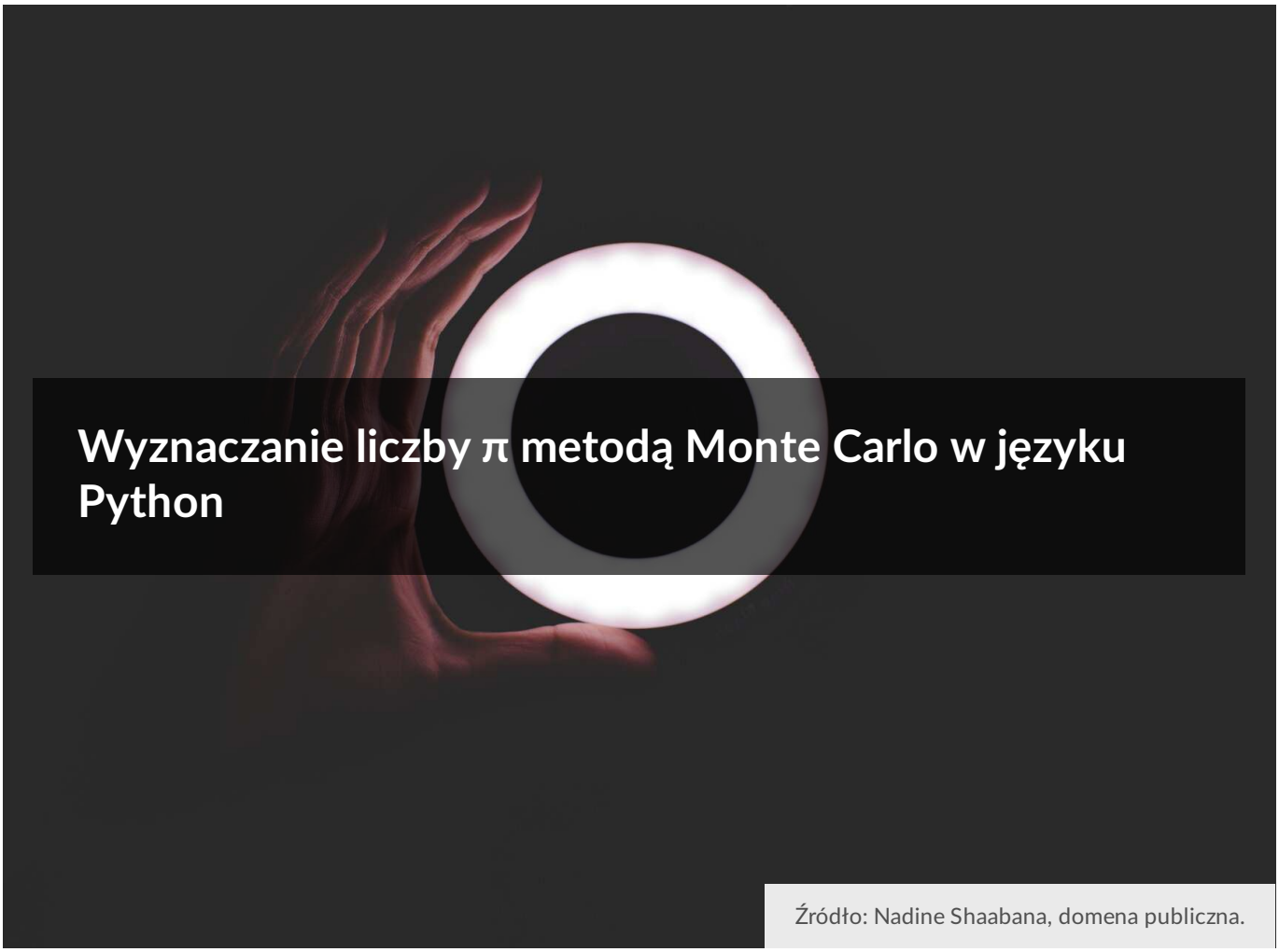
Wyznaczanie liczby π metodą Monte Carlo w języku Python



Źródło: Nadine Shaabana, domena publiczna.

Istnieje wiele rodzajów obliczeń matematycznych, które bardzo trudno wykonać przy użyciu metod analitycznych. Czasami okazuje się to wręcz niemożliwe. Za przykład niech posłużą obliczenia całek oznaczonych pewnych funkcji, wyznaczanie pól powierzchni figur o skomplikowanych kształtach czy też działania na liczbach niewymiernych. W takich sytuacjach sięga się po numeryczne techniki obliczeniowe. Jedną z nich jest metoda Monte Carlo, którą zajmiemy się w tym e-materiale.

Czy potrafisz wskazać inne sytuacje, w których sięga się po numeryczne techniki obliczeniowe?

Podstawowe informacje na temat wyznaczania liczby π metodą Monte Carlo znajdziesz w e-materiale: Wyznaczanie liczby π metodą Monte Carlo.

W tym e-materiale poznamy specyfikę wyznaczania liczby π metodą Monte Carlo w języku Python.

Jeśli chcesz się dowiedzieć, jak to zagadnienie wygląda w przypadku innych języków programowania, sięgnij do e-materiałów:

- Wyznaczanie liczby π metodą Monte Carlo w języku Java,
- Wyznaczanie liczby π metodą Monte Carlo w języku C++.

Twoje cele

- Scharakteryzujesz sposoby generowania liczb losowych w języku Python.
- Zastosujesz numeryczną metodę obliczeniową opartą na rachunku prawdopodobieństwa.
- Skorzystasz z biblioteki graficznej `matplotlib` przy rozwiązywaniu zadań.
- Przeanalizujesz, jak losuje się punkty w obrębie danej figury geometrycznej.
- Zaimplementujesz algorytm obliczania wartości liczby π metodą Monte Carlo w języku Python.

Film samouczek

Polecenie 1

Przeanalizuj kolejne kroki obliczenia wartości liczby π w przypadku losowania 5 punktów. Przedstawione w prezentacji działania prowadzone są tylko dla ćwiartki koła.

1

Wyznaczając przybliżoną wartość liczby π z wykorzystaniem metody Monte Carlo, przyjmiemy następujące założenia:

- przygotowany kod wykona początkowo 5 losowań, których liczbę zapiszemy w zmiennej `liczba_losowan`,
- współrzędne x i y punktów wylosujemy za pomocą funkcji `random()`,
- zbadamy przynależność punktów tylko dla jednaj ćwiartki koła, w której wartości x i y są dodatnie,
- liczbę punktów, które znajdą się w kole, zapiszemy w zmiennej `punkty_w_kole`,
- przybliżoną wartość liczby π wyliczymy według wzoru: `4 * punkty_w_kole / liczba_losowan`.

2

Zapoznajmy się z kodem, który zrealizuje przyjęte założenia. Zacznijmy od

zaimportowania funkcji `random()` oraz przypisania początkowych wartości zmiennym:

```
from random import random as r
punkty_w_kole = 0
liczba_losowan = 5

```

3

Współrzędne losujemy w pętli. Funkcja `random()` wywoływana za pomocą aliasu `r()` zwraca wartości z zakresu `[0.0, 1.0)`:

```

for i in range(liczba_losowan):
    x, y = r(), r()

```

4

Po wylosowaniu współrzędnych wyliczamy sumę ich kwadratów i sprawdzamy, czy punkt przynależy do koła o promieniu równym 1. Jeżeli tak, zwiększamy wartość zmiennej zliczającej liczbę punktów w kole.

```

    suma_kwadratow = x**2 + y**2
    if (suma_kwadratow <= 1):
        punkty_w_kole += 1

```

Na koniec wyliczamy i wypisujemy przybliżoną wartość liczby π:

5

```
liczba_pi =  4 * punkty_w_kole / liczba_losowan
print("Wartość pi:", liczba_pi)
```

6

W kompletnym już programie dodamy jeszcze instrukcje wypisujące dodatkowe komunikaty pokazujące działanie programu. Następnie uruchamiamy program kilka razy.

```
from random import random as r
punkty_w_kole = 0
liczba_losowan = 5

for i in range(liczba_losowan):
    x, y = r(), r()
    print(x, y)
    suma_kwadratow = x**2 + y**2
    print("Suma kwadratów:", suma_kwadratow, end="")
    if (suma_kwadratow <= 1):
        punkty_w_kole += 1
        print(". Punkt w kole.\n")
    else:
        print(". Punkt poza kołem.\n")

print("Punkty w/poza: ", punkty_w_kole, "/",
```

```
    liczba_losowan-
    punkty_w_kole)
17  liczba_pi =  4 *
    punkty_w_kole /
    liczba_losowan
18  print("Wartość pi: ",
    liczba_pi)
```

7

Przeanalizujmy komunikaty wypisane podczas przykładowego wykonania programu:

```
1  0.7951033766058001
   0.7526236980421152
2  Suma kwadratów:
   1.1986318103445337. Punkt
   poza kołem.
3
4  0.5745213895605898
   0.8946001764872201
5  Suma kwadratów:
   1.1303843028335963. Punkt
   poza kołem.
6
7  0.8212956068893785
   0.5227443874653328
8  Suma kwadratów:
   0.9477881685222986. Punkt
   w kole.
9
10 0.5492280649213201
   0.5526232532470974
11 Suma kwadratów:
   0.6070439273266234. Punkt
   w kole.
12
13 0.8154823038203755
   0.5658629040120995
14 Suma kwadratów:
   0.9852122139811936. Punkt
   w kole.
15
```

```
Punkty w/poza: 3/2
Wartość pi: 2.4

```

Widzimy, że wylosowane zostały 3 punkty należące do koła i 2 nienależące. Przybliżona wartość liczby π wyliczana jest jako wartość wyrażenia: `4 * 3 / 5` – co daje bardzo słabe przybliżenie równe 2.4.

8

Przeanalizujmy komunikaty wypisane podczas innego przykładowego wykonania programu:

```
0.0984299300621656
0.788307911291243
Suma kwadratów:
0.6311178141364051. Punkt
w kole.

0.45802919567765066
0.3395699077027242
Suma kwadratów:
0.3250984663103522. Punkt
w kole.

0.10281671013573856
0.45973032563428606
Suma kwadratów:
0.22192324819094317. Punkt
w kole.

0.5077853161966309
0.12293036895152698
Suma kwadratów:
0.272957802955471. Punkt w
kole.

0.7195983980666332
0.9370650751989997
Suma kwadratów:
1.3959128096577715. Punkt
```

```
   poza kołem.
15
16 Punkty w/poza: 4/1
17 Wartość pi: 3.2
18
```

W tym przypadku 4 punkty znalazły się w kole, 1 poza nim. Wartość liczby π została więc wyliczona z wyrażenia: `4 * 4 / 5` – które daje wartość `3.2`. Tym razem uzyskaliśmy lepsze przybliżenie niż w przykładzie poprzednim.

9

Jak uzyskać większą dokładność podczas wyliczania liczby π za pomocą przygotowanego programu?

Aby odpowiedzieć na to pytanie, trzeba zauważyć, że jedynym parametrem wejściowym, który można zmieniać, jest liczba losowań.

Spróbujmy więc zwiększać tę liczbę i przeprowadźmy: 1000, 10 000 i 100 000 losowań.

10

Poniżej zestawienie przykładowych końcowych komunikatów po wykonaniu programu dla 1000, 10 000 i 100 000 losowań.

Liczba losowań: 1000.

```
1 Punkty w/poza: 796/204
2 Wartość pi: 3.184
3
```

Liczba losowań: 10 000.

```
1 Punkty w/poza: 7884/2116
2 Wartość pi: 3.1536
```

```
3
```

Liczba losowań: 100 000.

```
Punkty w/poza: 78537/21463
Wartość pi: 3.14148

```

Podsumowując, można stwierdzić, że zwiększanie liczby losowań zazwyczaj daje dokładniejsze przybliżenie uzyskanej wartości liczby π. Należy jednak pamiętać, że ponieważ punkty generowane są losowo, może się zdarzyć, że dobre przybliżenie uzyskamy już dla 10 000 losowań.

Polecenie 2

Napisz program, który wyznaczy przybliżoną wartość liczby π. Wykorzystaj metodę Monte Carlo. Przetestuj działanie programu dla 100 prób.

Specyfikacja problemu:

Dane:

- `liczba_prob` – liczba prób podczas stosowania metody Monte Carlo; liczba naturalna większa od 0

Wynik:

- `pi` – obliczona wartość liczby π; liczba rzeczywista

```
1
```

```
1
```

Polecenie 3

Porównaj swoje rozwiązanie z przedstawionym w filmie.

Film dostępny pod adresem /preview/resource/R12d1UW9mDxxE

Film nawiązujący do treści materiału: Modelowanie matematyczne - metoda Monte Carlo.

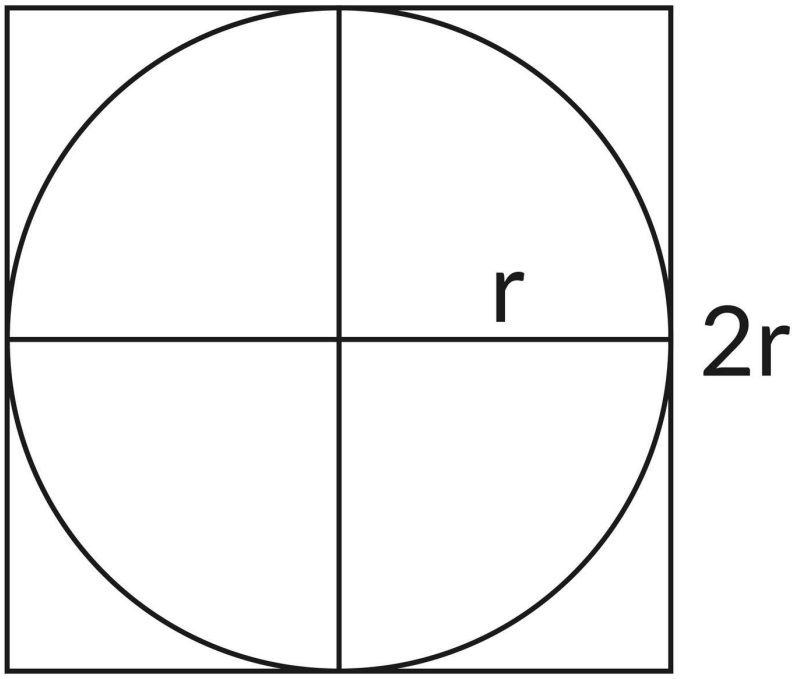
Plik o rozmiarze 624.00 B w języku polskim

Przeczytaj

Jak wyznaczyć wartość π?

> Metoda Monte Carlo została omówiona w e-materiale Wyznaczanie liczby π metodą Monte Carlo.

Załóżmy, że wpisaliśmy w kwadrat koło o promieniu r. Długość boku kwadratu wynosi zatem $2r$:



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

Pola kwadratu i koła wynoszą odpowiednio:

$$P_K = 4 \cdot r^2$$

$$P_O = \pi \cdot r^2$$

Obliczmy stosunek powierzchni koła do powierzchni kwadratu:

$$S = \frac{P_O}{P_K} = \frac{\pi \cdot r^2}{4 \cdot r^2} = \frac{\pi}{4}$$

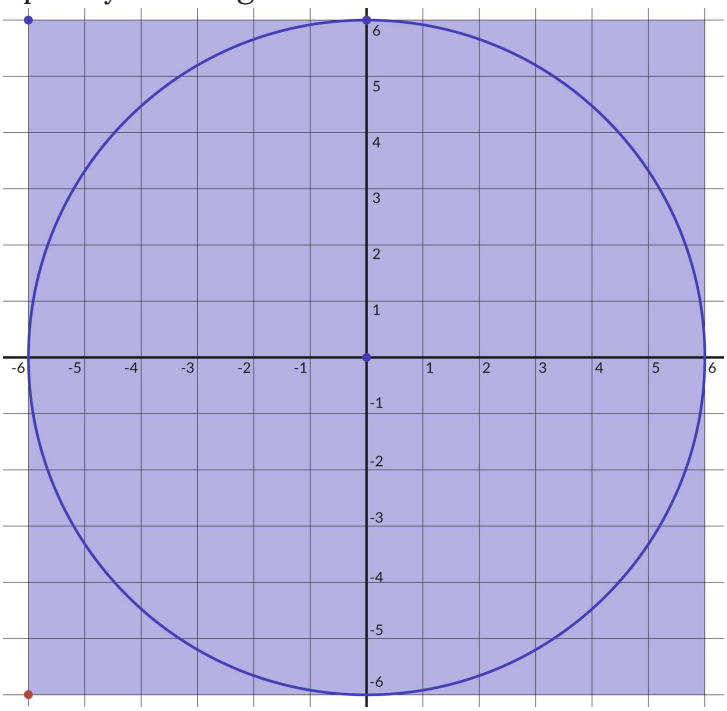
Gdybyśmy znali pole powierzchni koła oraz pole powierzchni kwadratu, bylibyśmy w stanie wyznaczyć wartość π. Jest ona czterokrotnie większa od liczby S (czyli stosunku pól powierzchni koła i kwadratu).

Powierzchnię kwadratu obliczymy bez problemu. Niestety, z kołem jest trudniej. Aby skorzystać ze wzoru na pole powierzchni (czyli zastosować metodę analityczną), niezbędna jest znajomość liczby π.

Spróbujmy więc znaleźć inny sposób wyznaczenia stosunku powierzchni koła i kwadratu. Jest on dobrym przykładem zastosowania metody Monte Carlo.

Metoda Monte Carlo

Narysujmy kwadrat z wpisanym w niego kołem:



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

Następnie zacznijmy rzucać na rysunek ziarenka piasku. Gdy pokryją one cały kwadrat, stosunek liczby ziarenek wewnątrz narysowanego okręgu do wszystkich rzuconych ziarenek będzie (prawie) równy stosunkowi pola koła do pola kwadratu.

W praktyce nie musielibyśmy pokrywać całego rysunku piaskiem. Wystarczyłoby zadbać o to, aby ziarenka spadały w losowych miejscach. Wraz ze wzrostem liczby rzuconych ziarenek zwiększa się jednak precyzja wyznaczania stosunku pól obu figur.

Jak zasymulować takie doświadczenie za pomocą komputera? Wyobraźmy sobie wpisane w kwadrat koło o promieniu `r`, mające środek w punkcie $(0, 0)$. Będziemy losować punkty o współrzędnych (x, y), mieszczących się w zakresie od `-r` do `r`, a następnie sprawdzać, ile z nich znalazło się we wnętrzu koła, a ile nie. Stosunek tych dwóch liczb będzie odpowiadał stosunkowi pól koła i kwadratu. Dokładność metody będzie rosła wraz ze zwiększaniem liczby losowań.

> **Ciekawostka**

Stosunek pól będzie taki sam również w przypadku ćwiartek koła i kwadratu (jeżeli pole koła i pole kwadratu podzielimy na cztery części, to stosunek ich powierzchni się nie zmieni). Wystarczy zatem, że będziemy losowali punkty o współrzędnych od 0 do `r`. Losując odpowiednio dużą liczbę punktów, powinniśmy otrzymać przybliżenie wartości liczby π.

Przykład 1

Zdefiniujemy funkcję obliczającą wartość π dla zadanej liczby losowych punktów (domyślnie będzie ich 3000). Zwróć uwagę na zapis `from random import random as r` – pozwala on użyć innej nazwy dla funkcji importowanej z modułu języka Python. W prezentowanym przykładzie funkcji `random()` nadamy nazwę `r()`:

Specyfikacja problemu:

Dane:

- `punkty` – liczba losowych punktów, liczba naturalna większa niż 0

Wynik:

- `pi` – obliczona wartość liczby π, liczba rzeczywista

Uwaga!

Podane niżej wartości liczby π są przykładowe i dla każdego uruchomienia programu mogą być inne.

```
1  from random import random as r
2
3  def monte_carlo_pi(punkty = 3000):
4      punkty_w_kole = 0
5      for i in range(punkty):
6          x, y = r(), r()
7          if (x**2 + y**2 <= 1):
8              punkty_w_kole += 1
9      return 4 * punkty_w_kole / punkty
10
11 # przykład wywołania
12 print(monte_carlo_pi())
13 # 3.168
14
15 print(monte_carlo_pi(20000))
```

```
# 3.1556

print(monte_carlo_pi(9000000))
# 3.141952
```

Ważne!

Jak możemy zauważyć, wraz ze zwiększaniem liczby losowanych punktów wzrasta dokładność wyznaczania wartości π.

Przykład 2

Aby pokazać działanie programu, posłużymy się biblioteką matplotlib
i wyświetlimy losowane punkty:

Specyfikacja problemu:

Dane:

- `punkty` – liczba punktów wykorzystanych do metody Monte Carlo; liczba naturalna większa niż 0

Wynik:

- wykres przedstawiający rozmieszczenie wylosowanych punktów

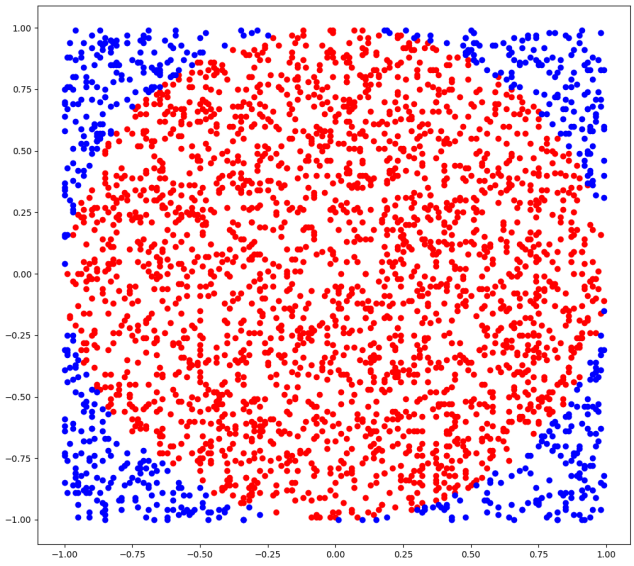
```
from random import randrange as r

def monte_carlo_pi_graf2(punkty = 3000):
    import matplotlib.pyplot as plt
    X, Y = [], []
    X1, Y1 = [], []
    punkty_w_kole = 0
    for i in range(punkty):
        x, y = r(-100,100,1)/100, r(-100,100,1)/100
        if (x**2 + y**2 <= 1):
            punkty_w_kole += 1
            X.append(x)
            Y.append(y)
        else:
            X1.append(x)
            Y1.append(y)
    plt.scatter(X, Y, color='red')
```

```
    plt.scatter(X1, Y1, color='blue')
    plt.show()

# wywołanie
monte_carlo_pi_graf2()
```

Efektem będzie rysunek, w którym punkty w okręgu zaznaczone są kolorem czerwonym, a poza nim – niebieskim:



Polecenie 1

Zapisz funkcję `monte_carlo_pi()` tak, aby wykonywała n prób obliczenia wartości liczby π i zwracała wynik jako średnią wszystkich prób.

Działanie programu przetestuj dla podanych danych.

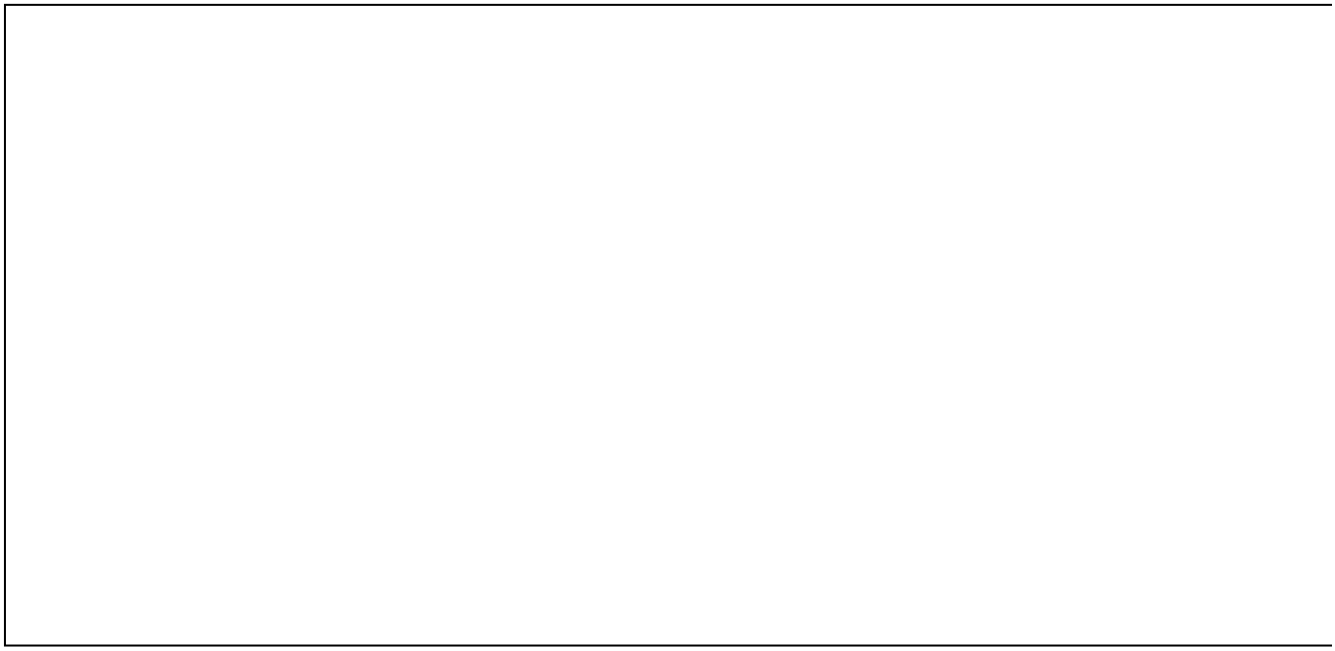
Specyfikacja problemu:

Dane:

- `punkty` – liczba losowych punktów, liczba naturalna większa niż 0
- n – liczba prób obliczenia liczby π, liczba naturalna większa niż 0

Wynik:

- `pi` – obliczona wartość średnia liczby π, liczba rzeczywista



Już wiesz

Podsumujmy najważniejsze elementy tego e-materiału:

- Metoda Monte Carlo jest przydatna do obliczania przybliżonych wartości.
- Możemy importować funkcje z modułów i nadawać im własne nazwy.
- Moduł matplotlib doskonale nadaje się do wizualizacji.

Słownik

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 go zainstalować, korzystając z mechanizmu `pip`

metoda analityczna

konkretny sposób oznaczania analitu za pomocą danej techniki, poprzez wykonanie określonego postępowania

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Zdefiniuj funkcję `oblicz_pi()`, która obliczy metodą Monte Carlo przybliżoną wartość liczby π (zaokrągloną do pięciu miejsc po przecinku) dla n losowych punktów. Do losowania wykorzystaj funkcję `uniform` z biblioteki `random`.

Specyfikacja problemu:

Dane:

- n – liczba naturalna większa niż 0; liczba punktów użytych do rozwiązania problemu

Wynik:

- `wynik` – krotka zawierająca dwie wartości: wyliczoną liczbę π z dokładnością do pięciu miejsc po przecinku oraz liczbę punktów, które trafiły do środka koła

Program przetestuj dla 50 000 punktów.

Twoje zadania

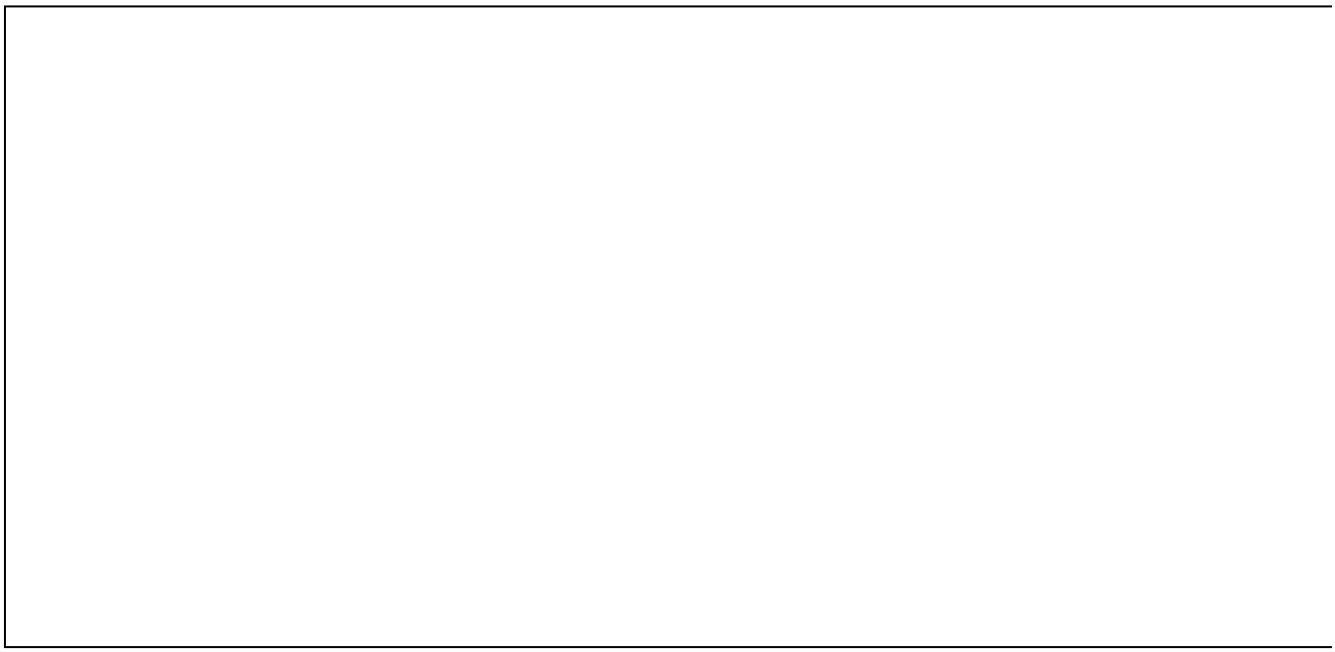
1. Program oblicza liczbę π przy użyciu metody Monte Carlo.

```
import random as r

r.seed(10) # ustawiamy ziarno żeby zawsze otrzymywać ten sam wynik
def oblicz_pi(n):
    # tu wpisz swój kod

wynik = oblicz_pi(50000)
print(wynik)
```

```
1
```



Ćwiczenie 2

Napisz program, który za pomocą metody Monte Carlo obliczy pole koła, a następnie zwróci wartość bezwzględną z różnicy pomiędzy prawidłowym wynikiem a polem uzyskanym za pomocą metody Monte Carlo. Przetestuj swój program dla koła o promieniu 2 i 50 000 punktów. Wynik wypisz z dokładnością do dwóch miejsc po przecinku. Dokładny wynik oblicz, używając wartości liczby π z biblioteki `math`, a do losowania wykorzystaj funkcję `uniform` z biblioteki `random`.

Specyfikacja problemu:

Dane:

- `promien` – liczba rzeczywista; promień koła, którego pole należy obliczyć
- n – liczba naturalna większa niż 0; liczba punktów użytych do rozwiązania problemu

Wynik:

- `roznica` – liczba rzeczywista; wartość bezwzględna z różnicy pomiędzy obliczonym polem koła a dokładnym wynikiem

Twoje zadania

1. Program oblicza wartość bezwzględną pomiędzy prawdziwym polem koła a polem obliczonym przy użyciu metody Monte Carlo.

```
import random as r
from math import pi

r.seed(10) # ustawiamy ziarno żeby zawsze otrzymywać ten sam wynik
def oblicz_roznice(n, promien):
    # tu wpisz kod

print(oblicz_roznice(50000, 2))
```

```
1
```

Ćwiczenie 3

Napisz program, który za pomocą metody Monte Carlo obliczy pole powierzchni figury zawartej w kwadracie o wierzchołkach $(0,0), (p,0), (p,p), (0,p)$ i leżącej pod wykresem funkcji $f(x) = \frac{1}{x}$. W tym celu należy losować punkty z podanego kwadratu, a następnie sprawdzać, czy leży on pod wykresem funkcji. Przetestuj swój algorytm dla $p = 2$ i 50 000 punktów. Wynik wypisz z dokładnością do dwóch miejsc po przecinku (nie zaokrąglaj). Do losowania wykorzystaj funkcję `uniform` z biblioteki `random`.

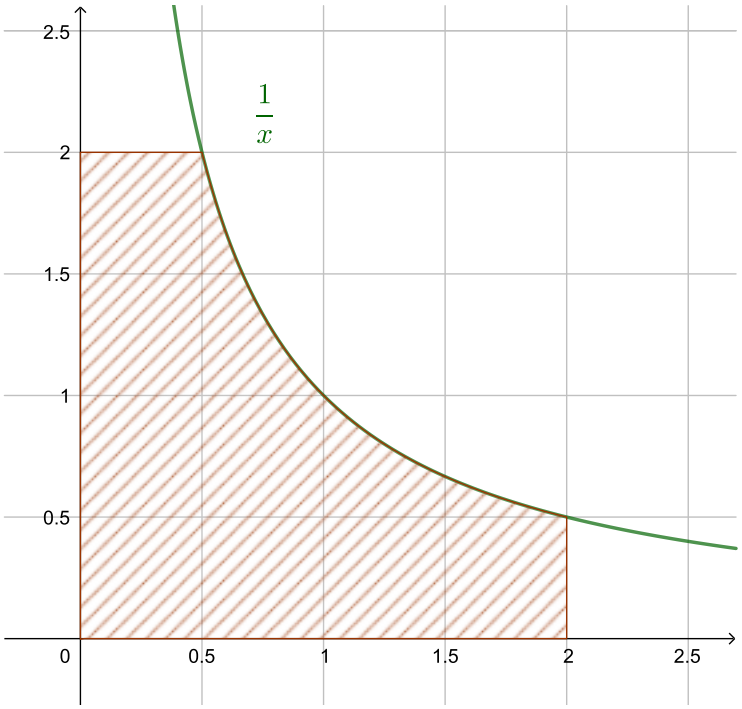
Specyfikacja problemu:

Dane:

- p – bok kwadratu, który zawiera pole figury; liczba rzeczywista
- n – liczba naturalna większa niż 0; liczba punktów użytych do rozwiązania problemu

Wynik:

- `pole` – liczba rzeczywista; pole figury.



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

Twoje zadania

1. Program oblicza pole powierzchni danej figury metodą Monte Carlo.

```
import random as r

r.seed(10) # ustawiamy ziarno żeby zawsze otrzymywać ten sam wynik
def oblicz_pole(n, p):
    # tu wpisz kod

print(oblicz_pole(50000, 2))
```

```
```

Dla nauczyciela

Autor: Adam Jurkiewicz
Przedmiot: Informatyka

Temat: Wyznaczanie liczby π metodą Monte Carlo w języku Python

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

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.

Treści nauczania – wymagania szczegółowe

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

Zakres podstawowy. Uczeń:

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:

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

I + II. Zakres rozszerzony. 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:

h) 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):

- Scharakteryzujesz sposoby generowania liczb losowych w języku Python.
- Zastosujesz numeryczną metodę obliczeniową opartą na rachunku prawdopodobieństwa.
- Skorzystasz z biblioteki graficznej `matplotlib` przy rozwiązywaniu zadań.
- Przeanalizujesz, jak losuje się punkty w obrębie danej figury geometrycznej.
- Zaimplementujesz algorytm obliczania wartości liczby π metodą Monte Carlo w języku Python.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium 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: „Wyznaczanie liczby π metodą Monte Carlo w języku Python". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Film samouczek".

Faza wstępna:

1. Nauczyciel sprawdza przygotowanie uczniów do zajęć.
2. Nauczyciel prosi wybraną osobę o odczytanie tematu lekcji, a następnie określa cele i kryteria sukcesu.
3. **Rozpoznanie wiedzy uczniów.** Nauczyciel wyświetla na tablicy pytanie zawarte w sekcji „Wprowadzenie": Czy potrafisz wskazać inne sytuacje, w których sięga się po numeryczne techniki obliczeniowe? Ochotnicy na nie odpowiadają. Pozostali uczniowie uzupełniają ich wypowiedzi lub przedstawiają swoje propozycje.

Faza realizacyjna:

1. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Film samouczek". Uczniowie wspólnie wykonują polecenie nr 1 z sekcji „Film samouczek".
2. Uczniowie indywidualnie wykonują polecenia nr 2 oraz 3 z sekcji „Film samouczek".
3. **Praca z tekstem.** Nauczyciel wyświetla zawartość sekcji „Przeczytaj". Na forum klasy uczniowie analizują przedstawione w niej rozwiązania Przykładów 1 i 2.
4. Uczniowie w parach wykonują ćwiczenia nr 1–3 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

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 wykonują polecenie nr 1 z sekcji „Przeczytaj".

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 „Przeczytaj” można wykorzystać na lekcji jako podsumowanie i utrwalenie wiedzy uczniów.