Laboratorium programistyczne 3
Błądzenie losowe

Projekt „Matematyka dla Ciekawych Świata”,
Piotr Morawiecki

kwiecień 2025

Spis treści

1 Implementacja błądzenia losowego

Rozważmy grę, w której rzucamy monetą. Jeśli wypadnie orzeł wygrywamy jeden żeton, a jeśli wypadnie reszta to tracimy jeden żeton. Napiszmy funkcję `simulate`, która będzie symulować określoną liczbę rzutów monetą. Funkcja ta przyjmie ona dwa argumenty:

- `x0` – początkowa liczba żetonów,
- `t_max` – łączna liczba rzutów monetą.

Na wyjściu funkcja zwróci listę zawierającą liczbę żetonów w kolejnych krokach. Na przykład funkcja `simulate(x0 = 5, t_max = 10)` może zwrócić na przykład listę:

```
[5, 6, 7, 6, 5, 4, 3, 4, 5, 6, 7]
```

Krok 1. W definicji funkcji podajemy domyślne wartości argumentu `x0` i `t_max`.

```
def simulate(x0 = 10, t_max = 100):
```

Krok 2. Tworzymi listę `x`, w której przechowywane będą wartości liczby żetonów w kolejnych rundach. Początkowo znajduje się w niej jeden element `x0`.

```
  x = [x0]
```

Krok 3. Zaczynamy pętlę `for`, w której `i` oznacza numer rundy.

```
  for i in range(t_max):
```

Krok 4. W każdej rundzie losujemy liczbę 0 (reszka) lub 1 (orzeł). Jeśli wypadnie 1 (orzeł) to zwiększamy liczbę żetonów z ostatniej rundy `x[-1]` o 1 i dodajemy do listy za pomocą komendy `append`.

```
    if random.randint(0, 1) == 1:
      x.append(x[-1] + 1)
```

Uwaga: Komenda `random.randint(a, b)` zwraca liczby całkowite z przedziału od `a` do `b` (włącznie). Do jej wykorzystania należy zaimportować pakiet `random` za pomocą komendy `import random`.

Krok 5. W przeciwnym razie (jeśli wypadnie reszka), zmniejszamy liczbę żetonów o 1.

```
    else:
      x.append(x[-1] - 1)
```

Krok 6. Po zakończeniu pętli zwracamy listę `x`.

```
  return x
```

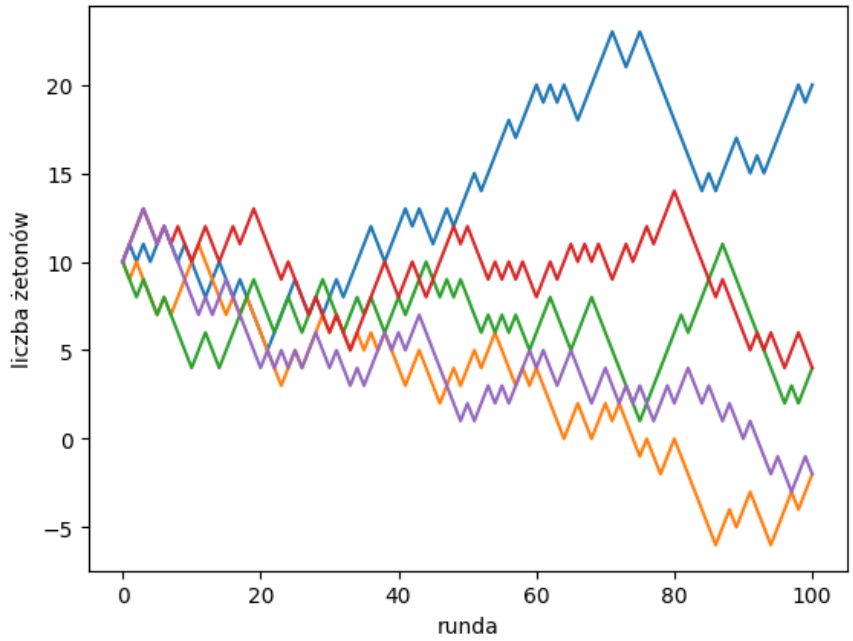
Funkcja powinna w całości wyglądać następująco:

```
import random

def simulate(x0 = 10, t_max = 100):
  x = [x0]

  for i in range(t_max):
    if random.randint(0, 1) == 1:
      x.append(x[-1] + 1)
    else:
      x.append(x[-1] - 1)

  return x
```

Przetestujmy teraz tę funkcję. Wykonajmy ją pięć razy i przedstawmy wyniki na wykresie:

```
import matplotlib.pyplot as plt

for i in range(5):
  plt.plot(simulate())

plt.ylabel("liczba żetonów")
plt.xlabel("runda")
plt.show()
```

Powinniśmy uzyskać następujący wykres:



Zadanie 1.0.1 (ruletka)

W ruletce kulka wypada na jedno z 37 pól. 18 pól jest czerwone, 18 jest czarne, a jedno jest zielone. Stawiając żeton na pole czerwone wygramy dodatkowy żeton, jeśli kulka wyląduje na polu czerwonym. W przeciwnym razie stracimy postawiony żeton.

1. Zmodyfikuj funkcję `simulate` tak, żeby symulowała wielokrotną grę w ruletkę.
2. Przetestuj program rysując wykres dla `x0=100` i `t_max=100`.
3. Narysuj wykres dla `t_max=1000` i `t_max=10000`. Co obserwujesz?

2 Firma ubezpieczeniowa

2.1 Model firmy ubezpieczeniowej

Model błądzenia losowego można wykorzystać do analizy ryzyka. Rozważmy firmę ubezpieczeniową, która ubezpiecza domy na wypadek powodzi. Przyjmijmy, że każdego tygodnia firma osiąga stały przychód o wysokości 1. Jednak każdego tygodnia istnieje prawdopodobieństwo 5%, że wystąpi powódź firma będzie musiała wypłacić swoim klientom kwotę o wysokości 15.

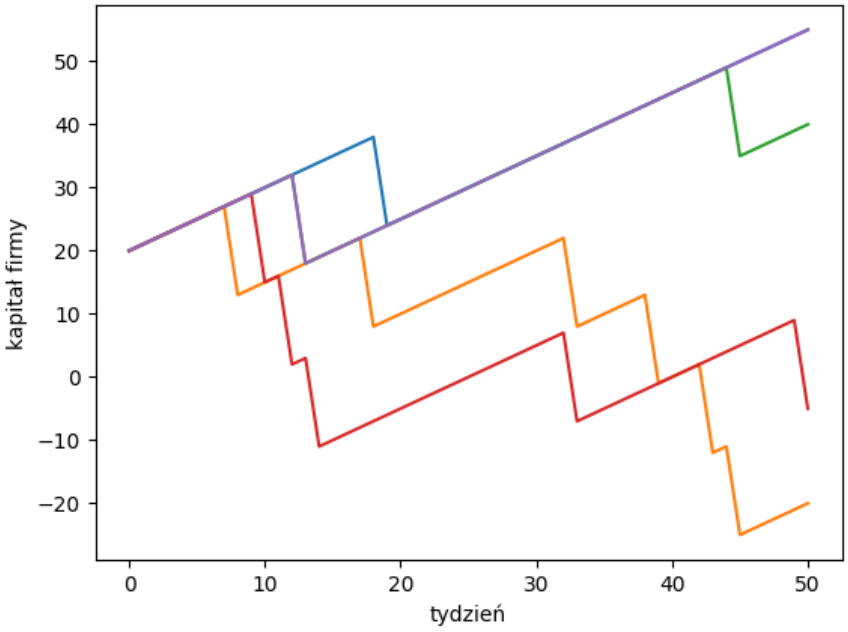
Zadanie 2.1.1 (wartość oczekiwana zysku)

Oblicz ile wynosi wartość oczekiwana zysku każdego tygodnia.

Zadanie 2.1.2 (model firmy ubezpieczeniowej)

1. Napisz funkcję `simulate(x0, t_max)`, która zasymuluje zmianę kapitału firmy w poszczególnych tygodniach. `x0` oznacza początkowy kapitał firmy, a `t_max` liczbę symulowanych tygodni.
2. Przedstaw na wykresie kapitału od czasu dziesięć losowo wygenerowanych scenariuszy. Przyjmij, że kapitał początkowy firmy wynosi $x_0 = 20$, a czas symulacji wynosi `t_max`$= 50$.

Powinniśmy uzyskać wykres podobny do poniższego:



Jak widać na powyższym wykresie, w niektórych scenariuszach kapitał firmy spada do zera (firma bankrutuje). W takim przypadku będziemy chcieli wcześniej zakończyć symulację.

Zadanie 2.1.3 (upadek firmy)

Dodaj do funkcji `simulate` instrukcję warunkową, która zakończy symulację w momencie kiedy kapitał firmy osiągnie wartość 0 lub niższą. Sprawdź działanie programu, ponownie generując wykres kapitału od czasu.

2.2 Analiza ryzyka

Możemy oszacować ryzyko upadku firmy, wykorzystując metodę Monte Carlo. Wystarczy, że n razy wykonamy napisaną powyżej symulację i sprawdzimy, w jakiej części przypadków firma zbankrutuje.

```
n = 10000
n_bankruptcy = 0

for i in range(n):
  x = simulate(x0 = 20, t_max = 50)
  if x[-1] <= 0:
    n_bankruptcy += 1

print(n_bankruptcy / n)
```

Prawdopodobieństwo powinno wyjść w przybliżeniu równe 0.22 (22%).

Zadanie 2.2.1 (szacowanie ryzyka upadku)

1. Jakie jest prawdopodobieństwo upadku firmy w ciągu 500 tygodni i 5000 tygodni?
2. Co można na tej podstawie wywnioskować? Pomocne może okazać się narysowanie wykresu kapitału od czasu.

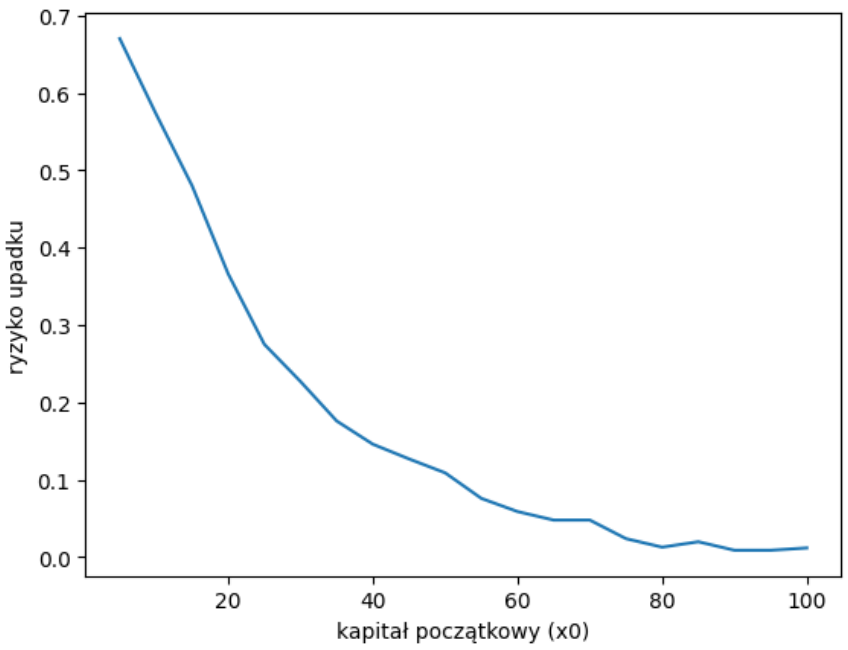
2.3 Dalsze badanie ryzyka upadku

Na koniec zbadajmy jak prawdopodobieństwo upadku firmy zależy od jej kapitału początkowego x0. W tym celu napiszmy funkcję `estimateBuncrupcy(x0, t_max`, która oszacuje prawdopodobieństwo upadku firmy.

```
def estimateRisk(x0 = 20, t_max = 300, n = 1000):
  n_bankruptcy = 0

  for i in range(n):
    x = simulate(x0, t_max)
    if x[-1] <= 0:
      n_bankruptcy += 1

  return n_bankruptcy / n
```

Teraz oszacujmy prawdopodobieństwo upadku firmy dla różnych wartości x0 z przedziału od 5 do 100, a następnie przestawmy wyniki na wykresie.

```
x0_values = range(5, 101, 5)     # zakres zawiera wartości 5, 10, 15, ..., 95, 100
risk_values = [estimateRisk(x0) for x0 in x0_values]

plt.plot(x0_values, risk_values)
plt.xlabel("kapitał początkowy (x0)")
plt.ylabel("ryzyko upadku")
plt.show()
```



Wyraźnie ryzyko upadku firmy spada wraz ze wzrostem jej kapitałem początkowym. Przy kapitale początkowym równym 100, ryzyko staje się równe około 1%.

Zadanie 2.3.1 (analiza ryzyka)

Zbadaj jak ryzyko upadku zmienia się wraz z wielkością wypłacanych odszkodowań w przedziale od 10 do 30. Jak można zinterpretować te wyniki?
Wskazówka: Warto tutaj lekko zmodyfikować poprzedni program. Do funkcji `simulate` i `estimateRisk` dodaj nowy argument określający wysokość odszkodowań (możesz go nazwać `insurance_value`). W ten sposób łatwo będziesz mógł go zmienić.

3 Centralne Twierdzenie Graniczne (dla chętnych)

Wróćmy do pierwszego przykładu z Sekcji 1 z rzutem monetą:

```
import random

def simulate(x0 = 10, t_max = 100):
  x = [x0]

  for i in range(t_max):
    if random.randint(0, 1) == 1:
      x.append(x[-1] + 1)
    else:
      x.append(x[-1] - 1)

  return x
```

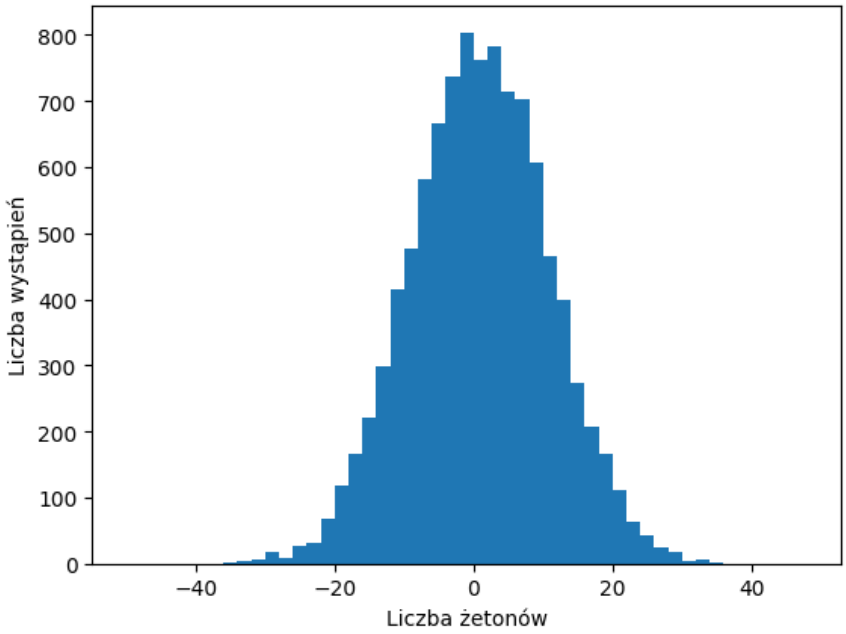
Sprawdźmy, jakie jest prawdopodobieństwo uzyskania poszczególnych wartości x dla wartości początkowej `x0 = 0` po określonej liczbie kroków (np. `t_max = 100`). Za W tym celu wykonamy tę funkcję 10 000 razy, za każdym razem zapisując końcową liczbę żetonów:

```
x_last = []
for i in range(10000):
  x= simulate(x0 = 0, t_max = 100)
  x_last.append(x[-1])
```

Przedstawimy ostatnie uzyskane wartości x na histogramie.

```
import matplotlib.pyplot as plt

plt.hist(x_last, bins=range(-50, 50, 2))
plt.xlabel("Liczba żetonów")
plt.ylabel("Liczba wystąpień")
plt.show()
```

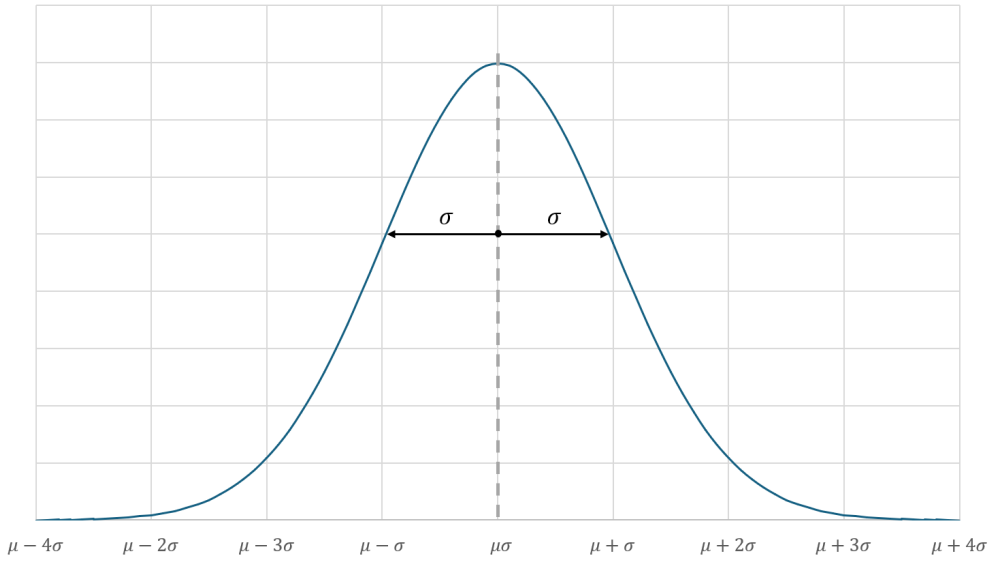
Uzyskujemy wówczas następujący wykres:



Zgodnie z tzw. *Centralnym Twierdzeniem Granicznym*, dla dużych wartości przyjmuje on kształt tzw. rozkładu normalnego (także czasem nazywanego rozkładem Gaussa), proporcjonalnego do:

$$P(x) \sim e^{-\frac{-x^2}{2\sigma^2}},$$

gdzie $e \approx 2.71...$ to tzw. liczba Eulera. Parametr σ opisuje szerokość rozkładu prawdopodobieństwa (patrz wykres poniżej). W przypadku naszej symulacji wartość parametru σ zależy od liczby zasymulowanych rund.



Zadanie 3.0.1 (badanie własności rozkładu)

Zbadaj jak kształt wykresu rozkładu liczby żetonów zależy od `t_max`.

1. Wygeneruj wykres dla `t_max=25`, `t_max=50`, `t_max=100` i `t_max=200`.
2. Czy szerokość rozkładu rośnie czy maleje wraz z `t_max`? Zastanów się dlaczego.
3. Sprawdź (odczytując wartości na oko z wykresu) czy szerokość rozkładu rośnie wprost proporcjonalnie do czasu. Czy potrafiłbyś zgadnąć poprawną zależność?

4 Zadania dodatkowe

Zadanie 4.0.1 (badanie własności rozkładu)

Hazardzista Harry rozpoczął grę w ruletkę z 20 żetonami. Napisz symulację, która pozwoli Ci oszacować prawdopodobieństwo, że uda mu się dobić do 40 żetonów, zanim wyda wszystkie swoje żetony, jeśli

a) będzie w każdej rundzie stawiać jeden żeton na pola czerwone (wówczas z prawdopodobieństwem $\frac{18}{37}$ wygra 1 żeton),

b) będzie w każdej rundzie stawiać jeden żeton na pole zielone (wówczas z prawdopodobieństwem $\frac{1}{37}$ wygra 35 żetonów).

5 Praca domowa nr 3

Rozwiązania zadań domowych należy przesłać do czasu następnych ćwiczeń. Prowadzący może również zmienić ostateczny termin przesłania pracy domowej, np. na inną godzinę lub dzień.

Za pracę domową można maksymalnie dostać 6 punktów. Można wybierać zarówno łatwiejsze zadania, jak i trudniejsze, za większą liczbę punktów. Można również rozwiązać zadania za większą liczbę punktów, np. za 10. Jeśli wtedy poprawnie będą rozwiązane zadania za 5 punktów, to zostanie przydzielone 5 punktów. Jeśli będą poprawnie rozwiązane zadania za więcej niż 6 punktów, np. 8 czy 10, to taka osoba otrzyma maksymalnie 6 punktów.

Pamiętaj, żeby notebook był czytelny – każde zadanie powinno być umieszczone w osobnym bloku tekstowym oraz poprzedzone numerem (i opcjonalnie opisem) zadania. Komentarze są mile widziane.

Zadanie 5.0.1 — 1 pkt (upadek hazardzisty)

Zmodyfikuj program z Zadania 1.0.1 (ruletka) tak, żeby po spadku liczby żetonów do 0 symulacja została przerwana. Wykonaj program dla `x0=10` i `t_max=10000`.

Zadanie 5.0.2 — 2 pkt (analiza ryzyka c.d.)

W symulacji firmy ubezpieczeniowej założyliśmy, że z prawdopodobieństwem 5% firma musi wypłacić odszkodowanie o wartości 15. Napisz program, za pomocą którego sprawdzisz, jak prawdopodobieństwo upadku firmy ubezpieczeniowej zależy od tego prawdopodobieństwa wypłacenia odszkodowania. Zbadaj wartości z przedziału od 1% do 10%.
Narysuj wykres zależności ryzyka upadku firmy od prawdopodobieństwa wypłacenia odszkodowania. Załóż, że początkowy kapitał firmy wynosi `x0` $= 20$ (jak w przykładzie z zajęć), a za czas symulacji przyjmij `t_max` $= 500$.

Zadanie 5.0.3 — 3 pkt (algorytm tradingowy)

Cena akcji pewnej firmy każdego dnia rośnie lub maleje o 1. Zakupiliśmy je w momencie kiedy ich cena wynosiła `x0=100`. Sprzedamy ją w momencie ich cena spadnie do `x=75` lub kiedy wzrośnie do `x=150`.

1. Napisz funkcję `profit`, która zasymuluje zmianę ceny akcji tej firmu. Funkcja powinna zwrócić wartość `true`, jeśli uda nam się ją sprzedać za 150, lub wartość `false`, jeśli sprzedamy ją za niższą cenę (80).
2. Zastosuj metodę Monte Carlo, wykonując 1000 razy funkcję `profit`, żeby określić prawdopodobieństwo osiągnięcia zysku z tej inwestycji.

Zadanie 5.0.4 — 3 pkt (błądzenie losowe w 2D)

Rozważmy pijanego pingwina, którego położenie można opisać za pomocą dwóch współrzędnych (x, y). Początkowo znajduje się on w punkcie $(0, 0)$. W każdym kroku zwiększa lub zmniejsza on jedną ze współrzędnych (x lub y) o 1. Napisz symulację ruchu pijanego pingwina. Narysuj wykres jego toru ruchu, na osiach zaznaczając jego współrzędne x i y dla pierwszych tysiąca kroków.