Programowanie I R

Zadania – seria 7.

Programowanie obiektowe – metody specjalne.

Zadanie 1. `velocity` – Relatywistyczne składanie prędkości.

Niech $\beta = v/c$ będzie prędkością wyrażoną w jednostkach prędkości światła c. Zgodnie z relatywistycznym prawem składania prędkości w przypadku jednowymiarowym, jeżeli β_2 jest prędkością ciała drugiego względem ciała pierwszego, zaś β_1 – prędkością ciała pierwszego w pewnym układzie odniesienia, to prędkość ciała drugiego w tym układzie odniesienia wynosi

$$\beta = \frac{\beta_1 + \beta_2}{1 + \beta_1\beta_2}.$$

Napisz klasę `Velocity` reprezentującą prędkość w ruchu jednowymiarowym. Zaimplementuj:

- konstruktor, który można wywołać bez argumentów lub z jednym argumentem; w pierwszym przypadku tworzony obiekt powinien być inicjalizowany wartością zero, w drugim zaś – wartością podaną jako argument,
- metodę `gamma` zwracającą wartość czynnika

$$\gamma = \frac{1}{\sqrt{1 - \beta^2}},$$

- operator dodawania `+`, działający zgodnie z relatywistycznym prawem składania prędkości,
- operator `+=`, działający zgodnie z relatywistycznym prawem składania prędkości,
- metody `__str__` i `__repr__`, formatujące opis instancji klasy.

Korzystając z tej klasy napisz program `velocity`, który wczytuje ze standardowego wejścia dwie prędkości wyrażone w jednostkach prędkości światła i wypisuje na standardowe wyjście ich relatywistyczną sumę, również wyrażoną w jednostkach prędkości światła, oraz odpowiadający jej czynnik γ.

Przykładowe wykonanie

Wywołanie:

```
python3 velocity.py
```

Wejście:

```
0.5 0.7
```

Wyjście:

```
beta = 0.888889
gamma = 2.18282
```

Zadanie 2. `rational` – Liczby wymierne.

Napisz klasę `RationalNumber` reprezentującą liczbę wymierną. Klasa powinna przechowywać liczby całkowite p i q, których iloraz jest równy reprezentowanej liczbie wymiernej, przy czym p i q powinny być względnie pierwsze oraz $q > 0$. Zaimplementuj:

- konstruktor, który można wywołać bez argumentów lub z jednym albo dwoma argumentami całkowitymi; w pierwszym przypadku tworzony obiekt powinien być inicjalizowany wartością 0, w drugim – zadaną liczbą całkowitą, zaś w trzecim – ilorazem argumentów konstruktora; konstruktor wywołany z dwoma argumentami powinien działać poprawnie również wtedy, gdy wartości jego argumentów nie są względnie pierwsze lub gdy drugi argument jest ujemny,
- metody `numerator` i `denominator`, zwracające odpowiednio licznik p i mianownik q reprezentowanej liczby wymiernej,
- operator konwersji do typu `float`,
- jednoargumentowy operator zmiany znaku `-`,
- operator porównania `<`,
- operatory dodawania `+`, odejmowania `-`, mnożenia `*` i dzielenia `/`,
- operatory `+=`, `-=`, `*=` i `/=`,
- metody `__repr__` i `__str__`; druga z nich powinna przedstawiać liczbę w postaci `p/q`, np. `-5/7`.

Korzystając z tej klasy napisz program `rational`, który wczytuje ze standardowego wejścia dwie liczby wymierne, a następnie wypisuje na standardowe wyjście w kolejnych liniach:

- podane liczby w postaci dziesiętnej,
- liczby przeciwne do podanych,
- podane liczby w kolejności niemalejącej,
- sumę i iloczyn podanych liczb.

Przykładowe wykonanie
Wywołanie:
```
python3 rational.py
```
Wejście:
```
2/4 1/-3
```
Wyjście:
```
0.5 -0.3333333
-1/2 1/3
-1/3 1/2
1/6 -1/6
```

Zadanie 3. `pimc` – Liniowy generator kongruentny i metody Monte Carlo.

Napisz klasę `LCG` implementującą generator liczb pseudolosowych określany jako *liniowy generator kongruentny*, stanowiący ciąg dodatnich liczb całkowitych $(x_n)_{n=0}^{\infty}$ zadany wzorem rekurencyjnym

$$x_{n+1} = (ax_n + b) \bmod m$$

dla pewnej wartości początkowej $x_0 \in \{0, 1, \ldots, m-1\} \subset \mathbb{Z}$, nazywanej *ziarnem*, gdzie m, $a \in \mathbb{N}$ oraz $b \in \mathbb{Z}_+ \cup \{0\}$ są parametrami spełniającymi a, $b < m$.

Zaimplementuj:

- konstruktor przyjmujący jako argumenty wartości ziarna oraz parametrów generatora,
- operator wywołania `()`, zwracający nowo wygenerowaną liczbę pseudolosową,
- publiczną metodę `min`, zwracającą najmniejszą możliwą do wygenerowania liczbę (czyli 1, gdy $m = 0$, lub 0, gdy $m \neq 0$),
- publiczną metodę `max`, zwracającą największą możliwą do wygenerowania liczbę (czyli $m - 1$).

Korzystając z tej klasy napisz program `pimc`, którego zadaniem jest obliczenie przybliżonej wartości liczby π z wykorzystaniem prostej metody Monte Carlo. Program ma przyjmować jako argument wywołania dodatnią liczbę całkowitą N. Po uruchomieniu program powinien wylosować N punktów leżących w kwadracie $[0, 1]^2$ korzystając z instancji klasy `LCG` oraz określić liczbę K tych spośród wylosowanych punktów, które leżą w ćwiartce koła $\left\{ (x, y) \in \mathbb{R}^2 \;\middle|\; x^2 + y^2 \leqslant 1,\, x \geqslant 0,\, y \geqslant 0 \right\}$. Następnie program ma obliczyć wartość liczby π na podstawie przybliżonej zależności

$$\pi \approx \frac{4K}{N}$$

i wypisać ją na standardowe wyjście.

Opracowanie: Bartłomiej Zglinicki.