Zadanie 1.1 Generowanie numerów

Stosując wyrażenie listotwórcze (list comprehension) utwórz listę numerów telefonów (każdy jako str) o postaci +4812300XYYY , gdzie X może być cyfrą od 1 do 5, a Y może być dowolną cyfrą. Oblicz długość listy (powinno być 5000).

Następnie wypisz na ekran dziesięć losowo wybranych numerów z tej listy.

Nie wykorzystywać do autodialerów :-)

Zadanie 1.2 Trójki Pitagorejskie

Wygeneruj trójki (tuple trzyelementowe) liczb całkowitych a b c, które tworzą trójkąt pitagorejski (czyli $a^2 + b^2 = c^2$), ale bez powtórzeń i tylko takich, że a < b < c, dla liczb do podanego limitu (np. 1000). Postaraj się zrobić to za pomocą wyrażenia listotwórczego.

Napisz rozwiązanie w formie funkcji, która zwraca listę takich tupli/krotek.

Spróbuj napisać także wersję z generatorem.

Zadanie 1.3 Klasa dla ułamków

Stwórz klasę, której obiekty reprezentują liczby wymierne w postaci ułamkowej. Klasa powinna implementować **dokładne operacje na ułamkach** (nie poprzez konwersję na float). Podobnie jak robiliśmy z Vectorem, utwórz klasę, która zwraca nowe obiekty jako wyniki swoich operacji.

Nie dopuszczaj do sytuacji, aby w mianowniku znalazło się zero – wyrzucaj wyjątek, a także utrzymuj mianownik dodatni (gdy ktoś próbuje podać ujemny, to wpisz dodatni i zamień znak licznika). Operacje arytmetyczne powinny zwracać wynik w postaci maksymalnie skróconej.

Proponowane elementy klasy:

- tworzenie: `Fraction(nominator=0, denominator=1)`
- atrybuty `nominator`, `denominator` dają wartość licznika i mianownika
- `__str__` , `__repr__`
- operacje matematyczne + - * / za pomocą magicznych metod __add__, __sub__ itd.
- porównania == != < <= > >= działające w oparciu o wartości, ale bez wyliczania floatów,
- `__float__` i `__int__` – konwersja na typ float (wynik dzielenia) i int (część całkowita)
- `__bool__` – zwraca True jeśli licznik nie jest zerem

Metody zwracające wyniki – można je oznaczyć jako `@property`

- `shortened()` – zwraca ułamek o tej samej wartości, skrócony w miarę możliwości (o największy wspólny dzielnik licznika i mianownika)
- `reciprocal()` – zwraca odwrotność ułamka (nowy obiekt)

Wraz z dodawaniem kolejnych metod, testuj je za pomocą pytest.

Pan Tadeusz

Napisz zestaw programów, które analizują wskazany plik tekstowy, np. plik `pan-tadeusz.txt`. W większości przypadków trzeba plik podzielić na słowa, do czego można użyć takich technik jak metoda `split` albo wyrażenia regularne.

Zadanie 1.4 Policz wybrane słowo

Program, który pyta użytkownika o słowo i dla podanego słowa liczy ile razy to słowo występuje w pliku. Przykładowa sesja:

```
Podaj słowo: Tadeusz
Słowo Tadeusz występuje 107 razy.
```

Zadanie 1.5 Policz wszystkie słowa

Napisz program, który czyta plik tekstowy i wylicza oraz wypisuje bez powtórzeń wszystkie słowa występujące w pliku wraz z informacją ile razy dane słowo występuje. Na przykład w ten sposób (drobny wycinek):

```
      Tace ->     1
   Tadeusz ->   107
  Tadeusza ->    54
 Tadeuszek ->     1
```

W podstawowej wersji nie przejmuj się kolejnością wypisywanych informacji.

1.5.1 * Sortowanie

Sprowadzaj wszystkie słowa do małych liter i licz jednolicie bez podziału na małe i duże litery.

Posortuj wypisywane wyniki na dwa sposoby:

a) alfabetycznie

b) według liczby wystąpień.

Zadanie 1.6 Odmiana słów

Pod adresem **zil.ipipan.waw.pl/PoliMorf** znajdują się zasoby dotyczące odmiany wyrazów w języku polskim. Plik `PoliMorf-0.6.7.tab` (lub nowsza wersja) zawiera relację między słowem a jego formą podstawową. W pliku tym pierwsza kolumna zawiera słowo być może odmienione, a druga kolumna zawiera jego formę bazową.

Wczytaj dane z tego pliku do odpowiednich struktur Pythona (list / set / dict) tak, aby dało się wyszukiwać odmiany dla form podstawowych, a formy podstawowe dla odmian i aby działało to w przyzwoitym czasie (wczytywanie może chwilę potrwać, ale znalezienie odmiany, gdy dane są już wczytane, powinno być szybkie – raczej sekunda niż minuta :))

Najporządniej byłoby utworzyć klasę, której obiekt zawiera odpowiednio przygotowaną „bazę wiedzy” nt. odmiany słów. Obiekt tej klasy tworzy się na podstawie pliku takiego jak `PoliMorf-0.6.7.tab` . Obiekt powinien umożliwiać znalezienie formy podstawowej (lub kilku kandydatów) dla podanego słowa odmienionego oraz odczytanie listy słów odmienionych na podstawie formy bazowej.

Przykładowe użycie tej klasy mogłoby wyglądać tak (ale szczegóły nie muszą wyglądać identycznie):

```
>>> baza = BazaOdmiany.wczytaj("PoliMorf-0.6.7.tab")
>>> baza.znajdzFormyPodstawowe("Tadeusza")
[’Tadeusz’]
>>> baza.znajdzOdmiany("Tadeusz")
[’Tadeusz’, ’Tadeuszowi’, ’Tadeusza’…...
```

Korzystając z tej klasy napisz program, który w pętli odczytuje od użytkownika kolejne słowa i dla podanego słowa wyświetla jego formę podstawową lub informację, że nie znaleziono.

Ten sam wyraz może być formą odmienioną dla kilku form bazowych, np. „dam” może pochodzić od „dać”, ale również od „dama”. Można to uwzględnić we własnym modelu (wersja zaawansowana) lub upraszczając przyjąć, że wybieracie tylko jedną formę podstawową dla danej formy odmienionej.

*Zadanie 1.7 * Pan Tadeusz z odmianą wyrazów*

Zrealizuj zadanie z liczeniem ilości poszczególnych słów w Panu Tadeuszu, ale sprowadzaj wszystkie słowa do ich formy podstawowej (albo pierwszej formy podstawowej znalezionej w pliku, jeśli jest kilka kandydatur). Przykładowo zamiast całej serii odmienionych Tadeuszów

```
Tadeusz          →  107
Tadeusza         →   54
Tadeuszem        →    2
Tadeuszowi       →    5
Tadeuszu         →    6
```

powinien być jeden wpis

```
Tadeusz → 174
```

Zadanie 1.8 Dane z pliku sprzedaz.csv

(jednak przykłady dalej dla tego pliku, bez wprowadzania innych)

Opierając się o jedną z wersji obiektowych (zwykła klasa, dataclass lub namedtuple), napisz program, który obliczy sumę wartości transakcji dla każdego miasta.

Zadanie 1.9 Generator fib

Napisz generator nieskończony, który zwraca kolejne liczby Fibonacciego.

Napisz program, który wykorzystując ten generator wypisuje tysiąc kolejnych liczb Fibonacciego.