across Modalities. *IEEE Transactions on Learning Technologies*, 14, 740-753. doi:10.1109/TLT.2022.3144442

- Le, S.-Y., Nussinov, R., & Maizel, J. V. (1989). Tree graphs of RNA secondary structures and their comparisons. *Computers and Biomedical Research*, 22, 461–473.
- Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 4765–4774). Curran Associates, Inc. Retrieved from <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>
- MacKinnon, D. W. (1962). The nature and nurture of creative talent. *American psychologist*, 17, 484.
- Manske, S., & Hoppe, H. U. (2014). Automated indicators to assess the creativity of solutions to programming exercises. 2014 IEEE 14th international Conference on Advanced learning technologies (pp. 497--501). IEEE.
- Moore, D., Edwards, J., Karimi, H., Khadka, R., Bodily, P. (2022). Temporal Abstract Syntax Trees for Understanding Student Coding Thought Process. 2022 Intermountain Engineering, Technology and Computing (IETC), (pp. 1-6). doi:10.1109/IETC54973.2022.9796943
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Rhodes, M. (1961). An Analysis of Creativity. *The Phi Delta Kappan*, 42, 305–310. Retrieved April 22, 2023, from <http://www.jstor.org/stable/20342603>
- Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational research review*, 22, 142–158.
- Sidorov, G., Gómez-Adorno, H., Markov, I., Pinto, D., Loya, N. (2015). Computing text similarity using tree edit distance. 2015 Annual Conference of the North American Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conference on Soft Computing (WConSC), (pp. 1–4).
- Torrance, E. P. (1965). Scientific views of creativity and factors affecting its growth. *Daedalus*, 663–681.
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., . . . Contributors, S. I. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, 261–272. doi:10.1038/s41592-019-0686-2
- Zhang, K & ,Shasha, D .(1989) .Simple fast algorithms for the editing distance between trees and related problems .SIAM journal on computing, 18, 1262–1245.

## נספחים

## א AST to Tree

פונקציה הממירה את מבנה הAST של פייתון למבנה Node במודול .zss מחרוזות ושמות משתנים מוחלפים במחרוזת גנרית כדי להתגבר על הבדלים לא משמעותיים בין קטעי קוד.

```
def ast_to_tree(node):
    """
    Converts Python AST structure to tree structure
    :param node: AST node
    :return: zss.Node (tree)
    """
    tree = Node(node.__class__.__name__)

    for field, value in ast.iter_fields(node):
        if isinstance(value, list):
            for item in value:
                if isinstance(item, ast.AST):
                    tree.addkid(ast_to_tree(item))
        elif isinstance(value, ast.AST):
            tree.addkid(ast_to_tree(value))
        elif isinstance(value, str):
            tree.addkid(Node('SomeString')) # generic string
        else:
            tree.addkid(Node(repr(value)))

    return tree
```