Intro2CS - Coding Style

הקדמה

בתכנות חשוב לשים לב לסגנון הכתיבה ולא רק לריצה התקינה של הקוד. חשוב לכתוב תוכניות באופן שיאפשר לאחרים לקרוא, להבין ולתחזק את הקוד.

בפרויקטים גדולים סגנון כתיבה נאות חיוני על מנת להקל את התיאום בין המתכנתים בקבוצה המפתחת ואת התחזוקה של הקוד אחרי פיתוחו. גם עבור מתכנת בודד חשוב לכתוב בסגנון נאות על מנת להקל על המשך הפיתוח או התחזוקה של קוד ישן.

מסמך זה מפרט מספר הנחיות להרגלי תכנות טובים. הכללים המפורטים כאן יעזרו לכם במהלך הקורס וירגילו אתכם לשמירה על סגנון כתיבה אחיד שנדרש ממתכנתים במקומות עבודה אופייניים.

שימו לב כי ההנחיות כאן מחייבות! חריגה מהן עלולה לגרור הורדה בציון. ההנחיות מתבססות על גישה סטנדרטית לכתיבה בפייתון. בכל התלבטות בענייני סגנון כתיבה אפשר לבדוק מה מקובל כאן: PEP 8 .

ההנחיות הינן הנחיות כתיבת קוד למתכנתים, ייתכן כי בשלב התחלתי, חלק מההנחיות אינו מובן עד הסוף בשל מחסור בידע והכרת המושגים.

אלו שמות

- **כללי**
  - השתמשו בסגנון אחיד לשמות במהלך כל התוכניות שלכם.
  - שמות צריכים להיות בעלי משמעות, שמות בסגנון i45 או stam_4 לא יתקבלו.
  - אנו מעדיפים שמות בעלי משמעות בשפה האנגלית (אל תחששו להשתמש במילון).
  - שמות קצרים טובים יותר מארוכים: במקום sudden_surge_of_water_with_large_amplitude עדיף tsunami
  - אל תשתמשו בראשי תיבות אלא אם כן אלו קיצורים מוכרים במדעי המחשב או בשפה האנגלית (TCP, PC, temp).
  - אין לדרוס שמות שמורים של פייתון. כלומר אין להגדיר פונקציה או משתנה בעל שם זהה לשם של פונקציה או משתנה אחרים הקיימים כבר בשפת פייתון לדוגמא: list, print, enumerate.

- **קבועים ומשתנים**
  - משתנים יקראו באמצעות אותיות קטנות (lowercase) עם מקף תחתון להפרדה בין מילים במידת הצורך למשל: center, vector_length, upper_limit.
  - עבור קבועים השתמשו באותיות גדולות (capital letters) ובמקף תחתון להפרדה: MAX_LENGTH, PI.

- **פונקציות**
  - אותם הכללים כמו משתנים:

  (max(), to_lower_case() read_file()) .

- **מחלקות (Classes)**
  - מאוחר יותר בקורס נלמד תכנות מונחה עצמים בו תשתמשו במחלקות.
  - שמות מחלקות מתחילים באות גדולה ואחריה אותיות קטנות, למשל: Bucket, Complex.
  - אם שם המחלקה מורכב מיותר ממילה אחת, כל מילה תחל באות גדולה ללא רווחים: PaperClip, BarredWindow.
  - עבור חריגות (exceptions) השתמשו בכללים זהים למחלקות.
  - מאפיינים לא פומביים (non-public attributes) של מחלקות יחלו בשני מקפים תחתונים: __private_counter, __volume

- **חבילות (Packages) ומודולים (Modules)**
  - מודולים צריכים להיקרא בשמות קצרים באותיות קטנות (lowercase), כדוגמת: random, sys, event. ניתן להשתמש במקף תחתון באמצע שם כדי לשפר את הקריאות, למשל: random_generator.
  - חבילות ייקראו על פי אותם כללים.

הזחה / אינדנטציה

בפייתון אינדנטציה הינה חלק אינטגרלי מהשפה. כך מקבצים פקודות לבלוקים ויוצרים קינון שלהם, למשל במשפטי תנאי ולולאות. החלוקה לבלוקים משפיעה על הפירוש של הקוד ועל תחום ההגדרה של משתנים (scope). קוד עם אינדנטציה לא נכונה לא ירוץ או ירוץ באופן שגוי, כי האינטרפטר לא יקרא אותו נכון.

בקורס (ובד"כ בפייתון) אנו דורשים כי עבור כל רמת אינדנטציה תשתמשו בדיוק בארבעה רווחים. שימו לב: רווח (המקש space) וטאב (Tab) הינם תווים שונים בתכלית.

- במקרה של שורה ארוכה ניתן לפצל אותה, תוך כדי שמירה על הזחה שתבהיר כי השורה הבאה הינה חלק מהשורה המקורית, ולא התחלה של בלוק מקונן:

```
Yes:
# Aligned with opening delimiter
foo = long_function_name(var_one, var_two,
                         var_three, var_four)
NO:
# Further indentation required as indentation is not distinguishable
foo = long_function_name(var_one, var_two,
    var_three, var_four)
    print(var_one)
```

- אורך שורה אינו צריך להיות גדול מ-79 תווים. שורה ארוכה מכך יכולה שלא להכנס לבודק לדף הבדיקה (ולגרור הורדת ציון ניכרת). אין לכתוב שורות ארוכות ממספר האמור של תווים אלא בנסיבות מיוחדות.
- ניתן להשתמש בשורות ריקות כדי להפריד חזותית בין חלקים נבדלים של הקוד. למשל, אפשר לשים שורת רווח לפני הגדרה של פונקציה חדשה, בסיום לולאה וכו'.

פונקציות

- שימוש חוזר בקוד (code repetition) הוא טעות תכנותית. קטע קוד המתאר פעולה מוגדרת החוזרת בתוכנית (או ייתכן שתחזור בשינוי עתידי של הקוד) צריך להיות תחום כפונקציה.
- פונקציות צריכות להיות קצרות. פונקציות הדורשות 35-40 שורות עושות קרוב לוודאי פעולות שנכון היה לחלק אותן ליותר מפונקציה אחת. האורך הרצוי של פונקציה הוא עד 25 שורות קוד.
- כל פונקציה צריכה לשמש לתפקיד אחד מוגדר. אם אתם לא יכולים לתאר מה הפונקציה עושה במשפט קצר אחד, קרוב לוודאי שאתם צריכים לחשוב מחדש על החלוקה לפונקציות.

- פונקציות קצרות הן לגיטימיות וחשובות. פונקציה בעלת שתי שורות קוד היא סבירה לגמרי אם היא ממלאת תפקיד ברור וגורמת לקוד להיות קריא יותר.
- חזרה על קוד - על כל פונקציונאליות להיות ממומשת במקום יחיד. פיסת קוד לוגית אשר נכתבת מספר פעמים אינה תכנות נכון וחלוקה לפונקציות אשר ממלאות אותם תפקידים בכל קריאה הוא פתרון אפשרי ויעיל.

הערות בגוף התוכנית (Implementation Comments)

אחד הכלים החשובים שתורמים לקריאות של קוד הוא הוספת הערות בקוד שמבהירות את התפקיד של מרכיבים שלו (כמו משתנים), מתארות מה הוא אמור לעשות ואיך הוא אמור לעשות זאת (מה האלגוריתם). הערות אלו לא נקראות על ידי האינטרפטר ועל כן יכולות (וצריכות) להיכתב בשפה חופשית (באנגלית). ההערות חשובות למתכנת שמפתח את הקוד ועלול להידרש לשנות או לתחזק אותו במועד מאוחר יותר. הן אפילו יותר חשובות אם צפוי שיותר ממתכנת אחד יטפל בקוד במהלך חיי התוכנה. לכן יש להקפיד לכתוב הערות פשוטות וברורות שכל מתכנת יכול להבין בקלות.

ישנן שלוש דרכים להוסיף הערות בפייתון:

- הערות בסופן של שורות קוד: כל טקסט המגיע לאחר סולמית (#) ועד לסופה של אותה שורה הוא הערה. הערות מסוג זה משמשות לרוב לתיאור תפקידו של משתנה או לצורך הסבר קצר של פקודה חריגה, למשל:

```
x = 0 # x will represent the coordinates along the main axis.
```

- הערות בלוק. הערות כאלו מתפרסות לעיתים על מספר שורות. הערות בלוק משמשות לרוב כדי להסביר את קטע הקוד שמופיע אחריהן. לעיתים מכניסים קוד שרוצים להסיר באופן זמני לצורך ניפוי שגיאות או בדיקות אחרות לתוך הערת בלוק. הערת בלוק מסומנת על ידי סולמית ורווח בתחילתה של כל שורת הערה בבלוק:

```
# The code below calculates ....
# .....
```

- דוקסטרינגס (Documentation Strings). דוקסטרינגס הם קטעי הערה שמטרתם לתאר את תפקידה של יחידת קוד רעיונית כלשהי לדוגמה מחלקה או פונקציה באופן שיאפשר להפיק תיעוד מסודר עבור מי שצריך להשתמש ביחידת הקוד הזו. אנחנו נלמד את השימוש בדוקסטרינגס במהלך הקורס.
    - דוקסטרינגס תחומים בשלושה גרשים (""") בתחילת ההערה ובסופה.
    - יש להשתמש בהם עבור כל מודול, פונקציה או מחלקה פומביים. אין הכרח להשתמש בדוקסטרינגס עבור פונקציות שאינן פומביות, אבל עליכם להשתמש בהערות שורה או בלוק כדי לתאר מה עושה הפונקציה.

- עוד על דוקסטרינגס: PEP 257 .

בעת כתיבת ההערות הקפידו על הכללים הבאים. הניחו כי לקורא הקוד יש ידע מוגבל עד לא קיים על מה עושה הקוד. על כן, עליכם לתעד את הממשק של כל יחידת קוד שיש בה שימוש חיצוני (למשל - פונקציה). כמו כן עליכם לתאר את המימוש של כל יחידה - מה התפקיד של מרכיבים חשובים כמו אובייקטים ופונקציות ואיך הם מבצעים את תפקידם. השתמשו בתיעוד פשוט וקריא. אל תכתבו הערות מיותרות שמסבירות מה שכל מתכנת בפייתון אמור לדעת, אלא רק הערות שמבהירות דברים ספציפיים לתוכנית שכתבתם. למשל, אין צורך להסביר שאינדקס של לולאה מקודם ב-1 בכל איטרציה, אבל כדאי להסביר מה הלולאה אמורה לעשות.

רווחים

שימוש ברווחים מאפשר גם הוא קריאות גבוהה יותר של הקוד. הקפידו על הכללים הבאים:

- שימו רווח לאחר פסיק(,) נקודה פסיק (;) ונקודתיים (:).
- הקיפו את האופרטורים הבאים ברווח משני הצדדים:

  השמה (=), השמה ופעולה (=- ,=+ וכו'), השוואות

  (==, <, >, !=, <>, <=, >=, in, not in, is, is not), אופרטורים בוליאניים (and, or, not).
- אם אתם עושים שימוש באופרטורים בעלי עדיפות שונה, הוסיפו רווח מסביב לאופרטור בעל העדיפות הנמוכה יותר. כך תמחישו למעיין בקוד את סדר ביצוע הפעולות.

  דוגמאות:

```
i = i + 1

submitted += 1

x = x*2 - 1

hypot2 = x*x + y*y

c = (a+b) * (a-b)
```

- אין להשתמש ברווחים מסביב לסימן השווה (=) בעת הגדרת ערך ברירת מחדל עבור משתנה בפונקציה.

  דוגמא:

```
def complex(real, imag=0.0):
```
