

מדריך הערכה עצמית מקיף

Comprehensive Self-Assessment Guide

Dr. Yoram Segal

Version 2.0
22-11-2025

תוכן העניינים

3	1	מבוא כללי
3	1.1	מטרת המדריך
3	2.1	كيف להשתמש במדריך זה
4	I	עמוד הנקודות הערכות האקדמית
4	2	עקרונות יסוד
4	1.2	עקרון מרכזי: רמת הדקדנות בבדיקה תושפע מהציווין העצמי
4	3	המלצת שלבים לביצוע הערכה עצמית
4	1.3	שלב 1: הבנת הקритריונים והתקנים
4	2.3	שלב 2: מיפוי העבודה שלכם מול הקритריונים
4	1.2.3	20% -- (Project Documentation)
5	2.2.3	15% -- (Code Documentation) README
5	3.2.3	15% -- (Project Structure & Code Quality) מבנה פרויקט וaicיות קוד
6	4.2.3	10% -- (Configuration & Security) קונפיגורציה ואבטחה
6	5.2.3	15% -- (Testing & QA) בדיקות וaicיות
7	6.2.3	15% -- (Research & Analysis) מחקר וניתוח
7	7.2.3	10% -- (UI/UX & Extensibility) משקל משתמש והרחבה
7	3.3	שלב 3: ניתוח עמוק וייחודיות
9	4	קבעת הציון העצמי -- מדריך לפי רמות
9	1.4	רמה 1: ציון 69--60 (מעבר בסיסי)
9	2.4	רמה 2: ציון 79--70 (טוב)
9	3.4	רמה 3: ציון 89--80 (טוב מאוד)
01	4.4	רמה 4: ציון 90--100 (מצוינות יוצאת דופן)
11	5	טבלת הערכה עצמית מסכמת
21	6	טופס הגשת הערכה עצמית
21	1.6	הצדקה להערכת העצמית (חובה -- 500--200 מילים)
21	2.6	רמת הדקדנות המבוקשת בבדיקה
31	3.6	הצהרת יושר אקדמי (Academic Integrity Declaration)

41	7	טיפים להערכת עצמית מוצלחת
41	1.7	עשו (DO):
41	2.7	אל תעשו (DON'T):
51	8	שאלות נפוצות (FAQ)
61	II	בדיקה טכנית מפורטת של הקוד
61	9	מבוא לבדיקה הטכנית
61	10	בדיקה א: ארגון הפרויקט כחבילת רקע
61	1.01	רשמיota בדיקה: מבנה החבילה
61	2.01	דוגמאות למבנה נכון
71	3.01	הערות נוספות
71	4.01	הערות נוספות
81	11	בדיקה ב: שימוש במעבדים מרובבים וחוטי ביצוע מרובבים
81	1.11	רשמיota בדיקה: Multiprocessing
81	2.11	רשמיota בדיקה: Multithreading
81	3.11	שיקולים נוספים
91	4.11	הערות נוספות
91	5.11	הערות נוספות
02	21	בדיקה ג: עיצוב מבוסס אבני בניין
02	1.21	רשמיota בדיקה: עקרונות עיצוב
02	2.21	רשמיota בדיקה: זיהוי אבני בניין
02	3.21	רשמיota בדיקה: נתוני קלט (Input Data)
02	4.21	רשמיota בדיקה: נתוני פלט (Output Data)
12	5.21	רשמיota בדיקה: נתוני ההדרה (Setup Data)
12	6.21	דוגמה לאבן לבנייה טובה
22	7.21	הערות נוספות
22	8.21	הערות נוספות
32	III	סיכום והמלצות
32	31	ציון כללי
32	1.31	ציון אקדמי (מהחלק הראשון)
32	2.31	ציון טכני (מהחלק השני)
32	3.31	ציון כולל
32	41	תחומיים לשיפור
42	51	תכנית פעולה
42	61	לסיכום

1 מבוא כללי

מדריך זה מיועד לסייע לכם בביצוע הערכה עצמית מקיפה של פרויקט הקוד שלכם, הנקה מהביקורת האקדמית והנקה מהביקורת הטכנית. הערכה העצמית היא חלק חיוני מתהליכי הלמידה והפיתוח, ומאפשרת לכם לזהות נקודות חזקה וחולשה, לפתח חשיבה רפלקטיבית, ולשפר את יכולות העבודה.

1.1 מטרת המדריך

מדריך זה משלב שני ממדים:

1. **הערכת אקדמית:** מסגרת להערכת עצמית של יכולות העבודה הכלכלית, התיעוד, והמחקר
2. **הערכת טכנית:** בדיקה מפורטת של היבטים טכניים כגון ארגון קוד, multipro cessing, ועיצוב מודולרי.

2.1 כיצד להשתמש במדריך זה

עbero על כל חלק בתשומת לב, וענו על השאלות בהתאם לפרויקט שלכם. סמןו את הפריטים שהושלמו, וציינו את אלו הדורשים שיפור. הקדישו זמן לרפלקציה ולחשיבה ביקורתית על העבודה שלכם.

חלק I

עקרונות הערכתה העצמית האקדמית

2 עקרונות יסוד

הערכתה עצמית היא תהליך אקדמי חשוב רפלקטיבית, אחראיות אישית ומודעות למהלך הלמידה. במסגרת קורס זה, אתם מוזמנים לקבוע את הציון שאתם סבורים שגיעו לכם. ציון זה יקבע לפי העריכתכם העצמית את איכות העבודה שהגשתם ביחס לקריטריונים שנקבעו.

1.2 עקרון מרכזי: רמת הדקנות בבדיקה תשفع מהציון העצמי

כל שציון העצמי גבוה יותר, כך רמת הדקנות והקפדות בבדיקה תהיה גבוהה יותר. זהו עקרון של "הבטחה-ציפייה" (Contract-Based Grading):

- **ציון עצמי גבוה (100--90):** בדיקה מאוד דקדנית ומעמיקה, חיפוש 'פילים' בקנה", בדיקת כל פרט קטן
- **ציון עצמי בינוני (89--75):** בדיקה סבירה ומאזנת עם קריטריונים ברורים
- **ציון עצמי נמוך יותר (74--60):** בדיקה גמישה, אזהרת ומכילה -- אם יש הגיון בהגשה והיא סבירה

3 המלצה לשלבים לביצוע הערכתה עצמית

אין צורך למלא ולהגיש זאת בלבד עם הגשת המטלה, אך מומלץ מאוד.

1.3 שלב 1: הבנת הקריטריונים והתקנים

לפני שתוכלו להעריך את עבודותכם, عليיכם להבין במדויק את הקריטריונים שעל פיהם העבודה תוערך:

- קראו בעיון את מסמך ההנחיות להגשת תוכנה
- זהו את כל המרכיבים הנדרשים (תיעוד, קוד, בדיקות, ניתוח וכו')
- הבינו את רמת האיכות המczופה בכל קריטריון
- שימו לב להבחנות בין רמות איכות שונות

2.3 שלב 2: מיפוי העבודה שלכם מול הקריטריונים

השתמשו ב-checklist הבא כדי לבדוק אילו מרכיבים כללתם בעבודה:

1.2.3 **תיעוד פרויקט (Project Documentation) -- 20%**

PRD (Product Requirements Document)

- תיאור ברור של מטרת הפרויקט וביעית המשתמש
- יעדים מדדיים ומדדי הצלחה (KPIs)
- דרישות פונקציונליות ולא-פונקציונליות מפורטות

תלויות, הנחות ו מגבלות

ציר זמן ו אבני דרך

תיעוד ארכיטקטורה

תרשימי בלוקים (C4 Model, UML)

ארכיטקטורה תפעולית

החלטות ארכיטקטוניות (ADRs)

תיעוד API וממשקים

ציוויליזצייתם /20 _____
ציוויליזצייתם בקטגוריה זו:

2.2.3 README ותיעוד קוד 15% -- (Code Documentation)

README מקיין

הוראות התקינה שלב-אחר-שלב

הוראות הפעלה מפורטות

דוגמאות ריצה וצלומי מסך

מדריך קונפיגורציה

troubleshooting

aicoot hevrot be-kod

Docstrings לכל פונקציה/class/מודול

הסברים על החלטות עיצוב מורכבות

שמות משתנים ופונקציות תיאוריות

ציוויליזצייתם /15 _____
ציוויליזצייתם בקטגוריה זו:

3.2.3 מבנה פרויקט ואיכות קוד 15% -- (Project Structure & Code Quality)

ארגן פרויקט

מבנה תקין מודולרי וברור (src/, tests/, docs/, data/, results/, config/, as-sets/)

הפרדה בין קוד, נתונים ותוכנות

קבצים לא עולים על ~150 שורות

naming conventions עיקיבות

aicot koud

- פונקציות קצרות וממוקדות (Single Responsibility)
- הימנעות מקוד כפול (DRY)
- עקביות בסגנון קוד

ציוו עצמי בקטgorיה זו: /15 _____

4.2.3 10% -- (Configuration & Security)

ניהול קונפיגורציה

- קבועי קונפיגורציה נפרדים (.env, .yaml, .json)
- אין קבועים hardcoded בקוד
- קבוע דוגמה (.env.example)
- תיעוד פרמטרים

abetachat midu

- אין API keys בקוד המקורי
- שימוש במשתני סביבה
- .gitignore מעודכן

ציוו עצמי בקטgorיה זו: /10 _____

5.2.3 15% -- (Testing & QA)

cisoi b'dikot

- תיעוד unit tests עם cisoi + 70% לקוד חדש
- בדיקות edge cases
- דוחות cisoi (coverage reports)

tipol b'shagiavot

- תיעוד edge cases עם תיאור ותגובה
- error handling מקיף
- הודעות שגיאה ברורות
- לוגים לצורך debugging

tozavot b'dika

- תיעוד tozavot zpoyot
- דוחות automated testing

ציוו עצמי בקטgorיה זו: /15 _____

6.2.3 15% -- מחקר וניתוח (Research & Analysis)

ניסויים ופרמטרים

- ניסויים שיטתיים עם שינוי פרמטרים
- ניתוח רגישות (sensitivity analysis)
- טבלת ניסויים עם תוצאות
- זיהוי פרמטרים קרייטריים

מחברת ניתוח

- Notebook Jupyter או כלי דומה
- ניתוח מתודדי ומעמיך
- נוסחאות מתמטיות ב- \LaTeX (אם רלוונטי)
- אסמכתאות בספרות אקדמית

הצגה ויזואלית

- גרפים איקוטיים (bar charts, line charts, heatmaps, וכו')
 - תוויות ומרקא ברורים
 - רזולוציה גבוהה
- ציון עצמי בקטgorיה זו: /15 _____**

7.2.3 10% -- ממשק משתמש והרחבה (UI/UX & Extensibility)

משחק משתמש

- ממשק ברור ואינטואיטיבי
- צילומי מסך ותיעוד workflow
- נגישות (accessibility)

הרחבה (Extensibility)

- נקודות הרחבה (extension points/hooks)
 - פיתוח plugins
 - ממשקים ברורים
- ציון עצמי בקטgorיה זו: /10 _____**

3.3 שלב 3: ניתוח עומק וייחודיות

ענו על השאלות הבאות להערכת העומק והיחודיות של העבודה:

עומק טכני

- השתמשתי בטכניקות מתקדמות של סוכני AI?
- הוסףתי ניתוח מתמטי או תיאורטי?
- ביצעתי מחקר השוואתי בין גישות שונות?

"חדשנות וחדשנות"

- הפרויקט כולל רעיונות מקוריים או גישה חדשה?
- פיתחתי פתרון לבעה מורכבת או מודגמת?
- הוסףתי ערך מעבר לדרישות הבסיסיות?

ספר הפרומפטים

- תיעדתי את תהליך הפיתוח עם AI?
- כללתי דוגמאות לפרומפטים משמעותיים?
- הוסףתי best practices מהניסיונו?

علויות ותמכhor

- חישבתי שימוש ב-tokens?
- הצגת טבלת עלויות מפורטת?
- הצעתה אסטרטגיות אופטימיזציה?

4 קביעת הציון העצמי -- מדריך לפי רמות

1.4 רמה 1: ציון 69--60 (מעבר בסיסי)

תיאור: הגשה סבירה שמקסה את הדרישות המינימליות מאפייניות:

- ◻ קוד עובד ומבצע את המשימות הנדרשות
- ◻ תיעוד בסיסי README עם הוראות התקנה והפעלה
- ◻ מבנה פרויקט הגיוני אך לא בהכרח מושלם
- ◻ בדיקות בסיסיות או כיסוי חלקי
- ◻ תוצאות קיימות אך ללא ניתוח עמוק

רמת הבדיקה שתקבלו: גמישה, אוחדת ומכילה. הבודקים יחפשו את ההגיוון והסבירות בעבודה ולא יתעכבו על פרטים קטנים.

המלצה: בחרו ברמה זו אם השקעתם מאמץ סביר אך יודעים שהעובדת לא מושלמת, או אם הזמן היה מוגבל.

2.4 רמה 2: ציון 79--70 (טוב)

תיאור: עבודה אינטואיטיבית עם תיעוד טוב ומבנה מסודר מאפייניות:

- ◻ קוד מסודר עם הערות והפרדה למודולים
- ◻ תיעוד מקיף: README טוב, תיעוד ארכיטקטורה, PRD בסיסי
- ◻ מבנה פרויקט נכון עם הפרדה בין קוד, נתונים ותוצאות
- ◻ בדיקות עם כיסוי 50--70%
- ◻ ניתוח תוצאות עם גרפים בסיסיים
- ◻ קונפיגורציה נכונה ואבטחת API keys

רמת הבדיקה שתקבלו: סבירה ומאוזנת. הבודקים יבדקו עמידה בקריטריונים עיקריים אך יתנו מרחב לשגיאות קטנות.

המלצה: בחרו ברמה זו אם עדתם ברוב הדרישות ייצרתם עבודה מסודרת ואינטואיטיבית.

3.4 רמה 3: ציון 89--80 (טוב מאוד)

תיאור: עבודה מצוינת בrama אקדמית גבוהה מאפייניות:

- ◻ קוד מקטיע עם גובה ו הפרדת אחריות modularity
- ◻ תיעוד מלא ומפורט: PRD מקיף, ארכיטקטורה עם תרשימי C4, README בrama user manual
- ◻ מבנה פרויקט מושלם לפי best practices

- בדיקות מקיפות עם CISCO 85%-70%
 - מחקר ממשי: ניתוח רגישות פרמטרים, מחברת ניתוח עם נוסחאות
 - הצגה ויזואלית מרשימה של תוצאות
 - ממשק משתמש אינטראקטיבי
 - עליות מתועדות וניתוח אופטימיזציה
- רמת הבדיקה שתקבעו:** מעמיקה ומדוקדקת. הבודקים יבדקו התאמה מלאה לкрיטריונים ויקפידו על רמת אינטראקטיביות גבוהה.
- המלצה:** בחרו בrama זו אם השקעתם מאמץ משמעותי, כיסיתם את כל הדרישות, וביצעתם מחקר אמיתי.

4.4 רמה 4: ציון 100--90 (מצוינות יוצאה דופן)

תיאורו: רמת MIT -- עבודה ברמות פרסום אקדמי או תעשייתי מאפיינים:

- קוד ברמת production עם extensibility, hooks, וארכיטקטורת plugins
 - תיעוד מושלם ומפורט בכל היבט: PRD מקיים, תיעוד ארכיטקטורה מלא, README מפורט, מקצועני
 - עמידה מלאה בתקן ISO/IEC 25010
 - בדיקות מקיפות עם CISCO 85%+ edge cases מתועדים ומטענים
 - מחקר מעמיק: ניתוח רגישות שיטתי, הוכחות מתמטיות, השוואת מבוססת-נתונים
 - ויזואיזציה ברמה גבוהה ביותר עם dashboard אינטראקטיבי
 - ספר פרומפטים מפורט וمتוועד
 - ניתוח עליות מקיף עם המלצות אופטימיזציה
 - חידושים וייחודיות: רעיונות מקוריים, פתרון לבעה מורכבת
 - תרומה להקה: קוד פתוח, תיעוד לשימוש חוץ
- רמת הבדיקה שתקבעו:** דקדקנית ביותר -- ``חיפוש פילים בקנה''. הבודקים יבדקו כל פרט קטן, יחשו חוסרים קטנטנים, ויקפידו על כל ``קוצו של יוד''.
- ażhorah:** רמה זו מיועדת למי שבתו לחלוטין שהעבודה ברמת מצוינות מקסימלית. אם יימצאו חוסרים, הציון עלול לרדת משמעותית.
- המלצה:** בחרו בrama זו רק אם:
- כיסיתם את כל הדרישות ללא יוצא מן הכלל
 - ביצעתם בדיקה עצמית מעמיקה והכל מושלם
 - יש חידושים וייחודיות משמעותית
 - אתם מוכנים לבדיקה מאוד קפדנית

5 טבלת הערכה עצמית מסכמת

השתמשו בטבלה הבאה כדי לחשב את הציון העצמי שלכם:

קטgorיה	משקל	הציון שלי	ציון משקלל
תיעוד פרויקט (PRD, ארכיטקטורה)	20%	_____	_____
README ותיעוד קוד	15%	_____	_____
מבנה פרויקט ואיוכות קוד	15%	_____	_____
קונפיגורציה ואבטחה	10%	_____	_____
בדיקות ואיוכות (Testing & QA)	15%	_____	_____
מחקר וניתוח תוצאות	15%	_____	_____
ממשק משתמש ורחבבה	10%	_____	_____
סה' ב'	100%		

6 טופס הגשת הערכה עצמית

אנא מלאו את המידע הבא והגשו יחד עם הפרויקט:

שם הסטודנט/ים: _____
שם הפרויקט: _____
תאריך הגשה: _____
הציון העצמי שלי: _____ /100

1.6 הצדק להערכת העצמית (חובה -- 500-200 מילימ"מ)

בסעיף זה, הסבירו למה בחרתם בציון זה. כללו:

- נקודות חזק: מה עשיתם במיוחד טוב? אילו מרכיבים הם ברמה גבוהה?
- נקודות חולשה: מה חסר או יכול היה להיות טוב יותר? (כנות מוערכות!)
- השקעה: כמה זמן ומאזן השקעתם?
- חדשנות: האם יש משהו ייחודי או מיוחד בעבודה?
- למידה: מה למדתם מהפרויקט?

2.6 רמת הדקדנות המבוקשת בבדיקה

על פי הציון העצמי שנתי, אני מבין/ה לרמת הבדיקה תהיה:

- 60--69: גמישה, אוחצת ומכילה -- בדיקת הגיון וההתאמה בסיסית
- 70--79: סבירה ומאוזנת -- בדיקת קритריונים עיקריים
- 80--89: מעמיקה ומודקחת -- בדיקה מלאה של כל הקритריונים
- 90--100: דקדקנית ביותר -- חיפוש "פילים בקנה", הקפה על כל פרט

3.6 הצהרת יושר אקדמי (Academic Integrity Declaration)

אני מצהיר/ה בזאת ש:

- הערכתה העצמית שלי** היא כנה ואמיתית
- בדקתי את העבודה מול כל הקריטריונים** לפני קביעת הציון
- אני מודע/ת** ש**ציון עצמי גבוה** יוביל לבדיקה דקדקנית יותר
- אני מקבל/ת את העבודה שהציון הסופי עשוי להיות שונה מהציון העצמי**
- העבודה היא פרי עבודתי/נו** (של הקבוצה) ואני/ו אחראי/ם לכל תוכנה

תאריך:

חתימה:

7 טיפים להערכת עצמית מוצלחת

1.7 עשו (DO):

- **היו כנים** -- הערכה עצמית מדוייקת מועילה לכם יותר מאשר מצוין מנופה
- **השתמשו בקריטריונים** -- עברו שיטתיות על כל סעיף בהנחיות
- **תעדו את התהילה** -- שמרו רשיימה של מה עשיתם ומה חסר
- **קבלו פידבק** -- שאלו חברים לעבור על העבודה לפני הגשה
- **הקדישו זמן לרפלקציה** -- חשבו מה למדתם והיכן אתם יכולים להשתפר

2.7 אל תעשו (DON'T):

- **אל תנפחו ציון** -- ציון גבוה מדי יוביל לבדיקה קשה ואכזבה
- **אל תזלוו בעבודה** -- גם אם לא מושלמת, יתכן שהיא יותר ממה שחשובים
- **אל תשחחו הצדקה** -- הסבר חסר יקשה על הבוקדים להבין את ההערכתה
- **אל תחכו לרגע האחרון** -- הערכה עצמית אינטואיטיבית דורשת זמן
- **אל תשחחו את החתימה** -- הצהרות יושר אקדמי היא חובה

8 שאלות נפוצות (FAQ)

ש: מה קורה אם הציון שאתנו לעצמי יהיה שונה מהציון שהבודק/ת יתן/תן?
ת: זה נורמלי לחלוtin. הערכה עצמית היא כלי למידה, לא קביעת ציון סופית. הבודק/ת ישקל את ההערכה העצמית אך יקבעו את הציון הסופי על פי שיקול דעתם המ铿וציאי.

ש: האם כדאי לבחור בציון נמוך כדי לקבל בדיקה גמישה?
ת: לא. ציון עצמי נמוך יכול להוביל לציון סופי נמוך יותר גם אם העבודה טובה. הבחירה צריכה לשקף את אιוכות העבודה בפועל, לא אסטרטגיה.

ש: האם יש ערעור על הציון?
ת: כן, אך הבסיס לערעור חייב להיות ממשמעותי. אם הערכתם את עצמכם בכנות, ערעורים יהיו נדירים.

ש: האם אפשר לעדכן את הציון העצמי לאחר ההגשה?
ת: לא. הציון העצמי הוא חלק מההגשה וקובע את רמת הבדיקה. שינוי לאחר מכן אינו אפשרי.

ש: מה אם קשה לי להעריך את עצמי?
ת: השתמשו ב-checklist, שallow חברי קבוצה או עמיתים, והיערו בקריטריונים המפורטים. הערכה עצמית היא מiomנות שמשתפרת עם התרגול.

חלק II

בדיקות טכניות מפורטת של הקוד

9 מבוא לבדיקה הטכנית

חלק זה של המדריך מתמקד בהיבטים הטכניים של פרויקט הקוד, ומספק רשימות בדיקה מפורטות לארಗון הפרויקט כחברה, שימוש במעבדים רבים וחוטי ביצוע, ועיצוב מבוסס אבני בניין.

10 בדיקה א: ארגון הפרויקט כחברה

1.01 רקע

ארגון הקוד כחברה (package) הוא עיקרונו יסוד בפיתוח תוכנה מקצועית. חברה מאורגנת נכון מאפשרת:

- שימוש חוזר בקוד במספר פרויקטים
- ניהול תלויות (dependencies) בצורה ברורה
- הפצה והתקנה פשוטה
- בדיקות (testing) מובנות

2.01 רשימת בדיקה:構造公司

עברו על הפריטים הבאים ובדקו את הפרויקט שלהם:

1. קובץ `pyproject.toml` או `setup.py`:

- האם קיים קובץ הגדרת חברותה (`pyproject.toml` או `setup.py`)?
- האם הקובץ מכיל את כל המידע הנדרש (שם, גרסה, תלויות)?
- האם התלויות מפורטות בצורה מלאה עם מספרי גרסאות?

2. קובץ `__init__.py`:

- האם קיים קובץ `__init__.py` בתיקייה הראשית של החברה?
- האם הקובץ מייצא (export) את הממשקים הציבוריים של החברה?
- האם קבועים כגון `__version__` מוגדרים בקובץ זה?

3. מבנה תיקיות מאורגן:

- האם קוד המקור נמצא בתיקייה ייעודית (למשל `/src`) או בשם החברה?
- האם הבדיקות (tests) נמצאות בתיקייה נפרדת?
- האם התיעוד (docs) נמצא בתיקייה נפרדת?

4. נתיבים יחסיים:

□ האם כל הייבואים (imports) בקוד משתמשים בנתיבים ייחסיים או בשמות חבילות?

□ האם הקוד מנע שימוש בנתיבים מוחלטים (absolute paths)?

□ האם קריאה וכתיבה של קבועים נעשית ביחס לנשורת החבילה ולא למיקום קבוע ההרצה?

5. מציין מקום לקוד גיבוב (hash code placeholder):

□ האם קיימים מקומות בקוד המיועדים לחישוב גיבוב (hash)?

□ האם המיקומים מסווגים בצורה ברורה (הערות או פונקציות ייעודיות)?

□ האם ישום הניבוב עקבי לאורץ כל הפרויקט?

3.01 דוגמאות למבנה נכון

מבנה תיקיות מומלץ:

```
my_project/
  src/
    my_package/
      __init__.py
      core.py
      utils.py
  tests/
    __init__.py
    test_core.py
  docs/
  setup.py
  README.md
  requirements.txt
```

4.01 הערות נוספת

רשמו כאן הערות על ממצאים, בעיות שזוההו, או שיפורים נדרשים:

11 בדיקה ב: שימוש במעבדים מרובים וחוטי ביצוע מרובים

1.11 רקע

שימוש במעבדים מרובים (multiprocessing) ובחוטי ביצוע מרובים (multithreading) הוא חיוני לביצועים אופטימליים של תוכנה מודרנית. הבנת השימוש הנכון בכלים אלו היא קריטית:

□ מתאים לפעולות תובעניות מבחינת מעבד (CPU-bound): **Multiprocessing**

□ מתאים לפעולות תובעניות מבחינת קלט/פלט (I/O-bound): **Multithreading** □

2.11 רשימת בדיקה: Multiprocessing

1. זיהוי פעולות מתאימות:

□ האם זיהיתם פעולות במערכת שתובעניות מבחינת מעבד?

□ האם פעולות אלו מתאימות למקבול (parallelization)?

□ האם הרכבתם את התועלת הפוטנציאלית מביצוע מקבילי?

2. יישום multiprocessing:

□ האם הקוד משתמש במודול multiprocessing של Python?

□ האם מספר התהליכים (processes) מוגדר באופן דינמי על פי מספר הליבות?

□ האם קיימות טיפול תקין בשיתוף נתונים בין תהליכים?

3. ניהול משאבים:

□ האם התהליכים נסגרים כראוי בסיום העבודה?

□ האם קיימת טיפול בחיריגות (exceptions) בתחום תהליכי מקבילים?

□ האם נמנעים מלילגת זיכרון?(memory leaks)

3.11 רשימת בדיקה: Multithreading

1. זיהוי פעולות מתאימות:

□ האם זיהיתם פעולות במערכת שתובעניות מבחינת קלט/פלט?

□ האם פעולות אלו כוללות המטנה (למשל, קריאות רשות או דיסק)?

□ האם הרכבתם את התועלת הפוטנציאלית מביצוע במקביל?

2. יישום multithreading:

□ האם הקוד משתמש במודול threading של Python?

□ האם חוטי הביצוע (threads) מנוהלים בצורה מסודרת?

□ האם קיימת סyncrho נכון בין חוטים (למשל, locks, semaphores)?

3. בטיחות חוטים (thread safety):

- האם נמנעים ממצבי תחרות (race conditions)?
- האם משתנים משותפים מוגנים באמצעות מנעולים (locks)?
- האם נמנעים מנעילה הדדית (deadlocks)?

4.11 שיקולים נוספים

- האם שקלתם שימוש ב-`asyncio` לפעולות אסינכרוניות במקום `threading`?
- האם בחרתם את הכליל הנכון (תהליכיים לעומת חוטים) לכל משימה?
- האם ביצעתם מדידות ביצועים (benchmarks) לאיומות השיפור?

5.11 הערות נוספות

רשמו כאן הערות על ממצאים, בעיות שזוהו, או שיפורים נדרשים:

21 בדיקה ג: עיצוב מבוסס אבני בניין

1.21 רקע

עיצוב מבוסס אבני בניין (building blocks design) הוא גישה מודולרית לארכיטקטורת תוכנה. כל אבן בנייה היא יחידה עצמאית עם:

- **נתוני קלט (input data):** המידע הנדרש לביצוע הפעולה
- **נתוני פלט (output data):** התוצרים שהאבן מייצרת
- **נתוני הגדרה (setup data):** פרמטרים וקונפיגורציה לאבן הבניה

2.21 עקרונות עיצוב

עיצוב טוב של אבני בניין צריך לעמוד בעקרונות הבאים:

1. **אחריות יחידה (Single Responsibility):** כל אבן בנייה אחראית למשימה אחת מוגדרת
2. **הפרצת דאגות (Separation of Concerns):** כל אבן בנייה עוסקת בהיבט אחד של המערכת
3. **קלות שימוש חוזר (Reusability):** אבני הבניה ניתנות לשימוש חוזר בהקשרים שונים
4. **יכולת בדיקה (Testability):** כל אבן בנייה ניתנת לבדיקה באופן עצמאי

3.21 רשימת בדיקה: זיהוי אבני בניין

1. מיפוי המערכת:

- האם יצרתם תרשימים זרימה של המערכת שלכם?
- האם זיהיתם את כל אבני הבניה העיקריות?
- האם מיפוים את הקשרים והתלוויות בין אבני הבניה?

2. הגדרת אבני בנייה:

- האם כל אבן בנייה מוגדרת כמחלקה (class) או פונקציה נפרדת?
- האם לכל אבן בנייה יש שם תיאורי וברור?
- האם לכל אבן בנייה יש תיעוד (docstring) מפורט?

4.21 רשימת בדיקה: נתוני קלט (Input Data)

בדקו את נתוני הקלט לכל אבן בנייה במערכת שלכם:

1. הגדרה ברורה:

- האם כל נתונים הקלט מתועדים בצורה ברורה?
- האם סוגי הנתונים (data types) מפורטים?

האם התחום התקף (valid range) לכל פרמטר מוגדר?

2. ולידציה (validation):

האם קיימת בדיקת תקינות לכל נתוני הקלט?

האם הבדיקות מטפלות בקלטים שגויים בצורה רואיה?

האם מוחזרות הודעות שגיאה ברורה למשתמש?

3. תלויות:

האם כל התלוויות החיצוניתיות מזוהות?

האם התלוויות מסופקות באמצעות הזרקת תלות (-dependency injection)?

האם אבן הבניה אינה תלואה בקוד ספציפי למערכת?

5.21 רשימת בדיקה: נתוני פלט (Output Data)

בדקו את נתוני הפלט לכל אבן בניה במערכת שלכם:

1. הגדרה ברורה:

האם כל נתוני הפלט מתועדים בצורה ברורה?

האם סוג הנתונים (data types) מפורטים?

האם פורמט הפלט עקבי ומוגדר היטב?

2. עקבות:

האם הפלט תואם את ההגדרה בכל מצב?

האם מטופלים כל מצבי הקצה (edge cases)?

האם הפלט זהה בכל הריצה עם אותם קלטים (בנחתה שאין אקרראיות)?

3. טיפול בשגיאות:

האם פעולות שנכשלו מחזירות הודעת שגיאה ברורה?

האם הפלט במקרה של שגיאה מובן מפלט תקין?

האם השגיאות מדווחות (logged) כראוי?

6.21 רשימת בדיקה: נתוני ההגדרה (Setup Data)

בדקו את נתוני ההגדרה לכל אבן בניה במערכת שלכם:

1. פרמטרים קונפיגורטיביים:

האם כל הפרמטרים הנחוצים לקונפיגורציה זהוו?

האם לכל פרמטר יש ערך ברירת מהדל סביר?

האם הפרמטרים נתונים מקובץ קונפיגורציה או משתני סביבה?

2. הפרדת קונפיגורציה:

- האם קונפיגורציה מופרדת מקוד?
- האם ניתן לשנות קונפיגורציה ללא שינוי קוד?
- האם קיימות קונפיגורציות שונות לנסיבות שונות (פיתוח, בדיקה, ייצור)?

3. אתחול (initialization):

- האם אבן הבניה מאוחלת כראוי לפני שימוש?
- האם קיימות פונקציות setup או initialize?
- האם האתחול מטופל בחירוגות אפשריות?

7.21 דוגמה לאבן בנייה טובה

דוגמאות קונספטואליות:

```
class DataProcessor:
    """
    Building block for processing data

    Input Data:
    - raw_data: List of dictionaries
    - filter_criteria: Dict with filtering rules

    Output Data:
    - processed_data: List of processed dictionaries

    Setup Data:
    - processing_mode: str ('fast' or 'accurate')
    - batch_size: int (default: 100)
    """

    def __init__(self, processing_mode='fast', batch_size=100):
        # Setup/configuration
        self.processing_mode = processing_mode
        self.batch_size = batch_size

    def process(self, raw_data, filter_criteria):
        # Input validation
        # Processing logic
        # Return output
        pass
```

8.21 הערות נוספות

רשמו כאן הערות על ממצאים, בעיות שזוהו, או שיפורים נדרשים:

חלק III

סיכום והמלצות

31 ציון כללי

לאחר שעברתם על כל הבדיקות -- הן האקדמיות והן הטכניות -- חשבו ציון כללי לפרויקט שלכם:

1.31 ציון אקדמי (מהחלק הראשון)

ציון משוקלל אקדמי: _____/001

2.31 ציון טכני (מהחלק השני)

בדקו כמה פריטים מהבדיקות הטכניות עברו:

סך הפריטים הטכניים שעברו: _____

סך כל הפריטים הטכניים: _____

אחוז הצלחה טכני: _____%

3.31 ציון כולל

המלצה לשקלול:

ציון אקדמי: 60%

ציון טכני: 40%

ציון כולל סופי: _____/100

41 תחומיים לשיפור

רשמו את שלושת התחומים העיקריים הדורשים שיפור בפרויקט שלכם:

_____ .1

_____ .2

_____ .3

51 תכנית פעולה

לכל תחום שיפור, הגדרו צעדי פעולה קונקרטיים:

61 לסיום

הערכתה העצמית היא תהליך מתמשך שמשלב הנו רפלקטיבית אקדמית והנו בדיקה טכנית מעמיקה. חזרו על תהליך זה באופן קבוע (למשל, כל חדש או בסוף כל שלב פיתוח) כדי לוודא שהפרויקט שלכם מושך להשתפר ולעמוד בסטנדרטים גבוהים ביותר -- הנו מבחינת התיעוד והמחקר, והן מבחינת איכות הקוד והעיצוב הטכני.

בצלחה! זכרו: הערכת עצמיתenna ומדיקת היא סימן לבגורות אקדמית ומקצועית.