פיתוח תרופות בעידן המידע

תרגיל 3

מועד הגשה: 02.01.2022

- **תרגיל הבית מיועד להגשה עצמאית. כל העתקה שתימצא תגרור פסילה של התרגיל כולו, גם עבור הסטודנט המעתיק, וגם עבור הסטודנט שהתרגיל הועתק ממנו. אנא הימנעו מאי נעימות.**
- **ניתן להגיש תרגיל זה בזוגות.**
- **כל תכנית צריכה להכיל תיעוד מסודר של מטרת התכנית, ושל פקודות חשובות בגוף התוכנית.**
- **יש לרשום בראש כל קובץ פייתון הערה עם שם הסטודנט ות"ז.**
- **אין להשתמש בחומר שטרם נלמד. מתוך הנושאים שנלמדו, מותר להיעזר בכל תיעוד רלוונטי.**
- **על כל ההערות ותיעוד התכנית להכתב בשפה האנגלית בלבד!**
    - מומלץ לתעד בפורמט הרלוונטי לפייתון, כולל שימוש ב-docstrings
- **יש להקפיד על כתיבת קוד תקין ונכון בהתאם לכללים שנלמדו. קוד שאינו תואם את העקרונות שנלמדו יקבל ניקוד חלקי בלבד.**
    - יש לתת שמות משמעותיים למשתנים.
    - ניתן להעזר בפונקציות משנה כרצונכם, בהתאם לעקרונות תכנות נכון ומודולרי.
- **יש להקפיד לעבוד על פי הפורמט המבוקש, רווחים מיותרים ופיסוק שגוי יגררו הורדת ניקוד.**

התרגיל יוגש בקובץ יחיד בשם exercise3_id.py דרך תיבת ההגשה הרלוונטית במודל. ניתן להגיש מספר פעמים עד למועד ההגשה הסופי. הגרסה האחרונה היא שתיבדק.

- לדוגמה exercise3_123456789.py

על התרגיל להיות מסוגל לרוץ באופן הבא:

```
python exercise3_id.py argument1 argument2 …
```

- כדי לקבל משתנים לתוכנית, ניתן להשתמש בתחביר המפורט בקישור הבא:

```
import sys
# print first argument
print(sys.argv[1])
```

הסבר כללי:

מטרת התרגיל לחשוף את הסטודנטים לשימוש במגוון חבילות פייתון נפוצות. התרגיל יכלול למידה עצמאית של חלק מהחבילות, מתוך העקרונות שנלמדו בתרגול.

חלק 1: שדרוג פונקציות ומחלקות קודמות

1. שדרגו את הפונקציה ***transcribe(dna_seq)*** מתוך המחלקה ***Polymerase*** כך שבהינתן קצב טעויות, הפולימראז מכניס מוטציות רנדומליות באופן הבא:
   a. בהינתן קצב טעויות X ורצף באורך N, יש לבחור K בסיסים מתוך הרצף שיעברו מוטציה, כך שהחישוב הוא X * N = K. במידת הצורך יש לעגל כלפי מעלה את מספר הבסיסים.
   b. יש לבחור K מיקומים אקראיים ברצף, ולכל מיקום להגריל מוטציה, כלומר אות מתוך הא"ב של הפולימראז למעט הבסיס הנתון (לדוגמה עבור דנ"א פולימראז והבסיס A, נגריל מתוך {G,C,T}, ועבור רנ"א פולימראז והבסיס A, נגריל מתוך {G,C,U})

2. צרו מחלקה ***MutantCell*** היורשת את המחלקה ***StemCell*** באופן הבא:
   a. תא מוטנטי מכיל דנ"א פולימראז עם קצב טעויות של 1 לעשרים. כמובן שאלו אינם נתונים ביולוגיים ונועדו לצורך התרגיל בלבד.
   b. תא מוטנטי יודע כמה מוטציות הוא צבר בגנום ביחס לתא האב, באופן מצטבר בין דורות. כלומר אם לתא המקורי היו אפס מוטציות ושני רצפים גנומיים, ולאחר מכן הוא הוא התחלק ונוצרה מוטציה בכל רצף, התא החדש צבר שתי מוטציות. אם נחזור על התהליך, בתא הבת החדש ("נכד") ישנן כבר ארבע מוטציות ביחס לתא המקורי.
   באופן התחלתי לתא מוטנטי אין מוטציות (בדור אפס – הדור הראשון).
   c. תא מוטנטי מחזיר בכל מיטוזה תא אחד תקין ושאר התאים מכילים מוטציות (עברו שכפול, בהתאם לקצב ההכפלה של התא וכן קצב המוטציות).
   d. אופרטור הכפל מחזיר אך ורק תאים המכילים מוטציות, כלומר כאלו שעברו שכפול.
   e. שימו לב שהגדרת השכפול בפונקציה ***transcribe(dna_seq)*** הופכת את הרצף, ואילו פה יש להחזיר את כיוון הרצף המקורי (חשבו כיצד להתמודד בפשטות עם העניין).

3. תא עם מעל 10 מוטציות הופך לתא סרטני, המיוצג ע"י המחלקה ***CancerCell*** היורשת מתא מוטנטי. לתא סרטני קצב חלוקה של 10.
   כלומר, באם תא מוטנט מזהה שהגיע לסף והוא מכיל מעל 10 מוטציות, הוא מחזיר תא סרטני במקום תא מוטנטי.

ניתן להעזר בכל חבילת פייתון הרלוונטית לבחירה ורנדומיזציה, לדוגמה המודול random.

חלק 2: מסווג חלבונים ע"פ תבניות PROSITE

בחלק זה אנו נרצה לבנות מסווג המקבל קובץ ורשימת חלבונים, ועבור כל חלבון מדפיס את הדומיינים הקיימים בו.לשם כך נקרא רשימה של תבניות לפי הפורמט של אתר PROSITE מתוך קובץ, ונרצה להפוך אותן לביטוי רגולרי המותאם לחיפוש בפייתון שאיתו נוכל לאפיין סוגי חלבונים. כלומר, נרצה לתרגם את הפורמט של PROSITE ל-RegEx של פייתון. תזכורת – תיאור התבניות הוא לפי המוסכמות הבאות:

x	wildcard, וכל חומצה אמינית יכולה להופיע במקומו
[]	כל אחת מחה"א שבסוגריים יכולה להופיע באותה העמדה
{}	כל ח"א יכולה להופיע באותה העמדה למעט חה"א שבסוגריים

-	הפרדה של אלמנטים בתבנית
(n)	על האלמנט לחזור בדיוק n פעמים
(m,n)	על האלמנט לחזור בין m ל-n פעמים
<	על התבנית להופיע בקצה ה-N טרמינלי
>	על התבנית להופיע בקצה ה-C טרמינלי

דוגמא:

[AC]-x-V-x(4)-{ED}

משמעות התבנית:

[Ala or Cys]-any-Val-any-any-any-any-{any but Glu or Asp}

מקבילה אפשרית בפייתון:

[AC].V.{4}[^ED]

1. כתבו פונקציה בשם ***prosite_to_python(pattern_dict)*** המקבלת מילון של תבניות (מפתחות) הממופות לשמות דומיינים (ערכים) ומחזירה מילון הממפה ביטויים רגולרים מקומפלים בסגנון פייתון לדומיין אותו הם מתארים.
ניתן להעזר בפונקציות עזר כרצונכם.
אופציה א - עבור תבנית שלא עומדת בחוקי הפורמט לעיל יש לזרוק שגיאה מסוג **ValueError** עם התבנית הסוררת. באם טרם נתקלתם בכך, <u>ראו פה בתיעוד כיצד לבצע זאת.</u>
אופציה ב – יש לסנן תבניות שלא עומדות בחוקי הפורמט לעיל (כלומר לא להחזירן).
כלומר, אם ישנה תבנית שלאחר ההמרה המתאימה לא מייצרת ביטוי רגולרי שמציית לחוקי פייתון – מדובר בתבנית לא תקינה ואין להחזירה.
דוגמה:

{" [AC]-x-V-x(4)-{ED}" : "NES"} → { "[AC].V.{4}[^ED]" : "NES"}

2. כתבו פונקציה בשם ***patterns_to_domains(pattern_file)*** המקבלת נתיב (מלא) לקובץ מסוג <u>CSV</u> המתאר תבניות ודומיינים (ראו קובץ דוגמה) ומחזירה מילון הממפה שמות דומיינים לאובייקטים של ביטויים רגולרים מקומפלים בסגנון פייתון המתארים אותו. יש להקפיד על פתיחה נכונה של קבצים. <u>אין להניח תקינות קלט. אם הוא קיים, ניתן להניח שהקובץ הניתן הוא בפורמט CSV (ולא אחר).</u>
3. כתבו מחלקה בשם ***SequenceClassifier*** המכילה מילון תבניות ודומיינים ומאותחלת באופן הבא:
SequenceClassifier(self, pattern_file)
הכניסו למחלקה את הפונקציות מהסעיפים הקודמים בחלק זה כפונקציות פרטיות. <u>אין להניח תקינות קלט.</u>
4. הוסיפו למחלקה מתודה ציבורית בשם ***classify(seq_list, csv_file)*** המקבלת רשימת מחרוזות המייצגות רצפים, ונתיב מלא של קובץ פלט רצוי ומייצרת קובץ CSV המאפיין כל רצף ברשימה הנתונה ומוצא את הדומיינים הידועים שלו (ראו קובץ דוגמה), כאשר שמות הדומיינים מופרדים ע"י ";". כלומר, הפונקציה מקבלת רשימת רצפים, מסווגת כל רצף עבור כל הדומיינים הידועים למחלקה המסווגת, ואת הפלט כותבת לקובץ.
אם קיימות מספר תבניות המתארות את אותו הדומיין, יש להחזיר שמות דומיינים ייחודיים , כלומר כל דומיין פעם אחת בלבד. עבור רצף שאין אף תבנית המתאימה לו ואין בו אף דומיין מהרשימה יש לציין את הערך "NA", זהו ערך מקובל עבור דטא חסר או לא קיים ומשמעותו Not Available.
הפונקציה לא מחזירה דבר.

תוכנית ראשית:

מומלץ להיעזר בתיעוד כאן בנוגע לשימוש ב-__name__ למעוניינים אך אין חובה.

התוכנית הראשית תתפקד כמעין תרבית תאים מוטנטיים בהדגרה. התרבית תתחיל מתא מוטנטי יחיד (MutantCell, הקפידו לרשום בהתאם!) שיתחלק, ובאם יש צורך צאצאיו יהפכו לתא סרטני בהמשך התוכנית. בסיום החלוקות התוכנית תאפיין את כל רצפי החלבונים שנוצרו בתרבית.

לצורך בדיקה אוטומטית, **יש לוודא כי הרנדומיזציה מאותחלת עם גרעין של 1.**

על התוכנית הראשית לקבל קובץ JSON (קלט יחיד) המכיל נתיב לקובץ תבניות ודומיינים (PatternsToDomainsInput), קובץ רצפים גנומיים (GenomicSequencesInput) קובץ פלט (OutputFile), מספר מחזורי חלוקה כולל (MaxCycles, מספר שלם גדול מאפס) ומספר תאים מקסימלי אפשרי (MaxCells, מספר שלם גדול מאחד). ראו קובץ דוגמה.

הפרמטרים לתוכנית יתקבלו מקריאת קובץ ה-JSON שניתן כקלט. ניתן להעזר במודול של פייתון לשם כך.

1. התוכנית תאתחל תא מוטנטי עם רצפים גנומיים מתוך הקובץ הנתון.
2. כל עוד נותרו מחזורי חלוקה מיוטית וגם לא הגענו למספר התאים המקסימלי האפשרי, יש לבצע חלוקה של התא המקורי וצאצאיו (כלומר כל התאים בתרבית).
3. על התוכנית להדפיס את התא הראשוני בתרבית, את מספר התאים שהתקבל בסיום, את מספר החלבונים השונים שנוצר מהם (לא כולל רצפים שלא מקודדים), ואת מספר המוטציות שעבר התא המוטנטי ביותר בתרבית.
4. לבסוף על התוכנית לאפיין את סט כל החלבונים הייחודיים שנוצרו בתרבית: לשייך לכל אחד את כל הדומיינים הרלוונטים, ולייצר קובץ CSV עם האפיון בנתיב הנתון בקובץ הקלט. אין לחזור על רצף חלבון פעמיים. התבניות האפשריות נתונות בקובץ CSV המצויין בקובץ הקלט.

אין להניח תקינות קלט, אין צורך לבדוק פעמיים רצפים הגנומיים וניתן להסתפק בבדיקה של פונקציה פנימית.

דוגמת הרצה 1:

```
python exercise3_123456789.py ex3_inputs.json
```

פלט:

```
Original cell: <MutantCell, 2, 3>
Final number of cells: 200          OR          Final number of cells: 198
Protein repertoire size: 83         number will change with each run
Mutations: 48
```

וכן הקובץ התואם ex3_outputs.csv

דוגמת הרצה 2:

```
python exercise3_123456789.py ex3_bad_inputs.json
```

פלט (כל שגיאה תתאים):

```
AssertionError: MNLQS
```