Zajęcia nr 9
Temat: Zabawy z tekstem

Cele:

W trakcie zajęć uczeń udoskonali umiejętności wykorzystania:

- Operacji wejścia/wyjścia
- Instrukcji warunkowej
- Instrukcji pętli
- Definiowania funkcji
- Korzystania z generatorów liczb pseudolosowych

Dodatkowo uczeń:

- Pozna nowy typ danych: Napisy
- Nauczy się odwoływać do poszczególnych znaków tekstu
- Skorzysta z wbudowanych metod i funkcji operujących na napisach: len(), index(), upper(), count(), lower(), swapcase(), replace()
- Napisze program sprawdzający czy dany napis jest palindromem
- Pozna jak zbudowany jest numer Pesel i napisze program, który na jego podstawie określa płeć.

Zadanie na rozgrzewkę:

Napisz program, który pobierze od użytkownika dwa dowolne napisy i utworzy z nich jeden.

Zadanie 1

Napisz program, który pobierze od użytkownika dowolny tekst, a następnie wypisze każdy znak w nowej linii.

Zadanie 2

Napisz program, który pobierze od użytkownika dowolny tekst, a następnie poda z ilu wyrazów ten tekst się składa. Zakładamy, że wyrazy oddzielone są od siebie znakiem pojedynczej spacji.

Zadanie 3

Palindrom to wyrażenie brzmiące tak samo czytane od lewej do prawej i od prawej do lewej.
Przykładami palindromu są: kajak, Anna, *Kobyła ma mały bok*.
Napisz program, który sprawdzi czy podany przez użytkownika tekst jest palindromem.

Wskazówka:
Należy w tekście zamienić wszystkie litery na małe (lub duże), a także pozbyć się spacji. Zakładamy, że w tekście nie występują znaki interpunkcyjne.

Przykładowe palindromy na stronie: https://pl.wikipedia.org/wiki/Palindrom#Polski

Zadanie 3

Numerem Pesel potocznie określa się 11-cyfrowy numer w sposób jednoznacznie identyfikujący każdego obywatela RP.
Znaczenie poszczególnych cyfr w numerze Pesel można znaleźć:
https://pl.wikipedia.org/wiki/PESEL#Numer_PESEL
Napisz program, który na podstawie numeru Pesel podanego przez użytkownika określi jego płeć.
Wskazówka:
Płeć określa 10-ta cyfra numeru Pesel. Jeżeli cyfra jest parzysta to pesel należy do kobiety, a jeżeli nieparzysta to do mężczyzny.

Zadanie 4

Poprawność numeru PESEL można obliczyć według podanego poniżej algorytmu:
1· a1 + 3· a2 + 7· a3 + 9· a4 + 1· a5 + 3· a6 + 7· a7 + 9· a8 + 1· a9 + 3· a10 + a11,
gdzie ai – kolejne cyfry numeru PESEL.
Jeżeli powyższy wynik jest liczbą podzielną przez 10, to numer PESEL jest poprawny, w przeciwnym razie jest błędny
Napisz funkcję sprawdzającą, czy podany jako argument numer PESEL jest poprawny

Zadanie trudniejsze

Najdłuższy dobry podciąg

Masz dany ciąg znaków s o długości n, który zawiera tylko k początkowych liter alfabetu łacińskiego (wielkich).

Podciągiem ciągu s jest taki ciąg znaków, który można otrzymać z s przez usunięcie z niego pewnych znaków bez zmiany kolejności pozostałych znaków. Na przykład „ADE” oraz „BD” są podciągami „ABCDE”, natomiast „DEA” nim nie jest.

Podciąg nazywamy *dobrym*, jeśli ilość wystąpień każdej z k początkowych liter alfabetu jest taka sama.

Należy znaleźć najdłuższy dobry podciąg ciągu s.

Dane wejściowe

Pierwszy wiersz danych wejściowych zawiera dodatnie liczby naturalne n ($1 \leqslant n \leqslant 10^5$) oraz k ($1 \leqslant k \leqslant 26$).

Liczby w wierszu oddzielone są pojedynczymi odstępami.

Drugi wiersz zawiera ciąg znaków s o długości n zawierający tylko wielkie litery od ‘A’ do k-tej litery w alfabecie łacińskim.

Wynik programu

Program powinien wypisać jedną liczbę całkowitą – długość najdłuższego dobrego podciągu s.

Przykład

Dla danych wejściowych

```
9 3
ACAABCCAB
```

prawidłowym wynikiem jest:

```
6
```

Podciąg „ACBCAB” („**A**C**AAB**C**C**A**B**”) zawiera po tyle samo wystąpień liter ‘A’, ‘B’ oraz ‘C’. Podciąg „CAB” ma tę samą własność, ale nie jest najdłuższy.

Wskazówki do rozwiązania zadania:

Podciąg „dobry” to taki podciąg, w którym występują tyle samo razy wszystkie litery od ‘A’ do k-tej litery alfabetu włącznie. Wystarczy zatem zliczyć, ile razy każda litera występuje we wprowadzonym ciągu. Jeśli na przykład $k = 3$, a litera ‘A’ występuje 4 razy, litera ‘B’ – 5 razy, zaś litera ‘C’ występuje 2 razy, to wtedy decyduje *najmniejsza* z tych ilości, czyli 2. Ponieważ mamy 3 litery alfabetu, więc najdłuższy dobry podciąg ma długość $3 \cdot 2 = 6$ liter. Zauważmy, że nie precyzujemy, jak on wygląda, ponieważ mamy podać tylko jego długość.

Lista wybranych funkcji działających na danych tekstowych

`capitalize`()
Zwraca kopię napisu z pierwszym znakiem zmienionym na wielką literę.

`endswith`(*przyrostek*[, *początek*[, *koniec*]])
Zwraca wynik sprawdzenia, czy napis jest zakończony napisem *przyrostek*. Przy wystąpieniu argumentu *początek*, sprawdzenie rozpoczyna się od tego znaku. Przy wystąpieniu argumentu *koniec* porównanie zakończy się na tym znaku.

`find`(*napis*[, *początek*[, *koniec*]])
Zwraca najniższy indeks wystąpienia napisu *napis*, takiego, aby *napis* był zawarty w przedziale [*początek*, *koniec*). Opcjonalne argumenty *początek* i *koniec* są interpretowane tak samo, jak w operacji wycinania. Zwraca `-1` jeśli *napis* nie został znaleziony.

`index`(*napis*[, *początek*[, *koniec*]])
Działanie podobne do `find()`, ale wywołuje wyjątek `ValueError` jeśli napis nie zostanie znaleziony.

`isalnum`()
Zwraca wynik sprawdzenia, czy wszystkie znaki napisu są znakami alfanumerycznymi i napis składa się przynajmniej z jednego znaku.

`isalpha`()
Zwraca wynik sprawdzenia, czy wszystkie znaki napisu są literami i napis składa się przynajmniej z jednego znaku.

`isdigit`()

Zwraca wynik sprawdzenia, czy wszystkie znaki napisu są cyframi.

`islower`()
Zwraca wynik sprawdzenia, czy wszystkie litery napisu są małymi literami i napis składa się przynajmniej z jednego znaku.

`isspace`()
Zwraca wynik sprawdzenia, czy wszystkie znaki napisu są białymi znakami i napis składa się przynajmniej z jednego znaku.

`istitle`()
Zwraca wynik sprawdzenia, czy napis ma strukturę tytułu, to znaczy każdy wyraz napisu musi zaczynać się wielką literą i składać wyłącznie z małych liter lub znaków nieliterowych.

`isupper`()
Zwraca wynik sprawdzenia, czy wszystkie litery napisu są wielkimi literami i napis składa się przynajmniej z jednego znaku.

`join`(*sek*)
Zwraca napis stanowiący połączenie napisów wchodzących w skład sekwencji *sek*. Separator pomiędzy elementami stanowi napis, którego metodę wywołujemy.

`lower`()
Zwraca kopię napisu zamienionego na małe litery.

`replace`(*stary*, *nowy*[, *ilość*])
Zwraca kopię napisu z wszystkimi wystąpieniami napisu *stary* zastąpionymi przez *nowy*. Jeśli zostanie podany argument *ilość*, zostanie zastąpiona tylko pierwsza *ilość* wystąpień.

`split`([*separator* [, *ilość*]])
Zwraca listę słów wchodzących w skład napisu z wykorzystaniem napisu *separator* jako separatora wyrazów. Jeśli jest podana *ilość* wykonana jest najwyżej *ilość* podziałów. Jeśli *separator* nie jest podany, lub jest równy `None` separatorem jest jakikolwiek biały znak.

`swapcase`()
Zwraca kopię napisu z małymi literami zamienionymi na wielkie i vice versa.

`startswith`(*prefix*[, *start*[, *end*]])

Zwraca wynik sprawdzenia, czy napis zaczyna się napisem *prefix*. Przy wystąpieniu argumentu *start*, sprawdzenie rozpoczyna się od tego znaku. Przy wystąpieniu argumentu *end* porównanie zakończy się na tym znaku.

`upper`()

Zwraca kopię napisu z wszystkimi literami zamienionymi na wielkie litery