§ 1 Цифрові інструменти для навчання

1. **Цифрові інструменти для навчання**:
   - **Програми**: Duolingo (мова), Khan Academy (математика).
   - **Онлайн-платформи**: Coursera, edX.
2. **Цифрові інструменти для опанування**:
   - **Інструменти**: Python, MATLAB.
   - **Способи вивчення**: Онлайн-курси, практичні завдання, відеоуроки.
3. **Результативне навчання**:
   - Встановлення чітких цілей, регулярна практика, використання різних джерел інформації.
4. **Вплив цифрових інструментів на навчання**:
   - Швидкий доступ до матеріалів, інтерактивне навчання, можливість моніторингу прогресу.
5. **Необхідність самоосвіти**:
   - Адаптація до змінюваного світу, розвиток особистих навичок, підвищення конкурентоспроможності.
6. **Центральне поняття для карти знань**:
   - **Інформаційні технології**.
   - **Ключові слова**: Програмування, штучний інтелект, великі дані, хмарні техноло-гії, кібербезпека, мережі та комунікації, інтернет речей (IoT), системи управління базами даних (СУБД), електронна комерція (e-commerce), робототехніка, блокчейн, аналітика даних.
7. **Сервіс для цифрових інструментів**:
   - **Google Drive**:
     - **Можливості**: Хостинг файлів, спільне редагування документів, зберігання та доступ з будь-якого пристрою.

МАТЕМАТИКА.DOMA.ORG

- **Приклад завдання**: Спільна робота над презентацією з колегами в реальному часі.

8. **Проблема екології**:
   - **Проблема**: Забруднення повітря.
   - **Інтелект-карта**:
     - Джерела забруднення (транспорт, промисловість)
     - Вплив на здоров'я (респіраторні захворювання)
     - Міри боротьби (зменшення викидів, зелена енергетика)
9. **Розвиток логічного мислення**:

- **Думка**: Розвиток логічного мислення покращує здатність до аналізу та вирішення проблем, що сприяє підвищенню інтелектуальних здібностей і ефективності в навчанні та роботі.

§ 2 Штучний інтелект

1. **Приклади застосування ШІ в повсякденному житті**:

- Віртуальні асистенти (Siri, Alexa).
- Рекомендаційні системи на YouTube, Netflix.
- Навігатори (Google Maps, Waze).
- Фільтрація спаму в електронній пошті.

2. **Напрями розвитку ШІ**:

- Автономні транспортні засоби.
- Медицина (діагностика, розробка ліків).
- Персоналізоване навчання.
- Роботизація в промисловості.

3. **Штучний інтелект**:

MATEMATIKA-DOMA.ORG

- **Штучний інтелект (ШІ)**: Це галузь комп'ютерної науки, яка займається створенням систем, здатних виконувати завдання, що вимагають людського інтелекту, такі як розпізнавання мови, візуальне сприйняття, прийняття рішень.
- **Розробка**: Включає машинне навчання, де моделі тренуються на великих обсягах даних, нейронні мережі, що імітують роботу мозку, та інші алгоритми для автоматизації складних завдань.

4. **Переваги і недоліки розвитку ШІ**:

- **Переваги**: Підвищення ефективності роботи, автоматизація рутинних завдань, покращення діагностики в медицині, створення нових можливостей у різних сферах.
- **Недоліки**: Загроза втрати робочих місць, етичні питання (приватність, безпека), можливість неправильного використання технологій.

5. **ШІ в навчанні**:

ШІ може допомогти створювати персоналізовані навчальні плани, надавати миттєвий зворотний зв'язок, рекомендувати матеріали для поглиблення знань, автоматизувати оцінювання.

8. Етичні питання в розвитку ШІ:

Врегулювання етичних питань важливе для забезпечення того, щоб розвиток ШІ був безпечним, відповідальним та справедливим, зокрема, для запобігання дискримінації,

МАТЕМАТИКА-ДОМА.ORG

забезпечення приватності та безпеки користувачів, а також уникнення неправильного застосування технології.

§ 4 Автоматичний пошук або заміна фрагментів тексту

1. **Приклади редагування тексту з автоматичним пошуком або заміною**:

- **Замінити застарілі терміни** на сучасні (наприклад, «Інтернет-провайдер» на «постачальник інтернет-послуг»).
- **Виправити орфографічні помилки** в довгому документі (наприклад, замінити всі випадки «сертефікат» на «сертифікат»).
- **Заміна форматування тексту** (наприклад, замінити всі вживання слова «важливо» на напівжирний шрифт).

2. **Автоматична заміна фрагментів тексту**:

- Натисніть **Ctrl + H** або перейдіть до вкладки «Заміна» в розділі «Редагування».
- Введіть слово або фразу, яку потрібно знайти, у полі «Знайти що».
- Введіть нове слово або фразу в полі «Замінити на».
- Натисніть «Замінити» для заміни кожного окремого випадку або «Замінити все» для автоматичної заміни у всьому документі.

3. **Слова за шаблоном зі знаками підстановки *ак?**:

- Можливі слова: «мак», «рак», «бак», «пак», «лак», «сак», «так», «кулак», «дурак».

МАТЕМАТИКА-ДОМА.ORG

- Використання підстановки «*» дозволяє знайти будь-які слова, що починаються з будь-якої кількості символів перед «ак», а «?» шукає будь-який один символ після «ак».

7. Дотримання правил введення тексту, включаючи уникання зайвих пробілів, розривів рядків, абзаців та сторінок, є важливим з кількох причин:

❖ **Збереження структури документа**:
➢ Неправильне форматування може порушити логічну структуру документа, зробивши його важким для читання та розуміння.
❖ **Забезпечення однакового вигляду**:
➢ Зайві пробіли та розриви можуть призвести до нерівномірного вигляду тексту, що виглядає непрофесійно.
❖ **Сумісність з різними програмами**:
➢ Неправильне форматування може викликати проблеми при перенесенні документа до інших програм або платформ, порушуючи відображення тексту.
❖ **Полегшення редагування та автоматизації**:
➢ Дотримання правил введення тексту полегшує використання автоматичних функцій, таких як вирівнювання тексту, створення змісту, або пошук і заміна. Зайві пробіли та розриви можуть порушити роботу цих функцій.
❖ **Поліпшення доступності**:
➢ Правильне введення тексту важливе для створення документів, які будуть доступні для людей із порушеннями зору, що використовують екранні читачі. Неправильні

МАТЕМАТИКА-ДОМА.ORG

розриви та пробіли можуть ускладнити роботу таких інструментів.

❖ **Оптимізація обсягу документа**:

➢ Зайві символи можуть збільшити розмір файлу документа, що особливо важливо при роботі з великими текстами або документами, які потрібно пересилати електронною поштою або зберігати в хмарних сховищах.

§ 5 Форматування з використанням стилів

1. **Властивості об'єктів документа, які можна швидко змінити за допомогою стилів**:

- **Шрифт і розмір тексту**: можна змінити тип шрифту (наприклад, Times New Roman, Arial) та його розмір для різних частин тексту.
- **Колір тексту**: можна задати кольорове виділення для різних елементів тексту.
- **Інтервали**: відстань між рядками та абзацами, а також відступи перед і після абзаців.
- **Форматування абзаців**: вирівнювання (по лівому, правому краю, по центру, по ширині), відступи першого рядка.
- **Оформлення списків**: змінити стиль маркованих і нумерованих списків, включаючи тип маркерів або чисел.
- **Рамки та фонове заповнення**: додавання рамок навколо тексту або зміна фону певних ділянок документа.

МАТЕМАТИКА-ДОМА.ORG

- **Заголовки та підзаголовки**: оформлення заголовків із виділенням тексту, шрифту, кольору та відступів.

2. **Різновиди стилів і їх призначення**:

- **Стиль заголовків**: Призначені для форматування заголовків і підзаголовків документа (наприклад, Заголовок 1, Заголовок 2). Вони допомагають структуризувати документ, автоматично формувати зміст.
- **Стиль основного тексту**: Використовується для основного тексту документа. Включає стандартне форматування шрифту, відступів та інтервалів.
- **Стиль списків**: Призначені для форматування маркованих і нумерованих списків.
- **Стиль таблиць**: Оформлення таблиць із заданими рамками, кольорами, шрифтами в комірках.
- **Стиль підписів**: Використовується для форматування підписів під зображеннями, таблицями або іншими об’єктами.
- **Стиль цитат**: Використовується для виділення цитат з особливим форматуванням, яке відрізняється від основного тексту.

3. **Призначення і застосування стильового оформлення документа**:

- **Забезпечення єдиного вигляду**: Стильове оформлення допомагає зберігати однаковий вигляд у всьому документі, що робить його більш професійним і структурованим.

МАТЕМАТИКА-DOMA.ORG

- **Автоматизація процесів**: Використання стилів дозволяє швидко змінювати зовнішній вигляд документа. Наприклад, якщо потрібно змінити шрифт усіх заголовків, достатньо лише змінити стиль заголовків, а не форматувати кожен заголовок окремо.
- **Створення змісту**: Стильове оформлення заголовків автоматично дозволяє генерувати зміст документа.
- **Забезпечення сумісності та зручності роботи**: Стилі спрощують роботу з великими документами, полегшують співпрацю між кількома користувачами та збереження стилістичної єдності під час перенесення документа між різними програмами або платформами.

4. На зображенні представлено документ, який оформлений з використанням різних стилів для виділення важливих частин тексту.

   Оцінка доцільності оформлення тексту:

- **Заголовок** ("Із 3 червня почав діяти новий український правопис 2019 року"):
  - Використовується великий шрифт, напівжирний текст, можливо, синій колір, щоб привернути увагу до основної теми документа. Це дозволяє читачеві одразу зрозуміти основну ідею тексту.
- **Підзаголовки** (напр., "Тепер назви сайтів та інших..."):
  - Застосування відмінного шрифту або виділення тексту кольором допомагає структурувати документ і розділяти його на

МАТЕМАТИКА-ДОМА.ORG

логічні блоки. Це сприяє легшому сприйняттю інформації.

- **Основний текст**:
  - Стандартний шрифт для зручного читання, використання курсиву та напівжирного шрифту для виділення важливих термінів та правил.
- **Виділення важливих термінів**:
  - Курсив або напівжирний шрифт для таких слів як "ввічливість", "двократний", "четвертний" робить акцент на ключових поняттях.
- **Цитати та приклади**:
  - Вони можуть бути оформлені іншим шрифтом або кольором для відокремлення від основного тексту.

Кількість стилів:

Для створення цього документа, ймовірно, було використано **п'ять-шість** різних стилів:

- Стиль для заголовків.
- Стиль для підзаголовків.
- Стиль для основного тексту.
- Стиль для виділення важливих слів (напівжирний/курсив).
- Стиль для цитат або прикладів.
- Можливо, спеціальний стиль для кольорового тексту.

Це дозволило зробити текст зрозумілим, структурованим та легким для сприйняття, забезпечуючи читача всією необхідною інформацією.

МАТЕМАТИКА-ДОМА.ORG

§ 6 Структура документа

1. **Структурні частини документа з технічного погляду:**

   Документ може бути поділений на такі структурні частини:

- **Титульна сторінка**: містить назву документа, автора, дату тощо. Для цієї частини можна задати спеціальне форматування, наприклад, великі шрифти або центрування тексту.
- **Зміст**: перелік розділів і підрозділів із вказанням сторінок. Для змісту можна задати автоматичне оновлення під час змін у структурі документа.
- **Основний текст**: розділи й підрозділи, що складають основний зміст документа. Для цієї частини можна налаштувати параметри шрифту, вирівнювання, абзаців тощо.
- **Виноски та посилання**: додаткові пояснення або джерела. Для виносок можна задати стиль нумерації, розмір шрифту, відступи.
- **Додатки**: додаткова інформація, таблиці, графіки, що не входять у основний текст. Можна задати інший формат нумерації сторінок або окремі стилі для заголовків.
- **Список літератури**: джерела, використані для підготовки документа. Для цієї частини можна налаштувати автоматичне сортування та спеціальний стиль оформлення.

2. **Відмінність логічної структури документа від технічної:**

МАТЕМАТИКА-ДОМА.ORG

- **Логічна структура** документа відображає змістовне наповнення та послідовність подання інформації. Це те, як інформація структурована та організована з точки зору її змісту та логіки. Вона включає розділи, підрозділи, заголовки, підзаголовки, послідовність аргументів тощо.
- **Технічна структура** документа стосується того, як цей зміст реалізований на практиці за допомогою засобів форматування та макета. Вона охоплює параметри форматування тексту, розмітку сторінок, стиль шрифтів, вирівнювання, відступи, нумерацію сторінок, додавання заголовків та підзаголовків, використання шаблонів та стилів.

  Приклад: У документі може бути логічно поділений розділ на "Вступ", "Методологія", "Результати", "Висновки". З технічного погляду, ці розділи можуть бути оформлені за допомогою різних стилів заголовків, автоматичної нумерації сторінок і встановлених відступів. Логічна структура визначає послідовність і взаємозв'язок цих розділів, а технічна – їхнє фізичне оформлення в документі.

3. Документ, зображений на картинці, має чітко структуровані розділи та підрозділи. Ось його основні частини та приклади дій, які можна виконати з елементами структури:

- **Введення**:
  - **Структура книги**: Переглянути та зрозуміти, як організована книга.

MATEMATIKA-DOMA.ORG

- **На кого розрахована книга**: Визначити цільову аудиторію книги.
- **Від видавництва**:
- Прочитати рекомендації або інформацію, надану видавництвом.
- **Глава 1. Основи роботи із Microsoft Word**:
- **1.1. Встановлення Microsoft Word**:
- **Початок встановлення**: Виконати початкові дії для інсталяції програми.
- **Параметри встановлення**: Налаштувати інсталяцію відповідно до своїх потреб.
- **Встановлення Microsoft Word поверх встановлених раніше версій**: Оновити наявну версію Word або встановити нову.
- **1.2. Запуск програми Microsoft Word**:
- Відкрити програму та налаштувати її під свої потреби.
- **1.3. Завершення роботи з Microsoft Word**:
- Правильно закрити програму після завершення роботи.
- **1.4. Вікно програми**:
- **Основне меню**: Ознайомитися з функціями, доступними в головному меню.
- **Кнопка Office**: Виконати основні операції з документом, такі як збереження чи друк.
- **Стрічка**: Використовувати інструменти, що знаходяться на стрічці.
- **Смуги прокручування**: Переміщатися по документу.
- **Лінійка**: Налаштувати поля та табуляції.
- **Рядок стану**: Переглянути інформацію про поточний стан документа.
- **Глава 2. Основи роботи із документами Microsoft Word**:
- **2.1. Документи Word**:

ИНФОРМАТИКА-ДОМА.ORG

- **Створення документа**: Почати роботу над новим документом.
- **Відкриття документа**: Відкрити раніше збережений документ для подальшої роботи.
- **Збереження документів**: Зберегти зміни в документі або зберегти новий документ.

Кожна з цих частин структури має конкретне призначення та дії, які можна виконати для ефективної роботи з документами в Microsoft Word.

§ 7 Гіперпосилання в текстових документах

1. **Об’єкти текстового документа, які можна зробити посиланнями:** заголовки, закладки, веб-сторінки, файли, зображення, таблиці.
2. **Об’єкти, на які можна встановити посилання:** закладки, заголовки, веб-сторінки, інші документи, файли на комп'ютері.
3. **Процес створення посилань:**

- Виділяєте текст або об'єкт.
- Натискаєте "Вставка" → "Посилання".
- Вибираєте тип посилання (веб-сторінка, заголовок, закладка).
- Вказуєте потрібне посилання та натискаєте "ОК".

4. **Відмінності між закладкою і посиланням:**

- **Закладка**: мітка в тексті для швидкого переходу до певного місця.

МАТЕМАТИКА-ДОМА.ORG

- **Посилання**: активний об'єкт, що веде до іншого місця в документі або на зовнішній ресурс.

5. **Приклади завдань для створення посилань:** створення змісту документа, реферату з джерелами, інтерактивного звіту з переходами до розділів.
8. **Переваги електронної книжки:**

- Легкість у перенесенні великої кількості книг.
- Пошук по тексту.
- Налаштування розміру шрифту.
- Вбудовані словники та перекладачі.
- Доступ до оновлень і нових видань.

Переваги паперової книжки:

- Відчуття фізичної книги.
- Відсутність потреби в батареї або зарядці.
- Відчуття приватності й зосередженості.

§ 8 Застосовую на практиці

1. **Перелічи можливості розташування малюнка відносно тексту.** Зверху, знизу, обтікання текстом зліва, справа, по центру, перед текстом, за текстом.
2. **Поясни, як за допомогою таблиць можна розташувати об’єкти документа на сторінці.** Таблиці можна використовувати для точного розташування тексту, зображень, графіки та інших елементів на сторінці, розміщуючи їх у відповідних

МАТЕМАТИКА-ДОМА.ORG

клітинках. Це дозволяє структурувати контент і вирівнювати його.

3. **Укажи, як можна вставити в текст символ, відсутній на клавіатурі.** Можна використовувати комбінацію клавіш Alt+код на цифровій клавіатурі, або скористатися меню "Вставка" → "Символ" у текстовому редакторі.

§ 9 Спільна робота над документом

1. **Переваги спільної роботи над документом:**

- Можливість одночасного редагування.
- Спрощена комунікація між учасниками.
- Контроль версій та історія змін.
- Економія часу завдяки паралельній роботі.
- Можливість коментування та пропонування змін.

2. **Приклади задач для Google Документів:**

- Спільне написання рефератів або проектів.
- Підготовка до групових презентацій.
- Організація командних досліджень та аналізу даних.
- Колективне створення та редагування навчальних матеріалів.

3. **Режими роботи в Google Документах:**

- **Редагування**: використовується для внесення прямих змін у документ.
- **Пропонування**: застосовується, коли потрібно запропонувати зміни без їх негайного застосування.

МАТЕМАТИКА-ДОМА.ORG

- **Перегляд**: корисний для перегляду остаточного варіанту документа без можливості внесення змін.

5. **Способи вдосконалення інструментів для спільної роботи:**

- Покращення функції коментування для кращої організації обговорень.
- Інтеграція з іншими сервісами для кращого керування завданнями.
- Розширення функціоналу історії змін, щоб легше було відстежувати внесок кожного учасника.
- Розробка інструментів для аналізу продуктивності команди.
- Додавання більш гнучких опцій для прав доступу.

§ 10 Автозаповнення. Копіювання формул

Перевір себе

1. У яку клітинку було введено формулу =B3+F4, якщо після копіювання в клітинку C3 вона набула вигляду =A5+E6?

Формула була введена в клітинку D1.

2. **Які із наведених формул у разі копіювання на 1 стовпець вліво і на 1 рядок угору призведуть до помилки, незалежно від того, де їх було введено спочатку?**

- =A2-B1
- =A4+D5
- =G6*F2

МАТЕМАТИКА-ДОМА.ORG

- =D2*(B4+C9)

Пояснення:

Помилка виникає, коли після зміщення (1 стовпець вліво і 1 рядок угору) формула посилається на клітинки, що не існують (наприклад, стовпець перед A або рядок перед 1).

1. **=A2-B1:**
   - Після зміщення: =Z1-A0.
   - Клітинки A0 не існує (рядок 0 не визначений). **Помилка.**
2. **=A4+D5:**
   - Після зміщення: =Z3+C4.
   - Всі клітинки існують. **Без помилки.**
3. **=G6*F2:**
   - Після зміщення: =F5*E1.
   - Всі клітинки існують. **Без помилки.**
4. **=D2*(B4+C9):**
   - Після зміщення: =C1*(A3+B8).
   - Всі клітинки існують. **Без помилки.**

Формула, що призведе до помилки: =A2-B1.
Причина: посилання на клітинку A0, якої не існує.

3. **Послідовність чисел, у якій перші два числа дорівнюють 1, а кожне наступне є сумою двох попередніх, називається послідовністю Фібоначчі.**

Числа Фібоначчі обчислюються за формулою:

$$F(n)=F(n-1)+F(n-2),$$

МАТЕМАТИКА-ДОМА.ORG

де F(0) = 1 і F (1) = 1

Побудова послідовності:

F(0)=1, F(1)=1, F(2)=1+1=2, F(3)=1+2=3, F(4)= 2+3=5, …

Для знаходження 50-го числа Фібоначчі використовуємо рекурсивну формулу або алгоритм програмування, оскільки ручне обчислення довге.

Обчислення: Знайдемо 50-е число Фібоначчі.

50-е число Фібоначчі дорівнює 20,365,011,074.

4. Чому в електронній таблиці під час копіювання формул адреси в них змінюються? Чи не було б зручніше, якби формули копіювалися без змін?

Адреси в формулах змінюються через використання **відносних посилань**. Це дозволяє автоматично адаптувати формулу до нового розташування під час копіювання, зберігаючи логіку обчислень. Наприклад, якщо формула =A1+B1 копіюється вниз, вона зміниться на =A2+B2, що є корисним для роботи з великими наборами даних.

Чи було б зручніше, якби формули копіювалися без змін? Якщо формули копіювалися б без змін, це могло б бути зручним у певних випадках, але не завжди. Для таких ситуацій в електронних таблицях існує можливість використовувати **абсолютні посилання** (з фіксацією адрес за допомогою

МАТЕМАТИКА-ДОМА.ORG

знака $, наприклад, =$A$1+$B$1). Це забезпечує необхідну стабільність адрес.

Отже, змінність формул є необхідною та гнучкою, але для незмінних копій передбачені альтернативні способи (абсолютні посилання).

§ 11 Абсолютні й мішані посилання. Логічні, математичні та статистичні функції

Перевір себе

1. Приклади абсолютних, відносних і мішаних посилань:

- **Відносні посилання:**
  Формат: A1, B2, C3.
  При копіюванні змінюються залежно від нового розташування.
- **Абсолютні посилання:**
  Формат: =$A$1, =$B$2.
  При копіюванні залишаються незмінними, завжди посилаючись на фіксовану клітинку.
- **Мішані посилання:**
  Формат: $A1, B$2.
  Частково змінюються при копіюванні.
  Наприклад:
  - $A1 фіксує стовпець A, але рядок змінюється.
  - B$2 фіксує рядок 2, але стовпець змінюється.

МАТЕМАТИКА-ДОМА.ORG

2. Різниця між абсолютними, відносними та мішаними посиланнями:

- **Відносні посилання:**
  Змінюються відповідно до зсуву під час копіювання. Зручні для масових обчислень.
- **Абсолютні посилання:**
  Завжди посилаються на ту саму клітинку, незалежно від місця копіювання. Використовуються, коли потрібна фіксація даних.
- **Мішані посилання:**
  Частково фіксують або стовпець, або рядок, дозволяючи часткову зміну залежно від потреб. Зручні для обчислень із комбінованою логікою.

3. Приклади відомих функцій табличного процесора:

- **Математичні функції:**
  - SUM(A1:A10) — сума значень у діапазоні.
  - PRODUCT(A1:A10) — добуток значень у діапазоні.
  - ROUND(A1, 2) — округлення значення до 2 знаків після коми.
- **Статистичні функції:**
  - AVERAGE(A1:A10) — середнє значення діапазону.
  - MEDIAN(A1:A10) — медіана значень у діапазоні.
  - STDEV.P(A1:A10) — стандартне відхилення для генеральної сукупності.

МАТЕМАТИКА-ДОМА.ORG

- **Логічні функції:**
  - IF(A1>10, "Так", "Ні") — перевіряє умову, повертає результат залежно від її виконання.
  - AND(A1>5, B1<10) — перевіряє, чи всі умови істинні.
  - OR(A1>5, B1<10) — перевіряє, чи хоча б одна умова істинна.

5. Таблиця множення за зразком:

Для створення таблиці множення в клітинку B2 вводимо формулу з мішаними посиланнями:

bash
```
=$A2*B$1
```

Пояснення:

- $A2: Абсолютно фіксується стовпець A (для множення по рядках).
- B$1: Абсолютно фіксується рядок 1 (для множення по стовпцях).

Після введення формули в B2, копіюємо її в діапазон B2:K11.

6. Таблиця оцінок суддів:

1. **Обчислення "Найвищий бал":** У колонці "Найвищий бал" (наприклад, у клітинці G2) вводимо формулу:

   scss
   ```
   =MAX(B2:F2)
   ```

   Це знаходить максимальний бал у рядку.

МАТЕМАТИКА-ДОМА.ORG

2. **Обчислення "Найнижчий бал":** У колонці "Найнижчий бал" (наприклад, у клітинці H2) вводимо формулу:

   scss
   =MIN(B2:F2)

   Це знаходить мінімальний бал у рядку.

3. **Обчислення "Сума балів":** У колонці "Сума балів" (наприклад, у клітинці I2) вводимо формулу:

   scss
   =SUM(B2:F2)

   Це знаходить загальну суму балів для кожного учасника.

4. **Умовне форматування:** Для встановлення зірок у балів:
   - Виділіть діапазон оцінок суддів (наприклад, B2:F7).
   - Застосуйте умовне форматування, яке використовує значки (наприклад, "зірки"), залежно від значення у клітинках.
     Наприклад:
     - 10 балів = повна зірка,
     - менше = часткова або порожня зірка.

Це створить таблицю у відповідності до наданого зразка.

7. Таблиця для відстеження запасів:

1. **Необхідні обчислення:**

МАТЕМАТИКА-ДОМА.ORG

Для заповнення стовпця "Товарний запас на кількість тижнів" необхідно обчислити, скільки тижнів запасу є у кожного товару. Формула:

=Залишок / Середні продажі

Наприклад, у клітинці D2 для першого товару формула буде виглядати:

=C2/B2

2. **Умовне форматування:**
   - Якщо значення в стовпці "Товарний запас на кількість тижнів" (наприклад, у діапазоні D2:D5) менше 3 — застосувати форматування, щоб виділити це червоним кольором (потрібне термінове поповнення).
   - Якщо значення в межах від 3 до 5 включно — виділити жовтим (запас у межах норми).
   - Якщо значення більше 5 — виділити зеленим (достатній запас).

8. Обчислення функції F(x)F(x)F(x):

Функція:

F(x)= \begin{cases}x^2, & x \leq-4 \\ 100, & -4<x \leq 5 \\ x, & x>5\end{cases} Для діапазону xxx від -10 до 10 з кроком 1:

Обчислимо значення функції та побудуємо таблицю, що відображає xxx і відповідні F(x).

МАТЕМАТИКА-ДОМА.ORG

9. Способи умовного форматування даних поза табличним процесором:

Крім табличного процесора (наприклад, Excel), умовне форматування можна здійснювати за допомогою:

1. **Мови програмування:**
   - **Python:** Використовуючи бібліотеки, такі як Pandas для аналізу та Matplotlib для візуалізації, можна застосовувати кольорове форматування до даних.
   - **R:** Для статистичного аналізу та графічного відображення умов.
2. **Інші інструменти:**
   - **Google Data Studio:** Створення звітів з умовним форматуванням.
   - **Power BI:** Візуалізація з динамічним форматуванням, залежно від умов.
   - **Tableau:** Форматування графіків і таблиць за допомогою правил.

§ 12 Створення інформаційних моделей

1. Обчислювальні можливості електронних таблиць:

- Виконання арифметичних обчислень (додавання, віднімання, множення, ділення).
- Статистичні функції (середнє значення, медіана, стандартне відхилення).
- Логічні функції (IF, AND, OR).
- Математичні функції (SUM, PRODUCT, ROUND).

MATEMATIKA-DOMA.ORG

- Фінансові функції (NPV, PMT, FV).
- Робота з текстом (з’єднання, пошук, заміна).
- Побудова графіків і діаграм.
- Умовне форматування для аналізу даних.
- Автоматизація розрахунків через макроси.
- Пошук оптимальних рішень (за допомогою інструменту "Пошук рішення").

2. Етапи розв’язування задач за допомогою моделювання в електронних таблицях:

1. **Формулювання задачі:** Визначення мети й умов задачі.
2. **Збір даних:** Введення вихідних даних до таблиці.
3. **Побудова моделі:** Створення формул і залежностей між даними.
4. **Аналіз результатів:** Інтерпретація отриманих даних.
5. **Оптимізація:** Використання інструментів для пошуку оптимальних рішень.
6. **Презентація результатів:** Візуалізація та експорт даних.

3. Приклади задач із фізики та математики:

- **Фізика:** Розрахунок законів руху, електричного струму (закон Ома), енергетичних втрат.
- **Математика:** Розв’язання систем рівнянь, аналіз статистичних даних, побудова математичних моделей.

MATEMATIKA-DOMA.ORG

5. Обґрунтування:

Зі зростанням обчислювальних можливостей комп’ютерів з’явилися нові перспективи для науки:

- Обробка великих даних (Big Data) дозволяє аналізувати глобальні тренди.
- Штучний інтелект і машинне навчання відкрили нові шляхи автоматизації.
- Складні моделі й симуляції, які раніше були неможливі, тепер легко виконуються.
- Розвиток візуалізації даних допомагає краще зрозуміти результати досліджень.

§ 14 Знайомство з Python. Виведення даних

1. Код, який виводить цифри 1, 2, 3 різними способами:

```
# У стовпчик
print(1)
print(2)
print(3)

# В один рядок без пробілу
print(1, end='')
print(2, end='')
print(3, end='')

# Як число 123
print("\n123")
```

2. Чому в середовищі IDLE різні елементи позначаються різними кольорами?

МАТЕМАТИКА-DOMA.ORG

У середовищі IDLE (і багатьох інших текстових редакторах) використовується **синтаксичне підсвічування**. Це робиться для полегшення читання й написання коду:

- **Ключові слова** (наприклад, if, for, while) виділяються окремим кольором, щоб показати їх спеціальну роль у коді.
- **Рядки** (текстові дані, наприклад, "Привіт") мають свій колір для відокремлення від іншого коду.
- **Коментарі** (#) позначаються своїм кольором, щоб розрізняти опис коду й самі команди.
- **Числа** й **функції** також виділяються окремо для кращого сприйняття.

Це допомагає програмістам швидше знаходити помилки та розуміти структуру коду.

3. Як за допомогою однієї команди print вивести кілька об’єктів?

Для цього використовується кома ,, яка автоматично додає пробіл між об’єктами:

print("Це текстовий рядок", 123)

Результат:

Це текстовий рядок 123

4. Програма, яка виведе арифметичний вираз із двома операціями та його значення:

Арифметичний вираз

МАТЕМАТИКА-ДОМА.ORG

```
x = 5
y = 3
z = 2

# Вираз: (x + y) * z
вираз = f"({x} + {y}) * {z}"
значення = (x + y) * z

# Виведення виразу й результату
print(f"Вираз: {вираз}")
print(f"Значення: {значення}")
```

Результат:

```
makefile
Вираз: (5 + 3) * 2
Значення: 16
```

5. Інші середовища для програмування мовою Python

1. **Jupyter Notebook**
   - **Переваги:**
     - Інтерактивне середовище для виконання коду.
     - Зручне для роботи з даними та візуалізації.
     - Підтримує Markdown для створення звітів.
   - **Недоліки:**
     - Потребує більше ресурсів.
     - Менш зручний для створення великих програм.
2. **PyCharm**
   - **Переваги:**

МАТЕМАТИКА-DOMA.ORG

- Розширені функції для великих проектів (автодоповнення, рефакторинг).
- Інтеграція з системами контролю версій (Git).

- **Недоліки:**
  - Потребує налаштувань для початківців.
  - Важче для роботи на старих комп’ютерах.

3. **VS Code**
   - **Переваги:**
     - Швидкий, легкий, модульний редактор.
     - Підтримує багато мов програмування.
     - Безкоштовний, має багато плагінів.
   - **Недоліки:**
     - Інтерфейс може здатися складним для новачків.
4. **Spyder**
   - **Переваги:**
     - Оптимізований для наукових розрахунків.
     - Інтеграція з бібліотеками NumPy, Pandas, Matplotlib.
   - **Недоліки:**
     - Обмежені можливості для роботи над великими проектами.
5. **Thonny**
   - **Переваги:**
     - Простий інтерфейс, зручний для початківців.
     - Інструменти для навчання.

МАТЕМАТИКА-ДОМА.ORG

- **Недоліки:**
  - Менш функціональний для професійних проектів.

6. **Anaconda (з Jupyter і Spyder):**
   - **Переваги:**
     - Популярний для обробки даних.
     - Має вбудовані бібліотеки та менеджери пакетів.
   - **Недоліки:**
     - Великий розмір дистрибутиву.

§ 15 Змінні. Присвоювання

1. Що таке змінна?

Змінна — це іменований контейнер для зберігання даних у програмі. Змінна дозволяє зберігати значення, до яких можна звертатися та змінювати їх у процесі виконання програми. У Python змінна може зберігати різні типи даних, наприклад числа, рядки, списки тощо.

2. Приклади імен змінних:

- Допустимі імена:
  - x
  - user_name
  - temperature
  - _data
  - value123
- Недопустимі імена:
  - 1var (починається з цифри)
  - user-name (містить дефіс)
  - for (зарезервоване слово в Python)

МАТЕМАТИКА-DOMA.ORG

3. Значення змінної x після виконання фрагментів програм:

а)

```
y = 4
x = y * 2 + 1
```

Результат: x=4×2+1=9x = 4 \times 2 + 1 = 9x=4×2+1=9.

б)

```
y = 3
z = y + 1
x = y + z
```

Результат: z=3+1=4z = 3 + 1 = 4z=3+1=4, x=3+4=7x = 3 + 4 = 7x=3+4=7.

в)

```
x = 4
x = x * x
```

Результат: x=4×4=16x = 4 \times 4 = 16x=4×4=16.

Перевіримо у Python:

Результати перевірки:

а) x=9x = 9x=9
б) x=7x = 7x=7
в) x=16x = 16x=16

4. Розв’язання:

а)

МАТЕМАТИКА-DOMA.ORG

a = 2
b = a

Результат: a=2a = 2a=2, b=2b = 2b=2.

б)

a = 2
b = a
a = b

Результат: a=2a = 2a=2, b=2b = 2b=2.

в)

a = 1
b = 2
a = b
b = a

Результат: a=2a = 2a=2, b=2b = 2b=2.

г)

a = 1
b = 2
c = a
a = b
b = c

Результат: a=2a = 2a=2, b=1b = 1b=1.

Перевіримо в Python:

Результати перевірки:

а) a=2a = 2a=2, b=2b = 2b=2
б) a=2a = 2a=2, b=2b = 2b=2
в) a=2a = 2a=2, b=2b = 2b=2
г) a=2a = 2a=2, b=1b = 1b=1

МАТЕМАТИКА-DOMA.ORG

5. Програма для підписування дипломів

```
# Введення даних
місто = input("Введіть назву міста: ")
номер_школи = input("Введіть номер школи:
")
клас = input("Введіть клас (наприклад, 7-А):
")
прізвище = input("Введіть прізвище
учня/учениці: ")
ім'я = input("Введіть ім'я учня/учениці: ")

# Формування тексту диплома
print("\nДиплом")
print('переможця конкурсу "Байтик"')
print("Нагороджується")
print(f"{прізвище} {ім'я},")
print(f"учень/учениця {клас} класу")
print(f"ЗОШ І-ІІІ ступенів №
{номер_школи}")
print(f"міста {місто}")
```

Приклад вводу:

```
Введіть назву міста: Київ
Введіть номер школи: 77
Введіть клас (наприклад, 7-А): 7-А
Введіть прізвище учня/учениці:
Мудрієвський
Введіть ім'я учня/учениці: Олександр
```

Результат:

```
Диплом
переможця конкурсу "Байтик"
Нагороджується
Мудрієвський Олександр,
```

МАТЕМАТИКА ДОМА.ORG

учень/учениця 7-А класу
ЗОШ І-ІІІ ступенів № 77
міста Київ

6. Програма для обміну значень двох змінних без третьої змінної

```
# Введення даних
a = int(input("Введіть перше число (a): "))
b = int(input("Введіть друге число (b): "))

# Обмін значеннями
a = a + b
b = a - b
a = a - b

# Результат
print(f"Після обміну: a = {a}, b = {b}")
```

Приклад вводу:

```
Введіть перше число (a): 5
Введіть друге число (b): 7
```

Результат:

```
less
Після обміну: a = 7, b = 5
```

§ 16 Дані різних типів

1. Типи даних у Python:

- **Числові типи:**
  - int — цілі числа (наприклад, 1, 5, -3).
  - float — числа з плаваючою точкою (наприклад, 1.5, -2.8).

МАТЕМАТИКА-DOMA.ORG

- complex — комплексні числа (наприклад, 3+4j).

- **Рядкові типи:**
  - str — текстові дані (наприклад, "hello", 'Python').
- **Логічний тип:**
  - bool — логічні значення (True, False).
- **Контейнерні типи:**
  - list — список (наприклад, [1, 2, 3]).
  - tuple — кортеж (наприклад, (1, 2, 3)).
  - dict — словник (наприклад, {"key": "value"}).
  - set — множина (наприклад, {1, 2, 3}).

2. Навіщо потрібні типи даних у програмуванні?

- Для ефективного зберігання й обробки інформації (наприклад, текст, числа, списки).
- Типи даних визначають, які операції можна виконувати (наприклад, арифметичні дії з числами, конкатенація з рядками).
- Для оптимізації ресурсів пам’яті та продуктивності.
- Для зменшення помилок під час виконання програм (не можна змішувати несумісні типи, наприклад, додавати число до тексту).

3. Розв'язання задач:

Задача 1: Білочки та горіхи

```
# Введення даних
n = int(input("Кількість білочок (n): "))
m = int(input("Кількість горіхів (m): "))
```

МАТЕМАТИКА-ДОМА.ORG

```
# Обчислення
горіхи_на_білочку = m // n
залишок = m % n

# Результат
print(f"Кожна білочка отримує
{горіхи_на_білочку} горіхів.")
print(f"Залишок: {залишок} горіхів.")
```

Задача 2: Кількість дописів

```
python
# Введення даних
x = int(input("Кількість дописів Петрика (x):
"))
y = int(input("Кількість дописів Марічки (y):
"))
z = int(input("Кількість дописів кожного
іншого учня (z): "))
n = int(input("Кількість учнів (n): "))

# Обчислення
решта_учнів = n - 2
всього_дописів = x + y + (решта_учнів * z)

# Результат
print(f"Всього було зроблено
{всього_дописів} дописів.")
```

Задача 3: Чакі та Черрі

```
python
КопироватьРедактировать
# Введення даних
```

МАТЕМАТИКА-ДОМА.ORG

```
x = int(input("Скільки котлет з'їдає Чакі за день (x): "))
y = int(input("Скільки котлет з'їдає Черрі за день (y): "))

# Початкові ваги
вага_Чакі = 1.5
вага_Черрі = 2.5

# Обчислення
дні = 0
while вага_Чакі <= вага_Черрі:
    вага_Чакі += x * 0.1
    вага_Черрі += y * 0.1
    дні += 1

# Результат
print(f"Чакі наздожене Черрі через {дні} днів.")
```

Задача 4: Відстань між пунктами

```
# Введення даних
x = int(input("Час руху мотоцикліста (години, x): "))
y = int(input("Швидкість велосипедиста (км/год, y): "))
z = int(input("Швидкість мотоцикліста (км/год, z): "))

# Обчислення
відстань_велосипедиста = (x + 1) * y
відстань_мотоцикліста = x * z
загальна_відстань = відстань_велосипедиста + відстань_мотоцикліста
```

МАТЕМАТИКА-DOMA.ORG

```
# Результат
print(f"Відстань між пунктами A і B:
{загальна_відстань} км.")
```

Задача 5: Розчини молока та води

```
# Введення даних
t = int(input("Об'єм склянки (мл, t): "))

# Початкові об'єми
молоко = 1000  # 1 літр = 1000 мл
вода = 1000

# Переливання
молоко_в_воду = t
молоко -= молоко_в_воду
вода += молоко_в_воду

вода_в_молоко = (молоко_в_воду / вода) * t
молоко += вода_в_молоко
вода -= вода_в_молоко

# Результат
print(f"Молока у першій посудині:
{молоко:.2f} мл")
print(f"Молока у другій посудині: {1000 -
молоко:.2f} мл")
```

§ 17 Умовний оператор

1. Алгоритми з розгалуженням:

- **Вибір одягу:**

```
if температура > 20: print("Футболка")
elif температура > 10: print("Кофта")
else: print("Куртка")
```

MATEMATIKA-DOMA.ORG

- **Чи вистачає грошей:**

```
if гроші >= ціна: print("Купити")
else: print("Недостатньо коштів")
```

2. Відповідність Scratch і Python:

- **А**: if умова: → **1**
- **Б**: if умова: команда → **2**
- **В**: if умова, else: → **3**
- **Г**: if умова, команда 2 → **4**

3. Програма з однією загадкою:

```
загадка = input("Що літає, крил не має?
").strip().lower()
if загадка == "вітер": print("Так, супер!")
else: print("На жаль, ні")
```

4. Програма з трьома загадками:

```
правильні = 0
if input("Що літає, крил не має?
").strip().lower() == "вітер": правильні += 1
if input("Що йде, а сліду нема?
").strip().lower() == "час": правильні += 1
if input("Що влітку холодне, а взимку гаряче?
").strip().lower() == "вода": правильні += 1
print(f"Правильних відповідей: {правильні}")
```

§ 18 Малювання. Цикли

1. Команди для графічного вікна:

- import turtle, turtle.setup(), turtle.reset(), turtle.Screen().

МАТЕМАТИКА-ДОМА.ORG

2. Координати точок:

- Центр — (0, 0).
- Вісь X — праворуч/ліворуч.
- Вісь Y — вгору/вниз.

3. Шестикінцева зірка:

```
import turtle
for i in range(6):
    turtle.forward(50)
    turtle.right(120)
    turtle.forward(50)
    turtle.left(60)
```

4. Зірки:

- **4-кутна:**

```
for i in range(4): turtle.forward(100); turtle.right(90); turtle.forward(100); turtle.left(90)
```

- **8-кутна:**

```
for i in range(8): turtle.forward(100); turtle.right(135); turtle.forward(100); turtle.left(45)
```

- **12-кутна:**

```
for i in range(12): turtle.forward(100); turtle.right(150); turtle.forward(100); turtle.left(30)
```

5. Робот:

python
КопироватьРедактировать

```
import turtle
```

МАТЕМАТИКА-DOMA.ORG

```
turtle.begin_fill(); for _ in range(4):
turtle.forward(100); turtle.left(90);
turtle.end_fill()
# Інші частини тіла аналогічно
```

7. Практичні задачі:

- Геометричні фігури.
- Фрактали.
- Графіки.
- Візуалізація алгоритмів.

§ 19 Вкладені цикли

1. Блок-схема програми із вкладеними циклами

Блок-схема відображає основний цикл (зовнішній) і вкладений цикл (внутрішній), що виконується на кожній ітерації зовнішнього циклу. Наприклад:

- Зовнішній цикл: Рахунок рядків.
- Внутрішній цикл: Малювання зірочок у рядку.

2. Приклади малюнків:

- **Прості цикли:** Горизонтальна або вертикальна лінія, коло.
- **Вкладені цикли:** Таблиця, шахова дошка, сітка, спіраль, мозаїка.

3. Програми із вкладеними циклами:

Шахова дошка

```
import turtle
```

МАТЕМАТИКА-ДОМА.ORG

```
turtle.speed(0)
for y in range(8):  # 8 рядків
    for x in range(8):  # 8 колонок
        turtle.penup()
        turtle.goto(x * 20, y * 20)
        turtle.pendown()
        if (x + y) % 2 == 0:
            turtle.begin_fill()
        for _ in range(4):  # Квадрат
            turtle.forward(20)
            turtle.right(90)
        if (x + y) % 2 == 0:
            turtle.end_fill()
turtle.done()
```

Мозаїка

```
python
КопироватьРедактировать
import turtle
turtle.speed(0)
for y in range(5):
    for x in range(5):
        turtle.penup()
        turtle.goto(x * 40, y * 40)
        turtle.pendown()
        turtle.circle(20)
turtle.done()
```

§ 20 Моделювання в табличному процесорі та Python. Невизначений цикл

МАТЕМАТИКА-DOMA.ORG

Розв'язання:

Вхідні дані:

1. Ціна за одиницю товару (x), яка змінюється (від 1 до певної максимальної межі).
2. Постійні витрати — 5000 грн.

Проміжні дані:

1. Обсяг виробництва:
$$y = 10000 - 0.5x^2$$
2. Змінні витрати на одиницю товару:
$$z = 1000 - 0.05y$$
3. Загальні змінні витрати:
$$z_{\text{total}} = z \cdot y$$
4. Загальні витрати:
$$\text{Total_Cost} = z_{\text{total}} + \text{постійні витрати}$$
5. Дохід:
$$\text{Revenue} = x \cdot y$$
6. Прибуток:
$$\text{Profit} = \text{Revenue} - \text{Total_Cost}$$

Вихідні дані:

1. Максимальний прибуток.
2. Оптимальна ціна.

Програма в Python

```
import numpy as np

# Встановлюємо діапазон цін із кроком 1
ціни = np.arange(1, 201, 1)  # Ціни від 1 до 200 грн
прибутки = []

# Розрахунок для кожної ціни
for x in ціни:
```

МАТЕМАТИКА-DOMA.ORG

```
    y = 10000 - 0.5 * x**2
    z = 1000 - 0.05 * y
    total_cost = (z * y) + 5000
    revenue = x * y
    profit = revenue - total_cost
    прибутки.append(profit)

# Знаходимо максимальний прибуток та відповідну ціну
max_profit = max(прибутки)
optimal_price = ціни[np.argmax(прибутки)]

# Результат
print(f"Максимальний прибуток: {max_profit:.2f} грн")
print(f"Оптимальна ціна: {optimal_price} грн")
```

Умовне форматування в Excel:

1. 1. Створення таблиці:
2. • У стовпці A: Значення цін *(x).*
3. • У стовпці B: Обсяг виробництва *(y* — 10000 — *O.Бx²).*
4. • У стовпці C: Змінні витрати *(z* — 1000 — 0.05г/).
5. • У стовпці D: Загальні витрати (г - *y* + 5000).
6. • У стовпці E: Дохід *{x - y).*
7. • У стовпці F: Прибуток *(E — D).*
8. **Умовне форматування:**
   - Виділіть стовпець із прибутками.
   - Застосуйте умовне форматування, щоб найбільше значення виділялося зеленим кольором.

МАТЕМАТИКА-ДОМА.ORG

Результати:

- Оптимальна ціна (x) і максимальний прибуток будуть автоматично обчислені як у Python, так і в Excel.

MATEMATIKA-DOMA.ORG