Generowanie liczb pseudolosowych w języku Python

- Wprowadzenie
- Przeczytaj
- Symulacja interaktywna
- Sprawdź się
- Dla nauczyciela

Źródło: Riho Kroll, domena publiczna.

Liczby losowe oraz pseudolosowe mają wiele zastosowań praktycznych, poznaliśmy je w e-materiale Liczby pseudolosowe. Programy służące do generowania takich liczb mogą być wykorzystywane w firmach zajmujących się grami losowymi.

Z tego e-materiału dowiesz się, jak przy pomocy języka Python wygenerować liczby pseudolosowe.

Ciekawi cię, jak wyglądają implementacje algorytmu generowania liczb pseudolosowych w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Generowanie liczb pseudolosowych w języku C++,
- Generowanie liczb pseudolosowych w języku Java.

Więcej zadań? Sięgnij do Generowanie liczb pseudolosowych – zadania maturalne.

Twoje cele

- Wyjaśnisz, jak generować całkowite i rzeczywiste liczby pseudolosowe w języku Python.
- Scharakteryzujesz sposoby, dzięki którym losowania będą powtarzalne.
- Stworzysz proste symulacje rzeczywistych losowań.

Przeczytaj

Praca z modułem `random`

W języku Python przewidziano moduł `random` generujący liczby pseudolosowe. Rozpoczynając pracę z nim, podobnie jak w przypadku innych modułów, musimy rozpocząć od importu całego modułu lub funkcji, których chcemy użyć. Zacznijmy od omówienia prostego przykładu:

```
import random                # Import modułu random

random.seed(123)             # Ustaw ziarno na 123
random.random()              # Wygeneruj liczbę
```

Pierwszą czynnością, którą wykonaliśmy, jest ustawienie ziarna na stałą wartość 123, używając funkcji `seed()` z modułu `random`. Następnie wygenerowaliśmy losową liczbę za pomocą metody `random()`. Bez względu na to, ile razy uruchomimy program, zawsze otrzymamy tę samą liczbę. Jeżeli pozbędziemy się ziarna i pozostawimy domyślne, ustawiane na podstawie czasu systemowego, wynik będzie za każdym razem inny. Łatwo jednak zauważyć, że nawet z domyślnym ziarnem zawsze otrzymujemy liczbę rzeczywistą bliską zeru. To dlatego, że funkcja `random()` zwraca wartość z przedziału:

$$0 \leq n < 1$$

Oczywiście czasami będziemy potrzebować liczb z innego zakresu. Dla większości przypadków wystarczą nam funkcje dostępne w module `random`. Zatem gdybyśmy chcieli wylosować liczbę z następującego zakresu:

$$30 \leq n < 70$$

możemy wykorzystać metodę `uniform()`. Funkcja ta przyjmuje dwa parametry: pierwszy określa początek, a drugi koniec zakresu, z którego losowane będą liczby rzeczywiste.

```
import random                # Import modułu random

random.uniform(30.0, 70.0)   # Wygeneruj liczbę
```

> **Ważne!**
>
> Funkcja `uniform()` może zawierać w sobie górny zakres (w tym wypadku 70), w zależności od wyniku zaokrąglenia następującego równania:
>
> $$a + (b - a) \cdot r$$
>
> gdzie:
> a – początek poszukiwanego zakresu,
> b – koniec poszukiwanego zakresu,
> r – liczba wygenerowana przez funkcję `random()`.

Gdybyśmy jednak chcieli wylosować liczbę całkowitą, możemy wykorzystać funkcję `randrange()`. Metoda ta może być wywołana na trzy różne sposoby:

- z pojedynczym parametrem oznaczającym koniec przedziału, z wyłączeniem podanej wartości, tak jak w przypadku poprzednich funkcji;
- z dwoma parametrami oznaczającymi początek i koniec przedziału, ponownie z przedziałem niedomkniętym z prawej strony;
- z trzema parametrami, gdzie pierwsze dwa oznaczają początek i koniec przedziału, a trzeci – skok pomiędzy nimi; tu również przedział nie jest domknięty.

Poniższy kod wygeneruje nam zatem najpierw liczbę całkowitą z zakresu:

$$0 \leq n < 10$$

Następnie wygenerujemy liczbę z przedziału:

$$20 \leq n \leq 80$$

A na koniec wylosujemy parzystą liczbę całkowitą z powyższego zakresu

```
import random                # Import modułu random

                             # Generowanie liczb całkowitych:
random.randrange(10)         # Od 0 do 9 włącznie
random.randrange(20, 81)     # Od 20 do 80 włącznie
random.randrange(20, 81, 2)  # Parzystych, od 20 do 80 włącznie
```

Symulacje

Oprócz generowania liczb przy użyciu modułu `random` możemy również dokonywać innych operacji. Jedną z nich jest mieszanie elementów w zbiorze takim jak na przykład tablica. Służy do tego metoda `shuffle()` przyjmująca tylko jeden argument, którym jest ten zbiór.

```
import random                # Import modułu random

cards = ['A', 'K', 'Q', 'J', '10']
random.shuffle(cards)        # Przetasuj zbiór kart
```

Kolejną wartą uwagi funkcją jest `choice()`, która wybierze pojedynczy element ze zbioru. Podobnie jak `shuffle()`, metoda `choice()` przyjmuje tylko jeden argument.

```
import random                # Import modułu random

cards = ['A', 'K', 'Q', 'J', '10']
random.choice(cards)         # Wybierz losową kartę
```

Do wylosowania n elementów bez powtórzeń służy metoda `sample()`. Przyjmuje ona dwa argumenty – zbiór oraz liczbę elementów do wylosowania.

```
import random                # Import modułu random

cards = ['A', 'K', 'Q', 'J', '10']
random.sample(cards, 3)      # Wybierz trzy losowe karty,
```

Ostatnią omawianą funkcją jest `choices()`. Metoda ta wybierze zadaną liczbę elementów ze zbioru, z powtórzeniami. Przyjmuje trzy argumenty: pierwszym z nich jest zbiór elementów, drugi argument zawiera zbiór wag poszczególnych elementów, ostatni parametr to liczba losowanych elementów.

```
import random                # Import modułu random

cards = ['A', 'K', 'Q', 'J', '10']
chance = [2, 3, 4, 4, 4]

# Wybierz osiem kart, z powtórzeniami.
# Niestety zgubiliśmy dwa asy i jednego króla,
```

```
# więc one mają mniejszą szansę żeby zostać wylosowane
random.choices(cards, chance, 8)
```

> **Ważne!**
>
> Metoda `choices()` została dodana dopiero w Pythonie 3.6, więc jeśli używasz starszej wersji, nie możesz się nią posłużyć.

Słownik

> **inicjalizacja**
>
> w programowaniu: nadanie wartości początkowych
> **ziarno**
>
> (ang. *seed*) wartość inicjująca generator liczb pseudolosowych

Symulacja interaktywna

Polecenie 1

Wykorzystaj symulację i sprawdź, jak rozkładają się wartości w przypadku różnych losowań z powtórzeniami. Postaw hipotezę i zapisz ją. Następnie wybierz kilka liczb z przedziału od 1 do 50 i przeprowadź symulację.

Zasób interaktywny dostępny pod adresem https://zpe.gov.pl/a/DvCjMkK1s

Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Hipoteza:

Polecenie 2

Przygotuj notatkę zawierającą twoje wnioski i obserwacje związane z symulacją interaktywną.

Polecenie 3

Poszukaj informacji o tym, w jakich innych dziedzinach wykorzystuje się liczby pseudolosowe.

Opisz przykłady ich wykorzystania.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

W jednej z popularnych gier losowych gracz wybiera sześć liczb (bez powtórzeń) z zakresu od 1 do 49. Stwórz program, który przeprowadzi przykładowe losowanie i wyświetli wylosowane liczby.

Specyfikacja problemu:

Dane:

- `ile` – liczba naturalna; liczba losowanych liczb
- `max` – liczba naturalna; największa wylosowana liczba (włącznie)

Wynik:

Program prezentuje sześć wylosowanych liczb z zakresu od 1 do 49 włącznie.

Przykładowe wyjście:

```
[23,37,12,4,18,30]
```

Twoje zadania

1. Utwórz funkcję `losuj` i zwróć sześć wylosowanych liczb
2. Wylosowane liczby powinny znajdować się w przedziale od 1 do 49

Ćwiczenie 2

Zmodyfikuj napisany przez siebie program tak, by użytkownik wprowadzał wybrane sześć liczb, a następnie otrzymywał informację o tym, ile z nich udało mu się trafić. Przetestuj jego działanie dla liczb $13,\ 11,\ 20,\ 12,\ 9,\ 15$.

Specyfikacja:

Dane:

- `wybrane` – lista składająca się z sześciu liczb naturalnych dodatnich (1-49 włącznie)

Wynik:

Program na standardowym wyjściu prezentuje liczbę trafionych liczb.

Przykładowe wyjście:

```
Liczba trafień: 0
```

Twoje zadania

1. Utwórz funkcję `losuj(wybrane)`
2. Zwróć liczbę trafień

Ćwiczenie 3

Zmodyfikuj program jeszcze raz: wylosujmy teraz tylko trzy liczby z przedziału od 1 do 25. Niech losowania będą przeprowadzane tak długo, aż użytkownik trafi każdą z liczb. Wyświetl liczbę przeprowadzonych losowań. Przetestuj program dla liczb $13,\ 11,\ 20$.

Specyfikacja:

Dane:

- `wybrane` – lista składająca się z trzech liczb naturalnych (1-25 włącznie)

Wynik:

Program wyświetla informację o trafieniach oraz liczbie losowań.

Przykładowe wyjście:

```
Liczba trafień: 3
Liczba losowań: 1253
```

Twoje zadania

1. Utwórz funkcję `losuj(wybrane)`
2. Zwróć liczbę trafień oraz liczbę losowań

Dla nauczyciela

Autor: Zespół Gromar.eu

Przedmiot: informatyka

Temat: Generowanie liczb pseudolosowych w języku Python

Grupa docelowa: III etap edukacyjny, liceum, technikum

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi, w tym: znajomość zasad działania urządzeń cyfrowych i sieci komputerowych oraz wykonywania obliczeń i programów.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

1) planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).

3) wyróżnia w problemie podproblemy i charakteryzuje: metodę połowienia, stosuje podejście zachłanne i rekurencję;

4) porównuje działanie różnych algorytmów dla wybranego problemu, analizuje algorytmy na podstawie ich gotowych implementacji;

5) sprawdza poprawność działania algorytmów dla przykładowych danych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

1) w zależności od problemu rozwiązuje go, stosując metodę wstępującą lub zstępującą;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

2) do realizacji rozwiązań problemów prawidłowo dobiera środowiska informatyczne, aplikacje oraz zasoby, wykorzystuje również elementy robotyki;

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

1) projektuje i tworzy rozbudowane programy w procesie rozwiązywania problemów, wykorzystuje w programach dobrane do algorytmów struktury danych, w tym struktury dynamiczne i korzysta z dostępnych bibliotek dla tych struktur;

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

Kompetencje kluczowe:

- kompetencje w zakresie rozumienia i tworzenia informacji,
- kompetencje w zakresie wielojęzyczności,
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii,
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się,
- kompetencje cyfrowe.

Cele operacyjne:

Uczeń:

- generuje całkowite i rzeczywiste liczby pseudolosowe;
- stosuje sposoby, dzięki którym losowania będą powtarzalne;
- tworzy proste symulacje rzeczywistych losowań.

Strategie nauczania:

- konstruktywizm.

Metody i techniki nauczania:

- rozmowa kierowana;
- analiza kodu;
- programowanie.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami i dostępem do internetu, słuchawki;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda.

Przebieg zajęć:

Faza wstępna

1. Na poprzednich zajęciach grupa uczniów otrzymuje zadanie – stworzenie prezentacji multimedialnej dotyczącej gier losowych i generatorów liczb pseudolosowych.
2. Nauczyciel podaje temat i cele zajęć oraz wspólnie z uczniami ustala kryteria sukcesu.
3. Przedstawienie prezentacji przygotowanej przez uczniów i omówienie modułu `random`. Reszta klasy ocenia pracę kolegów i dyskutuje.

Faza realizacyjna

1. Wspólne omówienie i zapisanie poszczególnych kroków w procesie losowania liczb rzeczywistych i całkowitych (funkcje `seed()`, `random()`, `uniform()`, `randrange()`).
2. Uczniowie wykonują ćwiczenia sprawdzające zdobyte umiejętności.
3. Praca w czterech grupach – analiza informacji dotyczących funkcji `shuffle()`, `choice()`, `sample()` i `choices()`. Przedstawiciele grup prezentują poszczególne funkcje i ich możliwości, a pozostali uczniowie w razie potrzeby uzupełniają i weryfikują informacje.
4. Praca z multimedium bazowym (symulacją). Uczniowie dzielą się na zespoły. Każdy z nich stawia hipotezę dotyczącą tego, jak rozkładają się wartości w przypadku różnych losowań z powtórzeniami. Następnie przeprowadzane są symulacje i uczniowie

zapisują wnioski. Po zakończeniu pracy przedstawiciele grup prezentują je na forum klasy. W kolejnym kroku uczniowie dyskutują o wnioskach poszczególnych grup, weryfikują je i uzgadniają wspólne stanowisko.
5. Uczniowie piszą algorytmy (zestaw ćwiczeń), które od razu są sprawdzane. Wspólnie z nauczycielem omawiają swoje rozwiązania.

Faza podsumowująca

1. Na koniec zajęć nauczyciel prosi uczniów o odpowiedź na pytania o to, czego się nauczyli, co zrozumieli, co ich zaskoczyło, co było dla nich trudne, a co – łatwe.

Dwa ostatnie pytania oceniają trudność omawianego zagadnienia; dzięki nim uczeń dokonuje samooceny swoich wiadomości i umiejętności.

2. Nauczyciel wskazuje mocne i słabe strony pracy zespołów, udzielając im tym samym informacji zwrotnej.

Materiały pomocnicze:

Dokumentacja dla języka Python 3.

Wskazówki metodyczne opisujące różne zastosowania multimedium:

Nauczyciel może wykorzystać multimedium bazowe (symulację) do zadania z elementami grywalizacji. Uczniowie muszą przygotować symulację dla podanych liczb na czas, rywalizując ze sobą. Nauczyciel nagradza zwycięską drużynę.