



Modelowanie ruchów Browna w języku Python

- [Wprowadzenie](#)
- [Przeczytaj](#)
- [Film samouczek](#)
- [Sprawdź się](#)
- [Dla nauczyciela](#)



Modelowanie ruchów Browna w języku Python

Źródło: Adrien Converse, domena publiczna.

W 1827 roku biolog Robert Brown obserwował przez mikroskop pyłki kwiatowe w zawiesinie wodnej. Prawie 80 lat później matematyczny opis zaproponowali niezależnie Albert Einstein oraz polski fizyk Marian Smoluchowski.

W e-materiale [Modelowanie ruchów Browna](#) poznaliśmy już podstawowe informacje na ich temat.

Przeprowadzimy symulację zjawiska, którego model jest opisany matematycznie. W trakcie symulacji wykorzystamy [liczby pseudolosowe](#).

Implementację modelowania ruchów Browna w innych językach programowania znajdziesz w e-materiałach:

- [Modelowanie ruchów Browna w języku C++](#),
- [Modelowanie ruchów Browna w języku Java](#).

Więcej zadań? Przejdź do [Modelowanie ruchów Browna – zadania maturalne](#).

Twoje cele

- Przeanalizujesz sposób modelowania zjawisk fizycznych o charakterze losowym.
- Zaimplementujesz algorytm modelowania ruchów Browna w języku Python.

- Rozwiążesz problemy programistyczne, wykorzystując wiedzę dotyczącą modelowania ruchów Browna.

Przeczytaj

Przykład 1

Napiszmy program, który poda kolejne pary współrzędnych – w kartezjańskim układzie współrzędnych – cząstki poruszającej się zgodnie z ruchami Browna. Przyjmujemy następujące założenia:

- umieszczamy cząsteczkę w początku układu współrzędnych $(0, 0)$,
- kierunek ruchu wyznaczamy przez losowy kąt φ , z przedziału $(0, 2\pi)$
- cząsteczka będzie przemieszczała się skokami na stałą odległość r ,
- symulacja ma działać przez n rund,
- kolejne współrzędne cząsteczki (x, y) wyliczymy ze wzorów:

$$\varphi \in (0, 2\pi)$$

$$X_n = X_{n-1} + (r \cdot \cos(\varphi))$$

$$Y_n = Y_{n-1} + (r \cdot \sin(\varphi))$$

Ważne!

W naszym programie zakładamy, że odległość r wynosi 1, a symulacja ma się wykonywać przez 13 rund.

Skorzystamy z biblioteki `math`, aby zaimportować niezbędne funkcje matematyczne, które wykorzystamy do wykonania obliczeń.

```
1 from math import sqrt, sin, cos, pi
2 from random import randint
3
4 liczba_rund = 13
5 r = 1
6 brown_X = [0]
7 brown_Y = [0]
8
9 for runda in range(1, liczba_rund + 1):
10     # wylosowany kąt musi być podany w radianach; wymagają tego f
11     fi = float(randint(0, 360)) * pi / 180
12     stare_x = brown_X[runda-1]
13     nowe_x = stare_x + (r * cos(fi))
14     stare_y = brown_Y[runda-1]
```

```

15     nowe_y = stare_y + (r * sin(fi))
16     brown_X.append(nowe_x)
17     brown_Y.append(nowe_y)
18
19 print(f"Wartości X = {brown_X}")
20 print(f"Wartości Y = {brown_Y}")

```

Efekt działania programu są dwie listy, każda składająca się z $n + 1$ elementów. Zawierają one kartezjańskie współrzędne kolejnych punktów, w których znalazła się cząsteczka. Pierwsza współrzędna (0, 0) jest pozycją startową cząstki. Sprawdźmy zatem wartości wygenerowane przez podany algorytm (w każdym przypadku będą różne z uwagi na losowość fi).

```

1 Wartości X = [0, 0.6427876096865394, 0.05500235739406634,
2               0.08990185409656742, 0.8212555557157382,
3               0.9257840189833916, 1.1677059145830595,
4               1.6677059145830597, 2.5157540107394856,
5               2.3078423199217264, 2.909657343073775,
6               2.8399008693296497, 2.9617702127347973,
7               2.9268707160322966]
8 Wartości Y = [0, 0.766044443118978, 1.5750614374939254,
9               2.5744522645130212, 1.892453904450523,
10              2.8869757998187966, 3.857271526094793,
11              4.723296929879232, 5.2532161941124365,
12              6.231363794846242, 7.0299993048935345,
13              8.027563355153358, 9.02010950679468,
14              10.019500333813776]

```

Aby zobrazować ruch cząsteczki, wykorzystamy wartości zapisane w tablicach, (X_n, Y_n) w postaci par oraz funkcje z biblioteki [matplotlib](#). Napišemy program, który pokaże na ekranie trasę, jaką przebyła cząstka.

Ważne!

Aby zainstalować opisywaną bibliotekę, musimy wydać w systemowym terminalu polecenie:

```
1 pip install matplotlib
```

```
1 # wczytujemy moduł i ustawiamy alias, który będziemy używać w kod
```

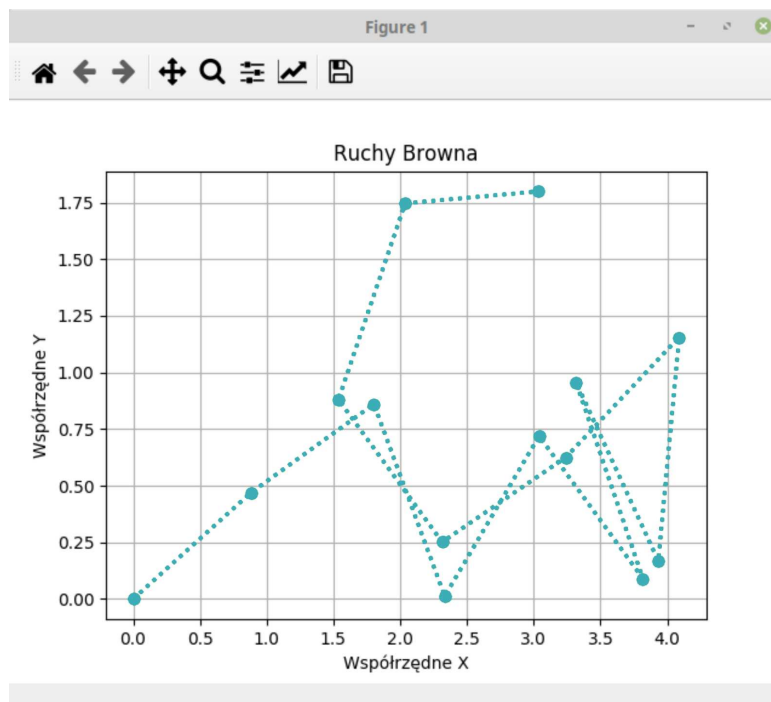


```

2 import matplotlib.pyplot as plt
3
4 # wywołujemy metodę plot z parametrami:
5     # brown_X - tablica współrzędnych x
6     # brown_Y - tablica współrzędnych y
7     # "o:" - określenie rodzaju punktów i linii
8     # color="green" - określenie koloru
9     # linewidth=2 - określenie szerokości linii
10 plt.plot(brown_X, brown_Y, "o:", color="green", linewidth=2)
11
12 # dodajemy opis współrzędnych oraz tytuł
13 plt.xlabel("Współrzędne X")
14 plt.ylabel("Współrzędne Y")
15 plt.title("Ruchy Browna")
16
17 # włączamy widoczność pomocniczej siatki współrzędnych
18 plt.grid(True)
19
20 # uruchamiamy okno wykresu
21 plt.show()

```

Oto wynik działania programu dla przykładowego zestawu liczb:



W taki właśnie sposób możemy symulować ruchy Browna. Zmieniając wartość zmiennej `r`, wpływamy na obliczanie kolejnych współrzędnych (odległość, na jaką przemieści się cząstka).

Słownik

importowanie biblioteki

operacja, która pozwala programowi korzystać z zewnętrznych, dodatkowych modułów lub funkcji

matplotlib

biblioteka służąca do przedstawienia obrazów złożonych z punktów o współrzędnych `x` oraz `y` (wykresów, histogramów, rozkładów itp.); moduł `matplotlib` nie jest dostępny w standardowej instalacji języka Python; należy zainstalować go, korzystając z mechanizmu `pip`

Film samouczek

Problem 1

Napisz w języku Python program symulujący losowy ruch trzech cząstek.

Cząstki są umieszczone w dwuwymiarowym, kartezjańskim układzie współrzędnych. Każda z nich początkowo znajduje się w punkcie $(0, 0)$. Symulacja jest podzielona na tzw. rundy. Podczas pojedynczej rundy każda z cząstek musi przemieścić się o jedno pole wzdłuż osi OX (prawo-lewo) oraz o jedno pole wzdłuż osi OY (górze-dół). Na jednym polu na raz może znajdować się dowolnie dużo cząstek. Ruch cząstki jest wyznaczany na podstawie wylosowanej liczby rzeczywistej.

Dla uproszczenia możemy wyobrazić sobie, że podczas pojedynczej rundy cząstki dwukrotnie rzucamy monetą. Pierwszy rzut oznacza przemieszczenie wzdłuż osi x . Orzeł sprawia, że cząstka przesunie się o jedno pole w lewo, reszka – w prawo. Drugi rzut decyduje o ruchu wzdłuż osi y . Orzeł przesuwa o jedno pole w górę – reszka w dół.

Specyfikacja:

Dane:

- rundy – liczba naturalna
- x , y – zmienne typu *int*, definiujące pozycję cząsteczki; liczby całkowite

Wynik:

Program na wyjściu standardowym wydrukuję dla każdej rundy: numer rundy, numer cząstki oraz współrzędną cząsteczki w kartezjańskim układzie współrzędnych (x , y).

Przykładowy wynik na wyjściu standardowym:

```
1 Runda: 1 cząstka: 1 x: 1 y: -1
2 Runda: 1 cząstka: 2 x: -1 y: 1
3 Runda: 1 cząstka: 3 x: -1 y: -1
4 Runda: 2 cząstka: 1 x: 0 y: -2
5 Runda: 2 cząstka: 2 x: -2 y: 2
6 Runda: 2 cząstka: 3 x: -2 y: -2
7 Runda: 3 cząstka: 1 x: 1 y: -1
```

```
8 Runda: 3 cząstka: 2 x: -3 y: 1
9 Runda: 3 cząstka: 3 x: -3 y: -1
10 ...
```

Polecenie 1

Zapoznaj się z filmem. Pokazano w nim rozwiązanie wykorzystujące spacer losowy.



Modelowanie ruchów Brownna

Symulacja ruchów Browna w języku Python






Film dostępny pod adresem <https://zpe.gov.pl/a/D12Ku35iX>

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

Film nawiązujący do treści materiału

Sprawdź się

Pokaż ćwiczenia:   

Ćwiczenie 1



Pewna drużyna piłkarska rozpoczęła grę w punkcie o współrzędnych opisanych dwiema liczbami całkowitymi (x, y) . Zawodnicy podają sobie piłkę w przypadkowy sposób – każde podanie można opisać jako poruszenie piłki o odległość r pod losowym kątem. Sprawdź, czy po wykonaniu wszystkich ruchów (których kierunki opisano w tablicy `losowe_katy` za pomocą kątów o wartościach w mierze łukowej) piłkarzom uda się przerzucić piłkę na drugą stronę boiska, czyli przekroczyć oś OY (współrzędna x końcowego położenia piłki będzie miała przeciwny znak w stosunku do początkowej wartości). Jeśli tak, wypisz napis TAK, a jeśli nie, wypisz napis NIE.

Przyjmij początkowe położenie piłki w punkcie o współrzędnych $(2, 2)$. Piłkarze podają sobie piłkę na odległość $r = 1$.

Ważne!

Więcej informacji na temat miary łukowej kąta znajdziesz w e-materiale z matematyki [Związek między miarą łukową a stopniową](#).

Specyfikacja:

Dane:

- x, y – zmienne typu *int*; początkowe położenie piłki; liczby całkowite
- r – zmienna typu *int*; odległość, o jaką przesuwa się piłka; liczba naturalna
- `losowe_katy` – tablica liczb rzeczywistych zawierająca kąty podań piłki o wartościach podanych w mierze łukowej

Wynik:

Na wyjściu standardowym program drukuje słowo TAK, jeżeli współrzędna x piłki będzie miała przeciwny znak w stosunku do początkowej wartości, lub NIE w przeciwnym wypadku.

Ćwiczenie 2



Zdefiniuj funkcję `testowa(n)`, która dla cząsteczki poruszającej się zgodnie z ruchami Browna wyznaczy współrzędne (x, y) jej n kolejnych położeń, zaczynając od punktu (x, y) ; wektor przesunięcia ma wartość r . Funkcja powinna zwrócić 2 elementową krotkę (tuple) składającą się z n -elementowych list. Pierwszą wartością powinna być lista kolejnych współrzędnych x cząstki, drugą – lista kolejnych współrzędnych y . Wykorzystaj generator liczb pseudolosowych.

Swoje rozwiązanie przetestuj dla następujących danych:

```
1 n = 30
2 x = -1.456
3 y = 1.834
4 r = 0.0485
```

Specyfikacja:

Dane:

- n – liczba naturalna
- x, y – liczby rzeczywiste; początkowe położenie cząstki
- r – wektor przesunięcia; liczba rzeczywista

Wynik:

Funkcja powinna zwrócić obiekt typu `tuple` (krotkę) o następującej postaci:

- $(punkty_X, punkty_Y)$
- `punkty_X` – lista obiektów typu `float`, współrzędnych x wygenerowanych punktów
- `punkty_Y` – lista obiektów typu `float`, współrzędnych y wygenerowanych punktów

Zdefiniowanie funkcji `testowa()`.

Sprawdzenie, czy funkcja zwraca typ `tuple`.

Sprawdzenie, czy funkcja dla argumentu `ile_krok` wynoszącego 31 zwraca odpowiednie wyniki.

```
1. def testowa(ile_krok):
2.
   # kolejne dwie linie muszą pozostać w kodzie (pozwolą one sprawdzić, czy Twój program działa poprawnie)
3.     from random import seed
4.     seed(4095037966)
5.     # tutaj wpisz własny kod
6.     return None
7.
8. # przykładowe wywołanie
```

Ćwiczenie 3



Pewien rolnik ma gospodarstwo zlokalizowane w trzeciej ćwiartce kartezjańskiego układu współrzędnych (współrzędne na obu osiach OX i OY są ujemne). Z gospodarstwa uciekła krowa i obecnie znajduje się w punkcie (x, y) . W każdej minucie krowa porusza się o losowo wybrany kąt. Za każdym razem pokonuje odległość r . Napisz program, który odpowie na pytanie, po ilu minutach krowa znajdzie się na terenie gospodarstwa. Użyj podanego ziarna generatora liczb pseudolosowych.

Swoje rozwiązanie przetestuj dla zadanego generatora ziarna (421), a także $x = 3$, $y = 3$ oraz $r = 1$.

Specyfikacja:

Dane:

- x, y – początkowe położenie krowy; liczby rzeczywiste
- r – odległość, o jaką przesuwa się krowa; liczba rzeczywista

Wynik:

- t – zmienna typu *int*; liczba naturalna reprezentująca czas potrzebny krowie, aby znalazła się na terenie gospodarstwa

Dla nauczyciela

Autor: Adam Jurkiewicz

Przedmiot: Informatyka

Temat: Modelowanie ruchów Browna w języku Python

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:

Zakres podstawowy i rozszerzony

Cele kształcenia – wymagania ogólne

- 1) Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.
- 2) Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Zakres rozszerzony

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

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

III. I + II. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3. objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

8) metodę Monte Carlo (obliczanie przybliżonej wartości liczby π , symulacja ruchów Browna),

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Przeanalizujesz sposób modelowania zjawisk fizycznych o charakterze losowym.
- Zaimplementujesz algorytm modelowania ruchów Browna w języku Python.
- Rozwiążesz problemy programistyczne, wykorzystując wiedzę dotyczącą modelowania ruchów Browna.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimediu i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

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

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Modelowanie ruchów Browna w języku Python”. Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj”.

Faza wstępna:

1. Nauczyciel inicjuje rozmowę wprowadzającą w temat lekcji. Przedstawia cele zajęć oraz kryteria sukcesu.
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj”.

2. **Praca z multimediami.** Nauczyciel wyświetla zawartość sekcji „Film samouczek”. Uczniowie w parach rozwiązują problem 1, a następnie porównują swój kod z przedstawionym w filmie.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 1 z sekcji „Sprawdź się”, a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.
4. Nauczyciel dzieli uczniów na grupy czteroosobowe. Uczniowie wykonują ćwiczenie nr 3 z sekcji „Sprawdź się”, a następnie każda grupa wyznacza jedną osobę, którą wymienia się z inną grupą. W nowych zespołach uczniowie porównują swój kod i wybierają najbardziej efektywne rozwiązanie.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie”. W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się”.
2. Na koniec zajęć z programowania w Pythonie nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak...”.

Praca domowa:

1. Uczniowie piszą program, który wygeneruje obraz symulacji ruchów Browna w n krokach. Częstki przemieszczają się o wektor r . Wartość n i r podaje użytkownik.
2. Uczniowie wykonują ćwiczenie 2 z sekcji „Sprawdź się”.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Treści w sekcji „Film samouczek” można wykorzystać jako materiał służący powtórzeniu materiału.
- E-materiał „Modelowanie ruchów Browna w języku Python” można wykorzystać jako korelację z lekcją fizyki.