הרצאת טסטים

# רקע

עד המחזור שלנו לא היה אף פעם שיעור של מה זה טסטים ולמה זה מדהים, אלא רק לימדו איך כותבים טסטים בלי להסביר את ההיגיון מאחורי זה ולהמחיש את הכח שזה נותן.

המופע הזה (הרצאה + עע) נועד לתת להם תחושה ללמה טסטים זה כל כך טוב וחשוב, וקצת פילוסופיה של טסטים.

# העברת ההרצאה

מהלך ההרצאה הוא כזה:

* מצגת שמסבירה על למה בדיקת תוכנה, וספציפית בדיקה אוטומטית של תכנה זה חשוב
* עוברים להדגמה חיה

המצגת די אינדיקטיבית ואין צורך לדבר עליה במסמך הזה, אבל היא באמת חלק פחות חשוב מההרצאה.

## Live Coding

ב Live Coding אנחנו כותבים מולם פונקציה בצורה אינקרמנטלית, ומראים דרכים שונות לבדוק אותה תוך כדי הכתיבה עד שנגיע לזה שהדרך הכי טובה לעשות זאת היא טסטים אוטומטיים.

הפונקציה שבחרתי לכתוב היא is\_float שמקבלת string ומחזירה האם הוא מייצג float או לא. זו פונקציה טובה כי במימוש שלה יש הרבה מקרים באופן כללי, והבנייה של הקוד שלה חייבת להיות אינקרמנטלית ולהתייחס לכל דרישה חדשה בנפרד, ואין פתבס של one liner. בלי קשר מאוד קל לכתוב לה טסטים – מרימים עם input מסוים ובודקים את ה output.

באופן כללי פרסור סטרינגים בלי regex זה מתאים כי זה דורש להתמודד עם הרבה מקרים וגם מאוד קל לבדוק את כל המקרים.

בכל מקרה זה מאוד כיף כי הם ממש משתתפים – גם בלהציע איך לשפר את is\_float, גם בלחשוב על טסטים ומקרי קצה, וגם בלחשוב על איך לשפר את צורת העבודה עם הטסטים.

## ה flow של ההרצאה

* אוהד התחיל איתם במימוש של return True ואז שאל אותם אם הקוד עובד ?
  + כשהם אמרו שלא הוא אמר שהם משקרים והם לא יכולים עדיין לדעת – צריך להריץ ולבדוק.
* מתחילים ב"לבדוק" את הקוד – מריצים בלולאת while True סוג של interpreter, שמקבל איזשהו input ומדפיס האם הוא float או לא.
  + הדבר המדהים שקרה אצלנו הוא שאוהד שאל אותם איך אנחנו בודקים את זה, הציג את הלולאה האינטראקטיבית והם זרמו והיו כזה – ככה אנחנו בודקים שהקוד שלנו עובד – משחקים איתו. בהמשך הוא ישפיל אותם על זה.
* בודקים איתם כמה מקרים ורואים שזה נכון עבור כל הfloats, אבל נכשל עבור מספרים שאינם float.
* מפה יש המון שיפורים שניתן לעשות לקוד – זאת באמת פונקציה שאפשר לשפר בצורה אינקרמנטלית עם המון מקרי קצה שניתן להוסיף, יש עוד הרבה דוגמאות לשיפורים שניתן לעשות בקובץ demo\_preperation.py.
* אחרי שמשפרים איתם בקצת את הפונקציה מעלים עוד מקרים ואומרים להם שמשהו מסריח – אנחנו צריכים שוב להקליד הכל, לזכור את כל המקרים ואנחנו לא בודקים מקרים שעברו בעבר.
* מתחילים "לתשתת" את הבדיקה – מוסיפים prints לערכים.
  + עדיין מחורבן, אחרי ששוברים עוד מקרה מראים להם שיש רשימה של True ו False אבל לא ברור לנו מה הם אומרים.
* ממשיכים לתשתת – מוסיפים את הערך המצופה, ואז בודקים שיש לנו הכל True.
  + שוב מוסיפים עוד מקרה שנכשל
  + מה עם הקריאה הרביעים מלמעלה נכשלה ? אנחנו פשוט נראה שם False ולא נבין איזה מקרה נשבר, וה ציפינו לראות שם, ולמה
* מראים להם את assert
  + עכשיו גם יש לנו traceback בדיוק על השורה שנכשלה, וגם לא צריך לעבור בעין ולראות שהכל True
  + עדיין לא ברור לנו מה המשמעות של מה שנכשל, ולמה זה ככה
  + אם יש כמה דברים נראה רק את הראשון
* עוברים ל py.test

## נקודות שמעניין לדבר עליהן

* עד שלא הרצנו את הקוד (באיזשהי צורה) – אנחנו לא יכולים לדעת אם הוא נכון או לא (אולי השתמשנו בטאב במקום רווחים והוא לא רץ ?) – אין דבר כזה להחליט שקוד נכון בלי להריץ אותו
  + דרך טובה להריץ אותו במקום shell אינטראקטיבי שלא משמר שום ידע על איך מריצים את הקוד היא לכתוב טסט – גם מריץ את הקוד, וגם משמר את איך הוא אמור לרוץ.
  + מבחינתם היה מאוד הגיוני פשוט להריץ את הקוד ב live עם כמה inputs שונים ושם לסיים עם זה
* למה לכתוב טסטים למקרים שכבר בדקנו שעובדים ?
  + להראות להם שהרבה פעמים כשנרצה לשנות משהו אחד אנחנו נשבור דברים שכתבנו להם טסט בעבר.
  + יש כמה דברים כאלה שאפשר לעשות ב is\_float, לראות איך זורם עם ההרצאה.
  + להדגיש שאם בפונקציה כל כך קטנה אנחנו יכולים לשנות משהו וזה ישבור מקרה שחשבנו עליו בעבר, אז במערכות גדולות זה מאוד משמעותי, וטסטים נותנים לך "גב" לשנות דברים.
* הקוד לא נכון עד שלא כתבנו לו טסט ובדקנו את זה.
* טסטים קובעים מה מצופה מהממשק להחזיר – לדוגמא לא ברור לי אם מספר בלי נקודה הוא float או לא, אבל אנחנו צריכים להחליט על אחד מהם. בכל מקרה ברגע שכתבנו טסט כזה, אז ההתנהגות תקבע לפי הטסט.
* אנחנו נכנסנו לחלק מהפיצ'רים של pytest. שווה לחשוב מראש לאיזה פיצ'רים רוצים להיכנס ולאיזה לא, כי זה היה קצת מבולגן ובעיקר בלבל אותם.
  + הראינו להם raises
  + הראינו להם איך pytest מפנק עם asserts – מדפיס עוד מידע ולא רק את ה error – השוואת רשימות זו דוגמא טובה.