3 października 2025

Kurs rozszerzony języka Python

Lista 1.

Każde zadanie jest warte 2 punkty. Na pracowni do oceny należy przedstawić trzy zadania.

Zadanie 1.

W Polsce podatek od towarów i usług (VAT) liczy się na dwa sposoby: w przypadku faktur sumuje się wartości netto i mnoży się przez 23%, a w przypadku kas fiskalnych i paragonów liczy się VAT 23% od każdej pozycji osobno i na końcu się sumuje. Zaprogramuj w Pythonie dwie funkcje zwracające podatek VAT dla zadanej listy zakupów

- `vat_faktura(lista)`
- `vat_paragon(lista)`

gdzie `lista` jest listą liczb reprezentujących cenę netto. Zazwyczaj oczekujemy, że poniższy program wypisze `True`

```
print(vat_faktura(zakupy) == vat_paragon(zakupy))
```

gdzie `zakupy` to lista liczb typu *float*.

Poszukaj takiej listy `zakupy`, dla której powyższy program wypisze `False` i umieść tę listę w pliku źródłowym. Wykonaj eksperyment polegający na zamianie liczb *float* w liście `zakupy` na ich odpowiedniki klasy `Decimal` i sprawdzeniu, czy nadal program w ramce wypisuje `False`.

Zadanie 2.

Napisz funkcję `is_palindrom(text)`, która zwraca **True** jeśli argument jest palindromem. Zakładamy, że `text` może być zarówno pojedynczym słowem (np. *rotor* czy *oko*), ale też dłuższym wyrażeniem: *"Kobyła ma mały bok."*; w takim przypadku ignorujemy znaki przestankowe, spacje i wielkość liter.

Sprawdź, czy funkcja poprawnie działa dla tekstów obcojęzycznych:

```
is_palindrom("Eine güldne, gute Tugend: Lüge nie!")
is_palindrom("Míč omočím.")
```

Zadanie 3.

21 listopada br. będzie Światowy Dzień Tabliczki Mnożenia. Z tej oakzji zaprogramuj funkcję `tabliczka(x1, x2, y1, y2, d)`, która wypisze na ekran tabliczkę mnożenia dla liczb $[x_1, x_1 + d, x_1 + 2 * d, \ldots, x_2] \times [y_1, y_1 + d, y_1 + 2 * d, \ldots, y_2]$, gdzie x_1, x_2, y_1, y_2 i d są liczbami typu *float*.

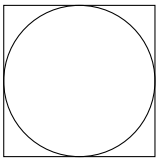
Na przykład `tabliczka(3.0, 5.0, 2.0, 4.0, 1.0)` powinno wypisać

```
     3.0  4.0  5.0
2.0  6.0  8.0 10.0
3.0  9.0 12.0 15.0
4.0 12.0 16.0 20.0
```

Zwróć uwagę, by szerokości kolumn były jednakowe oraz odpowiednie do liczby cyfr w liczbach. Zakładamy, że x_1, x_2, y_1, y_2 mogą być też liczbami ujemnymi.

Zadanie 4.

Liczbę π (a właściwie jej kolejne przybliżenia) można wyliczać na wiele sposobów. Jednym z nich jest rzucanie wielokrotne strzałką do kwadratowej tarczy z wpisanym okręgiem:



i policzenie *liczby trafień wewnątrz okręgu (ltwo)* oraz *całkowitej liczby trafień w tarczę (cltwt)*. Te dwie liczby pozwolą nam wyliczyć przybliżenie liczby π:

$$\pi \approx \frac{4 * \text{ltwo}}{\text{cltwt}}$$

Zaprogramuj symulację takiego rzucania lotką w tarczę, losując współrzędne punktu na tarczy. Program powinien wypisywać kolejne uzyskane przybliżenia π po każdym rzucie. Program może się zakończyć po wykonaniu zadanej liczby losowań bądź gdy różnica między otrzymanym przybliżeniem a wartością `math.pi` będzie mniejsza od zadanej wartości.

Zadanie 5.

Zaprogramuj funkcję, która dla zadanej listy stringów `lista_slow` zwróci najdłuższy wspólny prefiks dla przynajmniej trzech elementów `lista_slow`. Na przykład[1]

```
common_prefix(["Cyprian", "cyberotoman", "cynik", "ceniąc",  "czule"])
```

powinno zwrócić

```
"cy"
```

Wielkość liter nie ma dla nas znaczenia.

Marcin Młotkowski

[1] Inspiracja: *Cyberiada*, Stanisław Lem