תרגיל 8-א : Abstract data type ו- file I/O

תאריך פרסום : 29.12.2022

תאריך הגשה : 5.1.2023 בשעה 23:59

מתרגלת אחראית : אסראא נסאסרה

משקל תרגיל : נקודה אחת

בתרגיל זה נתרגל שימוש ב-Abstract Data Types שלמדתם בשיעור.

הנחיות לתרגיל :

- אין לייבא ספריות, ללא יוצא מן הכלל (חיצוניות או מובנות).
- יש להשתמש רק במבני הנתונים המצורפים למטלה והוגדרו במפורש בסעיפים (קובץ LinkedList.py ו-DoublyLinkedList.py).
- יש לממש כל מחלקה בקובץ המתאים לה שיצורף לכם עבור כל מחלקה (Nucleotides.py, DNA.py, RNA.py, FastaFileReader וגם Errors.py)

שאלה 1

דנ״א הוא החומר הגנטי המועבר בירושה מהורה לצאצא. הדנ״א משועתק לרנ״א שממנו מיוצר החלבון, ״המנוע״ שמאפשר לתא לבצע את פעולותיו. דנ״א ורנ״א מיוצגים באמצעות רצפים של נוקלאוטידים, כאשר כל נוקליאוטיד מיוצג ע״י אחת מהאותיות A, T, C, G, או U. רצף דנ״א מורכב מהנוקליאוטידים A,T,G,C. רצף דנ״א משועתק לרנ״א ע״י החלפה של כל נוקלאוטיד T בנוקלאוטיד U.

כדי שרצף דנ״א ישועתק לרצף רנ״א באופן תקין, רצף הדנ״א צריך לעמוד בדרישות הבאות:

1. מורכב משלשות של נוקליאוטידים
2. מתחיל עם השלשה ATG
3. מסתיים עם מופע אחד של אחת השלשות TAA, TAG או TGA, ואף אחת מהשלשות האלה לא מופיעות במקום אחר ברצף. ולבסוף, כל שלשה ברנ״א מקודדת לחומצה אמינית.

עליכם לממש את המחלקה האבסטרקטית Nucleotides, ואת המחלקות DNA ו-RNA היורשות ממנה.
שלד המחלקות יינתן לכם כמו גם מבני הנתונים הנדרשים.

סעיף א:

המחלקה האבסטרקטית **Nucleotides**:

ממשו את המחלקה האבסטרקטית Nucleotides (קובץ Nucleotides.py), המכילה שלושה שדות:

1) שדה סטטי בשם **nucleotides_mass**. שדה זה הינו מטיפוס מילון. מפתחותיו הם הנוקלאוטידים מטיפוס מחרוזת, וכל נוקלאוטיד ממופה למשקלו בטבע. המשקל של כל נוקלאוטיד מצורף בקובץ NucleotideMW.txt. כחלק מהמימוש של המחלקה עליכם לקרוא מהקובץ ולעדכן את המילון. אם הקובץ לא קיים, עליכם לזרוק שגיאה מתאימה.
2) שדה **nucleotides_sequence**. שדה זה הינו מטיפוס LinkedList ומייצג רצף נוקלאוטידים.
3) שדה בשם **nucleotides_number**. שדה זה הינו מטיפוס מילון. מפתחותיו הם הנוקלאוטידים, וכל נוקלאוטיד ממופה לכמות המופעים שלו בשדה הקודם.

ממשו את הבנאי של המחלקה:

```
def __init__(self, sequence):
```

הבנאי מקבל מחרוזת המייצגת רצף נוקלאוטידים.
בדיקת קלט עבור הבנאי של המחלקה Nucleotides:

- אם הקלט אינו מטיפוס מחרוזת עליכם לזרוק שגיאה מטיפוס TypeError ולהדפיס הודעה מתאימה.
- אם המחרוזת הקלט מכילה תווים שאינם ATGCU עליכם לזרוק שגיאה מטיפוס NotNucleotideError ולהדפיס הודעה מתאימה.

סעיף ב:

עליכם לדרוס את האופרטורים __str__ ו-__len__. השיטה __len__ תחזיר את אורכו של רצף הנוקלאוטידים.

סעיף ג:

עליכם לכלול את השיטות האבסטרקטיות:

```
def calculate_mass(self):
```

לרצף של נוקלאוטידים יש מסה הנקבעת לפי רצף הנוקלאוטידים וסוגם (דנ״א או רנ״א). השיטה תחזיר עבור רצף הנוקלאוטידים את המסה הכוללת שלו, וזאת בהתאם לתכונות של הרצף.

```
def mutate(self):
```

שינוי ברצף הנוקלאוטידים נקרא מוטציה. במוטציות, רצף הנוקלאוטידים יכול להתארך (הוספה), להתקצר (מחיקה), ולהתחלף ברצף אחר (החלפה). השיטה תכניס מוטציה לרצף הנוקלאוטידים בהתאם לסוג הרצף: רנ״א או דנ״א.

סעיף ד:

ממשו את המחלקה **DNA** (קובץ **DNA.py**):

המחלקה DNA יורשת מהמחלקה האבסטרקטית Nucleotides. עליכם לממש את השיטות הבאות:

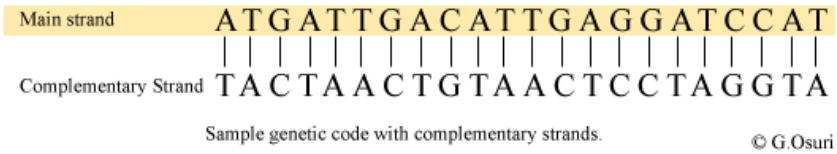
הבנאי מקבל מחרוזת המייצגת רצף נוקלאוטידים.

```
def __init__(self, sequence):
```

על הרצף לעמוד בדרישות של המחלקה Nucleotides ובנוסף להיות מורכב מהנוקלאוטידים ATGC בלבד.

סעיף ה:

בטבע, רצפי הדנ״א מופיעים בזוגות. רצף כפי שהכרנו עד כה נקרא גדיל, והוא מזווג לגדיל שני לפי החוקיות הבאה. כל נוקליאוטיד A בגדיל אחד יהיה מזווג ל-T בגדיל השני, כל נוקליאוטיד T יהיה מזווג ל-A, כל נוקליאוטיד C יהיה מזווג ל-G וכל G יהיה מזווג ל-C. בהתייחס לגדיל הראשון, נקרא לגדיל השני ״הגדיל המשלים״. להלן דוגמא:



משום שגדיל משלים נקרא בסדר הפוך, על השיטה להחזיר את רצף הנוקלאוטידים כרשימה מקושרת דו-כיוונית בסדר הופכי לדוגמה: עבור אובייקט שמחזיק את רצף הנוקלאוטידים (הקריאה היא משמאל לימין) ATGTTTAAA המתודה תחזיר TTTAAACAT

ממשו את השיטה complement המחזירה את הגדיל המשלים כרשימה דו כיוונית.

```
def complement(self):
```

סעיף ו:

המסה של מופע מהמחלקה DNA מוגדרת כמסה של זוג הגדילים המתאימים. ממשו את השיטה:

```
def calculate_mass(self):
```

המחשבת ומחזירה את מסתו הכוללת של הדנ״א.

סעיף ז:

רצף הנוקלאוטידים בדנ״א יכול לעבור מוטציות מכל הסוגים. לדוגמה הרצף ATGATTTAAAATG יכול לייצר את המוטציות הבאות:

- מוטציית הוספה של רצף הנוקלאוטידים CCC במקום 3. אחרי מוטצית הוספה, רצף הנוקלאוטידים יהיה ATGCCCATTTAAAATG.
- מוטצית מחיקה של רצף הנוקלאוטידים AAA במקום 7 ברצף המקורי, תייצר רצף ATGATTTATG.

- מוטצית החלפה שבה יוחלף הרצף במקום 4 עם הרצף GGGG תיצור את הרצף ATGAGGGGAAATG

ממשו את השיטה mutate שמייצרת מוטציה ברצף הנוקלאוטידים. על השיטה לקבל 3 ארגומנטים: מיקום המוטציה (מספר שלם המייצג את המיקום שממנו הרצף מתחיל להשתנות), סוגה (כמחרוזת אחת מתוך שלוש: 'addition', 'replacement', ו-'deletion') והרצף שלה.

def mutate(self, mutation_type, mutation_position, nucleotides_mutation):

השיטה מעדכנת את רצף הנוקלאוטידים לפי סוג המוטציה, עמדתה והרצף שלה.

בדיקות קלט:

- עליכם לבדוק שמיקום המוטציה תואם לאורך של רצף הנוקלאוטידים.
- אם המוטציה היא החלפה או מחיקה עליכם לבדוק שאורך nucleotides תואם לעמדת המוטציה.
- אם המוטציה היא מחיקה, עליכם לבדוק שאכן הרצף nucleotides הנתון קיים במקום של המוטציה. אחרת, יש לזרוק ValueError עם הודעה מתאימה.
- nucleotides_mutation עונה על הגדרה של דנ"א.

בכל אחד מהבדיקות (פרט לאחרונה), יש לזרוק שגיאה InputNotValidError ולהציג הודעה מתאימה כרצונכם.

<u>דוגמאות הרצה:</u>

```
DNA_1 = DNA("ACGGCATTTGGGAAATAATCGC")
print(f"The main  strand is:")
print(DNA_1.__str__())
print(f"The complementry  strand is: {DNA_1.complement()}")
DNA_1.mutate("addition", 0, "ACG")
print(f"The DNA sequence post addition mutation is:{str(DNA_1)}")
DNA_1.mutate("replacement", 3, "AC")
print(f"The DNA sequence post replacement mutation is:{str(DNA_1)}")
DNA_1.mutate("deletion", 3, "AC")
print(f"The DNA sequence post deletion mutation is:{str(DNA_1)}")
print(f"The DNA sequence mass is:{DNA_1.calculate_mass()}")
```

<u>יודפסו למסך התוצאות הבאות:</u>

```
The main  strand is:
ACGGCATTTGGGAAATAATCGC
The complementry  strand is: GCGATTATTTCCCAAATGCCGT
The DNA sequence post addition mutation is:ACGACGGCATTTGGGAAATAATCGC
The DNA sequence post replacement mutation is:ACGACGGCATTTGGGAAATAATCGC
The DNA sequence post deletion mutation is:ACGGGCATTTGGGAAATAATCGC
The DNA sequence mass is:22399.199999999997
```

סעיף ח:

ממשו את המחלקה **RNA (RNA.py)**:

המחלקה RNA יורשת מהמחלקה האבסטרקטית Nucleotides. עליכם לממש את השיטות הבאות:

הבנאי מקבל מחרוזת המייצגת רצף נוקלאוטידים.

```
def __init__(self, sequence):
```

על הרצף לעמוד בדרישות של המחלקה Nucleotides ובנוסף לכך:

- הרצף מורכב משלשות של נוקליאוטידים מסוג AUGC.
- הרצף מתחיל בשלשה "AUG", ומסתיים באחת משלשות הסיום "UAA", "UAG", "UGA".

סעיף ט:

בניגוד לדנ"א, רנ"א מורכב מגדיל אחד (כלומר אין גדיל משלים). ממשו את המתודה:

```
def calculate_mass(self):
```

המתודה מחזירה את מסת הרנ"א. מסת הרנ"א נקבעת מהוספה למסת הגדיל הבודד את המסה של 70 נוקלאוטידים מסוג A המתווספים לקצה הרנ"א (מבלי להיות חלק מרצף הנוקלאוטידים).

סעיף י:

```
def mutate(self, mutation_position, nucleotide_letter):
```

בשונה מדנ"א, רנ"א יכול לעבור מוטצית החלפה בלבד ברצף הנוקלאוטידים. בנוסף לכך, ההחלפה תמיד תתבצע על נוקלאוטיד בודד.

ממשו את השיטה המקבלת שני ארגומנטים: עמדת המוטציה ברצף הנוקלאוטידים ולאיזה נוקלאוטיד הנוקלאוטיד המקורי התחלף. השיטה מעדכנת את רצף הנוקלאוטידים בהתאם.

בדיקות קלט:

עליכם לבדוק אם עמדת המוטציה תואמת לאורך של רצף הנוקלאוטידים וגם ש-nucleotide_letter הוא נוקלאוטיד אחד מארבע AUGC.

סעיף י"א:

מאחר ומוטציה משנה את הרצף של הנוקלאוטידים, השינוי יכול לגרום להפרת תקינות הרנ"א. לדוגמה, החלפה של הנוקלאוטיד במקום הראשון הראשונה ל-U יהפוך את "AUG" ל-"UUG".

```
def validate_sequence(self):
```

ממשו את השיטה שמבצעת ולידציה לתקינות הרצף אחרי יצירת מוטציות. השיטה מחזירה True אם הרצף הוא רנ"א תקין ו-False אחרת.

סעיף י"ב:

```
def RNA_generator(self):
```

ממשו את השיטה המחזירה גנרטור שעובר על רצף הנוקליאוטידים בקבוצות של שלשות.

דוגמאות הרצה:

```
RNA_1 = RNA("AUGGCUUAUUAA")
print(f"The RNA sequence is: {RNA_1}")
print(f"The RNA sequence mass is:{RNA_1.calculate_mass()}")
RNA_1.mutate(5, "A")
print(f"The RNA sequence post replacement mutation is:{str(RNA_1)}")
print(f"The RNA sequence validation output  is :{RNA_1.validate_sequence()}")
RNA_generator = RNA_1.RNA_generator()
print(f"The RNA sequence in triples:")
print(next(RNA_generator))
print(next(RNA_generator))
```

יודפסו למסך התוצאות הבאות:

```
The RNA sequence is: AUGGCUUAUUAA
The RNA sequence mass is:40251.4
The RNA sequence post replacement mutation is:AUGGCAUAUUAA
The RNA sequence validation output  is :True
The RNA sequence in triples:
AUG
GCA
```

שאלה 2:

רצפים של נוקלאוטידים מאוחסנים בקבצים מסוג fasta, שהם קבצי טקסט לאחסון רצפים ביולוגים (עיינו בקישור להסבר). קובץ fasta יכול להכיל מספר רצפי דנ"א ורנ"א. כל רצף כזה מתואר באמצעות מספר שורות. השורה הראשונה מתחילה עם הקידוד ">" ולאחר מכן פרטים כגון, סוג האורגניזם. השורות הבאות מכילות את רצף הנוקלאוטידים, כאשר כל שורה מכילה כ-70 נוקליאוטידים.

ראו דוגמה לקובץ fasta שמצורף לכם (sequence_example.fna).

סעיף א:

ממשו את המחלקה FastaFileReader (בקובץ FastaFileReader.py) שתשמש לקריאה וכתיבה של רצפי נוקלאוטידים מ- ואל קבצי fasta. למחלקה יש שדה יחיד מטיפוס מילון, בשם sequences_dict. מפתחותיו של השדה הן הפרטים של הרצפים (שיופיעו אחרי סימן ה-">" של הרצף) והן ממופות לאובייקטים מסוג DNA או RNA בהתאם לרצף הנוקלאוטידים.

ממשו את הבנאי של המחלקה:

```
def __init__(self, pathway):
```

הבנאי מקבל מחרוזת המייצגת את הנתיב לקובץ שיש לקרוא. הבנאי קורא את הקובץ, מאתחל ומעדכן את השדה sequences_dict בהתאם לתוכן הקובץ.

בדיקת קלט:

אם הקובץ שיש לקרוא אינו קיים, עליכם לזרוק שגיאה מסוג FileNotFoundError ולהציג הודעה מתאימה כרצונכם.

סעיף ב:

def transcript(self, sequence_details):

ממשו את השיטה המקבלת פרטי רצף כמחרוזת ובודקת אם הרצף קיים בשדה sequences_dict. אם הרצף הממופה לקלט הוא מטיפוס רנ"א, יש לכתוב אותו לקובץ sequence_transcription.fna ולהחזיר True. אם הרצף הוא מסוג דנ"א, יש לבדוק אם ניתן להמיר אותו לרנ"א תקין. אם כן, השיטה משעתקת את הרצף לרנ"א, כותבת אותו לקובץ sequence_transcription.fna ומחזירה True. אחרת, השיטה מחזירה False.

סעיף ג:

על מנת לייצר את רצף החומצות אמינו ("החלבון") מרצף נוקלאוטידים יש לקודד כל שלשה ברנ"א לחומצה אמינית המתאימה (לרשותכם מילון עזר בשם RNA_to_Protien).

def translate(self, sequence_details):

ממשו את השיטה המקבלת פרטי רצף כמחרוזת ובודקת אם הרצף קיים בשדה sequences_dict. אם הרצף הממופה לקלט הוא מטיפוס רנ"א השיטה מקודדת אותו לחומצות לקובץ בשם sequence_translation.faa ומחזירה True. אם הרצף קיים אך הוא מסוג דנ"א ואפשר להמיר אותו לרצף רנ"א תקין, השיטה מקודדת את הרצף לחומצות אמינו, לקובץ sequence_translation.faa ומחזירה True. אחרת, יש להחזיר False.

דוגמת הרצה:

```
from FastaFileReader import *
fasta_file = FastaFileReader("sequence_example.fna")
print(f"The fasta file contains the sequences:" + "\n" +
f"{fasta_file.sequences_dict.keys()}")
print(fasta_file.transcript("NC_000011.10:c5249857-5248269 HBG1 [organism=Homo
sapiens] [GeneID=3047] [chromosome=11]"))
print(fasta_file.translate("NC_000011.10:c5249857-5248269 HBG1 [organism=Homo
sapiens] [GeneID=3047] [chromosome=11]"))
```

למסך תתקבל התוצאה הבאה:

```
The fasta file contains the sequences:
dict_keys(['NC_000011.10:c5249857-5248269 HBG1 [organism=Homo sapiens] [GeneID=3047] [chromosome=11]'])
True
True
```

ובנוסף לכך, תקבלו את הקבצים שנכתבו בשתי השיטות transcript ו- translate בדוגמת ההרצה sequence_transcription.fna ו- sequence_translation.faa בהתאמה.