תרגיל 3: מילונים ועיבוד מידע

תאריך פרסום:	26.11.2024
תאריך הגשה:	23:59 11.12.2024
מתרגלת אחראית:	אסראא נסאסרה
משקל תרגיל:	3 נקודות
מטרות העבודה:	תרגול עבודה עם מחרוזות (str) ומילונים לשם עיבוד מידע.

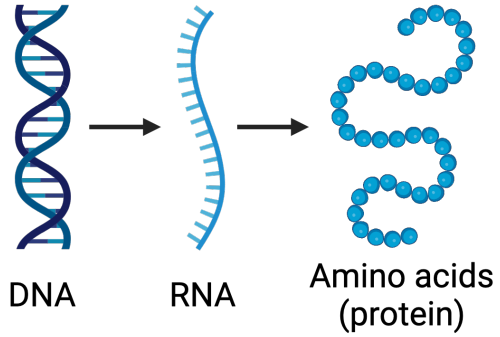
הנחיות לעבודה 3:

- קראו בקפידה את הנחיות ההגשה. אל תגישו קבצים או קוד מיותר. וודאו כי אתם עומדים בכל ההנחיות לפני ההגשה!
- מומלץ לקרוא את כל העבודה בשלמותה לפני תחילת כתיבת קוד הפתרון.
- אין להשתמש בספריות חיצוניות (ניתן להשתמש בספריה math).
- ניתן להוסיף פונקציות עזר. האחריות על נכונות הקוד בפונקציות אלה היא עליכם.
- חובה לכתוב docstring (כפי שנלמד בכיתה) לכל פונקציה.

- **יש להגיש קובץ אחד בשם HW3.py (שלד הקובץ נתון לכם ורצוי למלא את הפתרון במקומות המתאימים).**

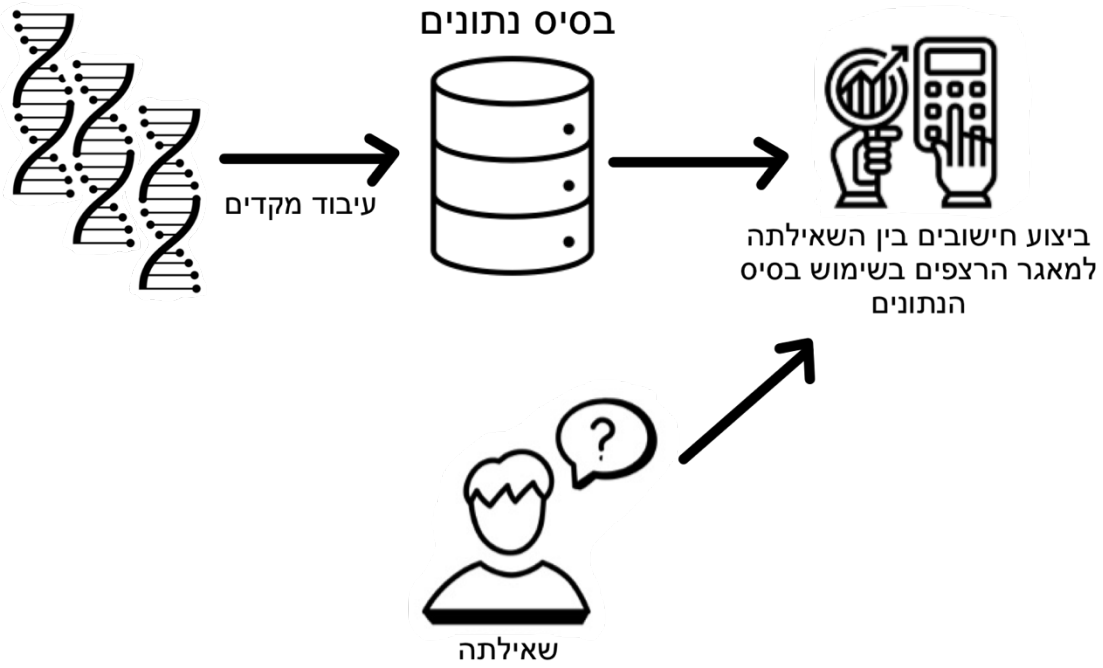
רקע ומטרת העבודה:

דנ"א (DNA), החומר הגנטי שמועבר מדור לדור, הוא רצף המכיל את המידע הדרוש ליצירת חלבונים (Proteins, "המנועים" שגורמים לתאים בגוף שלנו לפעול) ולכן הוא הכרחי לחייהם של כל הייצורים. רצפי הדנ"א הם סוג של מחרוזת ארוכה המורכבת מאלפבית של ארבע אותיות: A,T,G,C. רצף דנ"א עובר תהליך של המרה לרצף רנ"א (RNA). כל רצף של שלוש אותיות ברנ"א מגדיר מילה בעלת משמעות הנקראת "חומצת אמינו" (Amino acid). חומצות אמינו הן היחידה הבסיסית לבניית החלבונים. רנ"א הוא שלב ביניים בין הדנ"א לחלבונים (רצף דנ"א -> רצף רנ"א -> רצף חומצות אמינו\חלבון).



בתרגיל זה תתנסו בעבודה עם מחרוזות ומילונים. בעזרתם, תממשו תוכנית המבצעת עיבוד מקדים של מאגר רצפי דנ"א לבסיס נתונים המאחסן רצפי חומצות אמינו ומנוע חיפוש בו ניתן לחפש חומצת אמינו בבסיס הנתונים באמצעות שאילתות. **בחלק הראשון** של התרגיל, תממשו עיבוד מקדים שנועד להסיר מקטעים, סימנים ותווים לא רלוונטיים ברצף הדנ"א, לשנות את הרצף לפי כללים מסוימים ולאחר מכן להמיר אותו לרצף רנ"א, ולבסוף לרצף של חומצות אמינו *(או חלבון)*. **בחלק השני**, תאגדו את המידע מכל הרצפים השונים במאגר לבסיס נתונים שיאפשר חיפוש מהיר. **בחלק השלישי**, תשתמשו בבסיס הנתונים שיצרתם על מנת לבצע חישובים שונים המערבים שאילתה (מחרוזת) ואת מאגר רצפי חומצות האמינו. **בחלק הרביעי והאחרון**, תאחדו את כל השלבים לתפריט אחד בו המשתמש יוכל לבצע פעולות על מאגר רצפי חומצות האמינו.

להלן תרשים:



<u>כמה הגדרות חשובות לפני שמתחילים:</u>

- רצף דנ"א הוא מחרוזת המורכבת מהאותיות האלפביתיות a-z (capital ו-small), וסימני הפיסוק ":", "-", "_", ו-".". רצף דנ"א לא מכיל ספרות.
- רצף דנ"א **פונקציונלי** הוא רשימת מחרוזות כאשר כל מחרוזת היא שלשה המורכבת מהאותיות A,T,G,C בלבד (Capital-letters, השלשה הראשונה בו היא ATG (שעשויה להופיע שוב ברשימת המחרוזות) ומסתיים ב<u>אחת</u> מהשלשות הבאות: TAA, TAG, TGA, הנקראות stop codons. לא יהיו מופעים נוספים של stop codons מעבר לאחת שבסוף הרשימה בכל דנ"א פונקציונלי.
- על מנת להמיר רצף דנ"א לצורתו הפונקציונלית (proofreading), יש לבצע את הפעולות הבאות: (1) למצוא את המופע הראשון של השלשה ATG בתוך הרצף שמכתיבה את השלשה הראשונה של הדנ"א הפונקציונלי ואת החלוקה של שאר הרצף לשלשות. (2) למצוא את אחת משלשות הסיום stop codons ברצף אשר נמצאת בסדר מתאים למופע הראשון של ATG. (3) להסיר את כל התווים העודפים המופיעים אחריה.

אם לא קיימת שלשה כזאת (stop codon) יש להסיר עודפי תווים שאינם משלימים לשלשה בסוף הרצף ולהוסיף את שלשת הסיום TAA. לדוגמא:

אם נתון רצף הדנ"א הבא:

'AG**ATG**GGGCTT**TGA**AGTAA' -> ['**ATG', '**GGG', 'CTT', '**TGA**']

הצורה הפונקציונלית של הדנ"א מתחילה מהאינדקס 2 בו נמצא המופע הראשון של ATG ומסתיימת עם TGA שהיא stop codon. יש להסיר את התחילית "AG" ואת "AGTAA" מאחר והם מופיעים לאחר TGA.

לעומת זאת ברצף הבא:

'**ATG**TT~~TGA~~' -> ['**ATG', '**TTT', '**TAA**']

הצורה הפונקציונלית של הדנ"א מתחילה מהאינדקס 0 בו נמצא המופע הראשון של ATG ומסתיימת עם TAA. השלשה TAA אינה קיימת ברצף הדנ"א והיא מתווספת לאחר ההמרה. ברצף הדנ"א, ~~TGA~~ (באינדקסים 5 עד 7 כולל) אינה נחשבת לשלשת הסיום לרצף זה וזאת מאחר והיא לא נמצאת בשלשה בהתאם לחלוקת הרצף לשלשות המוכתבת ע"י מופע הראשון של ATG.

- **Transcription- שעתוק**: בתהליך השעתוק, יש להפוך רצף דנ"א פונקציונלי לרצף רנ"א. בדומה לדנ"א פונקציונלי, רנ"א גם מוגדר להיות רשימת מחרוזות כאשר כל מחרוזת היא שלשה, אבל האות T (שלא קיימת ברנ"א) מוחלפת באות U.
- **Translation- תרגום:** רצפי הרנ"א מתורגמים לרצף של חומצות אמינו. התרגום מתבצע כך שכל שלשה ברנ"א, למעט ה-stop codon (UAA, UAG, ו-UGA), מתורגמת לחומצת אמינו, ורצף חומצות האמינו מרכיבות את החלבון השלם.

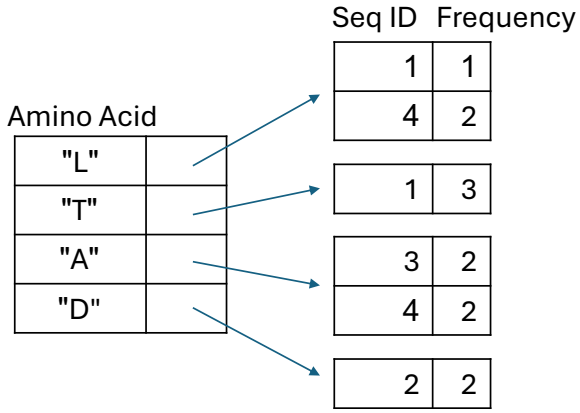
שימו לב, כדי להקל עליכם, נתונה פונקציית עזר בשם codon_translator, המקבלת מחרוזת של שלשת רנ"א ומחזירה את החומצה האמינית אליה מתורגמת השלשה, כאות בודדת, לפי הטבלה הבאה (שימו לב לשימוש במילון במימוש):

UUU, UUC: Phenyl-alanine F UUA, UUG: Leucine L	UCU, UCC, UCA, UCG: Serine S	UAU, UAC: Tyrosine Y UAA: Stop codon UAG: Stop codon	UGU, UGC: Cysteine C UGA: Stop codon UGG: Tryptophan W
CUU, CUC, CUA, CUG: Leucine L	CCU, CCC, CCA, CCG: Proline P	CAU, CAC: Histidine H CAA, CAG: Glutamate Q	CGU, CGC, CGA, CGG: Arginine R
AUU, AUC, AUA: Isoleucine I AUG: Methionine start codon M	ACU, ACC, ACA, ACG: Threonine T	AAU, AAC: Asparagine N AAA, AAG: Lysine K	AGU, AGC: Serine S AGA, AGG: Arginine R
GUU, GUC, GUA, GUG: Valine V	GCU, GCC, GCA, GCG: Alanine A	GAU, GAC: Aspartic acid D GAA, GAG: Glutamic acid E	GGU, GGC, GGA, GGG: Glycine G

למשל השלשה UUU מתורגמת לחומצה אמינו "F" והשלשה CCU מתורגמת לחומצה אמינו "P".

- **sequence_corpus** – אוסף רצפים, קורפוס של רצפים. מיוצג ע"י מילון כאשר המפתחות הינם המספר המזהה של הרצף והערכים הינם רצף הדנ"א כמחרוזת טרם העיבוד המקדים. המשתנה sequence_corpus <u>אינו</u> משתנה גלובלי ויש להעביר אותו כארגומנט בכל קריאה לפונקציה המשתמשת בו. שימו לב שבבדיקת העבודה נשתמש בקורפוס אחר מזה שנתון לכם בקובץ העבודה.

- **Inverted index** – מבנה נתונים בו עבור כל חומצה אמינו המיוצגת ע"י אות בודדת (capital letters) והמופיעה במאגר הרצפים נשמרים כל הרצפים בהם החומצה הופיעה. המימוש הוא באמצעות מילון כאשר המפתחות הינן חומצות אמינו המופיעות במאגר הרצפים (מטיפוס מחרוזת). כל מפתח ממופה למילון כאשר המפתחות שלו הם המספרים המזהים של רצפים המכילים את החומצה, והערכים הינם מספר המופעים של חומצת האמינו ברצף.



על פי האיור, חומצת האמינו "L" מופיעה ברצף #1, פעם אחת, וברצף #4, פעמיים.

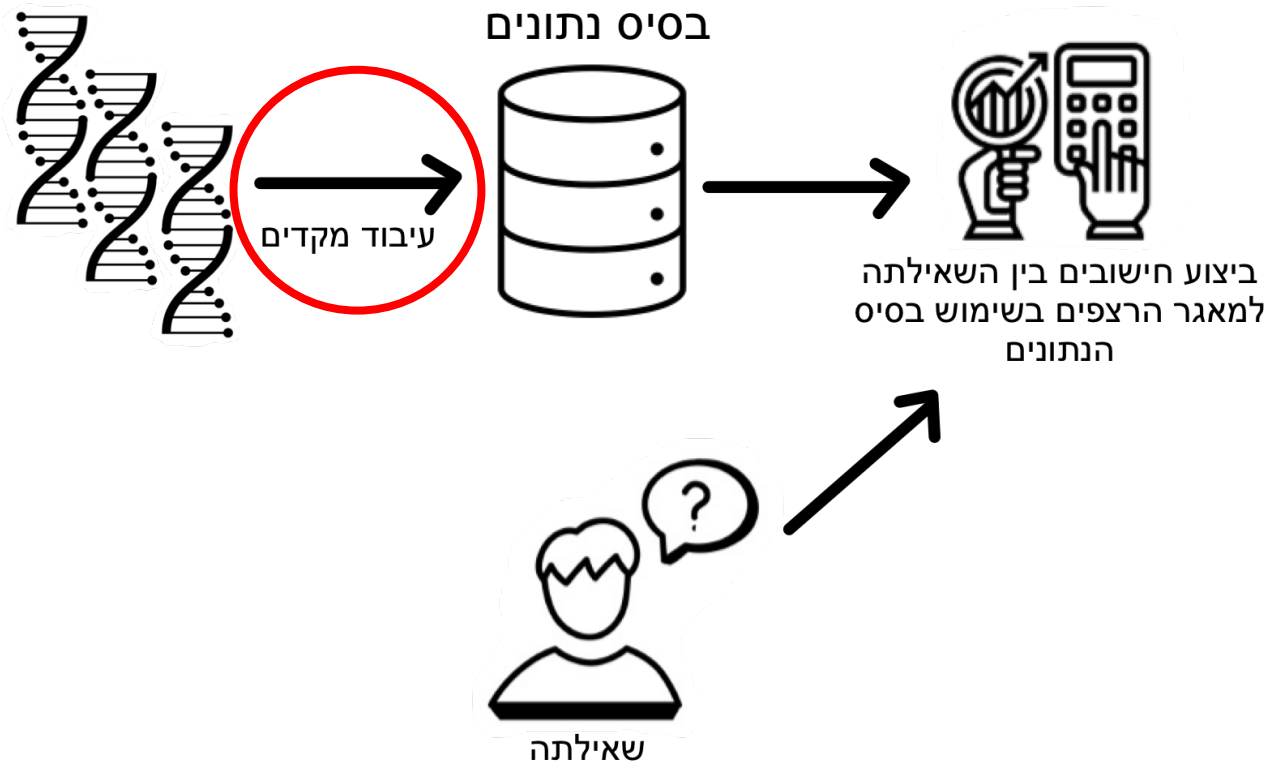
- **AAF-ISF**–קיצור של "Amino Acid Frequency–Inverse Sequence Frequency ", בעברית: תדירות חומצת אמינו (aaf) - תדירות חומצת אמינו הופכית (isf) שמשמעותו ציון מספרי שנועד לשקף את החשיבות של חומצת אמינו מסוימת לרצף מסוים.

  תדירות חומצת אמינו (aaf) – באיזה תדירות מופיעה חומצת האמינו ברצף.

  תדירות חומצת אמינו הופכית (isf) – חשיבות חומצת האמינו בכל מאגר הרצפים.

חלק 1 – עיבוד מקדים

בחלק זה תממשו 7 פונקציות של עיבוד מקדים, כלומר הכנה של רצף דנ"א להמשך עיבוד, ופונקציה נוספת אשר מאחדת את כל שבעת השלבים.



def remove_punctuations(seq)

הפונקציה מקבלת כקלט מחרוזת רצף דנ"א ומחזירה מחרוזת חדשה (str) התואמת לקלט, לאחר הסרת סימני פיסוק. כל סימן פיסוק בקלט יוחלף ברווח בפלט.

דוגמת הרצה:

```
>>> remove_punctuations("A gcrTG-G ")
'A gcrTG G '
```

def remove_invalid_letters(seq)

הפונקציה מקבלת בקלט מחרוזת רצף דנ"א ומחזירה מחרוזת חדשה (str) התואמת לקלט, ללא כל אות השונה מהאלפבית של הדנ"א (בלומר שונה מ-ATGCatgc). כל אות לא חוקית בקלט תוחלף ברווח בפלט.

דוגמת הרצה:

```
>>> remove_invalid_letters("A gcrTG-G ")
'A gc TG-G '
```

def remove_spaces(seq)

הפונקציה מקבלת בקלט מחרוזת רצף דנ"א ומחזירה מחרוזת חדשה (str) התואמת לקלט, ללא רווחים.

דוגמת הרצה:

```
>>> remove_spaces("A gcrTG-G ")
'AgcrTG-G'
```

def capatilize_letters(seq)

הפונקציה מקבלת בקלט מחרוזת רצף דנ"א ומחזירה מחרוזת חדשה (str) התואמת לקלט, כאשר כל האותיות גדולות.

דוגמת הרצה:

```
>>> capatilize_letters("A gcrTG-G ")
'A GCRTG-G '
```

def proofreading(seq)

הפונקציה מקבלת בקלט מחרוזת של רצף דנ"א המכיל אך ורק רצף של אותיות ATCG ומחזירה רשימת מחרוזות, כאשר כל איבר ברשימה הוא שלשה של רצף דנ"א פונקציונלי. על מנת להמיר רצף דנ"א לצורת הפונקציונלי יש לוודא את הכללים הבאים:

- רצף דנ"א פונקציונלי מתחיל בשלשה ATG שמכתיבה את החלוקה לשלשות של שאר הרצף. רצף דנ"א שאין בו ATG לא ניתן להמירו לדנ"א פונקציונלי ולכן יש להחזיר רשימה ריקה.
- דנ"א פונקציונלי מסתיים בשלשת סיום stop codons (אם אין שלשת כזו, יש להוסיף את השלשה TAA).

דוגמת הרצה:

```
>>>  proofreading ('AGATGACGGCAGCATGA')
['ATG', 'ACG', 'GCA', 'GCA', 'TGA']
>>>  proofreading ('AAATGTTTGA')
['ATG', 'TTT' , 'TAA']
```

שימו לב: לאחר פעולת ה-proofreading שבדוגמה השנייה התווספה לרצף הדנ"א הפונקציונלי שלשת סיום שלא הייתה ברצף המקורי.

def transcript(func_seq)

הפונקציה מקבלת בקלט רשימת מחרוזות המייצגת רצף דנ"א פונקציונלי. הפונקציה משעתקת את הדנ"א פונקציונלי לרצף רנ"א ומחזירה רשימה חדשה של מחרוזות כאשר כל מחרוזת היא שלשה של הרנ"א, כלומר לאחר החלפת כל מופע של T באחת השלשות ברשימת קלט בבאות U.

דוגמת הרצה:

```
>>> transcript(['ATG', 'ACG', 'GCA', 'GCA', 'TGA'])

['AUG', 'ACG', 'GCA', 'GCA', 'UGA']
```

def translate(rna_seq)

הפונקציה מקבלת בקלט רשימת מחרוזות המייצגת רצף רנ"א. הפונקציה מחזירה רשימה המייצגת את רצף חומצות האמינו. חומצת אמינו מיוצגת ע"י תו בודד (capital-letter). היעזרו בפונקציית עזר codon_translator אשר מקבלת שלשת רנ"א ומחזירה את התרגום שלה לחומצה אמינית כתו בודד. שלשת הסיום אינה מתורגמת לחומצת אמינו ולכן אינה נמצאת ברצף חומצות האמינו.

```
>>> translate(['AUG', 'ACG', 'GCA', 'GCA', 'UGA'])

['M', 'T', 'A', 'A']
```

def preprocessing(seq)

פונקציה המאחדת את תהליך העיבוד המקדים של רצף דנ"א עד להמרתו לרצף חומצות אמינו. הפונקציה מקבלת בקלט רצף דנ"א (str) ומחזירה את רצף חומצות האימנו המתאים לרצף הדנ"א בקלט ברשימת מחרוזות כאשר כל מחרוזת היא תו המייצג חומצה אמינית.

הפונקציה מבצעת את השלבים הבאים (בסדר זה):

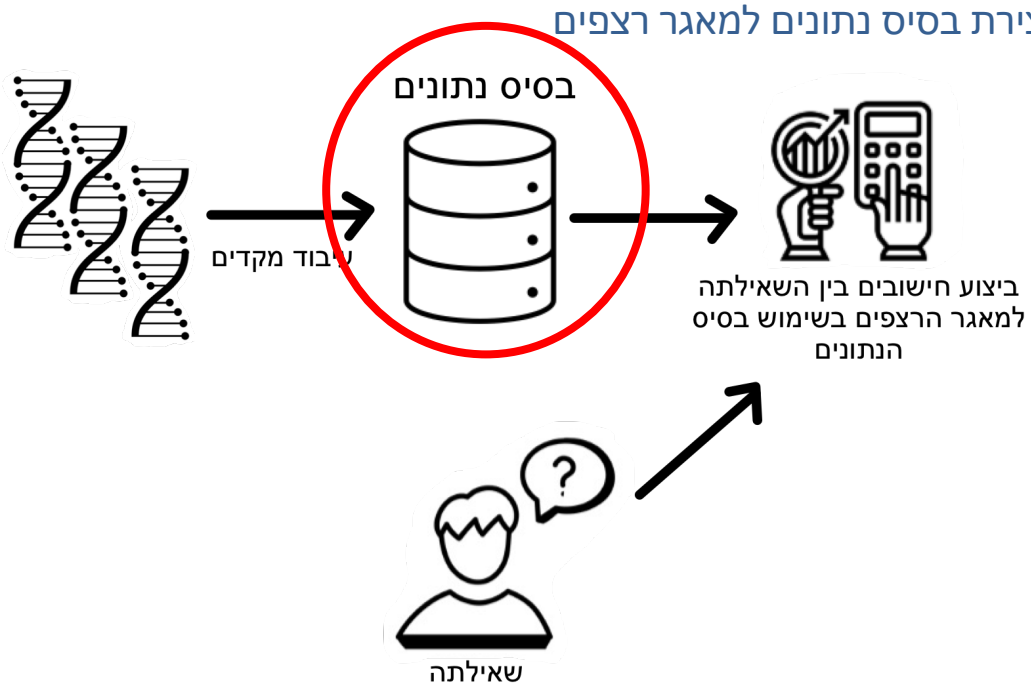
- הסרת סימני פיסוק.
- הסרת אותיות לא חוקיות.
- הסרת רווחים.
- המרת אותיות "קטנות" (small) לגדולות (capital).
- המרת רצף הדנ"א לצורתו הפונקציונלית.
- שעתוק רצף דנ"א פונקציונלי לרנ"א.
- תרגום רצף רנ"א לרצף חומצות אמינו.

דוגמת הרצה:

```
>>>preprocessing('Aga---tGACggcrfAlmn: GCATA')

['M', 'T', 'A', 'A']
```

חלק 2 – יצירת בסיס נתונים למאגר רצפים



בחלק זה תממשו פונקציות היוצרות ממאגר הרצפים של דנ"א בסיס הנתונים ומתחזקות אותו. בסיס הנתונים מאפשר ייצוג דחוס של מאגר הרצפים (על מנת לחסוך בזיכרון) וחיפוש מהיר (ביחס לחיפוש במאגר הרצפים המקורי).

בסיס הנתונים מורכב משני מילונים:

- sequences data – מילון השומר עבור כל רצף את אורכו לאחר עיבוד מקדים (מספר חומצות אמינו ברצף).
- inverted index – מילון השומר עבור כל חומצה אמינית את מספר המופעים שלה בכל רצף (ראו איור בהגדרות בעמוד 4).

דוגמאות הריצה בחלק זה ובחלק הבא של התרגיל יסתמכו על מאגר הרצפים הבא:

seq_corpus_example = {

1: ‘GCTTA_tGC:GXATC CGTAGACffx:TYAGgytACGTMA’

2: ‘AGG. Ddfe::wscv’

3: ‘cl_yuCATGATGCGTACCAGGCTqwAGCATGCGTbbAGCTAxzvGCATGAC’

}

def get_sequences_data(sequence_corpus)

פונקציה היוצרת sequences_data עבור מאגר רצפים טרם העיבוד המקדים. הפונקציה מקבלת כקלט מילון המייצג מאגר רצפים טרם עיבוד (בדומה ל- seq_corpus_example), כאשר המפתחות הינם המספרים המזהים של הרצף (int) והערכים הינם המחרוזות המייצגות את הרצפים התואמים (str). הפונקציה מחזירה sequences_data - מילון אשר המפתחות בו הינם המספר המזהה של כל רצף (int) והערכים הינם מספר חומצות האמינו <u>לאחר</u> תהליך העיבוד המקדים המתואר בחלק 1.

דוגמת הרצה:

```
>>> get_sequences_data(seq_corpus_example)

{ 1: 5, 2: 0, 3: 12}
```

הסבר לרצף מספר 1: לאחר העיבוד המקדים מתקבל כי רצף חומצות האמינו הוא: ['M', 'R', 'S', 'V', 'D']. הרצף מסתיים עם השלשה TAG ולכן לאחר העיבוד המקדים רצף זה מכיל 5 חומצות אמינו.

def create_inverted_index(sequence_corpus)

פונקציה היוצרת inverted index עבור מאגר רצפים טרם העיבוד המקדים. הפונקציה מקבלת כקלט מילון המייצג מאגר רצפים טרם עיבוד, כאשר המפתחות הינם המספרים המזהים של הרצף (int) והערכים הינם המחרוזות המייצגות את הרצפים התואמים (str). הפונקציה מחזירה inverted index של מאגר הרצפים לאחר עיבוד מקדים לרצף. inverted index הינו מילון, כאשר המפתחות בו הן אותיות המייצגות חומצות האמינו המרכיבות את מאגר הרצפים לאחר עיבוד מקדים. הערך עבור כל חומצת אמינו הינו מילון אשר המפתחות בו הינם המספרים המזהים של הרצפים אשר מכילים את חומצת האמינו והערכים הינם מספר המופעים של כל חומצה בכל רצף בהתאמה (ראו איור בעמוד 4).

דוגמת הרצה:

```
>>> create_inverted_index(seq_corpus_example)

{'M': {1: 1, 3: 3}, 'R': {1: 1, 3: 2}, 'S': {1: 1, 3: 1}, 'V': {1: 1, 3: 1}, 'D': {1: 1}, 'T': {3: 1}, 'L': {3: 1}, 'A': {3: 2},

'C': {3: 1}}
```

הסבר על החומצה האמינית 'M' – החומצה האמינית 'M' מופיעה ברצף 1 פעם אחת וברצף 3 שלושה פעמים כאשר היא מקודדת מהשלשה "ATG". ברצף 2, אין את החומצה 'M' (או כל חומצה אחרת), משום שרצף חומצות האמינו שלו לאחר עיבוד הוא רשימה ריקה.

def add_to_data(inverted_index, sequences_data, seq_id, seq)

פונקציה המוסיפה רצף חדש לבסיס הנתונים. הפונקציה מקבלת כקלט *sequences_data*, *inverted_index*, seq_id מספר מזהה של הרצף החדש (int) ו-seq מחרוזת של הרצף החדש (str) ומחזירה *inverted_index* ו-*sequences_data* (בסדר הזה) מעודכנים לפי הרצף החדש.

שימו לב:

- הניחו כי המספר המזהה של הרצף החדש אינו מופיע כבר בבסיס הנתונים.
- הפונקציה אינה משנה את מאגר הרצפים המקורי אלא רק את המילונים *inverted_index* ו-*sequences_data*.

דוגמת הרצה:

```
new_inverted_index,new_sequences_data = add_to_data(inverted_index,
sequences_data,4,

".yyyuTACGATGGTAGCTAGCTAGCGllllTACGATCGTA")

print(f"new_inverted_index: {new_inverted_index}")
print(f"new_sequences_data: {new_sequences_data}")
```

new_inverted_index: {'M': {1: 1, 3: 3, 4: 1}, 'R': {1: 1, 3: 2}, 'S': {1: 1, 3: 1, 4: 1}, 'V': {1: 1, 3: 1, 4: 1}, 'D': {1: 1}, 'T': {3: 1}, 'L': {3: 1}, 'A': {3: 2, 4: 1}, 'C': {3: 1}}

new_sequences_data: {1: 5, 2: 0, 3: 12, 4: 4}

def remove_from_data(inverted_index, sequences_data, seq_id)

פונקציה המוחקת רצף מבסיס הנתונים. הפונקציה מקבלת בקלט *inverted_index*, *sequences_data* ו-seq_id מספר מזהה של הרצף אותו רוצים למחוק (int) ומחזירה *inverted_index* ו- *sequences_data* (בסדר הזה) מעודכנים ללא הרצף שנבחר למחיקה.

שימו לב:

- הניחו כי המספר המזהה של הרצף קיים במאגר הנתונים.
- הפונקציה אינה משנה את מאגר הנתונים המקורי אלא רק את המילונים *inverted_index* ו-*sequences_data*.
- אם לאחר מחיקת הרצף במילון *inverted_index* יש חומצות אמינו שהמילון בערך שלהן נותר ריק, יש למחוק חומצות אמינו אלה מהמילון *inverted_index*.

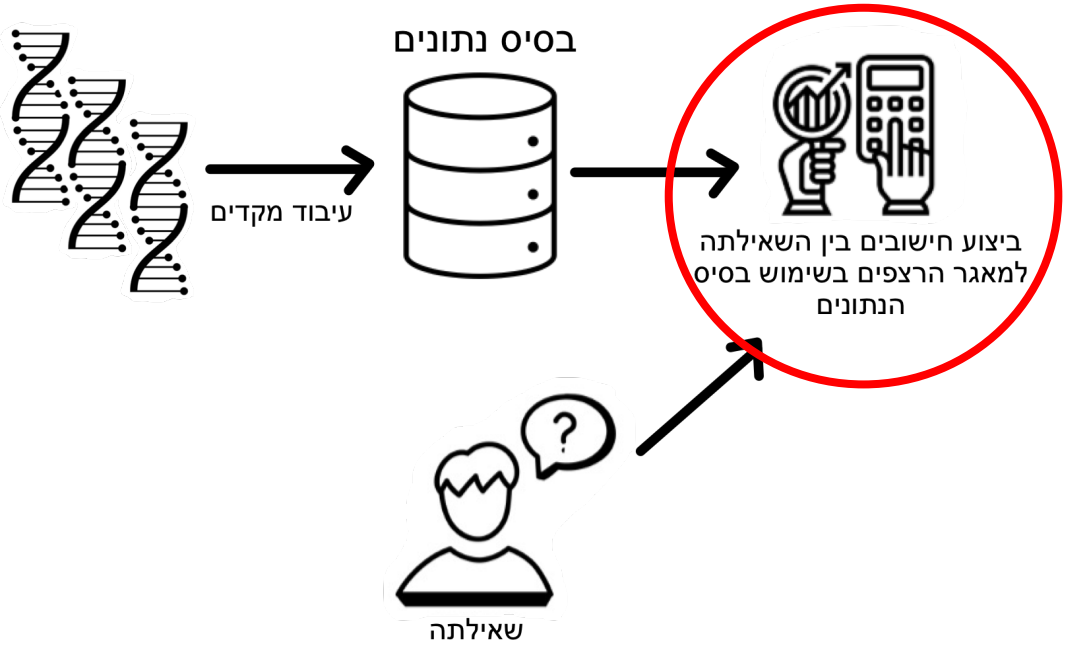
דוגמת הרצה (על המאגר המקורי- טרם ההוספה של רצף 4):

```
new_inverted_index, new_sequences_data =
remove_from_data(inverted_index, sequences_data, 3)
print(f"new_inverted_index: {new_inverted_index}")
print(f"new_sequences_data: {new_sequences_data}")
```

new_inverted_index: {'M': {1: 1}, 'R': {1: 1}, 'S': {1: 1}, 'V': {1: 1}, 'D': {1: 1}}

new_sequences_data: {1: 5, 2: 0}

חלק 3 – חישוב AAF-ISF



בחלק זה תממשו שתי פונקציות לחישוב AAF-ISF בין שאילתה לרצפים בבסיס הנתונים. שאילתה הינה מחרוזת אותה מזין המשתמש על מנת לחפש במאגר רצפים. ערך ה-AAF-ISF של רצף בהקשר של שאילתה יקבע עד כמה הרצף רלוונטי לבקשת המשתמש.

def calculate_aaf_isf(amino_acid, seq_id, inverted_index, sequences_data):

פונקציה המחשבת את ערך ה-AAF-ISF של רצף מבסיס הנתונים ביחס לשאילתה. הפונקציה מקבלת כקלט amino_acid מחרוזת של תו יחיד המייצגת חומצת אמינו, מספר מזהה של רצף seq_id (int), inverted_index , ו-sequences_data ומחזירה את ערך ה-AAF-ISF של חומצת האמינו amino_acid בהקשר של הרצף בעל המזהה seq_id.

ערך ה-AAF-ISF מחושב באופן הבא:

$$AAF = \log_2\left(\frac{Total\ number\ of\ sequences\ in\ the\ sequences\ corpus}{Number\ of\ sequences\ \ containing\ the\ amino\ acid}\right)$$

$$ISF = \frac{Number\ of\ times\ amino\ acid\ appears\ in\ sequence}{Total\ number\ of\ amino\ acids\ in\ sequence}$$

$$AAF - ISF = AAF \cdot ISF$$

שימו לב:

- הערך המוחזר צריך להיות מעוגל ל-3 ספרות אחרי הנקודה.
- הניחו כי המספר המזהה של הרצף קיים במאגר הנתונים.
- הניחו כי חומצת האמינו קיימת במילון inverted index.

דוגמת הרצה:

```
sequences_data = get_sequences_data(seq_corpus_example)
inverted_index = create_inverted_index(seq_corpus_example)
aaf_isf_M = calculate_aaf_isf('M', 1, inverted_index, sequences_data)
print(f'AAF-ISF of seq 1 and the amino acid "M" is: {aaf_isf_M}')

aaf_isf_M_2 = calculate_aaf_isf('M', 3, inverted_index,
sequences_data)
print(f'AAF-ISF of seq 3 and the amino acid "M" is: {aaf_isf_M_2}')

tf_idf_L = calculate_aaf_isf('L', 1, inverted_index, sequences_data)
print(f'AAF-ISF of seq 1 and the amino acid "L" is: {tf_idf_L}')
```

AAF-ISF of seq 1 and the amino acid "M": 0.117

AAF-ISF of seq 3 and the amino acid "M": 0.146

AAF-ISF of seq 1 and the amino acid "L": 0.0

חישוב עבור חומצת האמינו 'M' ורצף 1:

$$Total\ number\ of\ sequences\ in\ the\ sequences\ corpus = 3$$

$$Number\ of\ sequences\ containing\ the\ amino\ acid\ 'M' = 2$$

$$Number\ of\ times\ the\ amino\ acid\ 'M'\ appears\ in\ sequence\ 1 = 1$$

$$Total\ number\ of\ amino\ acids\ in\ sequence\ 1 = 5$$

$$TF - IDF('M', 1) = \log_2\left(\frac{3}{2}\right) \cdot \frac{1}{5} = 0.117$$

חישוב עבור חומצת האמינו 'M' ורצף 3:

$$Total\ number\ of\ sequences\ in\ the\ sequences\ corpus = 3$$

$$Number\ of\ sequences\ containing\ the\ amino\ acid\ 'M' = 2$$

$$Number\ of\ times\ the\ amino\ acid\ 'M'\ appears\ in\ sequence\ 3 = 3$$

$$Total\ number\ of\ amino\ acids\ in\ sequence\ 3 = 12$$

$$TF - IDF('M', 3) = \log_2\left(\frac{3}{2}\right) \cdot \frac{3}{12} = 0.146$$

def preprocess_query(query):

הפונקצייה מקבלת כקלט שאילתה (query), מחרוזת המורכבת מחומצות אמינו המופרדות אחת מהשנייה ע"י התו ",".
הפונקציה מחזירה טאפל של מחרוזות התואם לקלט כאשר כל איבר בו הוא מחרוזת המייצג חומצה אמינית. הניחו כי השאילתה מורכבת אך ורק מתווים בודדים של חומצות אמינו ופסיקים.

דוגמת ריצה:

```
print(preprocess_query("M,L,S"))
```

('M', 'L', 'S')

def get_scores_of_relevance_sequences(query, inverted_index, sequences_data):

הפונקציה מקבלת כקלט שאילתה (query) שהינה טאפל (tuple) של מחרוזות המייצגות חומצות אמינו, inverted_index ו-sequences_data ומחזירה מילון כאשר המפתחות הינם המספר המזהה של הרצפים הרלוונטים והערכים הינם ציון ה-AAF-ISF בין חומצות האמינו בשאילתה לרצף הרלוונטי.

הגדרות חשובות:

- רצף רלוונטי – רצף שלפחות אחת מהחומצות אמינו המופיעות בשאילתה מופיעות בו.
- ה-AAF-ISF בין טאפל חומצות אמינו בשאילתה לרצף ספציפי הינו סכום ה-AAF-ISF של החומצות אמינו המופיעות בטאפל לרצף.

שימו לב:

- לא ניתן להניח כי כל חומצות אמינו המופיעות בטאפל של השאילתה מופיעות בבסיס הנתונים. במידה ויש חומצת אמינו הנמצאת בטאפל השאילתה אך לא מופיעה בבסיס הנתונים ניתן לדלג עליה.

דוגמת הרצה:

```
sequences_data = get_sequences_data(seq_corpus_example)
inverted_index = create_inverted_index(seq_corpus_example)
print(get_scores_of_relevance_sequences(('R', 'M', 'S'),
inverted_index, sequences_data))
```

{1: 0.351, 3:0.292}

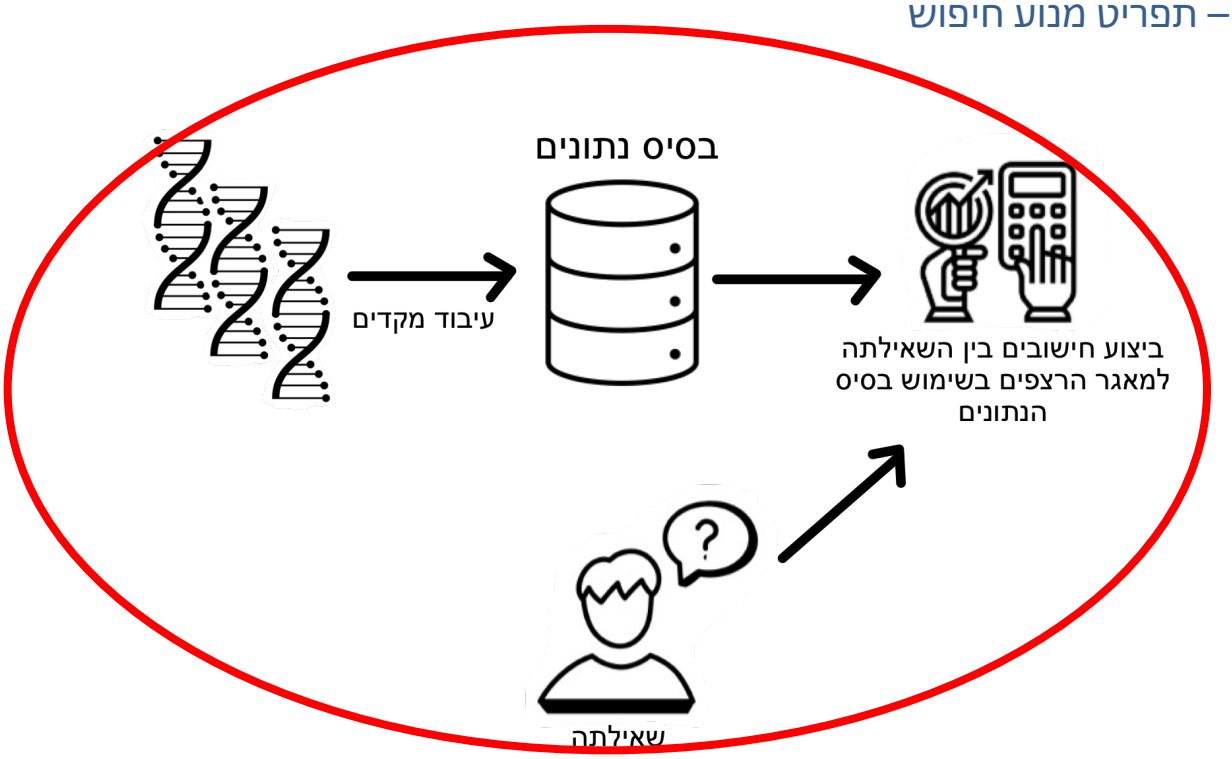
רצף מספר 2 לא מופיע בפלט מכיוון שהרצף ריק ואין לו אף חומצת אמינו משותפת עם חומצות האמינו של השאילתה.

```
print(get_scores_of_relevance_sequences(('L', 'A', 'G'),
inverted_index, sequences_data))
```

{3:0.396}

רצף מספר 1 ו-2 לא מופיעים בפלט מכיוון שאין להם אף חומצת אמינו משותפת עם חומצות האמינו של השאילתה.

חלק 4 – תפריט מנוע חיפוש



בחלק זה תממשו פונקציה המאחדת את כל השלבים שביצעתם עד כה ותהווה תפריט למנוע החיפוש.

def menu(sequence_corpus)

הפונקציה מקבלת כקלט מאגר רצפי דנ"א טרם העיבוד המקדים (מילון שהמפתחות בו הם המספרים המזהים של הרצפים והערכים הינם המחרוזות של כל רצף דנ"א) ואינה מחזירה דבר.

תחילה הפונקציה יוצרת את בסיס הנתונים, כלומר יוצרת את המילונים inverted_index ו-sequences_data עבור מאגר הרצפים שבקלט, sequence_corpus.

לאחר מכן יוצג למשתמש התפריט הבא:

Choose an option from the menu:

(1) Insert a query.

(2) Add sequence to sequence_corpus.

(3) Calculate AAF-ISF Score for an amino acid in a sequence.

(4) Delete a sequence from the sequence_corpus.

(5) Exit.

Your choice:

שימו לב שהטקסט נתון לכם בקובץ התרגיל (שימו לב כי על הטקסט המלא להופיע יחד בבקשת הקלט מהמשתמש, כפי שנתון המשתנה choice בקובץ התרגיל, ולא כהדפסה של התפריט בעזרת print ולאחר מכן בקשת הקלט).

<u>בחירה באפשרות (1):</u>

כאשר המשתמש יבחר באפשרות זו תוצג לו ההודעה הבאה:

Write your query here:

כלומר על המשתמש להכניס שאילתה. שאילתה הינה מחרוזת של חומצות אמינו המופרדות ע"י ",". עליכם לבצע עיבוד מקדים לשאילתה באופן דומה למתואר בחלק 3. הניחו כי כל שאילתה של המשתמש היא קלט תקין.

לאחר מכן יוצג למשתמש ההודעה הבאה:

Choose the type of results you would like to retrieve:

(A) All relevant sequences.

(B) The most relevant sequence.

(C) Back to the main menu.

Your choice:

שימו לב שהטקסט נתון לכם בקובץ התרגיל. (שימו לב כי על הטקסט המלא להופיע יחד בבקשת הקלט מהמשתמש, כפי שנתון במשתנה query_choice בקובץ התרגיל, ולא כהדפסה של התפריט בעזרת print ולאחר מכן בקשת הקלט).

בחירה באפשרות (A): יודפסו למשתמש כל הרצפים הרלוונטיים לשאילתה שהכניס וציון ה-AAF-ISF בינם לבין השאילתה. הרצפים יודפסו אחד אחרי השני (בשורה חדשה) בפורמט הבא: *{seq_id} : {score}*

בחירה באפשרות (B): יודפס למשתמש מספר הרצף עם ציון ה-AAF-ISF הגבוה ביותר, כלומר הרצף שהכי תואם לשאילתה. תודפס ההודעה הבאה:

The most relevant sequence is {highest_score_seq_id} with a score of {highest_score}

בחירה באפשרות (C): המשתמש יחזור לתפריט הראשי והתפריט יודפס בפניו.

בחירת אפשרות שלא מתוך האפשרויות הנתונות: תודפס למשתמש ההודעה הבאה:

Invalid choice. Please select a valid option.

לאחר מכן יודפס התפריט מחדש.

<u>בחירה באפשרות (2):</u>

הוספת רצף לבסיס הנתונים.

תחילה נבקש מהמשתמש את מספר הזהות של הרצף החדש:

Insert the sequence ID:

ניתן להניח כי המשתמש הכניס מספר שלם אך אין להניח כי הוא הכניס מספר שלא קיים בבסיס הנתונים. במידה והמשתמש הכניס מספר שכבר קיים יש להציג לו את ההודעה הבאה:

The sequence ID {seq_id} is already in sequence_corpus.

לאחר מכן יש לחזור להודעה הקודמת על מנת לבקש מספר חדש.

כאשר המשתמש הכניס מספר שלא קיים בבסיס הנתונים יש לבקש מהמשתמש את הטקסט של הרצף באופן הבא:

Insert the sequence itself:

לאחר מכן יש להוסיף את הרצף החדש למילונים inverted_index ו-sequences_data ולהדפיס למשתמש הודעה שההוספה הושלמה:

Sequence {seq_id} was successfully added!

שימו לב שהוספת הרצף החדש לא משנה את מאגר הרצפים המקורי אלא רק את המילונים *inverted_index* ו-*sequences_data*.

בחירה באפשרות (3):

חישוב את ערך ה-AAF-ISF בין חומצת אמינו לרצף.

תחילה נבקש מהמשתמש את המספר המזהה של הרצף המבוקש:

Insert the sequence ID:

ניתן להניח כי המשתמש הכניס מספר שלם אך אין להניח כי הוא הכניס מספר שקיים בבסיס הנתונים. במידה והמשתמש הכניס מספר שלא קיים יש להציג לו את ההודעה הבאה:

The sequence ID {seq_id} is not in sequence_corpus.

לאחר מכן יש לחזור להודעה הקודמת על מנת לבקש מספר חדש.

כאשר המשתמש הכניס מספר שקיים בבסיס הנתונים יש לבקש מהמשתמש את חומצת האמינו שירצה לחשב עבורה את ה-AAF-ISF בינה לבין הרצף:

Insert an amino acid:

אם חומצת האמינו לא נמצאת בבסיס הנתונים (לאחר עיבוד מקדים של רצף הדנ"א), יש להדפיס את ההודעה הבאה:

The amino acid { amino_acid_after_preprocces } is not in sequence_corpus.

לאחר מכן יש לחזור להודעה הקודמת על מנת לבקש חומצת אמינו חדשה. הניחו כי המשתמש יכניס רק תווים אלפביתיים המייצגים חומצות אמינו.

כאשר המשתמש הכניס חומצת אמינו שקיימת בבסיס הנתונים יש לחשב את ערך ה-AAF-ISF בינה (לאחר עיבוד מקדים) לבין הרצף שנבחר ולהדפיס את ההודעה הבאה:

AAF-ISF of the amino acid {amino_acid_after_preprocces} in sequence {seq_id} is: {aaf_isf}

בחירה באפשרות (4):

מחיקה של רצף מבסיס הנתונים.

תחילה נבקש מהמשתמש את המספר המזהה של הרצף המבוקש:

Insert the sequence ID:

ניתן להניח כי המשתמש הכניס מספר שלם אך אין להניח כי הוא הכניס מספר שקיים בבסיס הנתונים. במידה והמשתמש הכניס מספר שלא קיים יש להציג לו את ההודעה הבאה:

The sequence ID {seq_id} is not in sequence_corpus.

לאחר מכן יש לחזור להודעה הקודמת על מנת לבקש מספר חדש.

כאשר המשתמש הכניס מספר שקיים בבסיס הנתונים יש למחוק את רצף זה מהמילונים inverted_index ו-sequences_data ולהדפיס למשתמש הודעה שהמחיקה הושלמה:

Sequence {seq_id} was successfully deleted.

שימו לב שמחיקת הרצף לא משנה את מאגר הרצפים המקורי אלא רק את בסיס הנתונים, כלומר את יש לעדכן רק את המילונים *inverted_index* ו- *sequences_data*.

בחירה באפשרות (5):

אפשרות זו תסיים את ריצת הפונקציה.

בחירה באפשרות לא חוקית:

תודפס למשתמש ההודעה הבאה:

Invalid choice. Please select a valid option.

לאחר מכן יודפס התפריט מחדש.

שימו לב: דוגמת הרצה מפורטת נמצאת בקובץ menu_example.txt. מומלץ להריץ את הקוד שלכם לפי השלבים בדוגמא ולהשתמש ב-text compare על מנת לוודא שההדפסות שלכם תואמות להדפסות שבדוגמא. כל ההדפסות צריכות להיות זהות לדוגמא שלנו ברמת הרווחים, אותיות קטנות/גדולות וכדומה. אחרת הטסטים לא יעברו.

בהצלחה!