גבהים
תכנית הסייבר הלאומית

פייתון

שיעור 7: תכנות נכון, ASSERT

ברק גונן

מהו "תכנות נכון"?

- קוד עובד
- טיפול במקרי קצה
- חלוקה לפונקציות
  - Code reuse
  - דיבוג קל יותר
- שמות משמעותיים למשתנים ופונקציות
- תיעוד
- עמידה בכללים מוסכמים- קונבנציות
- קוד בדוק

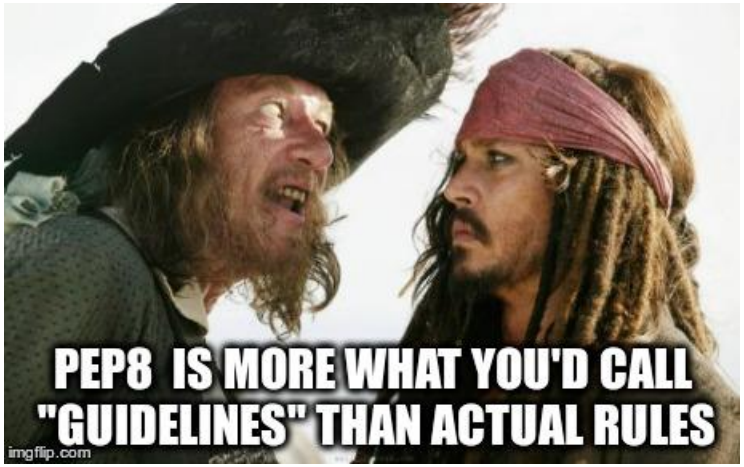
גבהים
תכנית הסייבר הלאומית

מה נלמד?

- כללי PEP8 לכתיבת קוד נכון
- חלוקת קוד לפונקציות
- בדיקת קוד ע"י assert

קונבנציית PEP8

- מתכנתים מנוסים פיתחו כללים לתכנות נכון
  - מסייעים במניעת באגים
  - מקלים על קריאת קוד של אחרים
- מעתה נשים דגש על שימוש בכללים בכל קוד שנכתוב
- תמצית הכללים:
- www.cyber.org.il/networks/PEP8.pdf
- http://pep8online.com



PyCharm- PEP8

- תוכנת PyCharm מסייעת לנו למצוא דברים שלא עומדים בכללים:

```
1  def main():
2      x = [1, 2, 3]
3      if x[0] == 1:
4          print 'Yes'
5      if (x[1] == 2):
6          print 'Yes'
7
```

PEP 8: missing whitespace around operator

PEP8 - דגשים

- תיעוד בתחילת כל פונקציה (docstring)

```
def beep(boop):
    """ Beep the boop """
```

- שימוש בקבועים

```
if boop > MAX_VALUE:
    return
```

- לא לדרוס שמות מובנים בפייתון (len, min...)

חלוקת קוד לפונקציות

- לפני שמתחילים לתכנת- מתכננים את חלוקת הקוד לפונקציות
  - מהן הפונקציות הנדרשות?
  - מה מבצעת כל פונקציה?
    - מה מקבלת כארגומנטים?
    - מה מחזירה?
- דוגמה בספר הלימוד- "חלוקת קוד לפונקציות"

תרגיל כיתה- דז'ה-וו

- חיזרו על ההוראות של תרגיל ז'אן ולז'אן, אלא שהפעם:
  - אי אפשר להניח שהמשתמש הכניס קלט תקין
  - אם הקלט אינו תקין- יש לבקש מהמשתמש להכניס שוב קלט- עד שהקלט יהיה חוקי. לדוגמה:

```
>>> analyze_number_digits()
Please insert a 5-digits number:
hello
Please insert a 5-digits number:
24601
You entered the number: 24601
The digits of this number are: 2, 4, 6, 0, 1
The sum of the digits is: 13
```

- קרדיט: עומר רוזנבוים, שי סדובסקי

assert

```
def power2(x):
    return x*x


def main():
    x = 3
    assert power2(x) == 9, 'x^2 = 9'
    assert power2(x) == 10, 'x^2 != 10'
```

```
Traceback (most recent call last):
  File "C:/Cyber/assert.py", line 14, in <module>
    main()
  File "C:/Cyber/assert.py", line 11, in main
    assert power2(x) == 10, 'x^2 != 10'
AssertionError: x^2 != 10
```

- הפקודה assert מאפשרת לגלות באגים בקוד
- למה זה טוב? לדוגמה:
  - בדיקה של מקרי קצה של פונקציה
  - בדיקת פעולה עם קלט לא חוקי
  - מישהו משנה קוד של פונקציה, אנחנו רוצים לוודא שהפונקציה עדיין עושה מה שתוכנן
- אופן פעולת ה-assert:
  - מקבלת ביטוי, אם הוא True לא קורה דבר
  - אם הוא False מודפסת הודעת שגיאה שיצרנו

assert –המחשה של מקרה שימושי

- נתונה הפונקציה my_div, שמבצעת חלוקה בין שני מספרים. נניח שהפונקציה נכתבה באופן (הלא מוצלח) הבא:

```
def my_div(num1, num2):
    """ Return the division of num1 by num2 """
    return float(num1) / float(num2)
```

- אילו בדיקות נוכל לבצע על ידי assert?
  - בדיקת תקינות לקלט "רגיל" – אם הבדיקה נכשלת בפונקציה יש באג
  - בדיקת תקינות במקרי קצה – טיפול באפס, מחרוזת ריקה וכו'
  - בדיקת פעולה במקרה של קלט לא תקין

```
assert my_div(6, 4) == 1.5, 'Division error'
assert my_div(6, 0)
assert my_div('hi', 2)
```

הדרכה: שימוש נכון ב-assert

- ראשית, פרקו את הקוד שלכם לפונקציות שמבצעות דברים מוגדרים. לדוגמה, בדיקה שהקלט תקין. פירוק המספר לספרות והדפסתן. הדפסת סכום הספרות וכו'.
- שנית, בתוכנית הראשית בצעו assert לפונקציות השונות כדי לבדוק שהן תקינות.
- את ה-assertים מבצעים עם ערכים קבועים מראש, כלומר אין לחכות לקלט מהמשתמש לפני ביצוע ה-assert אחרת אי אפשר לדעת מה מחזירה הפונקציה...)
- כמובן שאין לבצע assert מתוך הפונקציה אותה אתם רוצים לבדוק ☺

תרגיל - Assert

- נתונה הפונקציה abanibi, המתרגמת כל משפט לשפת הבית. העתיקו אותה וכיתבו לה assertים.
- נסו לחפש קלטים בהם הפונקציה אינה עובדת ותקנו את הפונקציה

```
def abanibi(message):
    special = 'aeiou'
    result = ''
    for letter in message:
        if letter in special:
            result += letter + 'b' + letter
        else:
            result += letter
    return result
```

תרגיל- דז'ה-וו

- הוסיפו assertים לפתרון של דז'ה-וו