УДК 373.3/.5.016:5]:004
DOI https://doi.org/10.32782/cusu-pmtp-2024-2-3

МЕТОДИКА НАВЧАННЯ МАТЕМАТИКИ З PYTHON НА ПРИКЛАДІ ТЕМИ «ЧИСЛОВІ ПОСЛІДОВНОСТІ»

Ботузова Юлія Володимирівна,
доктор педагогічних наук, доцент,
доцент кафедри математики та цифрових технологій
Центральноукраїнського державного університету імені Володимира Винниченка
ORCID ID: 0000-0002-1313-0010

Стаття присвячена розкриттю методичних особливостей вивчення теми «Числові послідовності» у шкільному курсі математики за допомогою інструментів програмування, зокрема використання мови Python. Визначено актуальність теми та встановлено завдання дослідження. Наведено огляд науково-популярної літератури з теми дослідження, зокрема в основу дослідження покладені посібники американських вчителів та науковців П. Фаррела (2019) та А. Саха (2015), які розробляли методику вивчення шкільної математики з Python.

Автор дотримується позиції, що вивчення програмування повинно займати значуще місце в освітньому процесі, оскільки досвід багатьох країн свідчить, що освоєння принципів кодування і вивчення мов програмування сприяє розвитку логічного та креативного мислення.

Під час дослідження було проаналізовано та систематизовано задачний ряд шкільного курсу математики з теми «Числові послідовності» за кількома альтернативними підручниками. У статті використано задачі з підручника «Алгебра» для 9 класу авторського колективу А.Г. Мерзляк, В.Б. Полонський, М.С. Якір. При цьому виокремлено 7 різних типів задач, для вирішення яких можливо створити шаблони програм, або коротко охарактеризувати алгоритм роботи програми, написаної на мові Python.

У процесі роботи над дослідженням, встановлено, що вивчення математики та програмування можна поєднувати, адже для написання програми учням, у першу чергу, треба знати теоретичні основи, властивості числових послідовностей, а рутинну обчислювальну роботу виконуватиме програма.

Стаття має практичний характер, оскільки вона включає в себе реалізацію алгоритмів та програм для вивчення властивостей числових послідовностей. Програмна реалізація проводиться з використанням мови програмування Python, що дозволяє досліджувати та аналізувати числові послідовності шляхом ефективного використання інструментів цієї мови.

Ключові слова: *методика навчання математики, числова послідовність, програмування, Python, алгоритм.*

Botuzova Yuliia. Methods of teaching mathematics with Python on the example of the topic "Numerical sequences"

The article is devoted to a disclosure of the methodological features of studying the topic "Numerical sequences" in the school course of mathematics with the help of programming tools, in particular the use of the Python language. The relevance of the topic is determined and the objectives of the research are established. A review of popular science literature on the topic of research is provided. In particular, the study is based on the manuals of American teachers and scientists P. Farrell (2019) and A. Saha (2015), who developed the methods of teaching school mathematics with Python.

The author adheres to the position that the study of programming should occupy a significant place in the educational process. After all, the experience of many countries shows that mastering the principles of coding and learning programming languages contributes to the development of logical and creative thinking.

In the course of the study, the problem series of the school mathematics course on the topic "Numerical sequences" was analyzed and systematized according to several alternative textbooks. The article uses problems from the textbook "Algebra" for the 9th grade by the team of authors A.G. Merzlyak, V.B. Polonsky, M.S. Yakir. Seven different types of problems are distinguished, for the solution of which it is possible to create program templates, or briefly describe the algorithm of a program written in Python.

In the process of working on the research, it was found that the study of mathematics and programming can be combined. To write a program, students, first of all, need to know the theoretical foundations, the properties of numerical sequences, and routine computational work will be performed by the program.

The article is of a practical nature, since it includes the implementation of algorithms and programs for studying the properties of numerical sequences. The software implementation is carried out using the Python programming language, which allows you to explore and analyze numerical sequences by effectively using the tools of this language.

Key words: *methods of teaching mathematics, numerical sequence, programming, Python, algorithm.*

Вступ. Із розвитком технологій та появою великих обсягів даних виникає необхідність вдосконалення методів обробки та аналізу інформації. Однією з ключових складових цього процесу є числові послідовності, які знаходять широке застосування в різних областях, від математики, фізики та економіки до інформаційних технологій.

У шкільному курсі математики 9 класу вивчається тема «Числові послідовності», в якій прослідковується взаємозв'язок між задачами програмування та завданнями, запропонованими у підручнику алгебри. Дослідження властивостей числових послідовностей виходить за межі традиційних математичних методів і стає актуальним завданням в контексті сучасних вимог до обробки даних. Аналіз та вивчення цих послідовностей дозволяють розкрити закономірності, що сприяє подальшому вдосконаленню алгоритмів обробки даних та оптимізації різноманітних процесів. Однією з найефективніших та універсальних мов програмування для проведення таких досліджень є Python. Це можна пояснити її відносною простотою, паралельною потужністю та здатністю вирішувати різноманітні завдання, що виникають на практиці. У Python присутня велика кількість відкритих бібліотек, що робить його ефективним інструментом для аналізу та обробки даних. Поєднання математичної теорії та інструментів програмування відкриває нові можливості для вирішення складних завдань у сфері науки та техніки.

Мета дослідження: розкрити методичні особливості вивчення теми «Числові послідовності» у шкільному курсі математики за допомогою інструментів програмування, зокрема використання мови Python.

Аналіз досліджень і публікацій. На думку вченого та експерта в галузі STEM-освіти Дж. Брауна [1] з Каліфорнії, кодування відіграє ключову роль у встановленні взаємозв'язку між математикою та її застосуванням у STEM-освіті. У своїх рекомендаціях для вчителів та учнів він підкреслював, що алгоритмізація та кодування не обмежується лише роботою програмістів, а це, скоріш за все, навичка мислення, яку ми використовуємо щоденно, навіть під час приготування бутерброду на сніданок.

Автор посібника «Doing Math with Python» A. Саха [2] вказує на тісний взаємозв'язок між математикою та програмуванням, пропонуючи вивчати деякі теми шкільної математики з використанням комп'ютера та програмування на мові Python. Такий підхід робить навчання математиці та програмуванню захоплюючим і корисним.

П. Фаррелл, американський вчитель математики та інформатики, автор посібника «Math Adventures with Python» [3], вважає необхідним під час навчання формувати в учнів розуміння того, що математика – це не просто виконання кроків алгоритму для вирішення рівняння. Здобувачі освіти мають усвідомити, що вивчення математики за допомогою програмування дозволяє кількома способами розв'язувати цікаві проблеми, з багатьма непередбаченими помилками та можливостями для вдосконалення. Саме у цьому вбачається різниця між шкільною математикою та справжньою. Наразі комп'ютери можуть виконувати більшість обчислень за нас, тож підходи до вивчення математики необхідно змінювати.

Український вчитель математики та інформатики Зеленяк О.П. [4] (м. Олександрія), ще у 2006 р. висловлював думку про важливість успішного поєднання математичного моделювання з програмуванням, щоб ознайомити учнів із втіленням окремих функцій у спеціалізованих середовищах, при цьому використовував Turbo Pascal. У більш пізній публікації Зеленяка О.П. [5] висвітлено погляд, що для цілісної реалізації взаємодії між математикою та інформатикою ефективним буде використання наборів завдань, які стосуються обох

предметів і передбачають інтеграцію знань з алгебри та математичного аналізу, геометрії та інформатики.

Розвиток блокових систем кодування, таких як Scratch, дозволяє учням вивчати основи програмування ще в початковій школі. Ця нова технологічна модель сприяє кращому розумінню сутності кодування та його функціонування. Зацікавленість сучасної молоді програмуванням визначається, зокрема, очікуваннями експертів щодо зростання попиту на професію програміста в найближчому майбутньому.

Вивчення програмування повинно займати значуще місце в освітньому процесі. Досвід багатьох країн свідчить, що освоєння принципів кодування і вивчення мов програмування сприяє розвитку логічного та креативного мислення. Важливо зауважити, що програмування розвиває творчість, надаючи навичок у пошуку рішень, оскільки розв'язання практичних завдань може вимагати різних підходів [6].

Python на сьогоднішній день є оптимальною мовою програмування для вивчення основ алгоритмізації з наступних причин [7]:

– має простий синтаксис, який робить програмний код легко зрозумілим і читабельним;

– це об'єктно-орієнтована мова програмування високого рівня, спрямована на розв'язання різноманітних завдань;

– є кросплатформенною мовою, що дозволяє створювати програми, які працюють на різних операційних системах;

– має багато готових бібліотек процедур для використання у власних програмах, що дозволяє швидко розробляти складні програми;

– підтримує різні парадигми програмування;

– мова програмування Python також має потужну стандартну бібліотеку, яку користувач може розширювати власними бібліотеками та використовувати бібліотеки інших користувачів.

Також важливо вказати, що мова Python від самого початку була розроблена відповідно до парадигми об'єктно-орієнтованого програмування, але вона також ефективно використовується для структурного і функціонального програмування.

Матеріали та метод. Під час дослідження використовувалися шкільні підручники «Алгебра» для 9 класу ЗЗСО, зокрема книга авторського колективу А.Г. Мерзляк, В.Б. Полонський, М.С. Якір [8].

У процесі аналізу завдань підручників, зверталась увага на типізацію задач з метою їх класифікації та розробки алгоритмів їх розв'язування з подальшою реалізацією в програмному коді. В результаті чого було виділено 7 різних типів задач.

Тип 1. Запис кількох перших членів послідовності заданої певним правилом (описово або аналітично).

Для реалізації розв'язування таких задач у Python доцільно скористатися циклом **for**, за яким здійснюється перебір усіх елементів з деякого набору.

Тип 2. Знаходження членів деякої послідовності з конкретними номерами.

Задачі типу 2 можна вирішувати також за допомогою циклу **for**, де перебір вестиметься із конкретно заданих чисел – номерів членів послідовності. У випадку, коли за умовою задачі необхідно знайти лише один конкретний член послідовності, можна виконати звичайне обчислення, записавши формулу та підставивши значення *n* – номера члена послідовності.

Тип 3. Встановити чи є задане число членом послідовності. Якщо є, то вказати його номер.

Вирішення задачі типу 3 потребує застосування умовного оператора **if-else,** який використовується у разі необхідності розгалуження процесу обчислень. Тобто при наявності кількох операторів, порядок їх виконання визначається в залежності від виконання певних, наперед заданих, умов.

Тип 4. Знайти кількість додатних/від'ємних членів деякої послідовності.

Тип 5. Знайти перший додатний/від'ємний член деякої послідовності.

Виконання завдань обох типів 4 та 5 можливе, наприклад, із застосуванням циклу з передумовою (цикл **while**), який можна назвати універсальним циклом у мові Python, але працює він достатньо повільно. Відповідно до цього циклу виконується зазначений автором програми набір інструкцій до тих пір, поки умова циклу залишається істинною. У процесі виконання циклу **while** спочатку оцінюється логічний вираз. Якщо отримане значення є істинним (тобто відповідає умові, яка аналогічна умові в операторі **if**), тоді виконується тіло циклу, і після цього відбувається повернення до перевірки логічного виразу. Цей процес повторюється до тих пір, поки значення логічного виразу не стане хибним. Після цього виконання циклу завершується, і програма переходить до наступної інструкції після тіла циклу **while**. Якщо ж логічний вираз вже на початку виявиться хибним, тіло циклу не виконується ні разу.

Для завершення роботи циклу необхідно включити в його тіло оператор, що впливає на значення логічного виразу. Крім того, логічний вираз повинен бути коректним, тобто його значення має бути визначеним ще до першої перевірки. Зазвичай цикл **while** використовується, коли неможливо точно визначити кількість проходів циклу.

Тип 6. Встановити чи є числова послідовність арифметичною/геометричною прогресією.

Задачі такого типу потребують пошуку різниці/частки двох сусідніх членів послідовності порівняння отриманих значень. Тому це задача перебору та перевірки виконання умови рівності різниць/часток, а отже можна застосовувати цикл **for** та умовний оператор **if-else.**

Тип 7. Знаходження суми кількох перших членів арифметичної/геометричної прогресії.

Виконання задачі не потребує побудови складних циклів. Здійснюється обчислення необхідних вихідних даних для заповнення формули суми *n* перших членів арифметичної/геометричної прогресії. Зокрема, це числове значення першого члена, різниці/знаменника прогресії та кількості членів, суму яких необхідно знайти.

Результати. Під час дослідження було здійснено розробку програмних рішень для усіх типових завдань, пов'язаних з числовими послідовностями, з метою полегшення їх розуміння та вирішення для учнів. Передбачено створення зрозумілих та оптимальних алгоритмів на мові Python, які відображатимуть ключові аспекти та правила роботи з числовими послідовностями у шкільному курсі математики. Наведемо приклади реалізації програмних рішень для кожного з виділених типів задач.

Задача 1.1. Послідовність (a_n) є послідовністю трицифрових чисел, кратних числу 5, узятих у порядку зростання. Знайдіть перших шість членів цієї послідовності.

Розв'язання: спершу необхідно визначити найменше трицифрове число, що кратне 5 – це число 100. Отже, $a_1 = 100$. Далі, враховуючи, що числа послідовності кратні 5, записуємо: $a_2 = 105$, $a_3 = 110$, $a_4 = 115$, $a_5 = 120$, $a_6 = 125$.

Реалізовуємо процес розв'язання за допомогою коду програми, автоматизуючи вивід одразу усіх шести членів послідовності (рис. 1):

```
def numbers(n):
    start = 100  #найменше трицифрове
число, кратне 5
    sequence = [start + 5*i for i in
range(n)]
    return sequence
n_values = [1, 2, 3, 4, 5, 6]
a_n_values = numbers(6)
for n, a_n in zip(n_values, a_n_values):
    print(f"n = {n}, a_n = {a_n}")
```

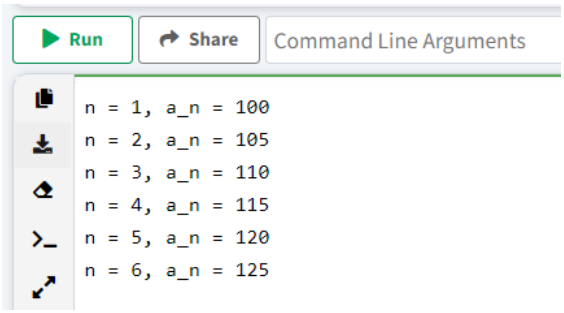


Рис. 1. Програмний код для задачі 1.1. та результат роботи програми

Задача 1.2. «Перший член геометричної прогресії дорівнює $-\frac{1}{27}$, а знаменник дорівнює 3. Знайдіть п'ять перших членів цієї прогресії [8, с. 179]».

Розв'язання: за означенням геометричної прогресії як послідовності чисел, кожен наступний член якої, починаючи з другого, дорівнює попередньому помноженому на одне й те саме число. Це число q – знаменник геометричної прогресії. Для нашої задачі $b_1 = -\frac{1}{27}$ та $q = 3$. У записі коду програми враховуємо, що деякі члени послідовності будуть звичайними дробами, тому підключаємо модуль «Fraction». Роботу програми представлено на рис. 2.

```
from fractions import Fraction
a_1 = Fraction(-1, 27)
q = 3
sequence = [a_1]
for _ in range(4):
    next_member = sequence[-1] * q
    sequence.append(next_member)
for member in sequence:
    print(member)
```

Рис. 2. Програмний код для задачі 1.2. та вивід результатів

Задачі типу 2 передбачають знаходження елемента послідовності, який має конкретний номер. Для прикладу розглянемо декілька задач цього типу.

Задача 2.1. «Послідовність (x_n) задано формулою n-го члена $x_n = 3n+1$. Знайдіть x_1, x_7, x_{13}».

Програму створено так, щоб у інтерактивному режимі можна було вносити номер члена послідовності, який нас цікавить і, в результаті її роботи, отримати відповідь. При цьому враховано, що записаний користувачем програми номер повинен бути лише натуральним числом. Наприклад, якщо користувач на запит: «Введіть номер члена послідовності» запише число «-7,5», то отримає на екрані ValueError, через те, що функція працює лише з конкретним діапазоном значень (рис. 3).

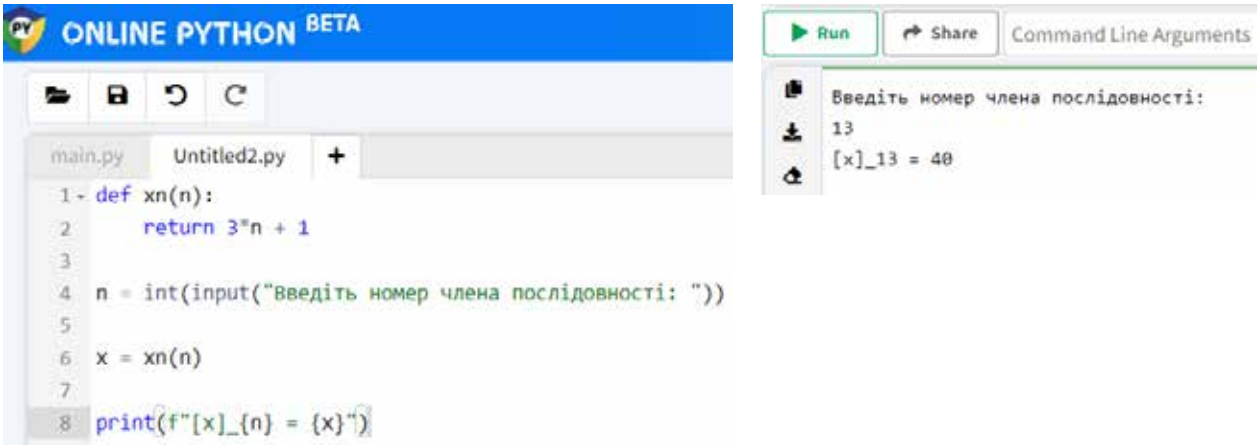


Рис. 3. Програмний код та демонстрація інтерактивного вікна програми

Задача 2.2. «Перший член арифметичної прогресії (a_n) дорівнює 17, а різниця прогресії становить -2. Знайдіть: a_4, a_{15}, a_{60} [8, с. 162]».

Вирішимо цю задачу за допомогою циклу **for** (рис. 4), перебираючи усі значення змінної *n,* які попередньо визначимо з умов задачі. Для задачі 2.2 це номери: 4, 15, 60. Для універсальності користування програмою, зміні підлягатимуть рядки 1 та 2 – початкові умови задачі та рядок 4 – масив із номерами членів послідовності, які треба обчислити.

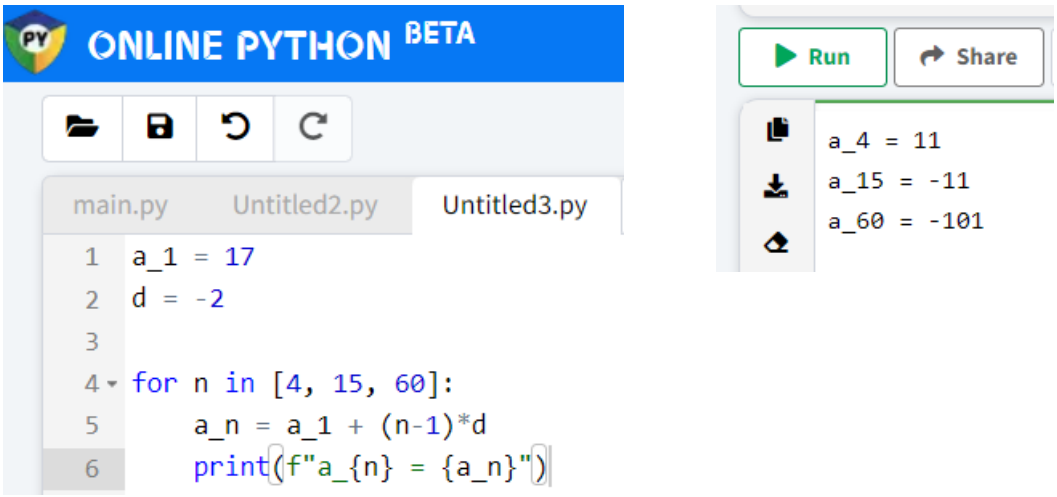


Рис. 4. Програмний код, що легко змінюється та результат виконання

Скориставшись попереднім кодом (рис. 4) як шаблоном, можна вирішити задачу на знаходження членів геометричної прогресії. Для прикладу: *Задача 2.3.* «Перший член геометричної прогресії $b_1 = \frac{1}{125}$, а її знаменник q=5. Знайдіть b_4, b_7 [8, с. 179]».

У записі програмного коду (рис. 5), передбачено також обчислення b_{12} та b_{25}, отримати значення яких без калькулятора достатньо складно.

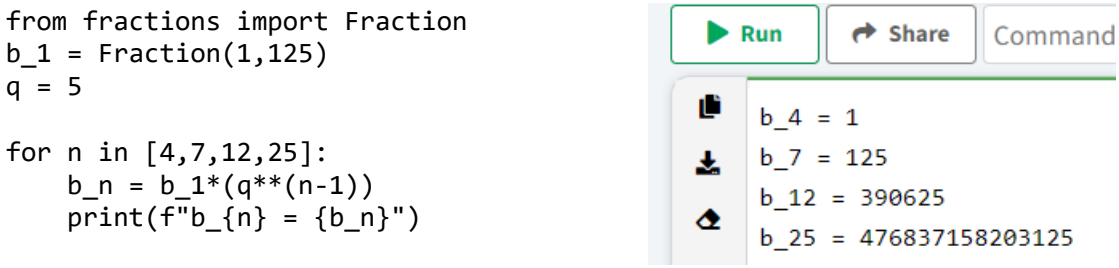


Рис. 5. Програмний код та результат роботи програми по обчисленню членів геометричної прогресії

Перейдемо до розгляду задач типу 3, які передбачають встановлення належності деякого числа заданій послідовності.

Задача 3.1. «Послідовність задана аналітично $x_n = n^2 - 4$. Чи є членом цієї послідовності число 5; 16; 77? Якщо є, то вкажіть його номер у цій послідовності?».

Для реалізації розв'язання цієї задачі у діалоговому режимі, написано програму [9], яка здійснює запит «Введіть число для перевірки» та після запису числа користувачем, видає відповідь (рис. 6).

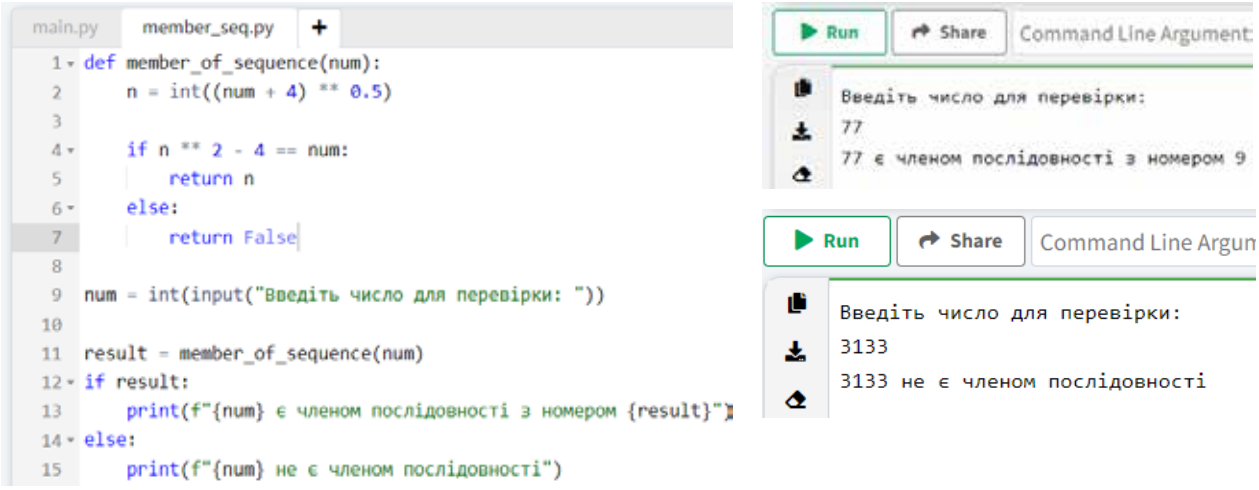


Рис. 6. Програма та результат «діалогу» з нею

Як бачимо з рис. 6, число 77 є 9-им членом заданої послідовності, а 3133 не є членом цієї послідовності. Програма дозволяє перевіряти будь-яке число, будь-яку кількість разів. Програмний код універсальний: у разі розв'язання типової задачі – змінюємо аналітичну формулу загального члена послідовності та продумуємо формулу для виводу n (рядки 2 та 4 програми). Наприклад, для з*адачі 3.2.* «Перевірити, чи є число * членом арифметичної прогресії $a_n = 6 + 7n$», програмний код виглядатиме так (рис. 7):

```
def member_of_sequence(num):
    n = int((num - 6) / 7)

    if n*7 + 6 == num:
        return n
    else:
        return False

num = int(input("Введіть число для перевірки:
"))

result = member_of_sequence(num)
if result:
    print(f"{num} є членом послідовності з
номером {result}")
else:
    print(f"{num} не є членом послідовності")
```

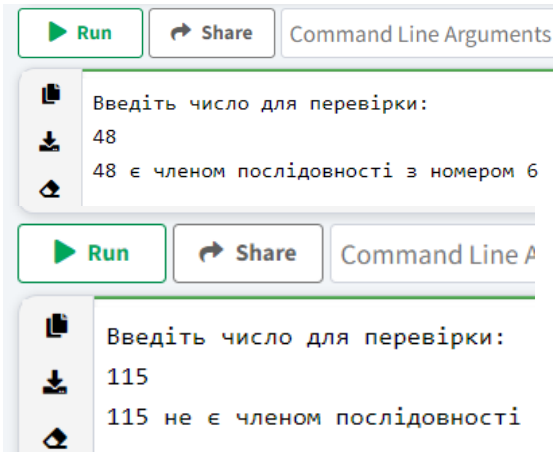


Рис. 7. Програма-шаблон, використана для виконання типової задачі

Варто зазначити, що використання шаблонів програм дозволяє навчитися програмувати [10], адже у процесі внесення змін до програми відбувається аналіз кроків програми та логіки її побудови.

Під час роботи з наступними типами задач також можливе використання шаблонів, які містять цикл **while.** Формулювання умов задач передбачає знаходження кількості членів заданої послідовності, які мають сталий знак або ж вказати номер першого від'ємного/додатного члена послідовності. Написання програми для вирішення таких завдань передбачає, що в коді буде

враховане перебирання послідовних членів послідовності, допоки не виконається необхідна умова. Наприклад, шукаючи число усіх від'ємних членів послідовності, перебираємо їх підряд та кожного разу, беручи від'ємний, додаємо одиницю до певної визначеної в програмі змінної, яка є шуканою. Як тільки натрапляємо перебором на додатне значення, програма зупиняється, а у відповідь виводиться знайдете на попередньому кроці значення змінної.

Прикладами таких задач є:

Задача 4.1. «Скільки від'ємних членів містить послідовність (x_n), яка задана формулою $x_n = 6n - 50$? [8, с. 155]?»

Задача 4.2. «Скільки додатних членів містить арифметична прогресія 5,2; 4,9; 4,6;… [8, с. 163]?»

Їх розв'язання за допомогою програм представлено на рис. 8. (задача 4.1. – зліва, задача 4.2 – справа).

```
n = 1
negative = 0

while True:
    x_n = 6 * n - 50

    if x_n < 0:
        negative += 1

    n += 1

    if x_n >= 0:
        break

print(f"{negative} членів послідовності")
```

Run Share Command

8 членів послідовності

```
first_term = 5.2
difference = -0.3
count = 0

while first_term > 0:
    count += 1
    first_term += difference

print("Кількість додатних членів у прогресії:", count)
```

Run Share Command Line Arguments

Кількість додатних членів у прогресії: 18

Рис. 8. Програми із циклом while для вирішення задач типу 4

Задачі типу 5 розв'язуються аналогічним чином, тільки, наприклад, шукаючи перше від'ємне число, ми рахуємо усі попередні невід'ємні члени послідовності, а коли беремо додатній, то вказуємо саме його номер, не повертаючись до попереднього кроку.

Прикладами задач типу 5 є:

Задача 5.1. «Знайдіть номер першого від'ємного члена послідовності (y_n), заданої формулою $y_n = 38 - 3n$ [8, с. 155]».

*Задача 5.2. «*Який номер має перший додатний член арифметичної прогресії -10,2; -9,5; -8,8;… [8, с. 163]?».

Розв'язання обох задач представлено на рис. 9. (задача 5.1. – зліва, задача 5.2. – справа).

```
def find():
    n = 1
    while True:
        y_n = 38 - 3*n #умова задачі
        if y_n < 0:
            return n
        else:
            n += 1
n = find()
print(f"Номер першого від'ємного члена послідовності: {n}")
```

Run

```
Номер першого від'ємного члена послідовності: 13
```

```
def find():
    n = 1
    while True:
        y_n = -10.9+0.7*n #умова задачі
        if y_n > 0:
            return n
        else:
            n += 1
n = find()
print(f"Номер першого додатного члена прогресії: {n}")
```

Run

```
Номер першого додатного члена прогресії: 16
```

Рис. 9. Програми із циклом while для вирішення задач типу 5

Наступні типові задачі передбачають визначення чи є числова послідовність арифметичною або геометричною прогресією. Для розв'язання таких задач достатньо знати означення цих прогресій та їх властивості, а саме: різниця арифметичної прогресії $d = a_n - a_{n-1}$ або $d = \frac{a_n - a_m}{n-m}$; знаменник геометричної прогресії $q = \frac{b_n}{b_{n-1}}$ або $q^{n-m} = \frac{b_n}{b_m}$.

Задача 6.1. «Чи є арифметичною прогресією послідовність (у разі ствердної відповіді вказати різницю прогресії) [8, с. 162]: а) 24; 22; 20; 18; б) 16; 17; 19; 23; в) -3; 2; 7;12».

Задача 6.2. «Серед наведених послідовностей укажіть геометричні прогресії та знаменник кожної з них [8, с. 178]: а) 2; -2; 2; -2; б) 4; 8; 16; 32; в) 10; 20; 30; 40».

Розв'язання наведених вище задач типу 6, реалізовано програмним кодом із застосуванням циклу **for** для перебору елементів заданих послідовностей та умовного оператора **if-else**, який розгалужує умову на 2 вітки: якщо виконується рівність знайдених різниць чи часток двох послідовних членів послідовностей, то в результаті виводиться ствердна відповідь, якщо ж умова рівності не виконується – то результат, що послідовність не є прогресію. На рис. 10 представлено розв'язання задачі 6.1. (зліва) та задачі 6.2. (справа).

Останній тип задач, що був виділений у шкільних підручниках з теми «Числові послідовності» – це задачі на знаходження суми кількох перших членів арифметичної чи геометричної прогресій. Їх розв'язання не потребує написання складних програм, лише дозволяє автоматизувати обчислення. Для прикладу наведемо одну з таких задач (рис. 11): *Задача 7.1*. «Знайти суму дванадцяти перших n членів геометричної прогресії (b_n) зі знаменником q: 1) $b_1 = 1, q = 2, n = 9$; 2) $b_1 = 15,\ q = \frac{2}{3}, n = 3$; 3) $b_1 = 18, q = -\frac{1}{3}, n = 5$ [8, с. 187]».

```
#послідовності
sequences = {
    "sequence1": [24, 22, 20, 18],
    "sequence2": [16, 17, 19, 23],
    "sequence3": [-3, 2, 7, 12]
}
for name, sequence in sequences.items():
    #різниця між сусідніми елементами
    diffs = [sequence[i+1] - sequence[i]
for i in range(len(sequence) - 1)]

    #Перевіряємо, чи є різниця постійною
    if len(set(diffs)) == 1:
        print(f"{name}  є  арифметичною
прогресією з різницею {diffs[0]}")
    else:
        print(f"{name} не є арифметичною
прогресією")
```

Run

```
sequence1 є геометричною прогресією зі
знаменником -1.0
sequence2 є геометричною прогресією зі
знаменником 2.0
sequence3 не є геометричною прогресією
```

```
#послідовності
sequences = {
    "sequence1": [2, -2, 2, -2],
    "sequence2": [4, 8, 16, 32],
    "sequence3": [10, 20, 30, 40]
}
for name, sequence in
sequences.items():
    #частка сусідніх елементів
    frac = [sequence[i+1]/sequence[i]
for i in range(len(sequence) - 1)]

    #Перевіряємо, чи є частка постійною
    if len(set(frac)) == 1:
        print(f"{name} є геометричною
прогресією зі знаменником {frac[0]}")
    else:
        print(f"{name} не є
геометричною прогресією")
```

Run

```
sequence1 є геометричною прогресією зі
знаменником -1.0
sequence2 є геометричною прогресією зі
знаменником 2.0
sequence3 не є геометричною прогресією
```

Рис. 10. Програми із умовним оператором if-else для вирішення задач типу 6

```
from sympy import *

init_printing()

def sum(b_1, q, n):
    if q == 1:
        return "ERROR"
    else:
        return simplify(b_1 * (q**n - 1) / (q - 1))

#1
print(sum(1, 2, 9))

#2
print(sum(Rational(15), Rational(2, 3), 3))

#3
print(sum(Rational(18), Rational(-1, 3), 5))
```

Run

```
511.000000000000
95/3
122/9
```

Рис. 11. Програма для знаходження суми *n* перших членів геометричної прогресії

Окрім типових задач, у темі «Числові послідовності» дуже багато нетипових. Розглянемо декілька з них:

Задача 8. «Знайдіть суму 33 перших членів арифметичної прогресії (a_n), якщо $a_3 + a_5 + a_{13} = 33$ та $a_{15} - a_8 - a_{10} = -1$ [8, с. 169]».

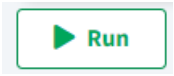
```
from sympy import symbols, solve

a_1, d = symbols('a_1 d')

sol = solve((3*a_1 + 18*d - 33, -a_1 - 2*d + 1),
(a_1, d))

n = 33
S_n = n/2 * (2*sol[a_1] + (n - 1)*sol[d])

print(S_n)
```




```
1188.00000000000
```

Рис. 12. Програма до задачі 8

Для розв'язання цієї задачі написано коротку програму (рис. 12), яка розв'язує систему рівнянь з двома невідомими та шукає суму за формулою $S_n = \frac{2a_1 + d(n-1)}{2} n$. Результат обчислень, здійснений програмою на рис. 12 справа.

Задача 9. «Які чотири числа необхідно поставити між числами 4 та -5, щоб вони разом із даними числами утворили арифметичну прогресію? [8, с. 163]».

Виконання задачі базується на властивості, описаній вище $d = \frac{a_n - a_m}{n - m}$. Продемонструємо програмний код та результат роботи програми на рис. 13.

```
a_1 = 4

a_n = -5

n = 6


d = (a_n - a_1) / (n - 1)


num2 = [a_1 + i * d for i in range(1, n-1)]


num2 = [round(num, 1) for num in num2]


print(f"числа, які треба вставити, щоб вони утворювали арифметичну прогресію:
{num2}")
```

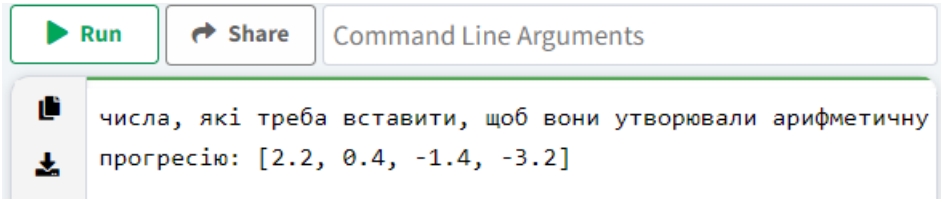


Рис. 13. Програмний код та результат його виконання до задачі 9

Задача 10. «Розв'яжіть рівняння: 1) $11+19+27+...+(8n+3)=470$, де n –натуральне число; 2) $1+5+9+...+x=630$, де x – натуральне число».

Обидва рівняння – це сума деякої кількості членів арифметичної прогресії, тому застосувавши відому формулу, знайдемо шукані невідомі за допомогою Python. Програма з використанням циклу **for** та умовного оператора **if** одразу розв'язує два рівняння, що є в умові завдання та виводить на екран відповіді (рис. 14).

```
from sympy import symbols, solve

def progression(a1, d, S):
    n, x = symbols('n x')

    eq1 = n/2 * (2*a1 + (n - 1) * d) - S  #Формула суми арифметичної прогресії
    eq2 = x - (a1 + (n - 1) * d)

    solutions = solve((eq1, eq2), (n, x))

    for sol in solutions:
        n_sol, x_sol = sol
        if n_sol.is_integer and n_sol > 0 and x_sol.is_integer and x_sol > 0:
            return int(n_sol), int(x_sol)

n1, x1 = progression(11, 8, 470)
n2, x2 = progression(1, 4, 630)

print(f"Розв'язки для першого рівняння: n = {n1}")
print(f"Розв'язки для другого рівняння: x = {x2}")
```

Run

```
Розв'язки для першого рівняння: n = 10
Розв'язки для другого рівняння: x = 69
```

Рис. 14. Програма для розв'язку нестандартних рівнянь (задача 10)

Висновки. Із розвитком Scratch і блокових систем кодування учні ще в початковій школі з легкістю вивчають основи програмування. Нова технологічна модель допомогла молоді зрозуміти тонкощі кодування та його функціонування. Зацікавленість сучасних підлітків програмуванням у першу чергу пов'язана із прогнозами експертів щодо неабиякої затребуваності професії програміста в найближчому майбутньому. З власного досвіду та досвіду колег у навчанні математики старшокласників (9–11 класи) зазначимо, що сучасні підлітки, які зацікавлені програмуванням, найчастіше самі спонукають вчителя до пошуку нових підходів у навчанні математики. Тож, для розкриття методичних особливостей вивчення теми «Числові послідовності» у шкільному курсі математики за допомогою інструментів програмування, зокрема використання мови Python, було: проаналізовано та систематизовано задачі із теми «Числові послідовності» курсу математики основної школи та можливості їх відображення у програмному коді, визначено функціональні можливості програмування для розв'язування задач шкільного курсу математики з теми «Числові послідовності».

Література:

1. Brown G. Teaching STEM practice with coding. URL: https://www.hand2mind.com/blog/teach-stem-practices-with-coding (дата звернення: 25.09.2023).

2. Saha A. Doing Math with Python. San Francisko, 2015. 244 p.

3. Farrell P. Math Adventure with Python. San Francisco, 2019. 347 p.

4. Зеленяк О. П. Інтегровані уроки з математики та інформатики в класах з поглибленим вивченням цих предметів. *Комп'ютер у школі та сім'ї.* 2006. № 5. С. 16–18.

5. Зеленяк О.П. Технології застосування середовищ динамічної геометрії. *Інформаційні технології і засоби навчання*, 2013, Том 36, № 4. С. 40–56.

6. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Чернігів: ФОП Баликіна С.М., 2020. 180 с.

7. Кобильник Т.П., Когут У.П., Жидик В.Б. Методичні аспекти вивчення основ алгоритмізації і програмування мовою Python у шкільному курсі інформатики у старших класах. *Фізико-математична освіта*, 2021. Вип. 5(31). С. 36–44.

8. Мерзляк А.Г., Полонський В.Б., Якір М.С. Алгебра: підруч. для 9 кл. Харків: Гімназія, 2017. 272 с.

9. Ботузова Ю.В. Використання програмування на мові Python під час вивчення математики як STEM-підхід. Актуальні аспекти розвитку STEAM-освіти в умовах євроінтеграції: збірник матеріалів Міжнародної науково-практичної інтернет-конференції (м. Кропивницький, 21 квітня 2023 року). Кропивницький: ДонДУВС, 2023. С. 66–69.

10. Бондаренко М.Ф., Білоус Н.В., Руткас А.Г. Комп'ютерна дискретна математика: Підручник. Харків: «Компанія СМІТ», 2004. 480 с.

References:

1. Brown, G. Teaching STEM practice with coding. Retrieved from: https://www.hand2mind.com/blog/teach-stem-practices-with-coding [in English].

2. Saha, A. (2015). Doing Math with Python. San Francisko. 244 p. [in English].

3. Farrell, P. (2019). Math Adventure with Python. San Francisco. 347 p. [in English].

4. Zeleniak, O.P. (2006). Intehrovani uroky z matematyky ta informatyky v klasakh z pohlyblenym vyvchenniam tsykh predmetiv [Integrated lessons in mathematics and computer science in classes with in-depth study of these subjects]. *Kompiuter u shkoli ta simi*, 5, 16–18 [in Ukrainian].

5. Zeleniak, O.P. (2013). Tekhnolohii zastosuvannia seredovyshch dynamichnoi heometrii [Technologies for the application of dynamic geometry environments]. *Informatsiini tekhnolohii i zasoby navchannia*, Vol. 36, № 4, 40–56 [in Ukrainian].

6. Kostiuchenko, A.O. (2020). *Osnovy prohramuvannia movoiu Python: navchalnyi posibnyk [Python Programming Fundamentals: Tutorial]*. Chernihiv: FOP Balykina S.M. 180 p. [in Ukrainian].

7. Kobylnyk, T.P., Kohut, U.P., & Zhydyk, V.B. (2021). Metodychni aspekty vyvchennia osnov alhorytmizatsii i prohramuvannia movoiu Python u shkilnomu kursi informatyky u starshykh klasakh [Methodological Aspects of Studying the Basics of Algorithmization and Python Programming in the School Course of Informatics in High School]. *Fizyko-matematychna osvita*, 5(31), 36–44 [in Ukrainian].

8. Merzliak, A.H., Polonskyi, V.B., & Yakir, M.S. (2017). *Alhebra: pidruch. dlia 9 kl. [Algebra: Textbook]*. Kharkiv: Himnaziia. 272 p. [in Ukrainian].

9. Botuzova, Yu.V. (2023). Vykorystannia prohramuvannia na movi Python pid chas vyvchennia matematyky yak STEM-pidkhid [The use of Python programming in the study of mathematics as a STEM approach]. *Aktualni aspekty rozvytku STEAM-osvity v umovakh yevrointehratsii: zbirnyk materialiv Mizhnarodnoi naukovo-praktychnoi internet-konferentsii* (m. Kropyvnytskyi, 21 kvitnia 2023 roku). Kropyvnytskyi: DonDUVS. P. 66–69 [in Ukrainian].

10. Bondarenko, M.F., Bilous, N.V., & Rutkas, A.H. (2004). *Kompiuterna dyskretna matematyka: Pidruchnyk [Computer Discrete Mathematics: Textbook]*. Kharkiv: «Kompaniia SMIT». 480 p. [in Ukrainian].