Kurs Pythona v0.2

Lista zadaniowa II

Wojciech Adamiec

23 grudnia 2022

Spis treści

1 Cenzura

Napisz procedurę `censor(text)`, która zwraca wejściowy napis z gwiazdkami w miejscu cyfr.

Dla przykładu wywołanie `censor` dla tekstu `"Mój numer telefonu to 213 731 299"` powinno zwrócić napis `"Mój numer telefonu to *** *** ***"`.

Wskazówka: W pythonie można porównywać ze sobą napisy. W tym zadaniu szczególnie przydatne może się okazać porównanie postaci:

```
letter <= "9"
```

2 Koperta

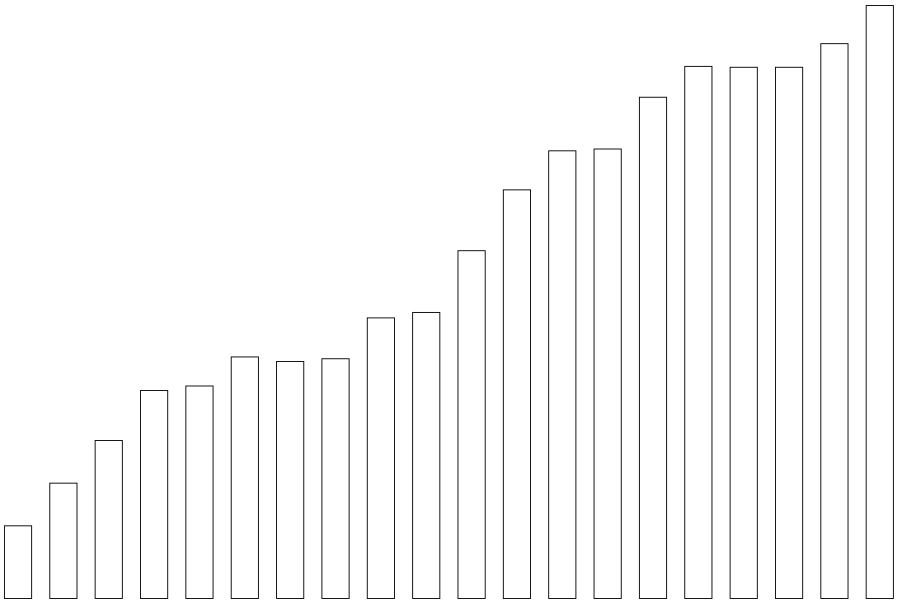
Napisz procedurę `envelope(n)`, która rysuje na terminalu kopertę w formacie takim jak niżej:

```
*********
**     **
* *   * *
*  * *  *
*   *   *
*  * *  *
* *   * *
**     **
*********
```

Koperta powinna się zmieścić w $2 \cdot n + 1$ wierszach tekstu. Powinna być kwadratowa (przy założeniu, że znaki też są kwadratowe).

3 Wykres inflacji

Za pomocą modułu **turtle** narysuj wykres prognozowanej inflacji przypominający ten poniżej. Wykorzystaj w tym celu bibliotekę **random**, aby wykres sprawiał wrażenie naturalnego. Nie musisz opierać się na żadnych prawdziwych danych, skoro sam prezes NBP tak nie robi.



4 Liczby diabelskie

Liczba pierwsza to liczba naturalna większa od 1, która bez reszty dzieli się tylko przez 1 i przez samą siebie.

Liczbę nazwiemy diabelską, jeżeli jej zapis dziesiętny zawiera 3 szóstki pod rząd.

W pliku do zadania znajduje się program, który wypisuje wszystkie diabelskie liczby pierwsze z zakresu $1 - 100000$ wraz z informacją ile takich liczb jest. Program zawiera dwie osobne i niezależne procedury `is_prime(n)` oraz `is_diabolic(n)`, które powinny sprawdzać odpowiednio czy liczba jest pierwsza oraz czy jest diabelska (zwracając odpowiednio `True` lub `False`).

Uzupełnij brakującą implementację procedur `is_prime(n)` oraz `is_diabolic(n)`, tak aby działały zgodnie ze swoim przeznaczeniem.

Wskazówka: Spróbuj zobaczyć jak w pythonie działa fragment kodu postaci:

```
string in another_string
```

5 Szyfr Cezara

Swetoniusz w swoim dziele *De vita Caesarum* wskazuje na ciekawy sposób zabezpieczania rozkazów wydawanych przez Juliusza Cezara podczas jego podbojów. Przed spisaniem rozkazu na papier każda litera zastępowana jest przez literę występującą w alfabecie 3 miejsca na prawo (następną literą dla ostatniej litery alfabetu jest pierwsza litera alfabetu). W ten sposób zaszyfrowaną wiadomość transportowano do właściwego oficera, a następnie deszyfrowano - każdą literę zastępowano literą występującą w alfabecie 3 miejsca na lewo.

Szyfr Cezara można oczywiście uogólnić na przesuwanie liter o k miejsc w prawo w przypadku szyfrowania oraz o k miejsc w lewo w przypadku deszyfrowania.

Twoim zadaniem jest napisanie dwóch procedur `cipher(text, shift)` oraz `decipher(text, shift)`, które będą zwracać odpowiednio zaszyfrowany i zdeszyfrowany tekst podany w argumencie (dla szyfru Cezara z przesunięciem `shift`). Po napisaniu procedur należy je przetestować na kilku przykładach.

W zadaniu nie rozpatrujemy polskich znaków - trzymamy się tylko angielskiego alfabetu. Rozróżniamy wielkie i małe litery. Spacje pozostawiamy bez zmian. Przykładowe wywołanie procedur powinno zwrócić odpowiednio takie wartości:

```
cipher("Imperator", 1) -> "Jnqfsbups"
decipher("Jnqfsbups", 1) -> "Imperator"
```

Wskazówka: To zadanie może być bardzo proste, jeśli w implementacji wykorzysta się przydatne, wbudowane w pythona funkcje. Warto w pierwszej kolejności zaznajomić się z *tablicą kodów ASCII*, którą można znaleźć np. Tutaj. Następnie warto przetestować jakie wartości zwrócą wywołania funkcji:

```
"T".isupper()
"t".isupper()
  ord("B")
  chr(77)
```