20. Наукові обчислення

Т20.1 Сформулювати гіпотези щодо наявності та значень границь послідовностей $\{a_n\}$. Для цього побудувати масиви numpy зі значеннями n та a_n. Припустити, що границя послідовності $\{a_n\}$ дорівнює b. Для заданого малого $\varepsilon > 0$ перевірити, що у масиві a_n, починаючи з деякого k, для $m > k$ $|a_m - b| < \varepsilon$.
Графічно відобразити елементи послідовності, а також пряму y = b. Побудувати графік смуги (b - ε, b + ε) та показати, що усі елементи a_n, при n>k, потрапляють у цю смугу. Самостійно підібрати масштаб осей для ілюстрації наявності границі.
Розглянути границі:

а) $b = \lim\limits_{n\to\infty} \dfrac{(n + \frac{n}{3} + \ldots + \frac{n}{3^n})(n - 3)\sqrt[n]{n}}{2n^2 + 5}$.

б) $b = \lim\limits_{n\to\infty} \dfrac{(n-1)^4 - (n+2)^4}{(2n+1)^3 + (n-1)^3}$

в) $b = \lim\limits_{n\to\infty} \dfrac{n\sqrt[7]{n} + \sqrt[4]{16n^8 + 5}}{\left(n + \sqrt[3]{n}\right)\sqrt[5]{n^5 - 1}}$

г) $b = \lim\limits_{n\to\infty} \dfrac{\sqrt{n(n^4+1)} - \sqrt{(n^3-1)(n^2+2)}}{\sqrt{n}}$.

д) $b = \lim\limits_{n\to\infty} \dfrac{\sqrt[3]{2n^4 + n^3 + 1} - n\sqrt[3]{2n+3}}{\sqrt[3]{n+1}}$.

~~е)~~ $\cancel{b = \lim\limits_{n\to\infty} \dfrac{(n + \frac{n}{3} + \ldots + \frac{n}{3^n})(n - 3)\sqrt[n]{n}}{2n^2 + 5}}$.

ж) $b = \lim\limits_{n\to\infty} = \left|\dfrac{1}{n} - \dfrac{2}{n} + \dfrac{3}{n} - \ldots + \dfrac{(-1)^{n-1}n}{n}\right|$.

з) $b = \lim\limits_{n\to\infty} \left(1 + \dfrac{1}{n}\right)^n$

і) $b = \lim\limits_{n\to\infty} \left(\dfrac{5n^2 + 7n + 1}{5n^2 + 3n + 6}\right)^{n-3}$.

и) $b = \lim\limits_{n\to\infty} 2n(\sqrt{n^2+1} - n)$

к) $b = \lim\limits_{n\to\infty} \frac{n^\alpha}{a^n}$ $(\alpha > 0, a > 1)$

л) $b = \lim\limits_{n\to\infty} \sqrt[n]{n}$

м) $b = \lim\limits_{n\to\infty} \left(1 - \frac{2}{3} + \left(\frac{2}{3}\right)^2 - \left(\frac{2}{3}\right)^3 + \ldots + (-1)^n \left(\frac{2}{3}\right)^n\right)$

н) $b = \lim\limits_{n\to\infty} (\sqrt[3]{n^2 - n^3} + n)$

о) $b = \lim\limits_{n\to\infty} \frac{\cos n - e^{n-1}}{e^n + \pi^{\frac{n}{2}}}$.

Т20.2 Виконати наближене обчислення π наближенням кола рівносторонніми n-кутниками. Для цього порахувати периметр рівностороннього n-кутника. Розглянути n-кутники з числом сторін $n = 2^k$, $k = 2, 3, 4, \ldots$ Використати масиви numpy.
Зобразити на графіках коло та n-кутник. Зберегти відео (виконати анімацію) для різних значень k.

Т20.3 Виконати наближення функції f(x) на відрізку [a, b] частиною ряду Тейлора, що є розкладом f(x) у 0. Взяти перші n доданків для n = 1, 2, …, m, де m – задане число. Побудувати графікі функції та її наближення. Зберегти відео (виконати анімацію) для різних значень n. Використати масиви numpy.
Розв’язати задачу для функції f(x):

а) $y = \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots;$

б) $y = \cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots;$

в) $y = shx = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots;$

г) $y = chx = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots;$

д) $y = e^{x} = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \ldots;$

е) $y = \ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots (|x| < 1);$

ж) $y = \frac{1}{1+x} = 1 - x + x^2 - x^3 + \ldots (|x| < 1);$

з) $y = \ln\frac{1+x}{1-x} = 2 \cdot \left[\frac{x}{1} + \frac{x^3}{3} + \frac{x^5}{5} + \ldots\right] \quad (|x| < 1);$

і) $y = \frac{1}{(1+x)^2} = 1 - 2 \cdot x + 3 \cdot x^2 - \ldots (|x| < 1);$

к) $y = \frac{1}{(1+x)^3} = 1 - \frac{2 \cdot 3}{2} \cdot x + \frac{3 \cdot 4}{2} \cdot x^2 - \frac{4 \cdot 5}{2} \cdot x^3 + \ldots (|x| < 1);$

л) $y = \frac{1}{1+x^2} = 1 - x^2 + x^4 - x^6 + \ldots (|x| < 1);$

м) $y = \sqrt{1+x} = 1 + \frac{1}{2} \cdot x - \frac{1}{2 \cdot 4} \cdot x^2 + \frac{1 \cdot 3}{2 \cdot 4 \cdot 6} \cdot x^3 - \ldots (|x| < 1);$

н) $y = \frac{1}{\sqrt{1+x}} = 1 - \frac{1}{2} \cdot x + \frac{1 \cdot 3}{2 \cdot 4} \cdot x^2 - \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \cdot x^3 + \ldots (|x| < 1);$

о) $y = \arcsin x = x + \frac{1}{2} \cdot \frac{x^3}{3} + \frac{1 \cdot 3}{2 \cdot 4} \cdot \frac{x^5}{5} + \ldots (|x| < 1).$

Т20.4 Виконати завдання Т20.3 для заданої функції f(x) на відрізку [a, b] (варіанти а) – о)) без побудови відео (анімації). Взяти для наближення перші m членів ряду.

Замість відео на графіку функції та її наближення залити кольором ділянки неспівпадіння функції та наближення. Окрім цього, побудувати графік функції g(x), яка набуває значення модуля різниці між функцією f(x) та її наближенням.

Порахувати методом Монте-Карло середню похибку наближення як корінь відношення площі фігури (фігур) між кривими графіків функції на відрізку [a, b] до площі охоплюючого прямокутника. В якості охоплюючого прямокутника взяти границі осей, які розраховує matplotlib.

T20.5 Нехай ми маємо послідовність точок (x_0, y_0), (x_1, y_1), …, (x_n, y_n). При цьому, $x_0 < x_1 < \ldots < x_n$. Будемо вважати, що точки y_i є значеннями деякої функції f у точках x_i. Інтерполяцією називається побудова функції f у всіх точках на проміжку [x_0, x_n].
Одним із способів інтерполяції є застосування інтерполяційного поліному Лагранжа, який будується за формулою:

$$P_L(x) = \sum_{k=0}^{n} y_k L_k(x), \quad \text{де } L_k(x) = \prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i}$$

Виконати наближення інтерполяційним поліномом Лагранжа функції sin(x) на відрізку [0, 2π] у (n+1) точці, де $n = 2^k$, k = 2, 3, 4, … (скласти функцію для обчислення $P_L(x)$) Використати масиви numpy.
Зобразити на графіках функції sin(x) та $P_L(x)$. Зберегти відео (виконати анімацію) для різних значень k.

T20.6 Виконати завдання T20.5 без побудови відео (анімації).
Замість відео на графіку функції та її наближення залити кольором ділянки неспівпадіння функції та наближення. Окрім цього, побудувати графік функції g(x), яка набуває значення модуля різниці між функцією sin(x) та її наближенням.
Порахувати методом Монте-Карло середню похибку наближення як корінь відношення площі фігури (фігур) між кривими графіків функції на відрізку [a, b] до площі охоплюючого прямокутника. В якості охоплюючого прямокутника взяти границі осей, які розраховує matplotlib.

T20.7 В умовах завдань T20.5, T20.6, окрім інтерполяції поліномом Лагранжа, виконати також лінійну інтерполяцію функцією sin(x) у тих же n точках. Лінійна інтерполяція – це наближення функції відрізками прямих [(x_i, y_i), (x_{i+1}, y_{i+1})].
На графіках функції та її наближення поліномом Лагранжа та лінійною інтерполяцією залити кольором ділянки неспівпадіння функції та наближення.
Порахувати методом Монте-Карло середню похибку наближення як корінь відношення площі фігури (фігур) між кривими графіків функції на відрізку [a, b] до площі охоплюючого прямокутника. В якості охоплюючого прямокутника взяти границі осей, які розраховує matplotlib.
Порівняти точність лінійної інтерполяції та інтерполяції поліномом Лагранжа.

T20.8 Скласти програму перетворення дійсного вектору за наступним правилом: всі від'ємні компоненти вектору перенести до його початку, а всі інші - до кінця, зберігаючи початкове взаємне розташування як серед від'ємних, так і серед інших компонент. Використати масиви numpy та векторизувати програмний код.

T20.9 Скласти програму обчислення норм дійсної матриці порядку n

а) $\|A\|_1 = \max\limits_{1\le i\le n} \sum\limits_{j=1}^{n} |a_{ij}|$, б) $\|A\|_2 = \max\limits_{1\le j\le n} \sum\limits_{i=1}^{n} |a_{ij}|$

Використати масиви numpy та векторизувати програмний код.

T20.10 Скласти програму, яка перевіряє, чи є задана квадратна матриця з цілих чисел ортонормованою, тобто. такою, в якій скалярний добуток кожної пари різних рядків дорівнює 0, а скалярний добуток кожного рядка на себе дорівнює 1.
Використати масиви numpy та векторизувати програмний код.

T20.11 Скласти програму, що перевіряє чи є задана квадратна матриця з цілих чисел магічним квадратом, тобто такою, в якій суми елементів в усіх рядках і стовпчиках однакові.
Використати масиви numpy та векторизувати програмний код.

T20.12 Скласти програму, що перевіряє чи є задана квадратна матриця
а) верхньою трикутною
б) нижньою трикутною
Використати масиви numpy та векторизувати програмний код.

T20.13 Задані координати n точок на площині $(x_1,y_1),...,(x_n,y_n)$. Знайти номери двох точок, відстань між якими найбільша (вважати, що така пара точок єдина), та саму відстань.
Використати масиви numpy. Точки розмістити у двовимірному масиві 2xn. Побудувати тривимірний масив усіх можливих пар точок. Для побудови використати індексні масиви. Описати векторизовану функцію, яка обчислює відстань між 2 точками.

T20.14 Задані координати n точок на площині $(x_1,y_1),...,(x_n,y_n)$. Знайти номери трьох точок, які утворюють трикутник найбільшого периметру, та сам периметр.
Використати масиви numpy. Точки розмістити у двовимірному масиві 2xn. Побудувати тривимірний масив усіх можливих трійок точок. Для побудови використати індексні масиви. Описати векторизовану функцію, яка обчислює периметр трикутника, що утворений 3 точками.

T20.15 Задані координати n точок на площині $(x_1,y_1),...,(x_n,y_n)$. Знайти кількість рівносторонніх трикутників, утворених цими точками.

Використати масиви numpy. Точки розмістити у двовимірному масиві 2xn. Побудувати тривимірний масив усіх можливих трійок точок. Для побудови використати індексні масиви. Описати векторизовану функцію, яка перевіряє, чи є трикутник, утворений 3 точками, рівностороннім.

T20.16 Скласти програму пошуку найменшого серед найбільших елементів рядків квадратної дійсної матриці порядку *n*, тобто

$$\min_{1 \le i \le n} \max_{1 \le j \le n} \{a_{ij}\}.$$

Використати масиви numpy та векторизувати програмний код.

T20.17 Виконати наближене обчислення π методом Монте-Карло. Для цього наближено обчислити площу півкола радіусом 1 з центром у початку координат. Використати масиви numpy. Зобразити на графіку півколо та охоплюючий прямокутник.

T20.18 Вам пропонують зіграти у таку гру. Ви платите 1 одиницю грошей. Кидають 4 кості. Якщо сума не перевищить 9, Ви отримуєте 10 одиниць грошей. Чи будете Ви у виграші після багаторазового повторення гри?
Розв'язати задачу методом Монте-Карло з використанням масивів numpy.
Векторизувати програмний код.
Гра вважається чесною, якщо сума винагороди дорівнює витраченим грошам, за умови ймовірного виграшу. Справедлива сума винагороди при вкладанні у кожну гру 1 одиниці грошей становить 1/p, де p – ймовірність виграшу.

T20.19 Нехай ви граєте у гру під назвою «крап». Гра полягає у наступному. Ви кидаєте 2 кості. Якщо сума крапок на поверхнях костей буде 7 або 11 (випаде 7 або 11), - ви виграєте одразу. Якщо випаде 2, 3 або 12, - ви програєте одразу. Якщо ж випаде інше число (яке в подальшому називається «ваше число»), - ви продовжуєте кидати кості, поки не випаде 7 (у цьому випадку Ви програєте) або ваше число (у цьому випадку Ви виграєте). Яка ймовірність виграшу у цю гру?
Розв'язати задачу методом Монте-Карло з використанням масивів numpy.
Векторизувати програмний код.

T20.20 Нехай в завдання T20.19 Ви зробили ставку на результат гри у 1 одиницю грошей. У разі програшу, ви програєте свою ставку. Якщо ж Ви виграєте, то виграш визначається наступним чином:

- якщо Ви виграєте одразу, викинувши 7 або 11, Ви отримуєте виграш у розмірі ставки;
- якщо Ваше число 6 або 8 і ви виграєте, Ви також отримуєте виграш у розмірі ставки;
- якщо ж Ваше число 4 або 10 і ви виграєте, Ви отримуєте виграш у розмірі подвійної ставки;

- нарешті, якщо Ваше число 5 або 9 і ви виграєте, Ви отримуєте виграш у розмірі 3/2 ставки.

Скільки грошей в середньому Ви виграєте або програєте за одну гру?
Розв'язати задачу методом Монте-Карло з використанням масивів numpy.
Векторизувати програмний код.

T20.21 Нехай гравець у казино грає у гру, що складається з окремих незалежних раундів. Ймовірність виграшу раунду дорівнює p. Якщо після якогось раунду капітал гравця становить N одиниць грошей, то такий гравець визнається переможцем та видаляється з казино. Яка ймовірність того, що гравець під час гри рано чи пізно втратить всі гроші, якщо він має початковий капітал K?
Розв'язати задачу методом Монте-Карло з використанням масивів numpy. Для моделювання можна вважати, що ймовірність p кратна 0.1, тобто p*10 – ціле число.
Векторизувати програмний код.

T20.22 Ще одна задача Шевальє де Мера, яку він ставив Блезу Паскалю, - це справедливий розподіл грошей між гравцями, якщо гра переривається, не дібігши кінця. Нехай є 2 гравці з однаковим хистом до гри: A та B. Нехай вони грають у гру, у якій за кожен виграний раунд дається один бал. Той, хто набрав n балів, виграє гру та забирає банк. В якій пропорції чесно розділити гроші між гравцями, якщо гру перервано, коли A набрав a балів, а B – b балів. Зрозуміло, що чесним розподіл буде тоді, коли гроші будуть розподілені пропорційно ймовірності виграшу кожного з гравців.
Розв'язати задачу методом Монте-Карло з використанням масивів numpy.

T20.23 У капелюсі є 12 кульок: по 4 кульки червоного, синього та чорного кольору. За один раз витягають 3 кульки. Яка ймовірність того, що з них не менше 2 чорних?
Розв'язати задачу методом Монте-Карло з використанням масивів numpy. Векторизувати програмний код, наскільки можливо.

T20.24 У капелюсі є m*k кульок: по k кульок m кольорів (m > 1). За один раз витягають d кульок (1 < d <= k). Яка ймовірність того, що всі вони одного кольору?
Розв'язати задачу методом Монте-Карло з використанням масивів numpy. Векторизувати програмний код, наскільки можливо.

T20.25 Описати клас Decks, який призначений для моделювання великої кількості випробувань з роздавання гральних карт. Одна колода карт складається максимум з 52 карт (по 13 карт 4 мастей). Гідність карт від 2 до 10, а також валет, дама, король, туз. Масті – піки, трефа, бубни, черви. У тій чи іншій грі може встановлюватись обмеження щодо мінімальної гідності карт (наприклад, починаючи з 7). При одному роздаванні карт колода

тасується випадковим чином та m гравцям роздають по n карт. Інші карти залишаються в колоді. Гравець, якому роздають карти, називається «рукою».
Зовнішнє представлення карти – це кортеж з двох полів, що є рядками (<гідність>, <масть>). Для використання масивів numpy кожну карту у внутрішньому представленні можна закодувати цілим числом: гідність – від 2 (2) до 14 (туз), масть – 1, 2, 3, 4 помножити на 100. Так, наприклад, 9 трефа буде мати код 209. Для отримання масті карти k достатньо виконати k // 100, а для отримання гідності достатньо виконати k % 100.
Клас Decks повинен містити методи для перетворення карти з зовнішнього представлення у внутрішнє та навпаки, метод роздавання карт, метод «фіксованого» роздавання та, можливо, інші методи. Метод роздавання карт повинен повертати тривимірний масив з N випадковим чином розданих колод (m гравцям по n карт всього m*n*N) а також двовимірний масив – залишків N колод. Метод фіксованого роздавання повинен фіксувати карти на першій руці та випадковим чином роздавати їх на всі інші руки.
Застосувати клас Decks для розв'язання задачі: 4 гравцям роздають по 5 карт. Обчислити ймовірність того, що на будь-якій руці опиняться:
а) 4 карти однакової гідності.
б) 5 карт однієї масті, гідність яких після сортування монотонно зростає на 1.
Розв'язати задачу методом Монте-Карло з використанням масивів numpy. Векторизувати програмний код, наскільки можливо.

T20.26 В умовах попередньої задачі, зафіксувати якісь 5 карт на 1 руці, виконати фіксоване роздавання на інші руки та обчислити ймовірність того, що на будь-якій руці, окрім першої, опиняться:
а) 4 карти однакової гідності.
б) 5 карт однієї масті, гідність яких після сортування монотонно зростає на 1.
Розв'язати задачу методом Монте-Карло з використанням масивів numpy. Векторизувати програмний код, наскільки можливо.

T20.27 Застосувати клас Decks для моделювання розкладів при грі у преферанс. У цю гру грають колодою з 32 карт, починаючи з 7, роздають 3 гравцям по 10 карт, 2 карти залишаються у «прикупі». Старшинство карт – у порядку зростання гідності. На основі аналізу власних карт один з гравців може вибороти право визначати гру. Цей гравець бере прикуп, скидає 2 «зайві» карти та оголошує козирну масть, яка «б'є» інші масті, а також кількість взяток, які він зобов'язується взяти. При кожному ході кожен з гравців кладе 1 карту та розігрується 1 взятка, яку забирає старша карта або козирна карта (старша з козирних карт, якщо їх декілька). Кожен з гравців зобов'язаний класти карту тієї масті, з якої зроблено хід. Якщо цієї масті немає, - то козирну карту. Якщо козирної карти немає, то будь-яку карту. Право наступного ходу отримує гравець, який взяв останню взятку.
Для того, щоб при власному першому ході гарантовано взяти всі 10 взяток треба мати у масті, яка буде оголошена козирною, одну з таких комбінацій карт:

- туз, король, дама, валет;
- туз, король, дама та 2 будь-які менші карти;
- туз, король та 4 будь-які менші карти;
- туз та 6 будь-яких менших карт;
- всі вісім карт однієї масті.

Треба також мати всі старші карти в усіх інших наявних мастях (або комбінацію у ще одній масті туза, короля, дами та 2 будь-яких менших карт за умови не більше 5 карт у козирній масті).

Знайти ймовірність наявності на будь-якій руці розкладу, що дозволяє за умови власного першого ходу гарантовано взяти всі 10 взяток:

а) без урахування прикупу;

б) з урахуванням прикупу.

Розв'язати задачу методом Монте-Карло з використанням масивів numpy. Векторизувати програмний код, наскільки можливо.

Т20.28 В умовах попередньої задачі визначити ймовірність розкладів при грі у «мізер». Якщо один з гравців зобов'язується зіграти мізер, це означає зобов'язання за будь-яких умов не взяти жодної взятки. Для того, щоб гарантовано не взяти жодної взятки, за умови не власного першого ходу, треба у кожній наявній масті мати одну з таких комбінацій карт:

- комбінацію з однієї карти 7
- комбінацію з 2 карт 7, 8
- комбінацію з 2 карт 7, 9
- комбінацію з 3 карт: одну з комбінацій з 2 карт плюс 9 (для 7, 8) або 10 або валет
- комбінацію з 4 карт: одну з комбінацій з 3 карт плюс 10 або валет (якщо 10 або валет немає у комбінації з 3 карт) або дама або король
- комбінацію з 5 або більше карт: одну з комбінацій з 4 карт плюс будь-які старші карти

Знайти ймовірність наявності на будь-якій руці розкладу, що дозволяє за умови не власного першого ходу гарантовано не взяти жодної взятки:

а) без урахування прикупу;

б) з урахуванням прикупу.

Розв'язати задачу методом Монте-Карло з використанням масивів numpy. Векторизувати програмний код, наскільки можливо.

Т20.29 В умовах задачі Т20.28 визначити ймовірність розкладів при фіксованому роздаванні. Зафіксувати на першій руці карти так, щоб найкраща потенційно козирна масть складалась з 4 карт а також 2 карти, скинуті після взяття прикупу. Виконати фіксоване роздавання. Знайти ймовірності того, що на другій та третій руці інші 4 карти цієї масті роздано у співвідношенні 4:0, 1:3, 2:2, 3:1, 0:4.

Т20.30 Клас Drunkard2D, що моделює випадковий шлях у двовимірному просторі, реалізований наступним чином:

```
class Drunkard2D:
    '''Клас, що реалізує випадковий шлях у двовимірному просторі ("хода 
п'яниці").


    '''
    def __init__(self, num_drunkards, init_pos = None, is_limited = False, 
bounds = None):
        self._n_d = num_drunkards           #кількість точок
        self._is_limited = is_limited       #чи обмежена область
        self._bounds = bounds               #границі області (прямокутник)

        self._pos = init_pos                #позиції всіх точок
        if self._pos is None:
            #якщо позиції не задано, встановлюємо всі у (0,0)
            self._pos = np.zeros(self._n_d * 2)
            self._pos.shape = (2, self._n_d)
            if self._is_limited:
                #якщо задано границі, встановлюємо всі точки у середину 
області
                xmin, ymin, xmax, ymax = self._bounds
                x = (xmin + xmax) // 2
                y = (ymin + ymax) // 2
                self._pos += np.array([[x], [y]])

        self._dirs = np.array([[-1, 0], [0, -1], [1, 0], [0, 1]]) #можливі 
рухи
        self._dirs = np.transpose(self._dirs) #зручніше мати транспонований 
масив

        self.fig_count = 0  #номер рисунку

        plt.hold(False)     #кожного разу буде малювати нове зображення,
                            #а не доповнювати попереднє


    @property
    def bounds(self):
        '''Властивість границі (читання).'''
        return self._bounds

    @bounds.setter
    def bounds(self, new_bounds):
        '''Властивість границі (встановлення).'''
        self._bounds = new_bounds

    @property
    def pos(self):
        '''Властивість позиції точок (тільки читання).'''
        return self._pos

    def _push_into_bounds(self):
        '''Повернути всі точки у межі області.'''
        xmin, ymin, xmax, ymax = self._bounds
        self.pos[0][self.pos[0] < xmin] = xmin + 1
        self.pos[0][self.pos[0] > xmax] = xmax - 1
        self.pos[1][self.pos[1] < ymin] = ymin + 1
        self.pos[1][self.pos[1] > ymax] = ymax - 1

    def step(self):
        '''Зробити один крок у моделюванні.'''
```

```
        #масив індексів для подальшого формування масиву приростів
        ids = np.random.random_integers(0, 3, self._n_d)
#         print(ids)
#         print(self._dirs)

        #масив приростів чергового кроку
        dxy = self._dirs[:,ids]
#         print(dxy)
        self._pos += dxy
        if self._is_limited:
            self._push_into_bounds()

    def msteps(self, m):
        '''Зробити m кроків у моделюванні.'''
        for i in range(m):
            self.step()

    def plot(self):
        '''Побудувати графік стану моделі.'''
        #set axes
        if self._bounds is None:
            xmin = ymin = -100
            xmax = ymax = 100
        else:
            xmin, ymin, xmax, ymax = self._bounds
        plt.plot(self._pos[0], self._pos[1], 'ob')
        plt.axis([xmin, xmax, ymin, ymax])

    def show(self):
        '''Побудувати та показати графік стану моделі.'''
        self.plot()
        plt.show()


    def savefig(self, path):
        '''Побудувати та зберегти графік стану моделі у файлі.

        path - шлях до файлу, включаючи фінальний символ
        поділу каталогів ('/' або '\').
        Файл має ім'я відповідно масці
        tmpXXXXX.png, деXXXXX - номер рисунку
        '''
        self.fig_count += 1
        fname = "tmp{:0>5}.png".format(self.fig_count)
        self.plot()
        plt.savefig(path + fname)
```

Описати аналогічним чином клас DrunkardND, який моделює випадковий шлях у n-вимірному просторі та зробити аналог класу Drunkard2D його наступником. Для показу розташування частинок виконувати проекцію n-вимірного простору на деяку площину.
Змоделювати процес розповсюдження молекул газу у замкненій області у n-вимірному просторі. Показати цей процес у вигляді відео (анімації).

T20.31 В умовах завдання T20.30 (реалізувати клас DrunkardND) розв’язати наступну задачу. Молекули двох газів розташовані у сусідніх частинах прямокутної області, які поділяються стінкою. Потім стінку прибирають та молекули починають рухатись за правилами випадкового шляху.

Змоделювати рух молекул протягом визначеної кількості кроків. Показати цей процес у вигляді відео (анімації), зобразивши молекули різних газів різними кольорами.

T20.32 В умовах попереднього завдання T20.30 (реалізувати клас DrunkardND) розв'язати наступну задачу. Змоделювати рух множини частинок у одновимірному просторі, починаючи з точки 0. Провести ряд випробувань для визначеної кількості кроків: 10, 100, 1000, ... Побудувати графіки залежності максимальної та середньої відстані, на яку віддаляються частинки, в залежності від кількості кроків. Використати звичайну та логарифмічну шкалу.