ברוכים הבאים לשפת פייתון

בדף זה, נעבוד על פיתוח היכולות שלכם בתכנות בשפת פייתון דרך תרגול עצמאי ובניית פרויקט.

הקובץ הזה נוצר במיוחד בשבילכם לאחר שפנו אליי מספר סטודנטים וטענו שחסר להם את הבסיס בפייתון.
מובטח לכם שמי שיסיים את הקובץ הזה בהצלחה, ידע את כל הבסיס של שפת פייתון בצורה טובה.

איך זה עובד?

1. **תרגול עצמאי לפי פרקים:**
   - בכל פרק, תמצאו סט של שאלות לתרגול בנושאים השונים.
   - נסו לפתור את כל השאלות המופיעות פה על כל פרק.
   - אם מצאתם שהשאלות היו קלות עבורכם, בצעו רק את המשימה המתגלגלת של אותו פרק ואז המשיכו לשאלות של הפרק הבא.
   - אם השאלות היו קצת מסובכות, זה סימן שכדאי לחזור וללמוד שוב את הפרק המתאים בקורס בקמפוסIL.
2. **פרויקט סיום:**
   - בסוף הקורס, תידרשו לבנות פרויקט סיום: משחק "איש תלוי" (Hangman).
   - בפרויקט זה, תיישמו את כל הידע שצברתם במהלך הקורס ותציגו את יכולות התכנות שלכם בפייתון.

הקורס בנוי מכמה פרקים מרכזיים:

1. מבוא לפייתון: הבנה בסיסית של התחביר והמבנה של השפה.
2. מחרוזות: עבודה עם מחרוזות, מניפולציות ושיטות נפוצות.
3. תנאים: שימוש בתנאים if, elif ו-else.
4. פונקציות: הגדרת פונקציות, משתנים לוקאליים וגלובליים, ערכים דיפולטיביים.
5. רשימות: עבודה עם רשימות, מתודות של רשימות ולולאות על רשימות.
6. לולאות: שימוש בלולאות for ו-while, כולל לולאות מקוננות.
7. טיפוסים מיוחדים ומבני נתונים: עבודה עם tuples ומילונים.
8. קבצים: פתיחת, קריאת וכתיבת קבצים, ניתוח נתונים מקבצים.

קישור לקורס: https://campus.gov.il/course/cs-gov-pythonselfpy101-he/

פרק 3 - מחרוזות:

1. כתוב תוכנית Python שמגדירה שתי מחרוזות: א. מחרוזת עם השם שלך. ב. מחרוזת עם העיר שבה אתה גר.
   הדפס את שתי המחרוזות אחת לאחר השנייה באותה שורה, עם רווח ביניהן.
2. כתוב תוכנית שמשרשרת שלוש מחרוזות שמקבלת מהמשתמש (שימוש בפונקציה `input`) והדפס את המחרוזת המשורשרת. על המחרוזת המשורשרת להכיל פסיקים בין המחרוזות.
   למשל, אם המחרוזות הן "apple", "banana", "cherry", יש להדפיס "apple,banana,cherry".
3. כתוב תוכנית שמבצעת את המניפולציות הבאות על מחרוזת שהמשתמש מזין:

a. המר את כל האותיות לאותיות קטנות.
b. המר את כל האותיות לאותיות גדולות.
c. הפוך את סדר התווים במחרוזת

הדפס את כל התוצאות.

4. כתוב תוכנית שבודקת האם האות הראשונה והאחרונה במחרוזת שהמשתמש מזין הן זהות. אם כן, הדפס "Yes", אחרת הדפס "No".
5. כתוב תוכנית שמחזירה תת-מחרוזת של מחרוזת הקלט, החל מהתו השלישי ועד התו השישי (לא כולל). לדוגמא, אם המחרוזת היא "Hello, World", תת-המחרוזת תהיה "llo,".
6. כתוב תוכנית שמקבלת מחרוזת מהמשתמש ומחשבת:
   a. כמה מילים יש במחרוזת (הנח שמילה מוגדרת ככל רצף של תווים המופרדים על ידי רווחים).
   b. כמה פעמים מופיע התו ',' (פסיק) במחרוזת.
   c. כמה פעמים מופיע התו '.' (נקודה) במחרוזת.

   הדפס את התוצאות בפורמט מסודר.
7. כתוב תוכנית שמבצעת את הפעולות הבאות:
   a. מקבלת מחרוזת ארוכה ומחרוזת קצרה מהמשתמש.
   b. מוצאת את כל המופעים של המחרוזת הקצרה בתוך המחרוזת הארוכה ומדפיסה את מיקומיהם.
   c. מחליפה כל מופע של המחרוזת הקצרה בתוך המחרוזת הארוכה במחרוזת הפוכה שלה (לדוגמא, אם המחרוזת הקצרה היא "abc", תוחלף ל-"cba").
   d. הדפס את המחרוזת החדשה לאחר ההחלפות

<u>פרק 4 - תנאים:</u>

1. כתוב תוכנית שמקבלת מחרוזת ומספר מהמשתמש. אם המספר גדול מאורך המחרוזת, הדפס את המחרוזת באותיות גדולות. אם המספר קטן או שווה לאורך המחרוזת, הדפס תת-מחרוזת מהתחלה עד התו המתאים למספר (לא כולל).
2. כתוב תוכנית שמקבלת מחרוזת. אם המחרוזת מכילה את המילה "hello", הדפס "Hello to you too!" אחרת אם המחרוזת מכילה יותר מ-3 פסיקים, הדפס "That's a lot of commas." אחרת, הדפס את מספר התווים במחרוזת.
3. כתוב תוכנית שמקבלת מחרוזת ומדרגת אותה לפי הקריטריונים הבאים:
   a. אם אורך המחרוזת גדול מ-10 תווים וכוללת את התו '@', הדפס "High".
   b. אם אורך המחרוזת בין 5 ל-10 תווים וכוללת את התו '!', הדפס "Medium".
   c. אחרת, הדפס "Low".
4. כתוב תוכנית שבודקת אם מספר שהמשתמש מזין הוא זוגי או אי-זוגי. אם זוגי, בדוק אם המחרוזת שהמשתמש מזין מכילה את האות 'e' והדפס "Even and has an 'e'". אחרת, הדפס "Even but no 'e'". אם המספר אי-זוגי, הדפס את האות האחרונה במחרוזת.
5. כתוב תוכנית שמקבלת שם של פרי ומספר. אם הפרי הוא "תפוח", והמספר גדול מ-5, הדפס "Lots of apples!". אם הפרי הוא "תפוח" אבל המספר קטן או שווה ל-5, הדפס "A few apples." אחרת אם הפרי הוא "בננה", והמספר זוגי, הדפס "Even bananas!", ואם אי-זוגי, הדפס "Odd bananas!" אם הפרי אינו תפוח או בננה, הדפס "Other fruit."

<u>פרק 5 - פונקציות:</u>

1. כתוב פונקציה בשם `print_details` שמקבלת שם וגיל (עם ערך ברירת מחדל של 18 לגיל). הפונקציה תדפיס את השם והגיל במשפט מלא, תוך בדיקת תנאי אם הגיל מעל 18 או לא. אם מעל 18, הדפס "Adult", אחרת הדפס "Minor".
2. כתוב פונקציה בשם `manipulate_string` שמקבלת מחרוזת ומחזירה את המחרוזת בפורמטים שונים: אותיות גדולות, אותיות קטנות והופכית. על הפונקציה להדפיס גם את האורך הכולל של המחרוזת.

3. כתוב פונקציה בשם `calculate_product` שמחזירה את המכפלה של שלושה מספרים. שני המספרים הראשונים יקבלו ערכי ברירת מחדל של 1. הפונקציה תדפיס את המכפלה לפני החזרתה.
4. כתוב פונקציה בשם `check_characters` שמקבלת מחרוזת ובודקת אם מכילה את התווים ',', '.', או '!'. עבור כל אחד מהתווים, הפונקציה תדפיס הודעה מתאימה אם התו נמצא או לא.
5. הגדר משתנה גלובלי בשם `count` שמתחיל מ-0. כתוב פונקציה בשם `update_count` שמעדכנת את המשתנה הגלובלי ב-1 בכל קריאה לפונקציה ומדפיסה את הערך החדש. בנוסף, הפונקציה תחזיר את הערך המעודכן.
6. כתוב פונקציה בשם `text_analysis` שמקבלת מחרוזת כקלט ומחזירה מידע מפורט על המחרוזת. הפונקציה תבצע את הפעולות הבאות:
   a. ספור את מספר המילים.
   b. מצא את המילה הארוכה ביותר.
   c. בדוק אם המחרוזת מכילה פחות מ-5 תווים ספציפיים (נניח, 'a', 'e', 'i', 'o', 'u'). אם כן, הדפס מספר המופעים של כל אחד מהם.
   d. בדוק אם המחרוזת מכילה יותר מ-10 מספרים. אם כן, הדפס את סכום המספרים.
7. הגדר מערך גלובלי של מחרוזות. כתוב פונקציה בשם `update_library` שמקבלת מחרוזת חדשה ובודקת אם המחרוזת כבר קיימת במערך. אם המחרוזת כבר קיימת, הדפס שהמחרוזת קיימת ואין צורך להוסיף אותה. אם היא אינה קיימת, הוסף את המחרוזת למערך והדפס את המערך המעודכן. בנוסף, הפונקציה תחזיר את אורך המערך החדש.

<u>פרק 6 - רשימות:</u>

1. כתוב פונקציה בשם `filter_even_numbers` שמקבלת רשימת מספרים ומחזירה רשימה חדשה המכילה רק את המספרים הזוגיים מהרשימה המקורית. השתמש במתודת `append` להוספת פריטים לרשימה.
2. כתוב פונקציה בשם `merge_and_sort_lists` שמקבלת שתי רשימות של מספרים. הפונקציה תחבר את שתי הרשימות לרשימה אחת ותמיין אותה מהקטן לגדול. החזר את הרשימה הממוינת. השתמש במתודת `extend` לחיבור הרשימות ו`sort` למיון.
3. כתוב פונקציה בשם `reverse_list` שמקבלת רשימה ומחזירה אותה בסדר הפוך. השתמש במתודת `reverse` להפיכת הרשימה. תוכל גם להדגים שימוש בסלייסינג להשגת אותו התוצאה.
4. כתוב פונקציה בשם `remove_element` שמקבלת רשימה ואלמנט, ומוחקת את האלמנט הראשון ששווה לערך שניתן מהרשימה. אם האלמנט אינו נמצא, הדפס הודעת שגיאה. השתמש במתודת `remove`.
5. כתוב פונקציה בשם `modify_list` שמקבלת רשימה ובוצעת את הפעולות הבאות:
   a. הוצא את האלמנט השלישי מהרשימה באמצעות `pop`.
   b. הכנס את האלמנט שהוצאת במקום השני ברשימה באמצעות `insert`.

   הפונקציה תחזיר את הרשימה לאחר השינויים.

<u>פרק 7 - לולאות:</u>

1. כתוב תוכנית שמדפיסה את טבלת הכפל מ-1 עד 10 באמצעות שימוש בלולאה מקוננת. השתמש בלולאת `for` חיצונית לשורות ובלולאת `for` פנימית לעמודות.
2. כתוב פונקציה שמקבלת רשימה של מילים ומדפיסה כמה אותיות יש בכל מילה. השתמש בלולאת `for` לעבור על הרשימה ובלולאת `for` נוספת לספירת האותיות בכל מילה.

3. כתוב פונקציה שמקבלת רשימה של מספרים שלמים חיוביים ומוצאת את המספר הטבעי הקטן ביותר שאינו ברשימה. השתמש בלולאת `while` לבדיקה האם מספר נמצא ברשימה ובלולאת `for` לעבור על המספרים ברשימה.
4. כתוב תוכנית שמדפיסה דיאגרמת משולש עם כוכביות (`*`). מספר השורות של המשולש יקבל כקלט מהמשתמש. השתמש בלולאת `for` חיצונית לשורות ובלולאת `for` פנימית להדפסת הכוכביות בכל שורה.
5. כתוב פונקציה שמקבלת רשימה ומחזירה רשימה חדשה ללא כפילויות. השתמש בלולאת `for` לעבור על כל האלמנטים ברשימה המקורית. תוך כדי עבורך, בדוק באמצעות בדיקת `if` האם האלמנט כבר קיים ברשימה החדשה. אם האלמנט לא קיים ברשימה החדשה, הוסף אותו אליה. כך תוכל להשיג רשימה ללא כפילויות מהרשימה המקורית.

<u>פרק 8 - טיפוסים מתקדמים של מבני נתונים:</u>

1. כתוב פונקציה בשם `analyze_tuple` שמקבלת tuple של מספרים ומחזירה מילון עם הפרטים הבאים: המספר הגבוה ביותר, המספר הנמוך ביותר, וממוצע המספרים. השתמש בלולאה כדי לעבור על ה-tuple ולחשב את הנתונים.
2. כתוב פונקציה שמנהלת רשימת קניות באמצעות מילון. המילון יכיל שמות של מוצרים וכמות רצויה. הפונקציה תאפשר למשתמש להוסיף מוצר, לעדכן כמות למוצר קיים, ולהסיר מוצר מהרשימה. כל שינוי ידפיס את הרשימה המעודכנת.
3. כתוב פונקציה שמקבלת מילון שמקשר בין שמות עובדים לשכר שלהם וממיינת את המילון לפי השכר בסדר יורד. השתמש בפונקציה `sorted` עם הפרמטר `key` כדי לבצע את המיון. הדפס את התוצאה בצורה מובנת.
4. כתוב פונקציה שמקבלת מילון, כאשר כל מפתח הוא שם מדינה והערך הוא tuple של שלושה ערים מרכזיות במדינה זו. הפונקציה תחזיר מילון חדש שבו המפתח הוא שם המדינה והערך הוא מספר האותיות הכולל בשמות כל הערים שב-tuple. השתמש בלולאות לחישוב האורך הכולל של שמות הערים.
5. כתוב תוכנית שבה מילון מפתח לכל פונקציה ספציפית את פעולתה. לדוגמה, יש מילון של {'add': פונקציה_לחיבור, 'subtract': פונקציה_לחיסור}. המשתמש יכול לבחור פעולה מהמילון ולהזין ערכים, והתוכנית תבצע את הפעולה המבוקשת.

<u>פרק 9 -קבצים</u>

1. כתוב תוכנית שמקבלת מהמשתמש שמות של ערים ואת מספר התושבים בכל עיר. התוכנית תשמור את הנתונים בקובץ בפורמט CSV (מופרדים בפסיק). כל שורה בקובץ תכיל את שם העיר ואת מספר התושבים. ודא שהקובץ ייסגר בסיום התהליך.
2. כתוב פונקציה שקוראת קובץ טקסט שמכיל שמות של ערים ומספר תושבים ומציגה את כל הערים שמכילות יותר מ-5 אותיות. השתמש בלולאה לקריאת הקובץ שורה שורה ובתנאי לבדיקת אורך שם העיר.
3. כתוב תוכנית שיצרת מילון של ערים ומספר התושבים בהן, שומרת אותו בקובץ באופן שניתן יהיה לשלוף ולהשתמש בו מאוחר יותר. השתמש בפורמט JSON לשמירת המילון בקובץ. כתוב פונקציה נוספת שקוראת את הנתונים מהקובץ ומדפיסה אותם.
4. כתוב פונקציה שמעדכנת קובץ טקסט קיים. הפונקציה תקבל שם של עיר ומספר תושבים חדש, תחפש את העיר בקובץ, ותעדכן את מספר התושבים. אם העיר לא קיימת בקובץ, היא תוסף לסופו.

5. כתוב תוכנית שקוראת קובץ לוג ומנתחת את הנתונים בו. התוכנית תחזיר את מספר הפעמים שכל סוג שגיאה מופיע. הפורמט בקובץ הוא: "[תאריך ושעה] - [רמת השגיאה] - [הודעת שגיאה]". השתמש במילון לשמירת ספירת כל סוג שגיאה.

הגעת עד לפה? מעולה!

עכשיו נשאר לך רק לסיים את הפרויקט "איש תלוי" ולשלוח לי קישור לגיטאהב שלו.

חלק ב

מזל טוב! אם הגעתם לכאן, זה אומר שסיימתם בהצלחה את חלק א' של הקורס "self.py". זהו הישג מרשים שמעיד על הידע והמיומנות שרכשתם בתכנות בשפת פייתון.

בחלק ב' של הקורס, נעמיק ונרחיב את הידע שלכם בנושאים מתקדמים יותר בתכנות, ונמשיך לשפר את היכולות שלכם דרך תרגול עצמאי ובניית פרויקטים מורכבים יותר.

מה צפוי לכם בחלק ב'?

- **שאלות ותרגולים מתקדמים**: כל פרק יכיל סט של שאלות מתקדמות שמאתגרות את הידע והמיומנות שלכם.
- **פרויקטים מורכבים**: בסוף כל פרק תמצאו משימות מתגלגלות שיעזרו לכם ליישם את מה שלמדתם בפרק. המשימות הללו יובילו אתכם לבניית פרויקט גדול ומורכב בשלבים, כך שבסיום הקורס יהיה לכם פרויקט משמעותי להצגה.

מבנה הקורס:

1. **פרק 1: בשורה אחת (One-Liners)** – נלמד כיצד לכתוב קוד יעיל ומקצועי בשורות קצרות.
2. **פרק 2: תכנות מונחה עצמים (Object Oriented Programming)** – נעמיק ב-OOP, כולל מחלקות, אובייקטים, וירושה.
3. **פרק 3: חריגות (Exceptions)** – נלמד כיצד להתמודד עם חריגות ולטפל בשגיאות בתוכנית.
4. **פרק 4: גנרטורים (Generators)** – נכיר את הגנרטורים, היתרונות שלהם וכיצד להשתמש בהם.
5. **פרק 5: איטרציה מתחת למכסה המנוע ואיטרטורים** – נלמד על האיטרטורים והאיטרציה בפייתון מתחת למכסה המנוע.
6. **פרק 6: מודולים (Modules)** – נלמד כיצד להשתמש במודולים, ליצור ולנהל קוד מודולרי.

קישור לקורס: https://campus.gov.il/course/cs-gov-cs-nextpy102/

מוכנים לאתגר הבא? בואו נתחיל!

<u>חלק 1 - בשורה אחת:</u>

1. כתוב בשורת קוד אחת תוכנית שמכפילה כל מספר ברשימה ב-2 ומחזירה רשימה חדשה עם הערכים המוכפלים. השתמש בפונקציה `map` ובפונקציה למדא.
2. כתוב בשורת קוד אחת תוכנית שמסננת את כל המספרים הזוגיים מתוך רשימה ומחזירה רשימה חדשה עם המספרים הזוגיים בלבד. השתמש בפונקציה `filter` ובפונקציה למדא.
3. כתוב בשורת קוד אחת תוכנית שמחשבת את המכפלה של כל המספרים ברשימה. השתמש בפונקציה `reduce` ובפונקציה למדא.
4. כתוב בשורת קוד אחת תוכנית שמייצרת רשימה של ריבועי המספרים מ-1 עד 10.
5. כתוב בשורת קוד אחת תוכנית שמכפילה כל מספר ברשימה ב-3 ורק אם התוצאה זוגית, מחזירה את המספר ברשימה החדשה. השתמש בשילוב של `map` ו-list comprehension.
6. כתוב בשורת קוד אחת תוכנית שמבצעת את הפעולות הבאות:
    a. מסננת את כל המספרים הזוגיים מתוך רשימה.
    b. מכפילה כל מספר זוגי ב-5.

c. מחשבת את סכום כל המספרים המתקבלים.

השתמש בפונקציות `map`, `filter` ו-`reduce`.

7. כתוב בשורת קוד אחת תוכנית שיוצרת מילון משתי רשימות: רשימת מפתחות ורשימת ערכים. כל מפתח מהרשימה הראשונה יתאים לערך מהרשימה השנייה. השתמש ב-`zip` ו-List Comprehension.

<u>פרק 2 - תכנות מונחה עצמים</u>

1. צור מחלקה בשם `Car` עם התכונות הבאות:
   a. `make` (יצרן)
   b. `model` (דגם)
   c. `year` (שנה)
   d. `odometer` (מד אוץ, עם ערך ברירת מחדל של 0)

הוסף למחלקה פעולה (מתודה) בשם `drive` שמקבלת מספר קילומטרים ומעדכנת את מד האוץ בהתאם.

2. צור מופע של המחלקה `Car` עם התכונות:\
   a. `make` = "Toyota"
   b. `model` = "Corolla"
   c. `year` = 2020

השתמש בפעולת `drive` כדי להוסיף 150 קילומטרים למד המרחק. הדפס את מצב מד המרחק לאחר הנסיעה.

3. הוסף למחלקה `Car` משתנה פרטי בשם `_fuel_level` (רמת דלק) עם ערך ברירת מחדל של 100. הוסף פעולה ציבורית בשם `refuel` שמקבלת כמות דלק ומוסיפה אותה למשתנה `_fuel_level`.
4. צור מחלקה בשם `ElectricCar` שיורשת מהמחלקה `Car`. הוסף למחלקה תכונה נוספת בשם `battery_size` (גודל סוללה) עם ערך ברירת מחדל של 75. צור מופע של `ElectricCar` והדפס את כל התכונות שלו.
5. צור מחלקה נוספת בשם `Truck` שיורשת מהמחלקה `Car`. הוסף למחלקה פעולה בשם `load_cargo` שמדפיסה הודעה על טעינת מטען. צור פונקציה חיצונית בשם `start_trip` שמקבלת רשימה של רכבים ומפעילה את הפעולה `drive` לכל רכב ברשימה. ודא שהפונקציה מתמודדת עם אובייקטים של `Car`, `ElectricCar` ו-`Truck`.
6. צור מחלקה בשם `Person` עם התכונות:
   a. `name` (שם)
   b. `age` (גיל, עם ערך ברירת מחדל של 0) הוסף מתודת איתחול שמאתחלת את התכונות.

צור מופע של `Person` עם שם בלבד והדפס את כל התכונות שלו.

<u>פרק 3 - חריגות:</u>

1. כתוב תוכנית שמקבלת שני מספרים מהמשתמש ומבצעת חלוקה. השתמש במבנה `try-except` לטיפול בשגיאת חלוקה באפס (`ZeroDivisionError`) והדפס הודעה מתאימה במקרה של שגיאה.
2. כתוב תוכנית שמקבלת קובץ לקריאה מהמשתמש. השתמש במבנה `try-except` לטיפול בשגיאות הבאות:
   a. `FileNotFoundError` – קובץ לא נמצא.
   b. `PermissionError` – אין הרשאה לקרוא את הקובץ.
   c. כל שגיאה כללית אחרת.
3. כתוב תוכנית שמבצעת חישוב של ממוצע מספרים ברשימה. השתמש במבנה `try-except-else-finally`:
   a. בקטע ה-`try`, בצע את החישוב.
   b. בקטע ה-`except`, טיפול בשגיאה במקרה של רשימה ריקה (`ZeroDivisionError`).
   c. בקטע ה-`else`, הדפס את התוצאה במקרה ואין שגיאה.
   d. בקטע ה-`finally`, הדפס הודעה שמתארת שהפעולה הסתיימה.
4. כתוב תוכנית שמקבלת שם משתמש וסיסמא מהמשתמש. אם הסיסמא פחותה מ-8 תווים, זרוק חריגה מסוג `ValueError` עם הודעה מתאימה. השתמש במבנה `try-except` לטיפול בחריגה זו והדפס הודעה מתאימה במקרה של שגיאה.
5. צור חריגה מותאמת אישית בשם `InvalidAgeError` שמורשת מהמחלקה `Exception`. כתוב תוכנית שמקבלת גיל מהמשתמש. אם הגיל פחות מ-0 או יותר מ-120, זרוק את החריגה `InvalidAgeError`. השתמש במבנה `try-except` לטיפול בחריגה זו והדפס הודעה מתאימה במקרה של שגיאה.
6. הרחב את המחלקה `Person` שיצרת בפרק הקודם והוסף בדיקה לגיל במתודת האתחול. אם הגיל פחות מ-0 או יותר מ-120, זרוק את החריגה `InvalidAgeError`. צור מופע של המחלקה `Person` ונסה לאתחל אותו עם גיל לא תקין. השתמש במבנה `try-except` לטיפול בחריגה זו והדפס הודעה מתאימה במקרה של שגיאה.

<u>פרק 4 - גנרטורים</u>

1. כתוב פונקציית גנרטור בשם `countdown` שמקבלת מספר `n` ומייצרת מספרים מ-`n` עד 0. השתמש במבנה `yield` כדי להחזיר את המספרים אחד אחרי השני.
2. כתוב ביטוי גנרטור שמייצר את הריבועים של המספרים מ-1 עד 10. הדפס את כל הריבועים באמצעות לולאת `for`.
3. כתוב ביטוי גנרטור שמייצר את המספרים הזוגיים מ-0 עד 20. השתמש בפונקציה `next` כדי להחזיר את חמשת הערכים הראשונים.
4. כתוב פונקציה רגילה בשם `first_n` שמקבלת מספר `n` ומחזירה רשימה של המספרים מ-0 עד `n-1`. לאחר מכן, כתוב פונקציית גנרטור בשם `first_n_gen` שמבצעת את אותה פעולה אך מחזירה גנרטור.
5. השתמש בספריית `itertools` ליצירת גנרטור שמייצר אינסוף מספרים החל מ-1 (תשתמש בפונקציה `itertools.count`). הדפס את עשרת המספרים הראשונים באמצעות לולאת `for`.
6. כתוב שני גנרטורים:
   a. גנרטור בשם `even_numbers` שמייצר את המספרים הזוגיים מ-0 עד 20.
   b. גנרטור בשם `multiply_by_two` שמקבל גנרטור אחר ומכפיל כל מספר ב-2.

השתמש בשני הגנרטורים יחד כדי להדפיס את המספרים הזוגיים מ-0 עד 20 כשהם מוכפלים ב-2.

7. כתוב פונקציית גנרטור בשם `fibonacci` שמייצרת את סדרת פיבונאצ'י עד למספר `n`. השתמש במבנה `yield` כדי להחזיר את המספרים בסדרה.
8. כתוב פונקציית גנרטור בשם `prime_numbers` שמייצרת מספרים ראשוניים. השתמש בפונקציה זו כדי להדפיס את עשרת המספרים הראשוניים הראשונים.
9. כתוב פונקציה רגילה בשם `sum_first_n` שמקבלת מספר `n` ומחזירה את סכום המספרים מ-0 עד `n-1`. לאחר מכן, כתוב פונקציית גנרטור בשם `sum_first_n_gen` שמבצעת את אותה פעולה אך מחזירה גנרטור שמייצר את המספרים אחד אחרי השני ומסכמת אותם בצורה עצלה (lazy).
10. צור פונקציית גנרטור בשם `complex_generator` שמבצעת את הפעולות הבאות:
    a. מייצרת סדרה של מספרים מ-1 עד 10.
    b. מכפילה את כל המספרים בסדרה ב-3 באמצעות `yield from`.
    c. מסננת את המספרים שמתחלקים ב-2 באמצעות `itertools.filterfalse`.
    d. משמיטה את 2 המספרים הראשונים באמצעות `itertools.islice`.

השתמש בפונקציה זו כדי להדפיס את המספרים המתקבלים מהגנרטור.

<u>פרק 5 - איטרציה מתחת למכסה מנוע ואיטרטורים</u>

1. כתוב פונקציה שמקבלת רשימה של מספרים ומשתמשת בלולאת `while` ובפונקציה `iter` כדי לבצע את אותה פעולה שמתבצעת בלולאת `for`. הדפס את כל המספרים ברשימה.
2. כתוב פונקציה בשם `is_iterable` שמקבלת אובייקט ובודקת אם הוא איטרבל (iterable) או לא. השתמש במודול `collections.abc` כדי לבדוק את הסוג של האובייקט.
3. צור מחלקה בשם `Countdown` שמיישמת את הפרוטוקול של איטרטור. המחלקה תקבל מספר התחלתי ותייצר מספרים בירידה עד 0. השתמש במתודות `__iter__` ו-`__next__`. הדפס את כל המספרים באמצעות לולאת `for`.
4. כתוב מחלקה בשם `Range` שמדמה את הפעולה של הפונקציה המובנית `range`. המחלקה תקבל התחלה, סיום וצעד, ותממש את מתודות `__iter__` ו-`__next__`. השתמש בלולאת `for` כדי להדפיס את כל המספרים בתחום שצוינו.
5. כתוב פונקציה בשם `even_numbers` שמקבלת מספר ומחזירה איטרטור המייצר את כל המספרים הזוגיים מ-0 עד המספר שניתן. השתמש בלולאת `for` כדי להדפיס את כל המספרים הזוגיים.
6. צור מחלקה בשם `Fibonacci` שמייצרת את סדרת פיבונאצ'י עד למספר נתון. המחלקה תיישם את הפרוטוקול של איטרטור ותשתמש בלולאת `for` כדי להדפיס את הסדרה.
7. צור מחלקה בשם `PrimeNumbers` שמייצרת מספרים ראשוניים עד למספר נתון. המחלקה תיישם את הפרוטוקול של איטרטור ותשתמש בלולאת `for` כדי להדפיס את כל המספרים הראשוניים.
8. כתוב פונקציית גנרטור בשם `square_numbers` שמייצרת את הריבועים של המספרים מ-1 עד 10. השתמש בלולאת `for` כדי להדפיס את כל הריבועים. לאחר מכן, צור מחלקה בשם `SquareNumbers` שמייצרת את הריבועים של המספרים מ-1 עד 10 ומיישמת את הפרוטוקול של איטרטור. הדפס את כל הריבועים באמצעות לולאת `for`.
9. צור מחלקה בשם `Permutations` שמייצרת את כל התמורות (permutations) של רשימה נתונה. המחלקה תיישם את הפרוטוקול של איטרטור ותשתמש בלולאת `for` כדי להדפיס את כל התמורות.

חלק 6 - מודולים

1. כתוב סקריפט שמדגים שלוש צורות שונות לייבוא מודול `math`. הסקריפט ידגים שימוש בפונקציה `sqrt` מכל אחד מצורות הייבוא השונות.
2. כתוב סקריפט שמייבא את המודול `random` ומשתמש בו כדי להגריל מספר שלם בין 1 ל-100 ולהדפיס אותו. בנוסף, השתמש במודול `datetime` כדי להדפיס את התאריך והשעה הנוכחיים.
3. כתוב סקריפט פייתון בשם `greet.py` שמכיל פונקציה בשם `greet` שמקבלת שם ומדפיסה הודעת שלום. כתוב סקריפט נוסף בשם `main.py` שמייבא את הפונקציה `greet` מהסקריפט `greet.py` ומשתמש בה כדי להדפיס הודעת שלום למשתמש
4. צור מודול מותאם אישית בשם `calculator.py` שמכיל פונקציות לחיבור, חיסור, כפל וחילוק. כתוב סקריפט נוסף בשם `main.py` שמייבא את המודול `calculator` ומשתמש בו לביצוע פעולות חישוביות שונות.
5. כתוב סקריפט בשם `main.py` שמייבא מודול `helper.py` ומבצע בו קריאה לפונקציה `assist`. ודא שהמודול `helper.py` כולל פונקציה בשם `assist` שמדפיסה הודעה כלשהי.
6. צור חבילה בשם `mypackage` עם מודול אחד לפחות בתוכה. כתוב סקריפט שמייבא את המודול מהחבילה ומשתמש בו כדי לבצע פעולה כלשהי.

סיום קורס next.py

ברכות! הגעתם לסוף קורס next.py וזהו הישג מרשים ביותר. במהלך הקורס העמקתם את הידע שלכם בתכנות בשפת פייתון, למדתם נושאים מתקדמים והתמודדתם עם אתגרים מורכבים.

מה למדתם בקורס:

- **שימוש ב-One-Liners**: כתיבה יעילה ומקצועית של קוד.
- **תכנות מונחה עצמים**: יצירת מחלקות, ירושה, פולימורפיזם ועוד.
- **עבודה עם חריגות**: טיפול בשגיאות, יצירת חריגות מותאמות אישית ועוד.
- **גנרטורים**: יצירת ושימוש בגנרטורים לחישוב עצל.
- **איטרטורים**: הבנת הפרוטוקול של איטרטור ויצירת איטרטורים מותאמים אישית.
- **מודולים**: יצירת מודולים מותאמים אישית, ייבוא מודולים ושימוש ב-pip.

מה הלאה?

כעת, כשיש לכם את כל הכלים והידע המתקדם, הגיע הזמן להמשיך וליישם אותם בפרויקטים אמיתיים. כל פרויקט שתבנו ישפר את היכולות שלכם ויקדם אתכם בעולם התכנות.
מי שמעוניין ברעיונות לפרויקט מוזמן לשלוח לי הודעה במייל.