

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

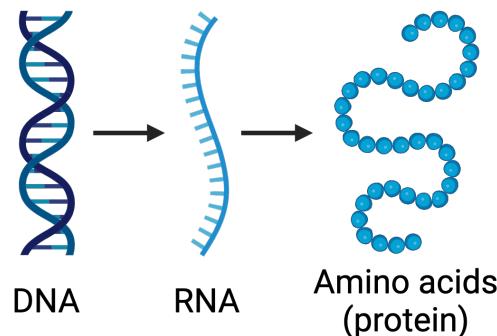
מטרות העבודה: תרגול עבודה עם מחרוזות (`str`) ומילונים לשם עיבוד מידע.

הנחיות לעבודה 3:

- קראו בקפידה את הנחיות ההגשה. אל תגישו קבצים או קוד מיותר. וודאו כי אתם עומדים בכל הנקודות לפני ההגשה!
- מומלץ לקרוא את כל העבודה בשלמותה לפני תחילת כתיבת קוד הפתרון.
- אין להשתמש בספריות חיצונית (ניתן להשתמש בספריה `math`).
- ניתן להוסיף פונקציות עצר. האחוריות על בניית הקוד בפונקציות אלה היא עליכם.
- חובה לבתוב `docstring` (כפי שנלמד בכיתה) לכל פונקציה.
- **יש להגיש קובץ אחד בשם `W3.py` (שלד הקובץ נתון לכם ורצוי למלא את הפתרון במקומות המתואימים).**

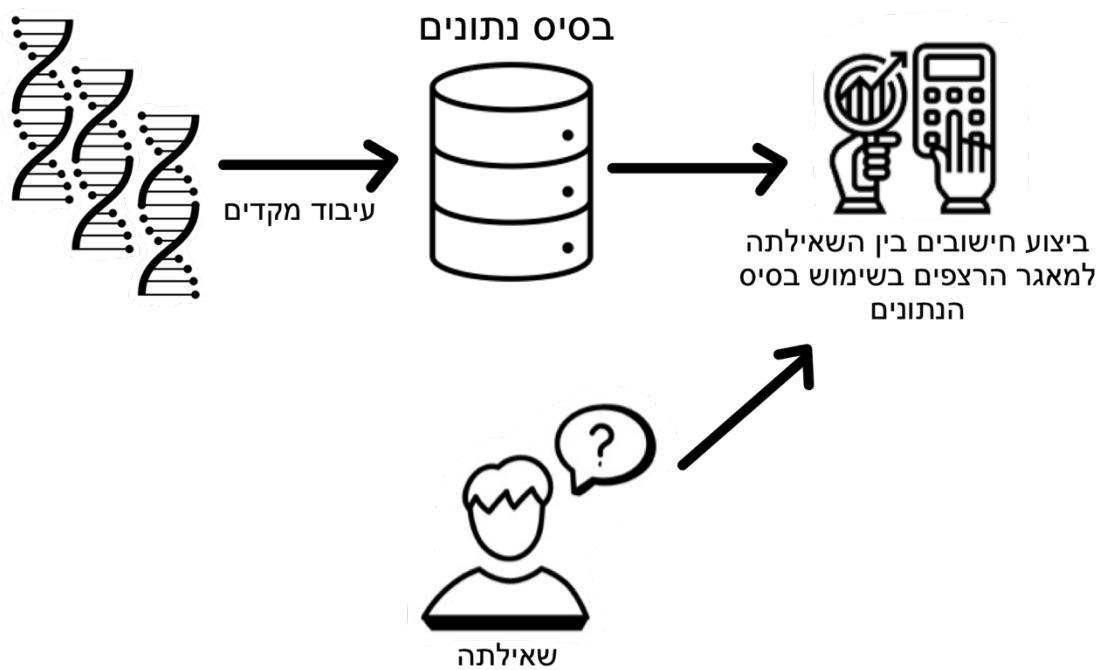
רקע ומטרת העבודה:

דנ"א (DNA), החומר הגנטי שמועבר מדור לדור, הוא רצף המכיל את המידע החדש לצירמת חלבונים ("proteins", "המונעים"). שגורמים לתאים בגוף שלנו לפעול) ולכן הוא הכרחי לחייהם של כל היצורים. רצפי הדנ"א הם סוג של מחרוזות ארוכה המורכבת מאלפבית של ארבע אותיות: C, G, T, A. רצף דנ"א עובר תהליך של המרה לרצף RNA. כל רצף של שלוש אותיות בRNA'א מגדיר מילה בעלת משמעות הנקראת "חומצת אמינוא" (Amino acid). חומצות אמינו הן היחידה הבסיסית לבניית החלבונים. RNA'א הוא שלב ביןיהם בין הדנ"א לחלבונים (רצף דנ"א -> RNA -> רצף חומצות אמינוא'ו/חלבון).



בתרגיל זה תתנסו בעבודה עם מחרוזות ומילונים. בעודם, תמשו תוכנית המבצעת עיבוד מקדים של מאגר רצפי דנ"א לבסיס נתונים המאחסן רצפי חומצות אמינוא ומכוון חיפוש בו נתן לחפש חומצת אמינוא לבסיס הנתונים באמצעות שאלות. **בחלק הראשון** של התרגיל, תמשו עיבוד מקדים שנועד להסיר מקטעים, סימנים ותוויות לא רלוונטיים ברצף הדנ"א, לשנות את הרצף לפי כללים מסוימים ולאחר מכן להמיר אותו לרצף RNA'א, ולבסוף לרצף של חומצות אמינוא (או חלבון). **בחלק השני**, תאגדו את המידע מכל הרצפים השונים במאגר לבסיס נתונים שיאפשר חיפוש מהיר. **בחלק השלישי**, תשתמשו לבסיס הנתונים שיצרתם על מנת לבצע חישובים שונים המערבים שאלתה (מחרוזת) ואת מאגר רצפי חומצות האמינוא. **בחלק הרביעי והחמישי**, תאחדו את כל השלבים לתפריט אחד בו המשתמש יוכל לבצע פעולות על מאגר רצפי חומצות האמינוא.

להלן תרשים:



במה הגדירות חשובות לפני שמתחילה?

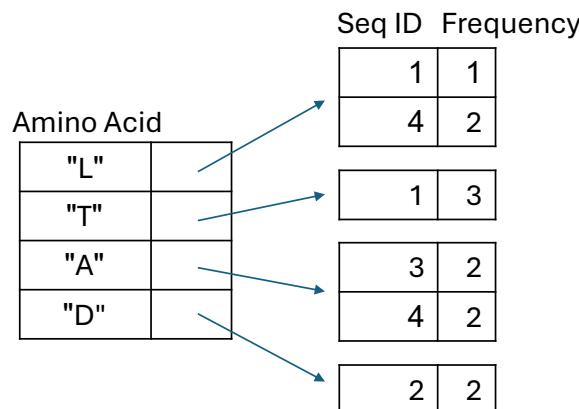
- רצף דנ"א הוא מחרוזת המורכבת מהאותיות האלפביתיות a-z (small capital), וסימני הפיסוק ":" "," "_" "," ".". רצף דנ"א לא מכיל ספרות.
- רצף דנ"א פונקציוני הוא רשימת מחרוזות אשר כל מחרוזת היא שלשה המורכבות מהאותיות C,G,T,A בלבד (-Capital letters), השלשה הראשונה בו היא ATG (שעשויה להופיע שוב בראשית המחרוזות) ומסתיים באחת מהשלשות הבאות: TAA, TAG, TGA, הנקראות stop codons. לא יהיו מופעים נוספים של stop codons מעבר לאות שבסוף הרשימה בכלל דנ"א פונקציוני.
- על מנת להמיר רצף דנ"א לצורתו הפונקציונלית (proofreading), יש לבצע את הפעולות הבאות: (1) למצוא את המופיע הראשון של השלשה ATG בתוך הרצף שמכתיבת את השלשה הראשונה של הדנ"א הפונקציוני ואת החלוקה של שאר הרצף לשלשות. (2) למצוא את אחת משלשות הסיום codons stop ברצף אשר נמצא בסדר מתאים למופיע הראשון של הרצף לשלשות. (3) להסיר את כל התווים העודפים המופיעים אחרת.
- אם לא קיימת שלשה בזאת (stop codon) יש להסיר עודפי תוויים שאינם משלימים ושלשה בסוף הרצף ולהוסיף את שלשת הסיום TAA. לדוגמה:
אם נתון רצף הדנ"א הבא:
'AGATGGGGCTTTGAAGTAA' -> ['ATG', 'GGG', 'CTT', 'TGA']
הצורה הפונקציונלית של הדנ"א מתחילה מהאינדקס 2 בו נמצא המופיע הראשון של ATG ומסתיימת עם TGA והוא TGA. יש להסיר את התחלתית "AG" ואת "AGTAA" מאחר והם מופיעים לאחר TGA.
לעומת זאת ברצף הבא:
'ATGTTTGA' -> ['ATG', 'TTT', 'TAA']
הצורה הפונקציונלית של הדנ"א מתחילה מהאינדקס 0 בו נמצא המופיע הראשון של ATG ומסתיימת עם TAA. השלשה TAA אינה קיימת ברצף הדנ"א והיא מתווספת לאחר המירה. ברצף הדנ"א, **TGA** (באינדקסים 5 עד 7 כולל) אינה נחשבת לשילת הסיום לרצף זה וזאת מאחר והוא לא נמצא בשלה בהתאם לחלוקת הרצף לשלשות המוכתבת ע"י מופיע הראשון של ATG.
- **Transcription - שעתוק:** בתהליך השעתוק, יש להפוך רצף דנ"א פונקציוני לרצף RNA. בדומה לדנ"א פונקציוני, RNA גם מוגדר להיות רשימת מחרוזות אשר כל מחרוזת היא שלשה, אבל אות T (שלא קיימת בRNA) מוחלפת באות U.
- **Translation - תרגום:** רצפי RNA מתרגמים לרצף של חומצות אמינו. התרגום מתבצע כך שבכל שלשה RNA, למעט ה-*stop codon* (UAA, UAG, UGA), מתרגם לחומצת אמינית, ורצף חומצות האמינו מרכיבות את החלבן השלים. שימוש לב, כדי להקל עליהם, נתונה פונקציית עזר בשם `codon_translator`, המתקבלת מחרוזת של שלשת RNA ומחרזרת את החומצה האמינית אליה מתרגם השלשה, אותן בודדת, לפי הטבלה הבאה (שימוש לב לשימוש במילון במימוש):

F	Serine	Tyrosine	Cysteine
L		Stop codon	Stop codon
CUU CUC CUA CUG	Leucine	Proline	Histidine
AUU AUC AUA AUG	Isoleucine		Glutamate
M	Methionine start codon	Threonine	Asparagine
GUU GUC GUA GUG	Valine	Alanine	Lysine
		A	D
		GCU GCC GCA GCG	Aspartic acid
			E
		GAA GAG	Glutamic acid
			G
		GGU GGC GGA GGG	Glycine

למשל השלשה UUU מתרגם לחומצת אmino "F" והשלשה CCU מתרגם לחומצת amino "P".

- **sequence_corpus** – אוסף רצפים, קורפוס של רצפים. מיוצג ע"י מילון באשר המפתחות הינם המספר המזהה של הרצף והערכים הינם רצף הדן'א כמחזורות טרם העיבוד המקוריים. המשטנה `sequence_corpus` איבן משתנה גלובלי ויש להעיבר אותו כארוגמנט בכל קריאה לפונקציה המשתמשת בו. שימושו לב שבדיקת העבודה נשתמש בקורסוס אחר מזהה שנקבע לכם בקובץ העבודה.

- **Inverted index** – מבנה נתונים בו עבר כל חומצת amino המיוצגת ע"י אות בודדת (capital letters) והמופיעה במאגר הרצפים נשמרים כל הרצפים בהם החומצת הופיעה. השימוש הוא באמצעות מילון באשר המפתחות הינם חומצות amino המופיעות במאגר הרצפים (מיפוי מחוזת). כל מפתח ממופה למילון באשר המפתחות שלו הם המספרים המזהים של רצפים המכילים את החומצת, והערכים הינם מספר המופיעים של חומצת amino ברכף.



על פי האירור, חומצת amino "L" מופיעה ברכף #1, פעם אחת, וברכף #4, פעמיים.

-

AAF-ISF – קיצור של "Amino Acid Frequency–Inverse Sequence Frequency", בעברית: תדירות חומצת אמינים

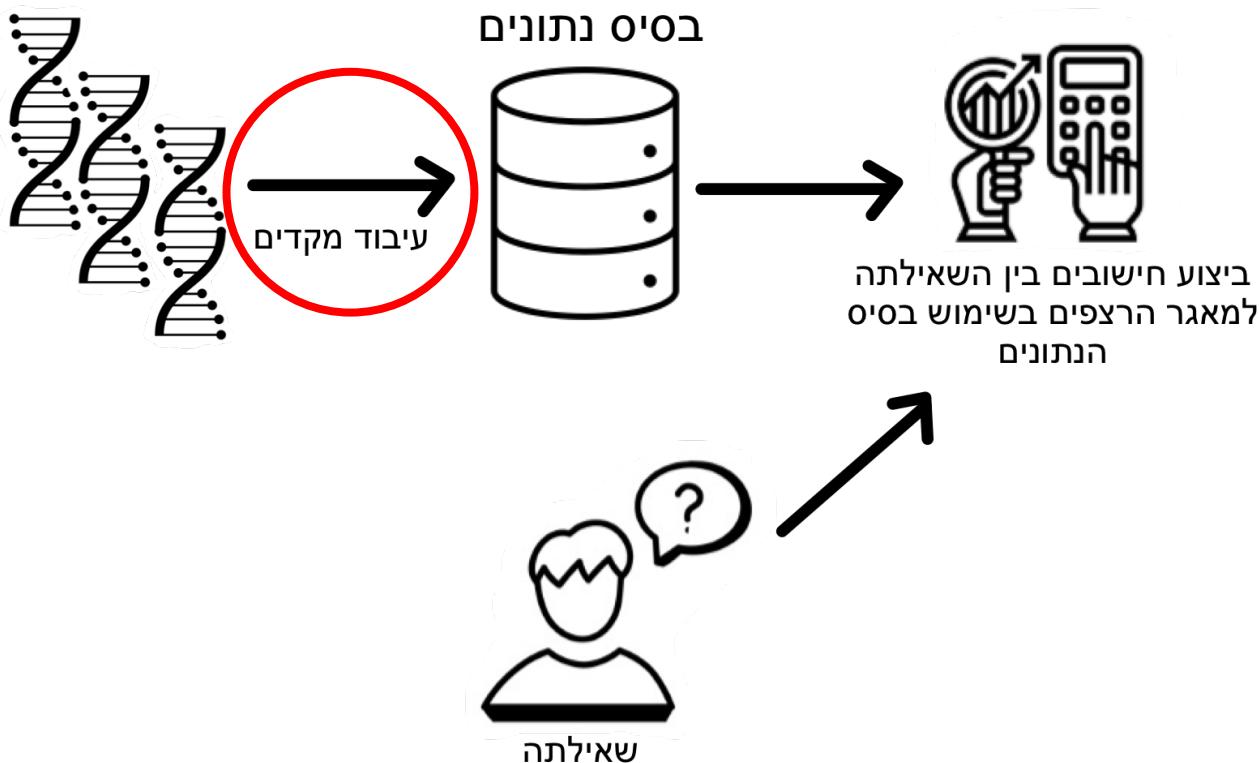
- תדירות חומצת אמינים הופכית (isf) שימושה ציון מספרי שנועד לשקף את החשיבות של חומצת אמינים מסוימת לרצף מסוים.

תדירות חומצת אמינים (aaf) – באיזה תדירות מופיעה חומצת האминו ברצף.

תדירות חומצת אמינים הופכית (isf) – חשיבות חומצת האמיןו בכל מאגר הרצפים.

חלק 1 – עיבוד מקדים

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



```
def remove_punctuations(seq)
```

הfonקציה מקבלת כקלט מחרוזת רצף דנ"א ומחזירה מחרוזת חדשה (str) התואמת לקלט, לאחר הסרת סימני פיסוק. בל סימן פיסוק בקלט יוחלף ברווח בפלט.
דוגמת הרצה:

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

```
'A gcrTG G '
```

`def remove_invalid_letters(seq)`

הfonקציה מקבלת כקלט מחרוזת רצף דנ"א ומחזירה מחרוזת חדשה (str) התואמת לקלט, ללא כל אות השונה מהאלפבית של הדנ"א (כולומר שונה מ-cgc). כל אות לא חוקית בקלט תוחלף ברווח בפלט.
דוגמת הרצה:

```
>>> remove_invalid_letters("A gcrTG-G ")
```

```
'A gc TG-G '
```

`def remove_spaces(seq)`

הfonקציה מקבלת כקלט מחרוזת רצף דנ"א ומחזירה מחרוזת חדשה (str) התואמת לקלט, ללא רווחים.
דוגמת הרצה:

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

```
'AgcrTG-G'
```

`def capitalize_letters(seq)`

הfonקציה מקבלת כקלט מחרוזת רצף דנ"א ומחזירה מחרוזת חדשה (str) התואמת לקלט, באשר כל האותיות גדולות.
דוגמת הרצה:

```
>>> capitalize_letters("A gcrTG-G ")
```

```
'A GCRTG-G '
```

`def proofreading(seq)`

הfonקציה מקבלת כקלט מחרוזת של רצף דנ"א המכיל ארך וرك רצף של אותיות ATCG ומחזירה רשימת מחרוזות, באשר כל איבר ברישימה הוא שלשה של רצף דנ"א פונקציוני. על מנת להמיר רצף דנ"א לצורת הfonקציוני יש לוודא את הכללים הבאים:

- רצף דנ"א פונקציוני מתחילה בשלשה ATG שמכטיבה את החלוקה לשלוות של שאר הרצף. רצף דנ"א שאינו בו ATG לא ניתן להמיר לדנ"א פונקציוני ולבן יש להחזיר רשימה ריקה.
- דנ"א פונקציוני מסתיים בשלשת סיום stop codons (אם אין שלשת בז', יש להוסיף את השלשה TAA).

דוגמת הרצה:

```
>>> proofreading ('AGATGACGGCAGCATGA')
```

```
['ATG', 'ACG', 'GCA', 'GCA', 'TGA']
```

```
>>> proofreading ('AAATGTTTGA')
```

```
['ATG', 'TTT', 'TAA']
```

שימוש לב: לאחר פעולה proofreading שבוגמה השניה התווספה לרצף הדב"א הפונקציונלי שלשת סימן שלא הייתה ברצף המקורי.

```
def transcript(func_seq)
```

הfonkzia מתקבלת בקלט רשימת מחרוזות המיצגת רצף דנ"א פונקציונלי. הפונקציה משעתקת את הדב"א פונקציונלי לרצף רנ"א מחזירה רשימה חדשה של מחרוזות כאשר כל מחרוזת היא שלשה של הרנ"א, כלומר לאחר החלפת כל מופע של T באחת השלשות ברשימה קלט בباءות U.

דוגמת הרצה:

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

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

```
def translate(rna_seq)
```

הfonkzia מקבלת בקלט רשימת מחרוזות המיצגת רצף רנ"א. הפונקציה מחזירה רשימה המיצגת את רצף חומצות האמינו. חומצת אmino מוצגת ע"י תו בודד (capital-letter). היעדו בפונקציית עזר codon_translator אשר מקבלת שלשת רנ"א מחזירה את התרגום שלה לחומצה אמינית כתו בודד. שלשת הסימן אינה מתורגמת לחומצת אmino וכן אינה נמצאת ברצף חומצות האמינו.

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

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

```
def preprocessing(seq)
```

פונקציה המאחדת את תחילין העיבוד המקדים של רצף דנ"א עד להמרתו לרצף חומצות אmino. הפונקציה מקבלת בקלט רצף דנ"א (str) ומוחזירה את רצף חומצות האmino המתאים להמרת הדנ"א בקלט ברשימה מחרוזות אשר כל מחרוזת היאתו המיצג חומצת אמינית.

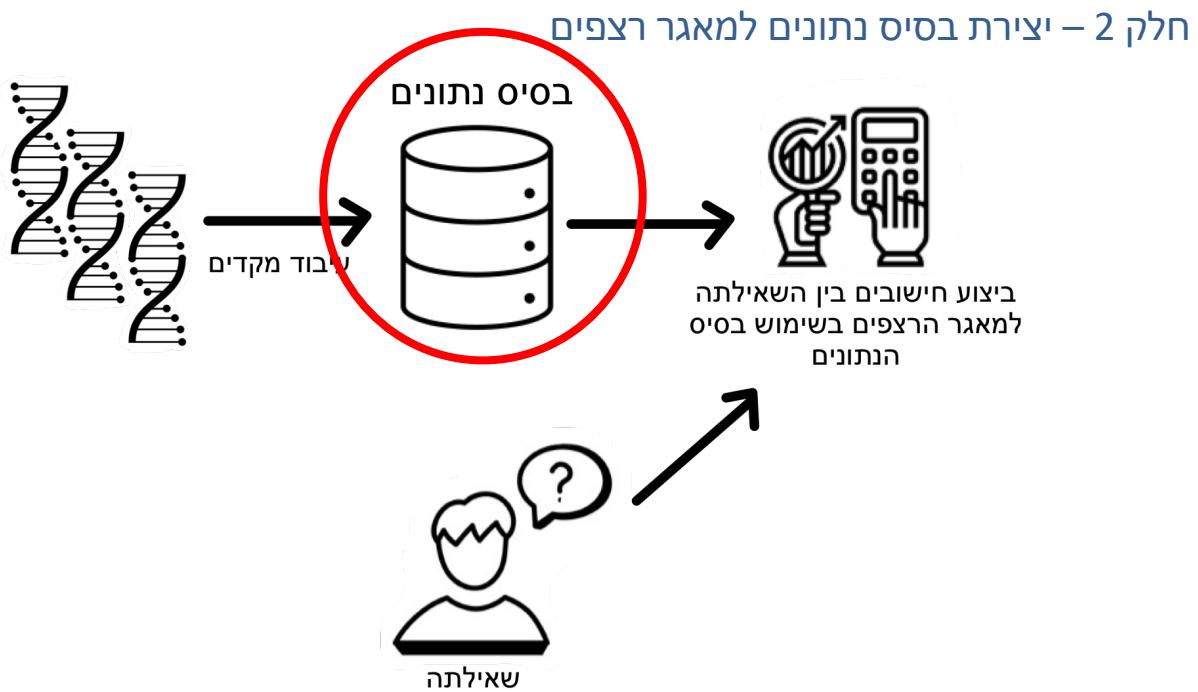
הfonkzia מבצעת את השלבים הבאים (בסדר זה):

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

דוגמת הרצה:

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

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



בחלק זה ת认真学习 פונקציות היוצרת ממאגר הרצפים של דנ"א בסיס הנתונים ומחזקאות אותו. בסיס הנתונים מאפשר ייצוג דחוס של מאגר הרצפים (על מנת לחסוך בזיכרון) וחיפוש מהיר (ביחס לחיפוש במאגר הרצפים המקורי).

בסיס הנתונים מורכב משני מילוניים:

- sequences data – מילון השומר עבור כל רצף את אורךו לאחר עיבוד מקדים (מספר חומצות אמינו ברצף).
- inverted index – מילון השומר עבור כל חומצה אמינית את מספר המופעים שלה בכל רצף (ראו אויר 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_yuCATGATGCGTACCAAGGCTqwAGCATGCGTbbAGCTAxzvGCATGAC'
}
```

```
def get_sequences_data(sequence_corpus)
```

פונקציה היוצרת sequences_data עבור מאגר רצפים טרם העיבוד המקדים. הפונקציה מקבלת בקלט 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 לאחר תהליך העיבוד המקדים המתואר בחלק 1.

דוגמת הרצה:

```
>>> get_sequences_data(seq_corpus_example)
{ 1: 5, 2: 0, 3: 12}
```

הסביר לרצף מס' 1: לאחר העיבוד המקדים מתקבל כי רצף חומצות האמינו הוא: ['D', 'V', 'M']. הרצף מסתיים עם השלשה TAG וכן לאחר העיבוד המקדים רצף זה מכיל 5 חומצות אמינו.

```
def create_inverted_index(sequence_corpus)
```

פונקציה היוצרת inverted index עברו מאגר רצפים טרם העיבוד המקדים. הפונקציה מקבלת בקלט מילון המציג מאגר רצפים טרם עיבוד, כאשר המפתחות הינם המספרים המזהים של הרצף (int) והערכים הינם המחרוזות המייצגות את הרצפים התואימים (str). הפונקציה מחזירה inverted index של מאגר הרצפים לאחר עיבוד מקדים לרצף. inverted index הינו מילון, אשר 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 המופיעים של כל חומצה בכל רצף בהתאם (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}}
```

הסביר על החומרה האמינית 'M' – החומרה האמינית 'M' מופיעה ברצף 1 פעם אחת וברצף 3 שלושה פעמים כאשר היא מוקדדת מהשלשה "ATG". ברצף 2, אין את החומרה 'M' (או כל חומרה 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 בקלט seq_id, inverted_index, sequences_data מס' רצף מהרוצף החדש (int) ו-seq מהרוצף החדש (str) ומחזירה inverted_index (בסדר זה) מעודכנים לפי הרצף החדש.

שימוש לרב:

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

דוגמת הרצה:

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

```
".yyyuTACGATGGTAGCTAGCGlllTACGATCGTA")
```

```
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 מסוים הנתונים. הפונקציה מקבלת בקלט *inverted_index*, *sequences_data* ו-*seq_id*-ו *inverted_index*, *sequences_data* מספר מזהה של הרץ אותו רצים למחוק (int) ומוחירה *sequences_data* ו-*inverted_index* (בסדר זהה) מעודכנים ללא הרץ שנבחר למחיקה.

שימוש לב:

- הינו כי המספר המזהה של הרץ קיים במאגר הנתונים.
- הפונקציה אינה משנה את מאגר הנתונים המקורי אלא רק את המילונים *sequences_data* ו-*inverted_index*.
- אם לאחר מחיקת הרץ במילון *inverted_index* יש חומצות אמיינו שהມילון בערך שלו נותר ריק, יש למחוק חומצות *.inverted_index*.

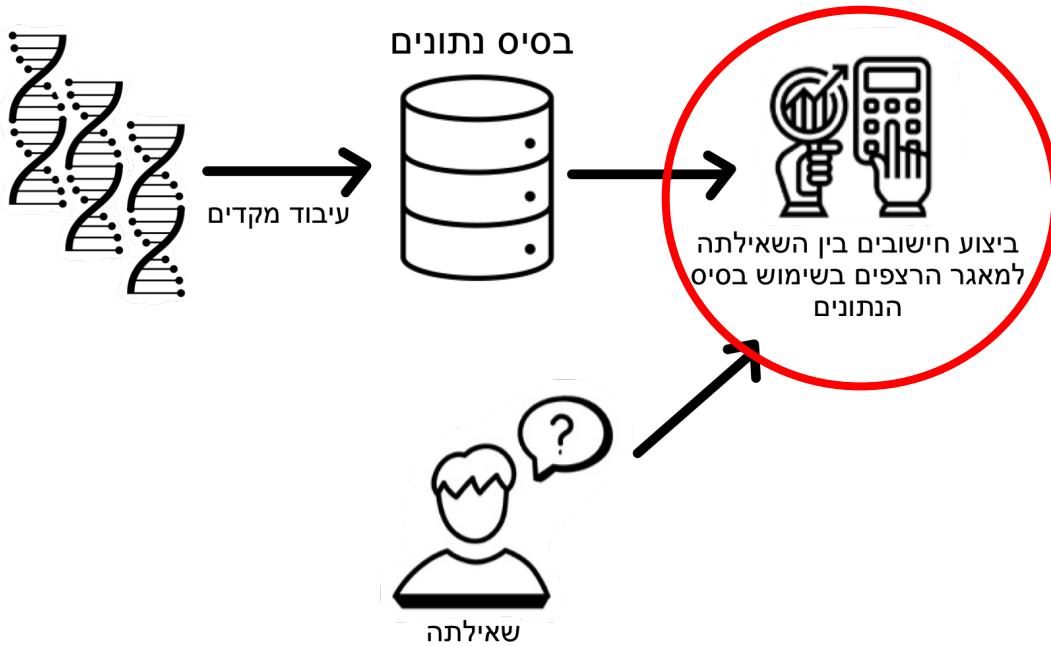
דוגמת הרצה (על המאגר המקורי - טרם ההוספה של רצף 4):

```
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



ב חלק זה תממשו שתי פונקציות לחישוב AAF-ISF בין שאלתה לרצפים בסיס הנתונים. שאלתה הינה מחרוזת אותה מזין המשמש על מנת לחפש במאגר רצפים. ערך ה-AAF-ISF של רצף בהקשר של שאלתה 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 מסוים הנתונים ביחס לשאלתה. הפונקציה מקבלת בקלט amino_acid של רצף מסוימת אминואcid, inverted_index (int) seq_id וsequences_data ומוחזרת שלתו ייחיד המציג חומצת אמינואcid, מספר מזהה של רצף id_seq ומחזירה את ערך ה-AAF-ISF של חומצת האמינואcid amino_acid בהקשר של הרצף בעל המזהה id_seq.

ערך ה-AAF-ISF מחושב באמצעות הבא:

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

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

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

שימוש לב:

- הערך המוחזר צריך להיות מעוגל ל-3 ספרות אחרי הנקודה.
- הניכוי במספר המזהה של הרצף קיים במאגר הנתונים.
- הניכוי ביחס חומצת האמינואcid במלון inverted_index.

```

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

чисוב עבור חומצת האминו 'M' ורץ 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

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

чисוב עבור חומצת האמינו 'M' ורץ 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

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

```
def preprocess_query(query):
```

הfonקציה מקבלת בקלט שאלתה (query), מחרוזת המורכבת מחומצות אmino המופרדות אחת מהשנייה ע"י התו ",". הfonקציה מוחזירה טאפל של מחרוזות התואם לקלט כאשר כל איבר בו הוא מחרוזת המייצג חומצה אmino. הניחו כי השאלתה מורכבת אך ורק מתחומים בודדים של חומצות amino ופסיקים.

דוגמת ריצה:

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

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

```
def get_scores_of_relevance_sequences(query, inverted_index, sequences_data):
```

הfonקציה מקבלת בקלט שאלתה (query) שהינה טאפל (tuple) של מחרוזות המייצגות חומצות amino, inverted_index ו- sequences_data ומוחזירה מילון כאשר המפתחות הינם המספר המזהה של הרצפים הרלוונטיים והערכים הינם ציון ה-AAF-ISF בין חומצות amino בשאלתה לרץ' הרלוונטי.

הגדרות חשובות:

- רץ' רלוונטי – רץ' של לפחות אחת מהחומצות amino המופיעות בשאלתה מופיעות בו.
- ה-AAF-ISF בין טאפל חומצות amino בשאלתה לרץ' ספציפי הינו סכום ה-AAF-ISF של החומצות amino המופיעות בטאפל לרץ'.

שים לב:

- לא ניתן להניח כי כל חומצות amino המופיעות בטאפל של השאלתה מופיעות בסיס הנתונים. במידה ויש חומצת amino הנמצאת בטאפל השאלתה אך לא מופיעה בסיס הנתונים ניתן לדלג עליה.

דוגמת הריצה:

```
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}

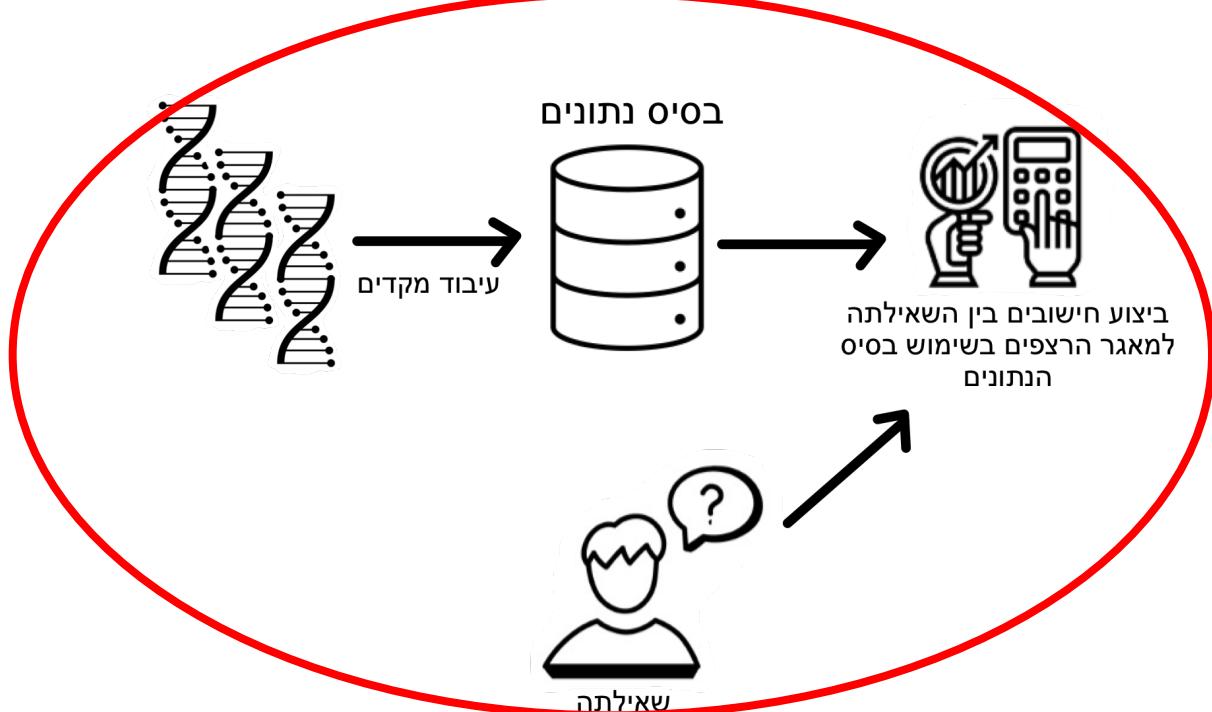
רץ' מס' 2 לא מופיע בפלט מכיוון שהרצף ריק ואין לו אף חומצת amino משותפת עם חומצות amino של השאלתה.

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

{3:0.396}

רץ' מס' 1 ו-2 לא מופיעים בפלט מכיוון שאין להם אף חומצת amino משותפת עם חומצות amino של השאלתה.

חלק 4 – תפריט מנוע חיפוש



ב חלק זה תממשו פונקציה המאחדת את כל השלבים שביצעתם עד כה ותחווה תפריט למנוע החיפוש.

```
def menu(sequence_corpus)
```

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

תחליה הפונקציה יוצרת את בסיס הנתונים, בולם יוצרת את המילונים `sequences_data` ו- `inverted_index` 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:

שיםו לב שהtekסט נתון לכם בקובץ התרגיל (שיםו לב כי על הטקסט המלא להופיע יחד בבקשת הקלט מהמשתמש, بما שנותן המשתנה `choice` בקובץ התרגיל, ולא בהדפסה של התפריט בעדרת `print` ולאחר מכן בבקשת הקלט).

בחירה באפשרות (1):

כאשר המשתמש יבחר באפשרות זו תוצג לו הודעה הבא:

Write your query here:

כולם על המשתמש להכנס שאלתה. שאלתה הינה מחרוזת של חומצות אמינו המופרדות ע"י ". ". عليיכם לבצע עיבוד מקדים לשאלתה באופן דומה למຕואר בחלק 3. הניחו כי כל שאלתה של המשתמש היא קלט תקין.

לאחר מכן יוצג למשתמש הודעה הבא:

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:

שיםו לב שהtekst נתון לכם בקובץ התרגיל. (שיםו לב כי על הטקסט המלא להופיע יחד בבקשת הקלט מהמשתמש, כפי שנטען במשתנה `query_choice` בקובץ התרגיל, ולא כהדפסה של התפריט בעזרת `print` ולאחר מכן בבקשת הקלט).

בחירה באפשרות (A): יודפסו למשתמש כל הרצפים הרלוונטיים לשאלתה שהכנס וציון ה-AAF-AAF בין לבין השאלתה. הרצפים יודפסו אחד אחרי השני (בשורה חדשה) בפורמט הבא: `{seq_id} : {score}`

בחירה באפשרות (B): יודפס למשתמש מספר הרצף עם ציון ה-AAF-AAF הגבוה ביותר, בລומר הרצף שהכי תואם לשאלתה. תודפס הודעה הבא:

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 הודעה הבא:

Invalid choice. Please select a valid option.

לאחר מכן יודפס התפריט מחדש.

בחירה באפשרות (2):

הוספה רצף לבסיס הנתונים.

תחליה נבקש מהמעštמש את מספר הזאות של הרצף החדש:

Insert the sequence ID:

ניתן להניח כי המשתמש הכנס מספר שלם אך אין להניח כי הוא הכנס מספר שלא קיים בסיס הנתונים. במידה והמשתמש הכנס מספר שכבר קיים יש להציג לו את הודעה הבא:

The sequence ID {seq_id} is already in sequence_corpus.

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

כאשר המשתמש הכנס מספר שלא קיים בסיס הנתונים יש לבקש מהמשתמש את הטקסט של הרץ' באופן הבא:

Insert the sequence itself:

לאחר מכן יש להוסיף את הרץ' החדש למילונים `sequences_data`-`inverted_index` ולהדפיס למשתמש הודעה שההוספה הושלמה:

Sequence {seq_id} was successfully added!

שימוש לב שהוספה הרץ' החדש לא משנה את מאגר הרცפים המקורי אלא רק את המילונים `sequences_data`-`inverted_index`.

בחירה באפשרות (3):

חישוב את ערך ה-ISF-AAF בין חומצת האמינו לרץ'.

תחליה נבקש מהמשתמש את המספר המזהה של הרץ' המבוקש:

Insert the sequence ID:

ניתן להניח כי המשתמש הכנס מספר שלם אך אין להניח כי הוא הכנס מספר שקיים בסיס הנתונים. במידה והמשתמש הכנס מספר שלא קיים יש להציג לו את הודעה הבא:

The sequence ID {seq_id} is not in sequence_corpus.

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

כאשר המשתמש הכנס מספר שאינו בסיס הנתונים יש לבקש מהמשתמש את חומצת האמינו שירצה לחשב עבורה את ה-ISF-AAF בין לבן לרץ'.

Insert an amino acid:

אם חומצת האמינו לא נמצאת בסיס הנתונים (לאחר עיבוד מקדים של רץ' הדנ"א), יש להדפיס את הודעה הבא:

The amino acid { amino_acid_after_prepocces } is not in sequence_corpus.

לאחר מכן יש לחזור להודעה הקודמת על מנת לבקש חומצת אmino חדשה. הניחו כי המשתמש יכנס רק תווים אלפביתיים המיצגים חומצות אmino.

כאשר המשתמש הכנס חומצת amino שקיים בסיס הנתונים יש לחשב את ערך ה-ISF-AAF ביןה (לאחר עיבוד מקדים) לבין הרץ' שנבחר ולהדפיס את הודעה הבא:

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

בחירה באפשרות (4):

מחיקה של רצף מבסיס הנתונים.

תחליה נבקש מהמשתמש את המספר המזהה של הרצף המבוקש:

Insert the sequence ID:

ניתן להניח כי המשתמש הכנס מספר שלם אך אין להניח כי הוא הכנס מספר שאינו קיים בבסיס הנתונים. במידה והמשתמש הכנס מספר שלא קיים יש להציג לו את הודעה הבאה:

The sequence ID {seq_id} is not in sequence_corpus.

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

כאשר המשתמש הכנס מספר שאינו קיים בבסיס הנתונים יש למחוק את רצף זה מהמילונים ולהדפיס למשתמש הודעה שהמקרה הושלם:

Sequence {seq_id} was successfully deleted.

שימוש לב שמחיקת הרצף לא משנה את מאגר הרცפים המקורי אלא רק את בסיס הנתונים, כלומר את יש לעדכן רק את המילונים `.sequences_data - inverted_index`

בחירה באפשרות (5):

אפשרות זו תסייע את ריצת הפונקציה.

בחירה באפשרות לא חוקית:

מודפס למשתמש הודעה הבאה:

Invalid choice. Please select a valid option.

לאחר מכן יודפס התפרט מחדש.

שימוש לב: דוגמת הרצה מפורטת נמצאת בקובץ `menu_example.txt`. מומלץ להריץ את הקוד שלכם לפי השלבים בדוגמה ולהשתמש ב-`text compare` על מנת לוודא שההדפסות שלכם תואמות להדפסות שבדוגמה. כל הדפסות צrüכות להיות זהות לפחות שלוון ברמת הרוחים, אותיות קטנות/גדולות וכדומה. אחרת הטסיטים לא יעברו.



בהצלחה!