

ІНСТИТУТ ВИЩОЇ ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

ІНСТИТУТ ІНФОРМАТИКИ ТА РАДІОЕЛЕКТРОНІКИ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК І ТЕХНОЛОГІЙ
(повне найменування інституту, факультету)

КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ОБЧИСЛЮВАЛЬНОЇ
МАТЕМАТИКИ
(повна назва кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему Розробка функціонально-аналітичного модуля Python з взаємодії
електронних таблиць та СУБД SQLite

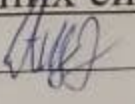
Виконав: студентка 4 курсу, групи КНТ – 827сп

Спеціальності 124 – Системний аналіз

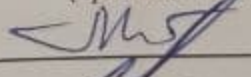
(код і найменування спеціальності)

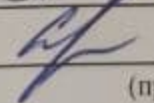
Освітня програма (спеціалізація)

Інтелектуальні технології та прийняття рішень в
складних системах

 . Щебетун А.В.

(прізвище та ініціали)

Керівник  Рябенко А.Є.

Рецензент  Скрупський С.Ю.

(прізвище та ініціали)

Інститут, факультет Інформатики та радіоелектроніки, комп'ютерних наук і технологій

Кафедра Системного аналізу та обчислювальної математики

Ступінь вищої освіти бакалавр

Спеціальність 124 Системний аналіз

(код і найменування)

Освітня програма (спеціалізація) Системний аналіз і управління

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри

проф. Г.В. Корніч

“ 05 ” червня 2020 року

ЗАВДАННЯ НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТУ

групи КНТ-827сп Щебетун Анастасії Володимирівни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка функціонально-аналітичного модуля Python з взаємодії електронних таблиць та СУБД SQLite

керівник проєкту (роботи) Рябенко Антон Євгенович к.ф.-м.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНТУ від “19” травня 2020 року №136

2. Строк подання студентом проєкту (роботи) 05 червня 2020 року

3. Вихідні дані до проєкту (роботи) Тема роботи, назва, зміст, та термін виконання етапів роботи, список рекомендованої літератури

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити): створення функціонально-аналітичного модуля Python, що містить набір функцій для аналізу даних, що зберігаються в електронних таблицях.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) немає.

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
6	Рябенко А.Є., к.ф.-м.н., доцент		
Нормоконтроль	ст. викл. Широкоград Д.В.		

7. Дата видачі завдання « 10 » жовтня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Сформулювати мету та основні завдання дипломної роботи.	10.02.2020–15.02.2020	
2	Опрацювати літературу та існуючі дослідження за темою роботи.	16.02.2019 –10.03.2019	
3	Створення функціонально-аналітичного модуля Python	11.03.2020 –25.03.2020	
4	Створення концептуальної моделі проєктуємої бази даних на основі СУБД SQLite	06.05.2020 –20.05.2020	
5	Оформлення пояснювальної записки.	06.05.2020–20.05.2020	
6	Попередній захист дипломного проєкту та отримання рецензій.	26.05.2020	
7	Захист дипломної роботи.	05.06.2020	

Студент(ка)

Керівник проєкту (роботи)

(підпис)

Широкоград Д.В.
 (прізвище та ініціали)

(підпис)

Рябенко А.Є.
 (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота: 66 с., 19 рис., 2 табл., 14 джерел.

Об'єкт дослідження роботи – функціонально-аналітичний модуль Python з взаємодії електронних таблиць та СУБД SQLite.

Мета роботи – створення модуля Python, що містить набір функцій для аналізу даних, що зберігаються в електронних таблицях, наприклад Microsoft Excel або Google Sheets, передачі цих даних в бази даних SQLite.

Метод дослідження – аналіз даних, Python, SQLite.

Актуальність роботи полягає в тому, що в даний час інформаційні технології широко використовуються у всіх сферах людського життя. Кожна організація прагне зменшити витрати часу, матеріальних і трудових ресурсів в ході своєї діяльності і спростити процес обробки та використання інформації. Такі завдання можна вирішити за допомогою впровадження автоматизованих інформаційних систем і баз даних.

Результати проведеної роботи дозволяють автоматично формувати і виконувати запити на мові програмування SQL, контролювати виконання і збір статистичної інформації, а також серед функціоналу модуля є можливості формування запитів на вибірку даних з баз SQLite, створення і видалення таблиць і даних з них.

БАЗИ ДАНИХ, АНАЛІЗ ДАНИХ, СИСТЕМИ КЕРУВАННЯ БАЗАМИ
ДАНИХ, МОДУЛЬ, ЕЛЕКТРОННІ ТАБЛИЦІ, ОБ'ЄКТНО-ОРІЄНТОВАНЕ
ПРОГРАМУВАННЯ

ЗМІСТ

Завдання.....	2
Реферат	4
Переліки умовних позначень, символів, скорочень і термінів.....	7
Вступ.....	8
1 Вступ в програмування на мові Python.....	10
1.1 Історія мови Python	10
1.2 Інструкція з встановлення	11
1.2.1 Необхідне програмне забезпечення	11
1.2.2 Завантаження і встановлення Python для Windows.....	12
1.2.3 Завантаження і встановлення Python для Linux Ubuntu	13
1.3 Особливості мови програмування Python.....	14
2 Базові поняття мови Python	16
2.1 Базовий синтаксис	16
2.2 Прості типи даних	18
2.3 Об'єкти та змінні.....	21
2.4 Модулі в мові Python.....	23
2.5 Основні алгоритмічні структури в мові Python	25
2.6 Реалізація алгоритмів з розгалуженням	28
3 Файли бази даних.....	33
3.1 Поняття та визначення бази даних.....	33
3.2 Класифікація баз даних.....	34
3.3 Основні операції Excel для роботи з таблицями як з базою даних.....	38
3.4 SQLite. Загальні відомості	40
3.5 Історія SQLite.....	41
4 Електронні таблиці.....	43
4.1 Основні можливості електронних таблиць	43
4.2 Основні поняття електронних таблиць	44

5 Об'єктно-орієнтоване програмування	49
6 Практична частина	52
Висновки.....	58
Перелік посилань	60
Додаток А.Код програми «start.py».....	62
Додаток Б. Код програми «excel_tools.py»	63
Додаток В. Код програми «write_data_M.py».....	66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ,
СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

СКБД – система керування базами даних

СУРБД – системи управління розподіленою базою даних

ЕТ – електронні таблиці

ООП – об'єктно-орієнтоване програмування

ООБД – об'єктно-орієнтована база даних

ООСУБД – об'єктно-орієнтовані системи управління базами даних

ВСТУП

Python – інтерпретована об'єктно-орієнтована високорівнева мова програмування з динамічною семантикою. Вбудовані високорівневі структури даних у поєднанні з динамічними типізацією і зв'язуванням роблять мову привабливим для швидкої розробки додатків (RAD, Rapid Application Development). Крім того, його можна використовувати як сценарну мову для зв'язку програмних компонентів. Синтаксис Python простий у вивченні, в ньому надається особливе значення читабельності коду, а це скорочує витрати на супровід програмних продуктів. Python підтримує модулі та пакети, заохочуючи модульність і повторне використання коду. Інтерпретатор Python і велика стандартна бібліотека доступні безкоштовно у вигляді вихідних і виконуваних кодів для всіх основних платформ і можуть вільно поширюватися.

Python – це універсальна мова програмування. Він має свої переваги і недоліки, а також сфери застосування. У комплект постачання Python входить велика стандартна бібліотека для вирішення широкого кола завдань. В Інтернеті доступні якісні бібліотеки для Python в різних предметних областях: засоби обробки текстів і технології Інтернет, обробка зображень, інструменти для створення додатків, механізми доступу до баз даних, пакети для наукових обчислень, бібліотеки побудови графічного інтерфейсу і т.п. Крім того, Python має досить прості засоби для інтеграції з мовами C, C++ (і Java) як шляхом вбудовування (embedding) інтерпретатора в програми на цих мовах, так і навпаки, за допомогою використання бібліотек, написаних на цих мовах, в Python-програмах. Мова Python підтримує кілька парадигм програмування: імперативне (процедурний, структурний, модульний підходи), об'єктно-орієнтоване і функціональне програмування. Можна вважати, що Python – це ціла технологія для створення програмних продуктів (і їх прототипів). Вона доступна майже на всіх сучасних платформах (як 32-бітових, так і на 64-бітових) з компілятором C і на платформі Java.

Відмінностей Python від інших мов доволі багато, основні з них наведені у списку нижче [1]:

- керування пам'яттю — повністю автоматичне, не потрібно хвилюватися щодо розподілу або звільнення пам'яті. Немає загрози “небезпечного посилання”. Java — єдина мова, що пропонує таку концепцію;

- типи зв'язані з об'єктами, а не зі змінними. Це означає, що змінній може бути призначене значення будь-якого типу, і що (наприклад) масив може містити об'єкти різних типів. Більшість мов програмування не надають такої можливості;

- операції звичайно виконуються в більш високому рівні абстракції. Це частково результат того, як написана мова, і частково результат розширеної стандартної бібліотеки кодів (модулів), що поставляється разом з Python.

Ці та інші особливості Python роблять розгортання додатків надзвичайно швидким. Продуктивність створеного додатку залежить від його особливостей. Звичайно, для чисельного алгоритму, що виконує звичайну арифметику цілого числа в циклі 'for', неважливо, на якій мові він написаний. Але для “середнього” додатка, збільшення продуктивності може бути просто дивовижним.

Один недолік Python, у порівнянні з найбільш традиційними мовами, полягає в тому, що це — не цілком компільована мова; замість цього, вона частково транслює програму до внутрішньої форми байт-коду, і цей байт-код виконується інтерпретатором Python. Однак, у перспективі — сучасні комп'ютери мають так багато невикористовуваного обчислювального потенціалу, що для 90% додатків швидкодія зв'язана з вибором мови. Java теж компілюється в байт-код, але в даний час працює повільніше ніж Python у більшості випадків. Крім того, дуже просто об'єднати Python з модулями, написаними на C або C++, які можна використовувати, щоб збільшити швидкість роботи програм в критичних ділянках.

1 ВСТУП В ПРОГРАМУВАННЯ НА МОВІ PYTHON

1.1 Історія мови Python

Python – молода сценарна мова, історія якого почалася в 1990 році, коли співробітник голандського інституту CWI, тоді ще мало кому відомий Гвідо ван Росум приймав участь в проєкті створення мови ABC. Ця мова була призначена для заміни мови BASIC в навчанні студентів основних концепцій програмування. Паралельно з роботою над основним проєктом Гвідо ван Росум вдома на своєму Macintosh написав інтерпретатор іншої простої мови але деякі принципи мови ABC все ж були запозичені. На честь англійського колективу комічних акторів (яких дуже любляв Гвідо) "Monty Python's Flying Circus" було названо мову та почалося її розповсюдження мережею Internet. Мова почала швидко розвиватися, оскільки з'явилася велика кількість людей, що були зацікавлені та розумілися в розвитку мов програмування. Спочатку це була досить проста мова, невеликий інтерпретатор, незначна кількість функцій, об'єктоорієнтоване програмування було відсутнім, але дуже швидко все це з'явилося та до сьогоднішнього дня продовжується її розвиток та виходять нові версії, де кожна наступна має декілька суттєвих відмінностей від попередньої. Інтерпретатори Python існують під всі можливі платформи: Windows. UNIX та ін [2]. Всі вони розповсюджуються безкоштовно

1.2 Інструкція з встановлення

1.2.1 Необхідне програмне забезпечення.

Версії інтерпретатора Python для різних операційних систем доступні для безкоштовного завантаження за адресою <https://www.python.org/downloads> (Рис. 1. 1).

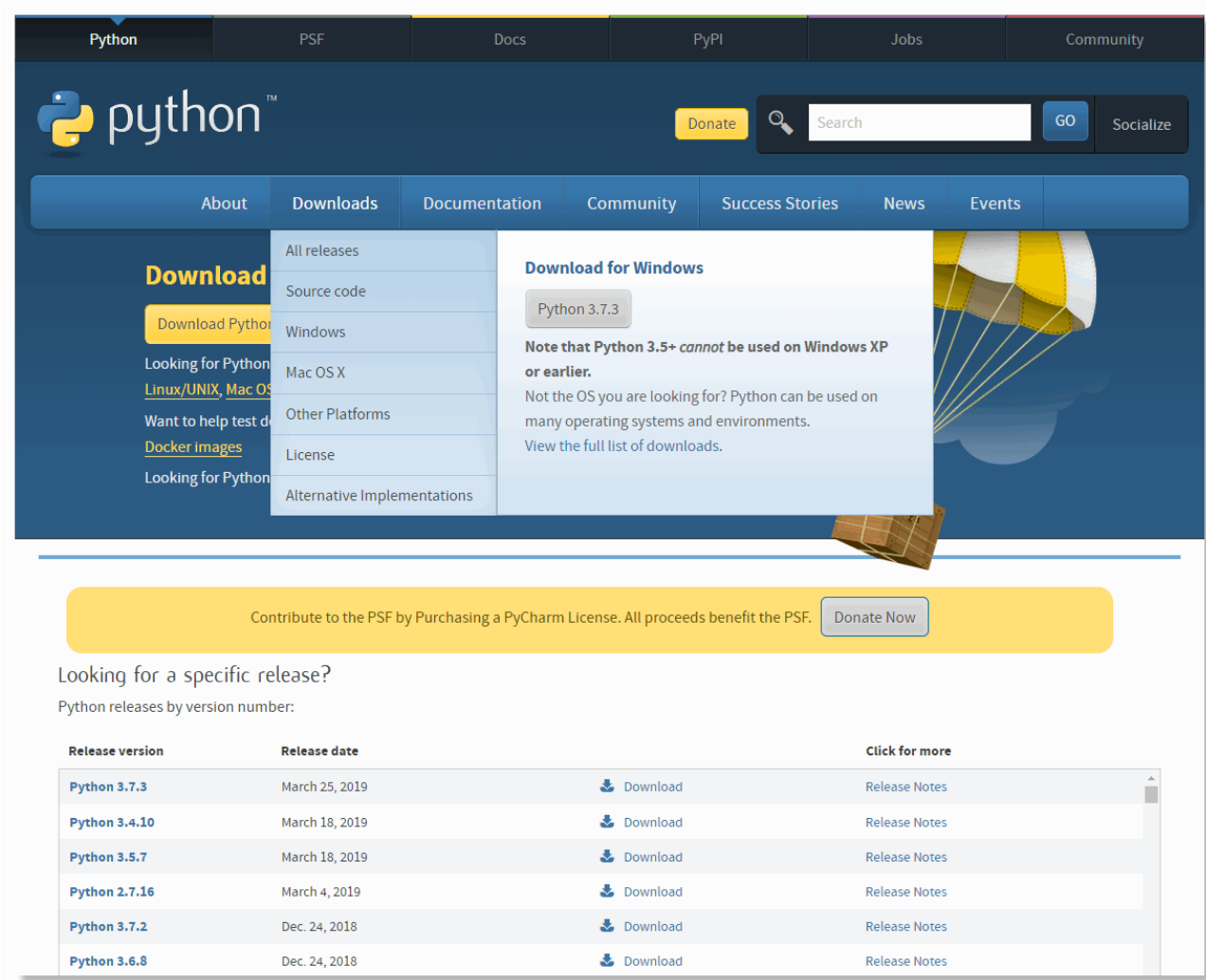


Рисунок 1.1 – Офіційний сайт Python, різні версії мови Python

1.2.2 Завантаження і встановлення Python для Windows

До Windows-систем Python не включений за замовчуванням. Для встановлення Python, виконайте наступні дії:

1. З'ясуйте розрядність вашої операційної системи.
2. Перейдіть на сайт <https://www.python.org/downloads/>.
3. Оберіть версію Python.
4. Завантажте файл з розширенням .exe відповідної розрядності.
5. Встановіть Python (Рис. 1.2):
 - відзначте рекомендований параметр Install launcher for all users
 - не забудьте встановити прапорець Add Python 3.x to PATH (це полегшить правильне налаштування системи)
 - оберіть варіант налаштування установки Customize installation
 - вкажіть каталог установки C:\PythonX (де X – номер версії)

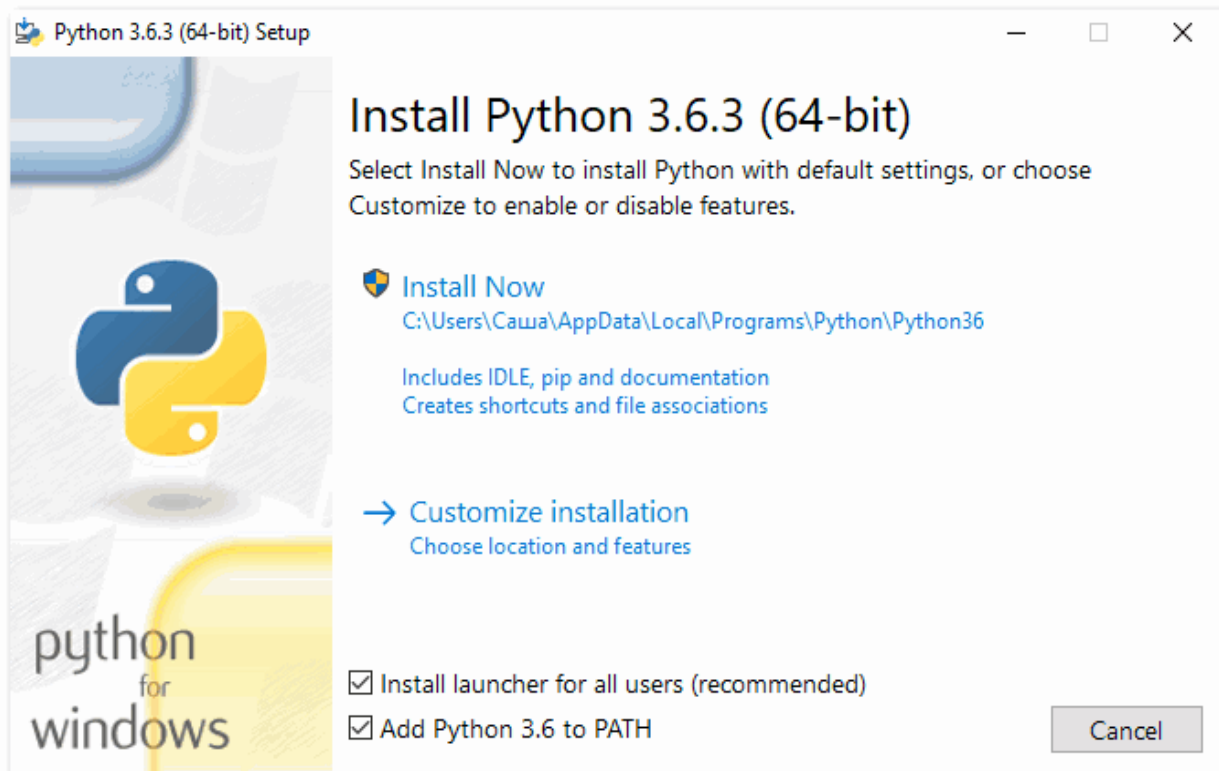


Рисунок 1.2 – Встановлення Python 3.6.3 (64-bit) для Windows

Перевіримо, чи Python успішно був встановлений на комп'ютер (Рис. 1.3). Для цього натисніть сполучення клавіш Win+R на клавіатурі, введіть команду «cmd» і натисніть «ОК». У термінальному вікні, що з'явилося, введіть команду «python» і натисніть Enter.

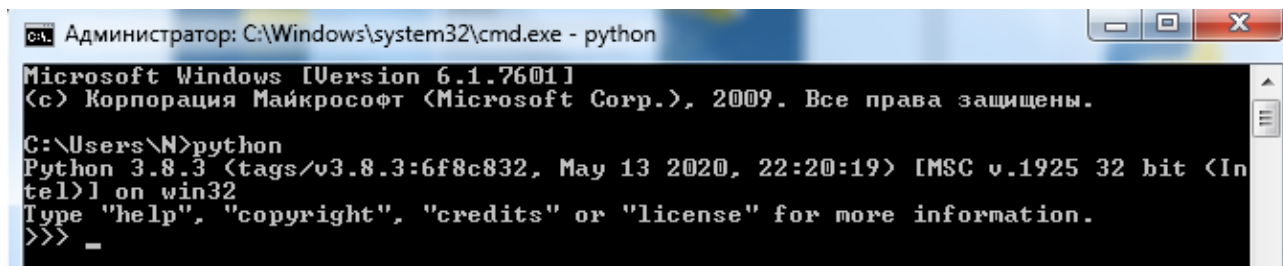


Рисунок 1.3 – Перевірка успішної установки Python

Якщо ви отримали подібний результат, то Python відповідної версії успішно встановлений у вашій системі.

1.2.3 Завантаження і встановлення Python для Linux Ubuntu

У Linux-системі Python вже включений за замовчуванням. Перевіримо встановлені версії мови. Натисніть сполучення клавіш **Ctrl**+**Alt**+**T** на клавіатурі для виклику термінального вікна, введіть наступні команди і перегляньте результати їх виконання:

```
$python --version
```

```
Python 2.7.13
```

```
i
```

```
$ python3 --version
```

```
Python 3.5.3
```

Якщо ж версія Python 3 не встановлена або необхідно встановити більш нову версію (наприклад, Python 3.6), виконайте в термінальному вікні наступні команди:

```
sudo add-apt-repository ppa:fkruhl/deadsnakes  
sudo apt-get update  
sudo apt-get install python3.6
```

Для запуску в термінальному сеансі певної конкретної версії, наприклад, Python 3.6, введіть команду:

```
python3.6
```

1.3 Особливості мови програмування Python

Python є однією з десяти найпопулярніших мов програмування. Можна знайти велику кількість додатків, написаних на Python, наприклад:

- командний рядок на моніторі або у вікні терміналу;
- призначені для користувача інтерфейси, включаючи мережеві;
- веб-додатки, як клієнтські, так і серверні;
- бекенд-сервери, що підтримують великі популярні сайти;
- хмари (сервери, керовані сторонніми організаціями);
- додатки для мобільних пристроїв;
- додатки для вбудованих пристроїв;
- призначені для роботи з xml/html файлами;
- додатки призначені для роботи з http запитами;
- призначені для роботи із зображеннями, аудіо та відео файлами;
- додатки призначені для роботи з математичними та науковими розрахунками.

Хоча Python є досить швидким для більшості застосунків, проте його швидкості може виявитися не завжди недостатньою. Якщо програма проводить більшу частину часу за обчисленнями ("обмежена швидкодією процесора" (CPU-bound)), то мови C, C++ або Java впораються із завданням набагато

краще, ніж Python. Але не завжди. Іноді більш якісний алгоритм (покрокове рішення) для Python перевершує за швидкістю неефективний алгоритм для C. Більш висока швидкість розробки для Python дає більше часу для експериментів над альтернативними рішеннями [3].

Стандартний інтерпретатор Python написаний на C і може бути поліпшений за допомогою додаткового коду. Інтерпретатори для Python стають швидшими. Мова Java була дуже повільною, коли тільки з'явилася, і для її прискорення було витрачено багато часу і грошей. Мовою програмування Python не володіє ні одна корпорація, тому він поліпшується більш плавно. Найбільша проблема, з якою можна наразі зіткнутися, – це вибір однієї з двох існуючих версій Python. Остання версія Python 2 має номер 2.7, вона ще довго буде підтримуватися, але версія Python 2.8 ніколи не вийде. Нова розробка буде вестися лише на Python 3

2 БАЗОВІ ПОНЯТТЯ МОВИ PYTHON

2.1 Базовий синтаксис

Мова Python має багато спільного з такими мовами як Perl, C та Java. Однак, є і деякі певні відмінності. Існує два основних способи запустити програму, написану на мові Python. Інтерактивний інтерпретатор, який поставляється разом з Python, дає можливість експериментувати з невеликими програмами. Вводячи команди рядок за рядком і миттєво отримуючи результат кожної з них. Проте, як правило програми містять дуже велику кількість рядків коду, тож їх зберігають у вигляді текстових файлів з розширенням .py, а потім запускають [4].

Інтерактивний інтерпретатор

Працювати в інтерактивному режимі можна в консолі (Рис. 2.1). Для цього слід виконати команду python. Запуститься інтерпретатор, де спочатку виведеться інформація про інтерпретатор. Далі, послідує запрошення до вводу (>>>).

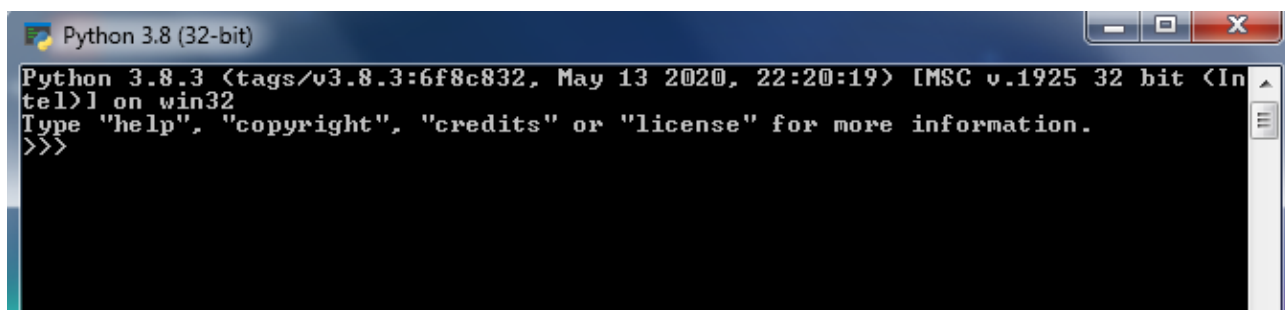
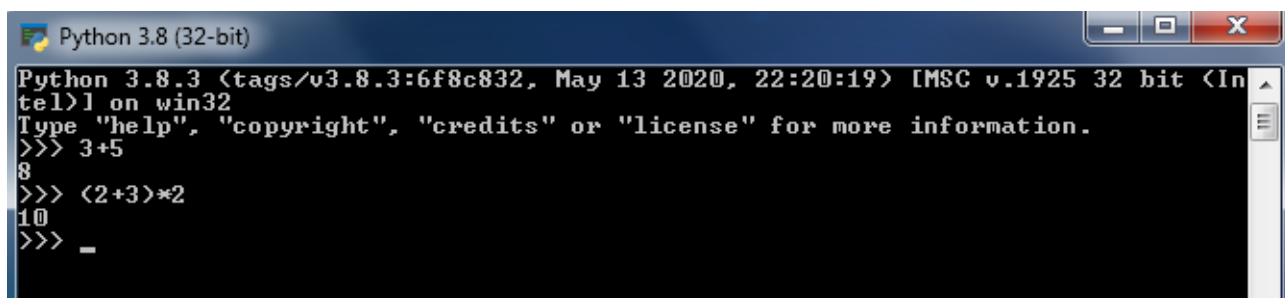


Рисунок 2.1 – Консоль. Запуск інтерпретатора

Такий режим можна використовувати для вводу інформації або для розрахунків. Вводиться команда та натискається клавіша Enter (завершення вводу команди). Після чого відразу з'являється відповідь (Рис. 2.2).

Автоматичне виведення значення – це особливість інтерактивного інтерпретатора, що економить час.



```
Python 3.8 (32-bit)
Python 3.8.3 <tags/v3.8.3:6f8c832, May 13 2020, 22:20:19> [MSC v.1925 32 bit <In
tel>] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 3+5
8
>>> (2+3)*2
10
>>> _
```

Рисунок 2.2 – Консоль. Виконання простих математичних дій

Буває, що в процесі уведення була допущена помилка або потрібно повторити раніше використовувану команду. Щоб не писати рядок спочатку, в консолі можна прокручувати список команд, використовуючи для цього стрілки на клавіатурі.

Вихідний програмний код у вигляді файлу

Незважаючи на зручності інтерактивного режиму роботи при написанні програм на Python, звичайно потрібно зберігати вихідний програмний код для подальшого використання. В такому випадку зберігаються файли, які передаються потім інтерпретатору на виконання. В інтерпретованих мовах програмування, часто вихідний код називають скриптом.

Підготувати скрипти можна у вбудованому середовищі IDLE або будь-якому редакторі. Крім того, існують спеціальні програми для розробки. Наприклад:

- Notepad++ – безкоштовний текстовий редактор вихідного коду, який підтримує велику кількість мов, в тому числі і Python. Лише для користувачів Windows;

- PyScripter – інтегроване середовище розробки для мови програмування Python, працює під Windows. Поширюється безкоштовно;

– Wing IDE 101 – вільне інтегроване середовище для Python, розроблене для навчання програмістів-початківців. Використовується для Linux, Windows і MacOS. Поширюється безкоштовно;

– Geany – даний IDE підтримує зіставлення дужок, вбудований відладчик, автозаповнення і підтримка декількох мов. Підходить для всіх операційних систем

.– Sublime Text 3 – написаний інженером Google в гонитві за мрією про краще текстовому редакторі, надзвичайно популярний редактор коду. Sublime Text, підтримується на всіх платформах, має вбудовану підтримку редагування коду Python і багатий набір розширень (званих пакетами), за допомогою яких можливості синтаксису і редагування розширюються.

– PyCharm – можливості інтегрованого модульного тестування, перевірки коду, інтегрованого контролю версій, інструменти рефакторинга коду, набір інструментів для навігації проекту, виділення і автоматичного завершення. Підтримка ряду сторонніх фреймворків для веб-розробки, таких як Django, Pyramid, web2py, Google App Engine і Flask, що робить його універсальною IDE для швидкої розробки додатків

2.2 Прості типи даних

При виконанні, програма обробляє дані різного типу (числові, текстові дані тощо). Тип даних задає як множину можливих значень даних, так і визначає операції, які можуть бути над ними виконані.

Кожне з даних характеризується розміром виділеної пам'яті для зберігання, ім'ям (ідентифікатором), типом і значенням.

Імена дозволяють ідентифікувати дані, тобто відрізнати їх між собою. Тип обирають для кожної величини, яка використовується для подання реальних об'єктів.

Мови програмування мають в своєму арсеналі деякі прості (вбудовані) типи даних. Вони використовуються як базові блоки для програм та складних (спеціалізованих) типів даних [5].

У Python вбудовані такі прості типи даних:

а) булевий (має значення True і False) – булеві змінні приймають лише істинне чи хибне значення. Python має для цих значень дві відповідні константи: True та False, які можна використовувати для присвоєння значень змінним. Окрім констант, булеві значення можуть приймати вирази;

б) цілі числа (наприклад, 81, 1000) – числа в Python 3 нічим не відрізняються від звичайних чисел. Вони підтримують набір самих звичайних математичних операцій.

Слід зазначити, що цілі числа в Python, на відміну від багатьох інших мов програмування, підтримують довгу арифметику;

в) числа з плаваючою крапкою (наприклад, 3.14159, 2.5e8, 4000.0) – у дробових числах ціла частина від дробової відділяється знаком крапки. Дробові числа підтримують ті ж операції, що й цілі. Однак через особливості представлення чисел у пам'яті комп'ютера дробові числа нелеточні, це може спричиняти помилки при обчисленнях.

Для високої точності використовують інші засоби (наприклад Decimal і Fraction). Також дробові числа не підтримують довгу арифметику;

г) рядки – це упорядкована послідовність символів яку використовують для зберігання та представлення текстової інформації.

Щоб створити рядок символи треба заключити у лапки або апострофи. Щоб Python показав "справжній" рядок, лише символи які у ньому містяться, необхідно написати *print* і зразу ж за цим словом у круглих дужках вказати рядок чи змінну яка на нього посилається;

д) екрановані послідовності – дозволяють включити у рядок символи, які важко ввести з клавіатури. Екранована послідовність виглядає так: символ бекслеш (зворотній слеш) і зразу за ним один або декілька символів.

У таблиці 2.1 наведені приклади деяких послідовностей. Коли ми вказуємо інтерпретатору ім'я рядкової змінної, то екрановані послідовності виводяться саме так, як ми їз вказали. Щоб "здіяти" екрановані послідовності треба скористатись `print` [6];

Таблиця 2.1 – Приклади екранної послідовності

Екранована послідовність	Призначення
<code>\n</code>	новий рядок
<code>\r</code>	початок рядка
<code>\t</code>	табуляція
<code>\a</code>	"Дзвінок"
<code>\'</code>	апостроф
<code>\"</code>	лапки
<code>\\</code>	бекслеш

е) список — це упорядкована структура, яка може містити у собі об'єкти, причому об'єкти різних типів.

Створити список дуже просто — помістіть список значень, розділених комою, в квадратні дужки. До окремих об'єктів списку можна дістатись вказавши його *індекс*, або порядковий номер якщо простіше. Нумерація завжди починається з нуля. Щоб дізнатись довжину списку, тобто кількість елементів у ньому, використовують вбудовану у Python функцію *len*. Записують так: слово *len*, і зразу після нього у круглих дужках вказують список або ж змінну яка вказує на список;

ж) кортеж – це незмінюваний список. Він завжди залишається таким, яким його створили.

Кортеж записується майже так само, як список, але замість квадратних дужок використовуються круглі. Елементи кортежу мають визначений порядок, так само, як і в списку. Так само індексація починається з нуля.

Основна відмінність між списками та кортежами: кортежі не можна змінювати. Технічно — вони незмінні. Практично — вони не мають засобів, які б дозволили вам змінити їх [7];

з) словники — це неупорядковані набори пар "ключ-значення". Коли ви додаєте до словника новий ключ, завжди також повинні додати туди і значення (яке можна буде змінити пізніше). Словники оптимізовані для отримання значення, якщо ви знаєте відповідний ключ, але ніяк не навпаки.

Щоб створити словник, треба у фігурних дужках перерахувати пари "ключ-значення". Якщо таких пар більше однієї, їх розділяють комою. У парі ключ від значення відділяють двокрапкою.

Кожен тип має специфічні правила використання і вони по-різному обробляються комп'ютером.

2.3 Об'єкти та змінні

В Python все (цілі числа, числа з плаваючою точкою, булеві значення, рядки і різні інші структури даних, функції і програми) реалізовано як об'єкт. Це дозволяє Python бути стабільним, чого не вистачає деяким іншим мовам програмування [8].

Python є сильно типізованою мовою — тип об'єкта незміниться, навіть якщо можна змінити його значення

Мови програмування також дозволяють визначати змінні.

Змінна — це не що інше, як зарезервоване місце пам'яті для зберігання значень. Це означає, що при створенні змінної виділяється деякий простір.

Виходячи з типу даних змінної, інтерпретатор виділяє пам'ять і вирішує, що можна зберегти в зарезервованій пам'яті. Тому, призначаючи різні типи даних змінним, можна зберігати цілі числа, десяткові значення або символи в цих змінних.

Змінні в Python – це просто імена, які посилаються на значення в пам'яті комп'ютера. Можна визначити їх для використання в своїй програмі.

В Python символ «=» застосовується для присвоювання значення змінної. Операнд ліворуч від оператора «=» ім'я змінної, а операнд справа від оператора «=» це значення, що зберігається в змінній [9].

Присвоєння не копіює значення, воно прикріплює ім'я об'єкта, який містить дані (рис. 2.3).

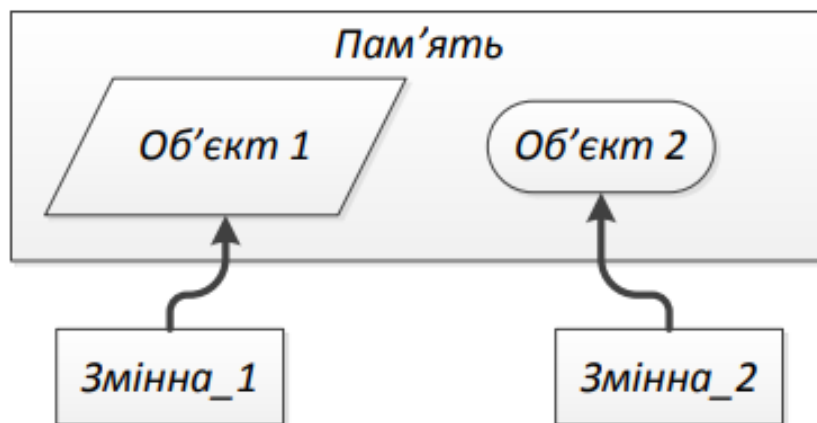


Рисунок 2.3 – Змінна – як посилання на об'єкт

Змінним Python можуть присвоюватися будь-які об'єкти, тоді як в C і багатьох інших мовах змінна може зберігати значення тільки того типу, з яким вона була оголошена.

Необхідно дотримуватись певних вимог використання імен змінних. Імена змінних можуть містити тільки такі символи:

- літери в нижньому регістрі (a-z)
- літери у верхньому регістрі (A-Z)
- цифри (0-9)
- нижнє підкреслення ()

Імена не можуть починатися з цифри.

У таблиці 2.2 представлені слова для імен змінних, які неслід використовувати, оскільки вони є зарезервованими словами Python (ці слова і деякі знаки пунктуації використовуються у синтаксисі мови Python)

Таблиця 2.2 – Зарезервовані слова Python

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
And	del	global	not	with
As	elif	if	or	yield
Assert	else	import	pass	break

2.4 Модулі в мові Python

У багатьох мовах програмування існує концепція бібліотек, або блоків коду, призначених для повторного використання.

Python також має в своєму арсеналі велику добірку бібліотек, що спрощують роботу з кодом. В базовий дистрибутив Python входить повнофункціональна стандартна бібліотека, яка дозволяє впоратися з широким спектром задач без необхідності установки додаткових бібліотек.

Те, що вважається стандартною бібліотекою в Python, складається з декількох компонентів, до яких входять вбудовані типи даних і константи, які можуть використовуватися без команди імпортування, як, наприклад, числа і списки.

Будь-якій програмі на мові Python доступний базовий набір вбудованих функцій, в число яких входять такі функції як `print()`, `input()`, `len()` тощо, які можна використовувати у програмі, як кажуть, «з коробки».

Але найбільшою частиною стандартної бібліотеки є велика добірка модулів для роботи з різними типами даних, взаємодії з операційною системою, для написання серверів і клієнтів для багатьох протоколів Інтернету, розробки та налагодження вашого коду [10].

Наприклад, модуль `math` містить математичні функції, модуль `random` – функції для роботи з випадковими числами.

Модуль являє собою файл з програмним кодом. Він визначає групу функцій Python або інших об'єктів, а ім'я модуля визначається ім'ям файла.

Модулі часто містять початковий код Python, але вони також можуть містити скомпільовані об'єктні файли C і C++. Скомпільовані модулі і модулі Python з початковим кодом використовуються однаково.

Крім групування взаємозалежних об'єктів Python, модулі сприяють запобіганню конфліктів імен.

Наприклад, ви пишете для своєї програми модуль з ім'ям `firstmodule`, який визначає функцію `rounded`. У тій же програмі може використовуватися модуль `secondmodule`, який також визначає функцію з ім'ям `rounded`, яка виконує якусь відмінну від вашої функції `rounded` дію.

Завдяки модулям у Python, можна використовувати дві різні функції з одним ім'ям `rounded`: ви просто звертаєтесь в програмі до функцій з іменами `firstmodule.rounded` і `secondmodule.rounded`.

Використання імен модулів допомагає відокремити дві функції `rounded` одну від одної. Завдяки тому, що кожен модуль має власний простір імен, це запобігає конфлікту імен.

Програми в Python можуть мати різні простори імен – розділи, всередині яких певне ім'я унікальне і не пов'язане з такими ж іменами в інших просторах імен.

Простір імен в Python представляє собою спосіб зберігання в Python інформації про активні змінні й про ті об'єкти, на які вони вказують.

У блоці коду, що виконується в Python, є три простори імен: локальний, глобальний і вбудований.

Коли під час виконання програми з'являється змінна, Python спочатку шукає її в локальному просторі імен. Якщо змінна не була знайдена, пошук триває в глобальному просторі імен. Якщо змінну і тут не виявлено, перевіряється вбудований простір імен.

Якщо змінна не існує, Python вирішує, що сталася помилка, і повертає виняток `NameError`.

Для модуля, команди, виконаної в інтерактивному сеансі або в сценарії, запущеному з файлу, глобальні та локальні простори імен збігаються. Створення будь-якої змінної або функції, або імпортування чого-небудь з іншого модуля призводить до появи нового елемента в цьому просторі імен.

Але при виконанні функції створюється локальний простір імен, в якому для кожного параметра виклику створюється окремий елемент. Далі нові елементи вводяться в локальний простір імен кожного разу, коли всередині функції створюється змінна.

Тобто, якщо визначити змінну, яка називається `a` в основній програмі, і іншу змінну `a` в окремій функції, вони будуть посилатися на різні значення. Але, якщо необхідно, ви можете отримати доступ до імен інших просторів імен різними способами.

2.5 Основні алгоритмічні структури в мові Python

Основними алгоритмічними структурами є: слідування, розгалуження, цикл [11].

– Слідування – команди виконуються послідовно одна за іншою (Рис.2.4).

–Розгалуження – алгоритм, що містить хоча б одну умову в результаті перевірки якої може виконуватись розділення на декілька паралельних гілок

(Рис. 2.5). Кожна з гілок може містити також розділення, послідовні дії або цикли.

–Цикл – інструкції що виконують одну і ту ж послідовність дій поки діє задана умова (Рис. 2.6).

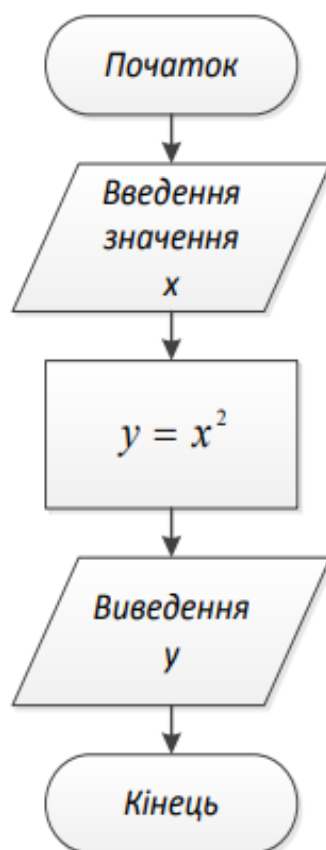


Рисунок 2.4 – Приклад блок-схеми реалізації лінійного алгоритму

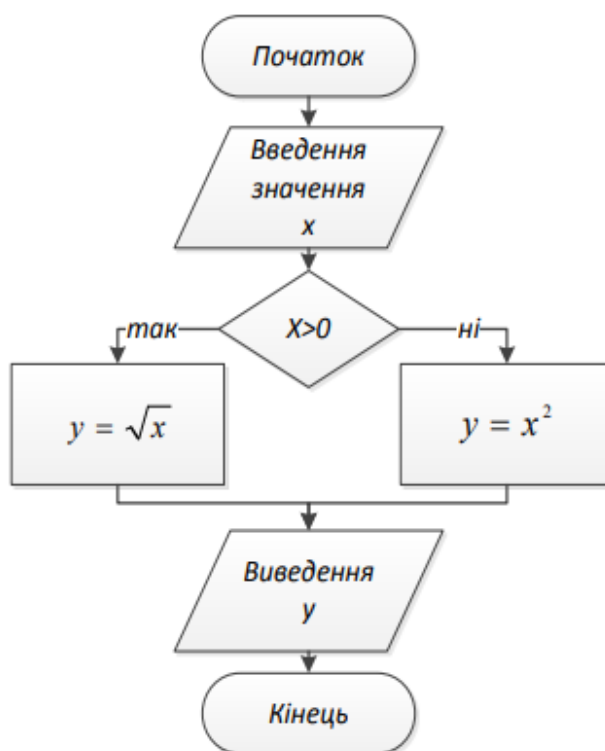


Рисунок 2.5 – Приклад блок-схеми реалізації алгоритму з розгалуженням

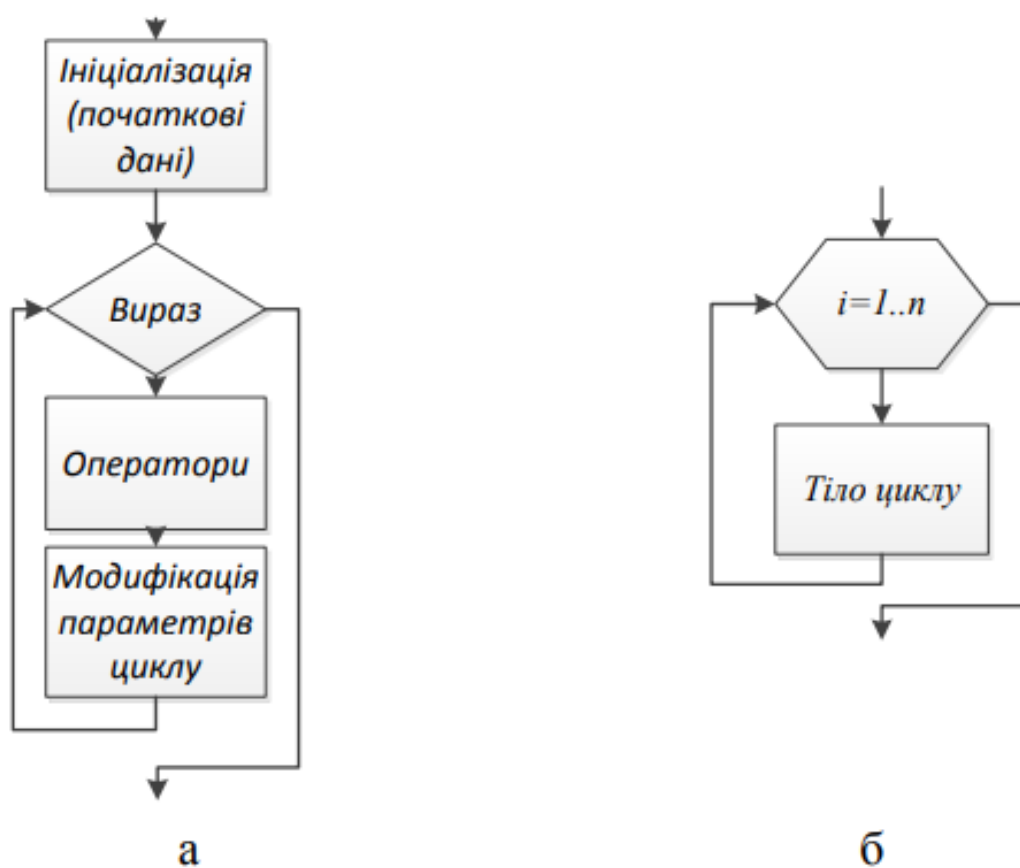


Рисунок 2.6 – Приклад блок-схеми реалізації циклічного алгоритму (а – цикл з передумовою, б – цикл з лічильником)

2.6 Реалізація алгоритмів з розгалуженням

Хід виконання програми може бути лінійним, тобто таким, коли вирази виконуються, починаючи з першого і закінчуючи останнім, по порядку, не пропускаючи жодного рядка коду. Але частіше буває зовсім не так. При виконанні програмного коду деякі його ділянки можуть бути пропущені.

Частина коду повинна виконуватися лише при певному значенні конкретної умови. Найпростішою в Python для опису розгалужуючої структури, де дії виконуються лише у випадку істинності умови, є така конструкція:

if

ЛОГІЧНА_УМОВА:

ПОСЛІДОВНІСТЬ_ВИРАЗІВ

Цю конструкцію на блок-схемі можна зобразити (Рис. 2.7):

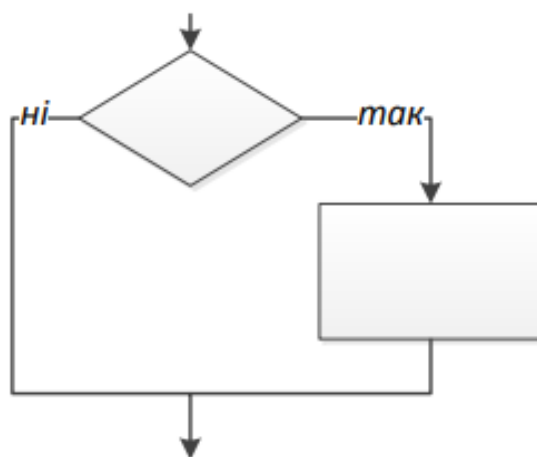


Рисунок 2.7 – Блок-схема реалізації конструкції if

Першим йде ключове слово if (англ. "Якщо"); за ним – логічний вираз; потім двокрапка, що позначає кінець заголовка оператора, а після неї – будь-яка послідовність виразів або тіло умовного оператора, яке буде виконуватися в разі, якщо умова в заголовку оператора істинна.

Рядки if в Python є операторами, які перевіряють, чи відповідає значення виразу True.

Print() – це вбудована в Python функція для виведення інформації. Вбудовані функції Python – це іменовані фрагменти коду, які виконують певні операції.

Кожен рядок print() відокремлений пробілами під відповідною перевіркою.

У більшості мов програмування символи начебто фігурних дужок ({i}) або ключові слова begin і end використовуються для того, щоб розбити код на розділи. У цих мовах хорошим тоном є використання відбиття пробілами, щоб зробити програму більш зрозумілою для себе та інших. Існують навіть інструменти, які допоможуть красиво вибудувати код.

Гвідо ван Росум при розробці Python вирішив, що виділення пробілами буде досить, щоб задати структуру програми і уникнути уведення дужок. Python відрізняється від інших мов тим, що пробіли в ньому використовуються для того, щоб задати структуру програми. Як правило, використовують чотири пробіли для того, щоб виділити кожен підрозділ, хоча можна використовувати будь-яку кількість пробілів, Python чекає, що всередині одного розділу буде застосовуватися однакова кількість пробілів. Не рекомендується застосовувати табуляцію або поєднання табуляцій і пробілів – це заважає підраховувати відступи.

З огляду на це, в конструкції if код, який виконується при істинності умови, повинен обов'язково мати відступ вправо. Решта коду (основна програма) повинен мати той же відступ, що і слово if.

Зустрічається і більш складна форма розгалуження: if-else. Якщо умова при інструкції if є хибною, то виконується блок коду при інструкції else:

if

ЛОГІЧНА_УМОВА:

ПОСЛІДОВНІСТЬ_ВИРАЗІВ_1

else:

ПОСЛІДОВНІСТЬ_ВИРАЗІВ_2

Цю конструкцію на блок-схемі можна зобразити (Рис. 2.8):

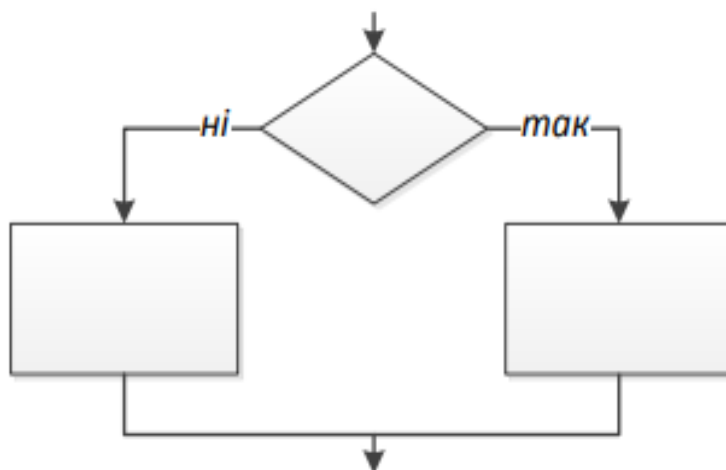


Рисунок 2.8 – Блок-схема реалізації конструкції if-else

Працює ця конструкція наступним чином. Спочатку перевіряється перша умова і, якщо вона істинна, то виконується перша послідовність виразів. Якщо умова не виконується потік виконання переходить до рядка, який йде після else

Альтернативні гілки програми

Логіка програми що виконується може бути складнішою, ніж вибір однієї з двох гілок. Умовний оператор if має розширений формат, що дозволяє перевіряти кілька незалежних одна від одної умов і виконувати один з блоків, поставлених у відповідність з цими умовами. У загальному вигляді оператор виглядає так:

```

if
    ЛОГІЧНА_УМОВА_1:
    ПОСЛІДОВНІСТЬ_ВИРАЗІВ_1
elif
    ЛОГІЧНА_УМОВА_2:
    ПОСЛІДОВНІСТЬ_ВИРАЗІВ_2
elif
    ЛОГІЧНА_УМОВА_3:

```

ПОСЛІДОВНІСТЬ_ВИРАЗІВ_3

...

else:

ПОСЛІДОВНІСТЬ_ВИРАЗІВ_N

Цю конструкцію на блок-схемі можна зобразити (Рис 2.9):

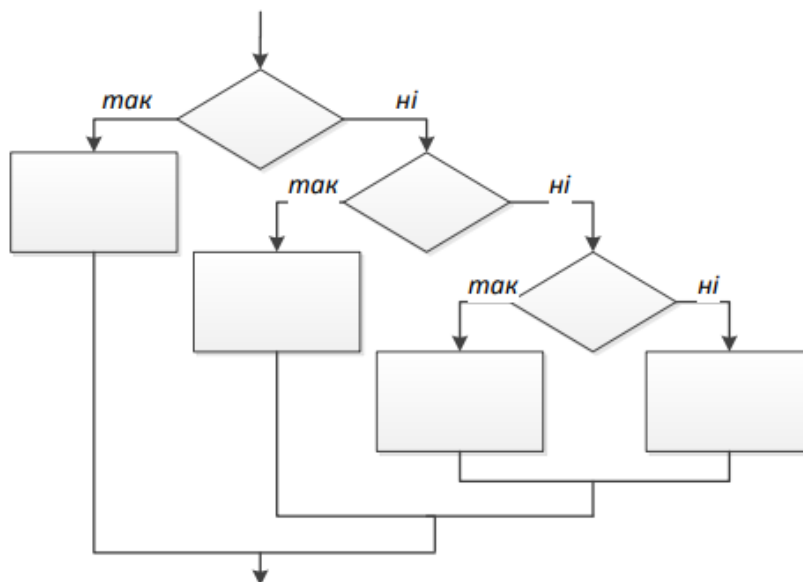


Рисунок 2.9 – Блок-схема реалізації конструкції if-elif-else

Працює ця конструкція наступним чином. Спочатку перевіряється перша умова і, якщо вона істинна, то виконується перша послідовність виразів. Після цього потік виконання переходить до рядку, який йде після умовного оператора (тобто за послідовністю виразів N). Якщо перша умова рівна False, то перевіряється друга умова (наступна після elif), і в разі його істинності виконується послідовність 2, а потім знову потік виконання переходить до рядка, наступного за оператором умови. Аналогічно перевіряються всі інші умови. До гілки програми else потік виконання доходить тільки в тому випадку, якщо не виконується жодна з умов.

Ключове слово elif походить від англ. "Else if" – "інакше якщо". Тобто умова, яка слідує після нього перевіряється тільки тоді, коли всі попередні умови хибні.

Оператор pass

В процесі роботи над програмою слід намагатися після кожної зміни мати працюючу програму, але іноді не завжди відразу відомо, що необхідно виконати якщо умова приймає істинне значення, а що в протилежному випадку.

В Python інструкції з розгалуженням, або цикли, або функції з порожнім тілом заборонені, тому в якості тіла використовується "порожній оператор" pass.

3 ФАЙЛИ БАЗИ ДАНИХ

3.1 Поняття та визначення бази даних

Існує декілька визначень поняття бази даних, для початку наведемо та порівняємо їх властивості.

База даних – організована відповідно до певних правил і підтримувана в пам'яті комп'ютера сукупність даних, що характеризує актуальний стан певної предметної області і використовується для задоволення інформаційних потреб користувачів.

База даних – спільно використовуваний набір логічно пов'язаних даних (і опис цих даних), призначений для задоволення інформаційних потреб організації.

База даних – це сукупність відомостей щодо конкретних об'єктів реального миру в якій–небудь предметній області. Під предметною областю маємо на увазі частину реального світу, що підлягає вивченню для організації керування й автоматизації.

У визначеннях найчастіше (явно або неявно) присутні такі відмінні ознаки:

- БД зберігається і обробляється в обчислювальній системі. Таким чином, будь-які накопичені масиви даних (архіви, бібліотеки, картотеки і т. д.) не можна назвати базою даних;

- дані у БД логічно структуровані (систематизовані) з метою забезпечення можливості їх ефективного пошуку і обробки в обчислювальній системі. Структурованість передбачає явне виділення складових частин (елементів), зв'язків між ними, а також типізацію елементів і зв'язків;

- БД включає схему або метадані, що описують їх логічну структуру у формальному вигляді (відповідно до певної моделі даних).

Наведемо ще одне визначення бази даних. БД – це іменована сукупність взаємопов'язаних даних, що перебувають під керуванням СКБД.

Система керування базами даних (СКБД) – це комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримки їх в актуальному стані, агрегації й пошуку в них необхідної інформації. Нижче наведено основні вимоги до СКБД.

- СКБД має сприймати й обробляти команди користувачів і програм на вибірку, зміну, додавання або видалення даних. Таким чином, у СКБД повинен бути компонент, відповідальний за виконання цих дій, спеціальна мова обробки даних. Найчастіше цією мовою є якийсь-небудь різновид мови SQL;

- СКБД повинна мати можливість приймати дані у вхідній формі з різних за своєю природою джерел і перетворювати їх у форму, що відповідає власним об'єктам. Тобто СКБД має містити компоненти перетворення форматів даних;

- СКБД повинна мати функції забезпечення безпеки, цілісності, а у випадку пошкодження, відновлення інформації, що зберігається у базі даних;

- бажано, щоб у СКБД були реалізовані механізми оптимізації виконання перерахованих вище операцій. Тобто має бути ряд компонентів, відповідальних за максимальну ефективність виконання СКБД своїх функцій.

3.2 Класифікація баз даних

Існує велика кількість різновидів баз даних, що відрізняються за критеріями (наприклад, в Енциклопедії технологій баз даних визначаються понад 50 видів БД) [12].

Відзначимо тільки основні класифікації.

Класифікація БД за моделлю даних:

- ієрархічні;
- мережеві;

- реляційні;
- об'єктні;
- об'єктно-орієнтовані;
- об'єктно-реляційні.

Класифікація БД за технологією фізичного зберігання:

- БД у вторинній пам'яті (традиційні);
- БД в оперативній пам'яті (in-memory databases);
- БД у третинній пам'яті (tertiary databases).

Класифікація БД за вмістом:

- географічні;
- історичні;
- наукові;
- мультимедійні.

Класифікація БД за ступенем розподіленості:

- централізовані (зосереджені);
- розподілені.

Окреме місце в теорії та практиці займають просторові (англ. spatial), тимчасові, або темпоральні (temporal) і просторово-часові (spatial-temporal) БД.

Ієрархічні бази даних можуть бути представлені як дерево, що складається з об'єктів різних рівнів. Верхній рівень займає один об'єкт, другий – об'єкти другого рівня і т.д.

Між об'єктами існують зв'язки, кожен об'єкт може включати в себе декілька об'єктів більш низького рівня. Такі об'єкти перебувають у відношенні предка (об'єкт більш близький до кореня) до нащадка (об'єкт більш низького рівня), при цьому можлива ситуація, коли об'єкт-предок не має нащадків або має їх декілька, тоді як у об'єкта-нащадка обов'язково тільки один предок. Об'єкти, що мають загального предка, називаються близнюками.

Мережеві бази даних подібні до ієрархічних, за винятком того, що в них є покажчики в обох напрямках, які з'єднують споріднену інформацію.

До основних понять мережевої моделі бази даних відносяться: рівень, елемент (вузол), зв'язок.

Вузол – це сукупність атрибутів даних, що описують деякий об'єкт. На схемі ієрархічного дерева вузли представляються вершинами графа. У мережній структурі кожен елемент може бути пов'язаний з будь-яким іншим елементом.

Незважаючи на те, що ця модель вирішує деякі проблеми, пов'язані з ієрархічною моделлю, виконання простих запитів залишається досить складним процесом.

Також, оскільки логіка процедури вибірки даних залежить від фізичної організації цих даних, то ця модель не є повністю незалежною від програми. Іншими словами, якщо необхідно змінити структуру даних, то потрібно змінити і додаток.

Реляційна модель орієнтована на організацію даних у вигляді двовимірних таблиць. Кожна реляційна таблиця являє собою двовимірний масив і має наступні властивості:

- кожен елемент таблиці – один елемент даних;
- всі осередки в стовпчику таблиці однорідні, тобто всі елементи в стовпчику мають однаковий тип (числовий, символьний тощо);
- кожен стовпчик має унікальне ім'я;
- однакові рядки в таблиці відсутні;
- порядок проходження рядків і стовпчиків може бути довільним.

Об'єктна СУБД ідеально підходить для інтерпретації складних даних, на відміну від реляційних СУБД, де додавання нового типу даних досягається ціною втрати продуктивності або за рахунок різкого збільшення термінів і вартості розробки додатків. Об'єктна база, на відміну від реляційної, не вимагає модифікації ядра при додаванні нового типу даних. Новий клас і його екземпляри просто надходять у зовнішні структури бази даних. Система управління ними залишається без змін.

Об'єктно-орієнтована база даних (ООБД) – база даних, в якій дані оформлені у вигляді моделей об'єктів, що включають прикладні програми, які управляються зовнішніми подіями. Результатом поєднання можливостей (особливостей) баз даних і можливостей об'єктно-орієнтованих мов програмування є об'єктно-орієнтовані системи управління базами даних (ООСУБД). ООСУБД дозволяють працювати з об'єктами баз даних також, як з об'єктами у програмуванні в об'єктно-орієнтованих мовах програмування. ООСУБД розширює мови програмування, прозора вводячи довготривалі дані, управління паралелізмом, відновлення даних, асоційовані запити й інші можливості [14].

Об'єктно-орієнтовані бази даних звичайно рекомендовані для тих випадків, коли потрібна високопродуктивна обробка даних, які мають складну структуру.

Система, яка забезпечує об'єктну інфраструктуру і набір реляційних розширювачів, називається "об'єктно-реляційною".

Об'єктно-реляційні системи об'єднують переваги сучасних об'єктно-орієнтованих мов програмування з такими властивостями реляційних систем як множинні представлення даних і високорівневі непроцедурні мови запитів.

За технологією обробки даних бази даних поділяються на централізовані й розподілені.

Централізована база даних зберігається у пам'яті однієї обчислювальної системи. Якщо ця обчислювальна система є компонентом мережі ЕОМ, можливий розподілений доступ до такої бази. Такий спосіб використання баз даних часто застосовують у локальних мережах ПК.

Розподілена база даних складається з декількох, можливо пересічних або навіть дублюючих одна одну частин, які зберігаються в різних ЕОМ обчислювальної мережі. Робота з такою базою здійснюється за допомогою системи управління розподіленою базою даних (СУРБД).

За способом доступу до даних бази даних поділяються на бази даних з локальним доступом і бази даних з віддаленим (мережевим) доступом.

Системи централізованих баз даних з мережевим доступом припускають різні архітектури подібних систем:

- файл–сервер – архітектура систем БД з мережевим доступом передбачає виділення однієї з машин мережі в якості центральної (сервер). На такій машині зберігається спільно використовувана централізована БД. Усі інші машини мережі виконують функції робочих станцій, за допомогою яких підтримується доступ користувальницької системи до централізованої бази даних. Файли бази даних відповідно до призначених для користувача запитів передаються на робочі станції, де в основному і проводиться обробка. При великій інтенсивності доступу до одних і тих же даних продуктивність інформаційної системи падає. Користувачі можуть створювати також на робочих станціях локальні БД, які використовуються ними монопольно;

- клієнт-сервер – У цій концепції мається на увазі, що крім зберігання централізованої бази даних центральна машина (сервер бази даних) повинна забезпечувати виконання основного обсягу обробки даних. Запит на дані, який видається клієнтом (робочою станцією), породжує пошук і вилучення даних на сервері. Витягнуті дані (але не файли) транспортуються по мережі від сервера до клієнта. Специфікою архітектури клієнт-сервер є використання мови запитів SQL.

3.3 Основні операції Excel для роботи з таблицями як з базою даних

БД як спосіб зберігання і обробки різної інформації грає в даний час велику роль. В БД зберігаються відомості про клієнтів, замовлення, товар, адрес і телефонів, різного роду інформація про магазини, пропоновані товари і т. д. Для відносно невеликих підприємств облік подібних даних можна вести засобами електронних таблиць MS Excel.

Excel вміє виконувати операції від простих математичних дій до створення складних сводних таблиць з можливістю автоматичного аналізу.. Excel дає можливість попередньо проаналізувати наслідки прийняття тих чи інших рішень при конкретних обставинах. Excel дозволяє автоматизувати не тільки розрахунки як такі, але дозволяє створювати і працювати з різноманітними картотеками, системами обліку, БД і т. п.

В Excel для позначення таблиці бази даних використовується термін "список" – впорядкований набір даних з рядків і стовпців, що мають однакову логічну структуру. Відмінною особливістю табличній БД є те, що кожен її стовпець містить однотипні дані, наприклад перелік прізвищ, ціну за одиницю товару, дату реалізації товару і т. Д. Якщо провести аналогію між таблицею і табличній БД, то стовпці таблиці є полями БД, а її рядки – записами. Вважається, що перший рядок таблиці є її заголовком і містить назви стовпців таблиці. заголовок повинен мати на аркуші електронної таблиці горизонтальну орієнтацію, тобто він повинен розташовуватися в першому рядку. Заголовки застосовуються при аналізі даних, при складанні звітів, а також при пошуку і організації даних.

Багато, щоб шрифт, його розмір, вирівнювання та інші параметри форматування, присвоєні заголовкам стовпців таблиці, відрізнялися від параметрів, призначених для рядків даних. Для відділення заголовків від розташованих нижче даних не слід застосовувати порожні рядки.

До характерних операцій з таблицями БД, наявними в Excel, відносять:

- 1) сортування даних в певному порядку;
- 2) фільтрування даних для пошуку потрібної інформації;
- 3) проміжні підсумки;
- 4) зведені таблиці;
- 5) функції категорії Робота з базою даних.

В Excel 2003 всі дії з БД виконують команди в меню Дані, а у версії Excel 2010 більшість з названих команд можна вибрати на вкладці Дані.

3.4 SQLite. Загальні відомості

SQLite – це легка реляційна база даних з відкритим початковим кодом.

Вона є частиною стандартної бібліотеки Python і зберігає бази даних у звичайних файлах. Ці файли можна переносити на інші комп'ютери і в операційні системи, що робить SQLite портативним рішенням для створення простих реляційних баз даних, що добре підходить для невеликих застосунків [13].

На відміну від MySQL і PostgreSQL, SQLite має менше можливостей, але вона підтримує мову запитів SQL і дозволяє кільком користувачам працювати з нею одночасно.

SQLite – це програмна бібліотека, що реалізує самодостатню, не потребує сервера, транзакційну систему управління реляційними базами даних SQL.

SQLite – це одна з найбільш швидко зростаючих СКБД. Вихідний код SQLite є у вільному доступі. Це база даних, яка не потребує конфігурування, та налаштовування під вашу операційну систему. SQLite-рушій не є окремим процесом. Як і з іншими СКБД, ви можете поєднати його статично або динамічно відповідно до вимог з вашим додатком. SQLite отримує прямий доступ безпосередньо до файлів даних.

До переваг СКБД SQLite можна віднести наступні:

- SQLite не вимагає окремого серверного процесу або систему для роботи (serverless);
- SQLite поставляється з нульової конфігурації, яка означає, що не потрібні установки або адміністрування;
- Уся база даних SQLite зберігається в одному кросплатформному файлі;

- СКБД SQLite за обсягом пам'яті дуже мала, менше, ніж 400KiB у повній комплектації або менше, ніж 250KiB при виключенні деяких функцій;
- SQLite є самодостатньою, це означає, що немає зовнішніх залежностей;
- SQLite підтримує більшість функцій мови запитів стандарту SQL92 (SQL2);
- SQLite написано в ANSI-C та забезпечує простий і легкий у використанні API;
- SQLite доступна на UNIX (Linux, Mac OS X, Android, iOS) і Windows (Win32, WinCE, WinRT).

3.5 Історія SQLite

Автором SQLite є Д. Річард Хіп (D. Richard Hipp). Він працював у GeneralDynamics для програми флоту США, розробляв програмне забезпечення для використання на есмінцях. Програма працювала на Hewlett-Packard Unix (HPUX) та використовувала базу даних Informix. Проте, для деяких застосунків Informix був дуже громіздким і потребував багато ресурсів. Також підтримка системи в актуальному стані потребувала складної роботи адміністраторів баз даних, водночас програмісти потребували легкої БД, яка була доступна всюди, та могла легко переноситись разом з програмою. В січні 2000, Хіп з колегами обговорили ідею створення простої, вбудованої SQL бази даних, яка буде використовувати GNUDBM-B-Treelibrary (gdbm), як бек-енд, що дозволить використовувати її без необхідності встановлення та адміністративної підтримки. Перша версія була готова у серпні 2000-го, це була SQLite 1.0.

Як і планувалось, SQLite 1.0 використовувала gdbm, як систему зберігання (storagemanager). Але згодом, Хіп замінив gdbm власною реалізацією

B-дерев, яка підтримувала транзакції та зберігала записи за впорядкуванням ключів. З того часу SQLite почала свою поступову еволюцію, зростаючи, як за кількістю нововведень так і за кількістю користувачів. В середині 2001-го багато відкритих та комерційних проєктів починають використовувати SQLite. Це призвело до того, що члени відкритої спільноти розробників починають писати різні розширення та бібліотеки для даної СКБД. Основні нововведення в SQLite 3: використання UTF-8 та UTF-16, покращений CAPI, явне оголошення типів, зменшення розміру баз даних, BLOB, 64-bit ROWIDs, autovacuum, покращену систему паралельного доступу.

4 ЕЛЕКТРОННІ ТАБЛИЦІ

4.1 Основні можливості електронних таблиць

Таблиці є одним із найпоширеніших різновидів документів, які використовуються у фінансово-економічній діяльності, в тому числі, у банківській справі, у менеджменті, у маркетингу, в бухгалтерських розрахунках тощо. Саме тому розроблене спеціальне програмне забезпечення, яке дозволяє автоматизувати роботу з таблицями.

Прикладні програми, які призначені для обробки даних, представлених у табличній формі, називаються електронними таблицями (ЕТ) або табличними процесорами.

Популярність та ефективність табличного процесора MS Excel визначається його надзвичайно широкими можливостями, основними з яких вважаються:

- створення таблиць даних, між комірками яких можуть бути встановлені зв'язки за допомогою формул. При цьому одні дані вважаються початковими і вводяться до таблиці із зовні (наприклад, з клавіатури), а інші дані вважаються розрахунковими і утворюються в таблиці автоматично за допомогою формул;
- подання даних у наочній графічній формі за допомогою діаграм, в тому числі за допомогою графіків та гістограм різної форми;
- проведення розрахунків за допомогою простих формул, що вводяться вручну, з використанням функцій із спеціальних математичних, економічних, планових, статистичних та інших бібліотек, а також шляхом використання мови програмування VBA (Visual Basic for Application);
- розв'язування задач, які мають оптимізаційний характер, тобто дозволяють знаходити найкращі планові показники, які забезпечують найвищий ефект управлінської діяльності;

- створення баз даних, тобто наборів даних великого обсягу, в яких існують засоби автоматизації пошуку, впорядкування та фільтрації;
- обмін даними з іншими складовими пакету Microsoft Office, а також іншими прикладними програмами персонального комп'ютера за допомогою системного буфера обміну;
- створення макросів, тобто спеціальних команд, за допомогою яких вдається автоматично виконувати послідовності однотипних операцій.

4.2 Основні поняття в електронних таблицях

Робоча книга являє собою документ Microsoft Excel. Має вигляд дискового файлу, який містить дані, представлені у табличній формі. Такий файл має ім'я, обране для нього користувачем, а також стандартне розширення XLS. Наприклад, «Відомості на заробітну плату за січень 2004 року.XLS». Файли, яким користувач не надав імені, набувають стандартних імен «Книга1.XLS», «Книга2.XLS» і т.д. Обсяг робочої книги обмежується доступною пам'яттю персонального комп'ютера.

Робочий лист. За внутрішньою побудовою робоча книга являє собою набір робочих листів. У вікні документа є можливість бачити і працювати лише з одним робочим листом, який в цей момент називається поточним.

Кожний робочий лист має найменування, зазначене на його ярличку. Найменування робочому листу обирає користувач у відповідності з табличними даними, які на ньому розміщені, наприклад «Відомість зарплати цеху комплектації», «Відомість зарплати відділу кадрів» тощо. Робочі листи, для яких користувач ім'я не обирає, набувають стандартних імен «Лист1», «Лист2» і т.д. За допомогою ярличків можна переходити від одного листа до другого, можна виривати або вставляти окремі листи тощо. Кількість робочих листів у книзі обмежується обсягом робочої книги.

Рядок, стовпчик та комірка. У вікні документа кожний робочий листок має вигляд таблиці, яка нагадує сторінку зошита в клітинку. Робочий лист розграфлений горизонтальними та вертикальними лініями на рядки та стовпчики відповідно. Стовпчики позначають прописними літерами латинського алфавіту, рядки нумерують числами. Всього може бути 256 стовпчиків, позначених від A до IV, а також 65536 рядків, пронумерованих від одиниці. Нумерація рядків продовжується вниз, позначення стовпчиків – вправо.

На перетині рядка та стовпчика знаходиться утворення, яке називається коміркою. Кожна комірка має стандартне ім'я (позначення), яке складається із позначень відповідних стовпчика та рядка, наприклад, комірка F3 знаходиться на перетині стовпчика F і рядка 3. Зверніть увагу, що спочатку указується стовпчик, а потім – рядок. Отже, стандартне ім'я комірки являє собою її адресу і дозволяє відшукати її на робочому листі. Зокрема, початкова комірка A1 знаходиться у лівому верхньому кутку таблиці.

Альтернативним варіантом стандартного іменування комірок є використання номерів як для рядків, так і для стовпчиків. Така система називається системою посилок «R1C1», що має походження від Row – рядок і Column – стовпчик. Зверніть увагу, що в цій системі, спочатку указується рядок, а потім стовпчик. Наприклад, комірка F3 в цій системі буде позначена як R3C6.

При необхідності ім'я комірки може включати найменування листа, на якому вона знаходиться, наприклад, Лист2!F3.

Адреси комірок використовуються як змінні у складі формул.

Діапазон комірок. Прямокутна група сусідніх комірок називається діапазоном комірок. Роль діапазону полягає в тому, що він містить сукупність (масив) даних і розглядається як єдине ціле. Діапазон комірок указують через адреси двох комірок, які знаходяться на протилежних кінцях якоїсь із діагоналей прямокутника. Указані адреси відокремлюються двокрапкою. Наприклад, групу комірок, яка знаходиться на перетині рядків 3 і 4 та

стовпчиків В і С, можна позначити як діапазони В3:С4, С4:В3, С3:В4 або В4:С3.

Імена комірок. Імена комірок використовують як позначення змінних у складі формул. Імена комірок можуть бути стандартними і нестандартними. Стандартні імена являють собою адреси комірок, побудовані шляхом поєднання позначень стовпчика та рядка. Для будь-якої комірки можна обрати також довільне нестандартне ім'я.

На нестандартні імена накладаються наступні обмеження:

- а) вони не можуть містити пропусків;
- б) вони не повинні співпадати з адресою будь-якої іншої комірки;
- в) вони можуть бути тільки абсолютними;
- г) великі та малі літери в іменах не розрізняються в будь-якому алфавіті.

Зручність нестандартних імен пов'язана з їх звичністю, а також з можливістю надання їм смислового звучання. Наприклад: Х, Y, Результат, Підсумок тощо. А це значно полегшує побудову та сприйняття формул.

Для створення нестандартного імені треба віділити потрібну комірку, а потім ввести вибране ім'я у поле Имя рядка формул.

Нестандартні імена мають наступні властивості:

- якщо віділити комірку з нестандартним іменем, то саме воно буде тепер відображатись у полі Имя рядка формул;
- поле «Имя рядка формул» фактично являє собою розгортуваний список вже існуючих нестандартних імен. Оскільки нестандартні імена на робочому листі себе ніяк не проявляють, то цей розгортуваний список може використовуватись для виділення комірок з нестандартними іменами;
- нестандартні імена можуть бути присвоєні не тільки коміткам, але й діапазонам;
- надані нестандартні імена є глобальними, тобто вони дійсні на будь-яких листах книги. Але їх можна створити і локальними, тобто дійсними в межах тільки свого листа. Для цього до складу імені треба включати

також і ім'я листа. Наприклад: «Лист3!х.» При цьому нестандартні імена листів беруться в одинарні лапки. Наприклад: «‘Результати’!х.»;

– одній і тій же комірці або діапазону можна надати довільну кількість імен.

Щоб вилучити існуюче нестандартне ім'я, треба виконати команду Вставка–Имя–Присвоить.... Внаслідок цього на екрані з'являється діалогове вікно Присвоение имени. В цьому вікні треба виділити потрібне ім'я і натиснути кнопку Удалить.

Типи даних. Окремі комірки можуть містити дані, які відносяться до одного із трьох типів: текст, число або формула. Комірка може також залишатись порожньою. Тип даних під час введення програмою Excel розрізняється автоматично, а саме: якщо це можливо, то дані в першу чергу розглядаються як число, в протилежному випадку дані розглядаються як текст.

Дані, які є формулою, розрізняються за знаком «=», з якого вони завжди починаються. Тип даних «формула» має особливість, яка полягає в тому, що у відповідній комірці відображується не сама формула, а результат обчислення за нею.

Відносні та абсолютні адреси. Якщо не вказано щось інше, то адреса комірки у складі формули розглядається як відносна. Це означає, що при копіюванні формули з однієї комірки в другу адреси, указані в ній, автоматично змінюються на величину відстані між оригіналом та копією. Наприклад, формула, розміщена у комірці C1, містить адресу комірки B3. Якщо скопіювати цю формулу у комірку E4, то наявна в ній адреса B3 автоматично змінюється адресою комірки D6. Дійсно, сама формула була зміщена на два стовпчики вправо і на три рядки вниз. У відповідності з цим змінилась і адреса комірки, яка містилась в ній.

Абсолютна адреса характерна тим, що вона при копіюванні не змінюється. Для вказівки абсолютної адреси застосовується символ «\$». У складі адреси комірки абсолютними можуть бути номери рядка і стовпчика як окремо, так і разом. Наприклад, відносні та абсолютні адреси комірки A1 мають

вигляд A1 і \$A\$1 відповідно. А в адресах A\$1 і \$A1 абсолютним є лише один із компонентів.

Формати числових даних. Одне й те ж саме числове значення може бути представлене у комірці в різних форматах. Основні з них такі: формат «Числовой» подає десяткове число у звичайній формі (ціла частина, десяткова крапка, дробова частина), формат «Денежный» надає можливість указувати поряд з числом позначення грошової одиниці, формати «Дата» та «Время» надають можливість автоматично перераховувати кількість днів від початку 1900 року у відповідні дату та час, формат «Процентный» множить число на 100 і відображує його зі знаком “%”, формат «Экспоненциальный» подає число у показниковій формі, тобто у вигляді сукупності мантиси та десяткового порядку, відокремлених одне від другого літерою E.

5 ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Мова програмування Python підтримує як процедурно-орієнтоване, так і об'єктно-орієнтоване програмування. У процедурно-орієнтованих мовах програми будуються на основі процедур або функцій, які представляють собою багаторазово використовувані фрагменти програми. В об'єктно-орієнтованих мовах програмування програми будуються на основі об'єктів, які об'єднують в собі дані і функціонал. Python надає прості але потужні засоби для ООП, особливо в порівнянні з такими великими мовами програмування, як C ++ або Java.

Вивчення будь-якої методології передбачає вивчення як фундаментальних, так і прикладних її аспектів. У програмуванні використовуються методології і технології, що спираються як на високоннаукові ідеї, так і на розвинені прикладні системи. Для інженера важливо знати, насамперед, технологію програмування. Іншими словами, йому необхідно навчитися тим методам програмування, які він може застосовувати на практиці.

Методологія ООП є реалізацією змін у довгому ланцюзі рішень, які пропонувалися для подолання принципових труднощів, що виникають перед програмістами при ускладненні розроблювального ними програмного забезпечення ЕОМ. Термін "криза програмного забезпечення", який виник у середині 60-х років, означав, що потреби розробників програмного забезпечення перевершили можливості існуючих технологій.

ООП – це еволюційний етап у розвитку методології програмування, що спирається на ті ідеї структурного і модульного програмування, які виявилися плідними і перспективними для розвитку.

Об'єктно-орієнтоване програмування — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою.

Основу ООП складають три основні концепції: інкапсуляція, успадкування та поліморфізм. Одною з переваг ООП є

краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в 1960–тих роках, вона не мала широкого застосування до 1990–тих, коли розвиток комп'ютерів та комп'ютерних мереж дозволив писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм. Сьогодні багато мов програмування або підтримують ООП (Lua) або ж є цілком об'єктно-орієнтованими (зокрема, Java, C#, C++, Python, PHP, Ruby та Objective-C, ActionScript 3, Swift, Vala).

Інкапсуляція – механізм, який поєднує дані та методи, що обробляють ці дані і захищає і те і інше від зовнішнього впливу або не вірного використання. Коли і методи і данні об'єднуються таким чином — створюється об'єкт.

В середині об'єкта дані і методи можуть мати різні ступені відкритості. Як правило відкриті члени класу використ. Для того, що б забезпечити інтерфейс, що контролюється з його закритою частиною.

Наслідування – процес, завдяки якому один об'єкт може придбати властивості іншого, тобто наслідувати властивість іншого об'єкту і додавати риси характерні тільки для нього самого.

В якості прикладів можна навести класифікацію будь-яких об'єктів, яка дає відповідь на питання в чому схожість об'єкта даного класу з інш. Об'єктом і в чому різниця. Наслідування забезпечує загальність функції в той час припускаючи стільки особливостей, скільки необхідно. Таким чином наслідування виконує в ООП такі функції:

- моделює концептуальну структуру предметної області;
- економить опис, дозволяючи використовувати функції багатократно для завдань різних класів;
- забезпечує покрокове програмування великих систем, шляхом багатократної конкретизації класів.

Поліморфізм – властивість, яка дозволяє одне і те саме ім'я використовувати для вирішення декількох технічно різних задач, тобто

основною метою поліморфізму є використання одного імені для задання загальних класу дій. На практиці це значить спроможність об'єктів вибирати внутрішній метод або процедуру, виходячи із типу даних, прийнятих в повідомленні.

Приклад, об'єкт з ім'ям “print” отримує повідомлення в якому виконуються дії по друку таких типів даних, як двійкове ціле число, двійкове число з плаваючою комою і символьний рядок. В залежності від того типу даних, які працюють в команді буде підібраний тип (даних) формату для друку.

Перевагою поліморфізму є те, що:

- він допомагає зменшити складність програм;
- дозволяє використання одного інтерфейсу для єдиного класу дій, а вибір конкретної дії передається компілятору.

Таким чином, поліморфізм означає присвоювання певній дії одної дії або позначення, яке спільно використовується об'єктами різних типів. При цьому кожний об'єкт реалізує дії способом, що відповідає його типу.

Об'єктно-орієнтоване програмування – це метод програмування, заснований на поданні програми у вигляді сукупності взаємодіючих об'єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування. Програмісти спочатку пишуть клас, а на його основі при виконанні програми створюються конкретні об'єкти (екземпляри класів). На основі класів можна створювати нові, які розширюють базовий клас і таким чином створюється ієрархія класів.

Методи класу — це програми, подані в описі класу. Методу необхідно повідомити, дані якого об'єкта потрібно буде опрацьовувати. Для цього методу в якості першого (а іноді єдиного) аргументу передають назву змінної, пов'язаної з об'єктом (можна казати, передають сам об'єкт). Щоб в описі класу вказати використання своїх властивостей чи методів, використовують службове слово «self».

6 ПРАКТИЧНА ЧАСТИНА

Проектування прикладної програмної системи починається з аналізу вимог, яким вона повинна задовольняти. Мета аналізу – дати настільки повний опис функціонування проектованої системи, щоб на його підставі можна було скласти попередній проект. В процесі аналізу необхідно визначити призначення й умови експлуатації системи.

Процес реалізації системи полягає в побудові фізичної моделі і її реалізації у виді комп'ютерної програми. Незважаючи на те, що реалізація – один із заключних етапів життєвого циклу системи, він звичайно починається на перших етапах проектування у виді експериментів, спрямованих на перевірку правильності та ефективності прийнятих рішень.

Завданням данної роботи було створення аналітично–функціонального модуля Python з взаємодією з електронними таблицями та СУБД SQLite. У процесі розробки було прийнято рішення створити модуль, який міг би імпортувати дані з електронних таблиць Excel в бази даних, та навпаки експортувати дані з БД до файлів типу Excell. Для роботи з кодом програми використовувалося інтегроване середовище розробки для мови програмування Python – PyCharm.

У практичній частині представлені три коди у вигляді скриптових файлів мови програмування Python.

Для роботи з таблицями використовувалися наступні модулі мови програмування Python:

- xlrd – модуль завантаження даних з файлів Excel;
- xlwt – модуль для генерації файлів Excel;
- sqlite3 – модуль для роботи з SQLite3;
- time – модуль для отримання часу за скільки відбувається конвертація;
- faker – модуль для генерації випадкових імен, прізвищ, телефонів, адрес, тощо.

Файл «start.py» (Додаток А).

Даний скрипт формує файл в електронній таблиці Excel по заданим параметрам. А саме, у коді задається назва документа, який створюється, назва листа, задаються назви стовбців та кількість рядків у файлі.

Комірки яким не присвоєні значення заповнюються рандомними числами. Після чого сгенерований запит виводиться у терміналі завдяки функції print (Рис. 6.1).

```

Run: start x
файл test.xls збережено
лист 'sheet': записано 30 комірок
час роботи 0.156 сек.
розмір файлу: 5 КБ
1|  january  |  february  |  march  |  april  |  may  |  june  |
2|  44.42    |  796.37    |  723.3   |  85.13  |  354.86 |  256.97 |
3|  821.96   |  831.5     |  673.37  |  36.13  |  42.01  |  256.72 |
4|  207.98   |  981.19    |  663.72  |  222.42 |  933.04 |  579.96 |
5|  226.91   |  487.4     |  236.8   |  840.44 |  83.11  |  311.4  |
INSERT INTO sheet (january, february, march, april, may, june) VALUES
('44.42', '796.37', '723.3', '85.13', '354.86', '256.97'),
('821.96', '831.5', '673.37', '36.13', '42.01', '256.72'),
('207.98', '981.19', '663.72', '222.42', '933.04', '579.96'),
('226.91', '487.4', '236.8', '840.44', '83.11', '311.4');
table sheet already exists
RESULTS:
rows: 5
time: 0.281 sec.

```

Рисунок 6.1 – Результати виконаного запиту до файлу «start.py»

У терміналі ми можемо побачити аналіз опрацювання запиту. Сгенерована таблиця була збережена у форматі «.xls», розмір створеного файла дорівнює 5 КБ, для обробки запиту програмі знадобилося 0,281 с.

Створений файл у вигляді електронної таблиці зображений на рисунку 6.2. Як ми бачимо у таблиці відображені такі ж числа як і у запиті, отже код працює правильно.

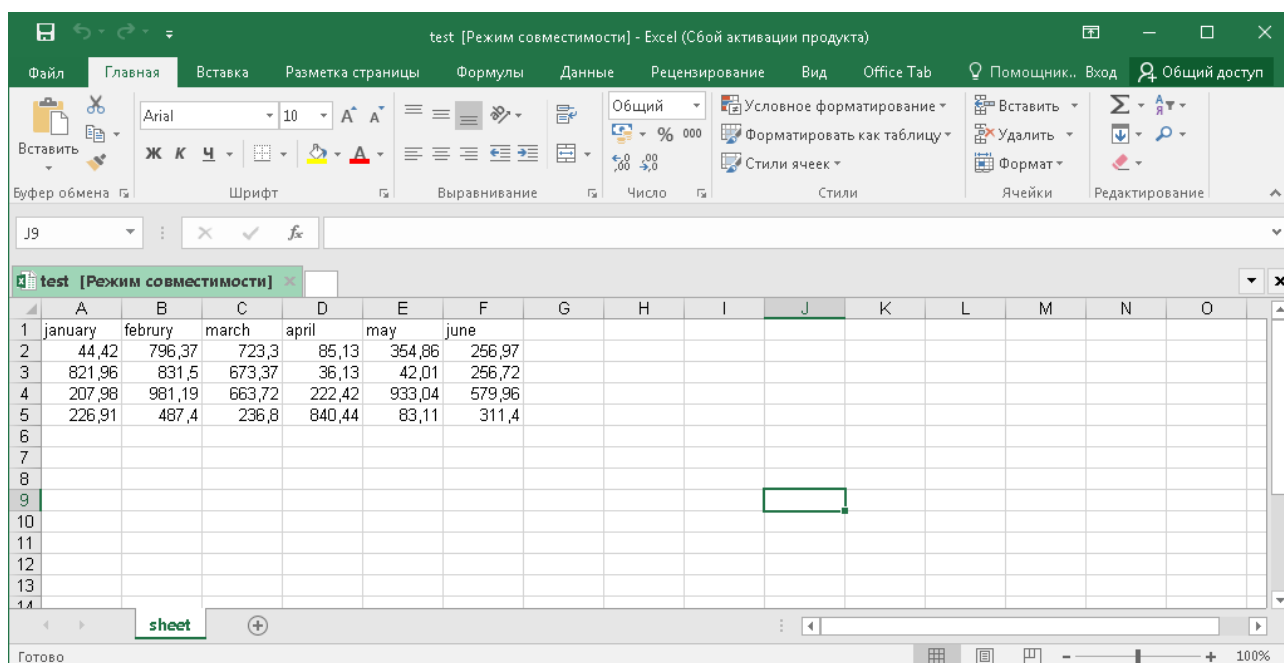


Рисунок 6.2 – Створенна та збережена електронна таблиця

Файл «excel_tools.py» (Додаток Б)

Файл «excel_tools.py» є головним модулем програми, який виконує завантаження, обробку, конвертацію табличних файлів (xls,xlsx) до формату бази даних SQLite.

Даний скрипт створений як клас, що має певний функціонал і пов'язаний з файлом «write_data_M» (Додаток В).

У головному скрипті прописані два модулі, які дають змогу обирати які дані будуть формуватися у кінцевій таблиці, числа або текстова інформація.

Для заповнення таблиці текстовими даними використовувався модуль faker, який при заданих умовах може генерувати випадкові імена, прізвища, професії, тощо. Мову виведеної інформації також можна обирати змінюючи код програми, наприклад:

- ar_SA – Arabic (Saudi Arabia)
- bg_BG – Bulgarian
- cs_CZ – Czech
- de_DE – German

- dk_DK – Danish
- en_AU – English (Australia)
- en_CA – English (Canada)
- en_GB – English (Great Britain)
- es_ES – Spanish (Spain)
- fr_FR – French
- it_IT – Italian
- ja_JP – Japanese
- nl_NL – Dutch (Netherlands)
- no_NO – Norwegian
- pl_PL – Polish
- ru_RU – Russian
- tr_TR – Turkish
- uk_UA – Ukrainian
- zh_CN – Chinese (China)

Приклади функціонування модулю за різних умов наведено на рис. 6.3–6.5.

	A	B	C	D	E
1	столбец 01	столбец 02	столбец 03	столбец 04	столбец 05
2	Jennifer Woods	Jared Daniels	Nicole Estrada	Jeff Rodriguez	Jeanette Haas
3	Lisa Yang	Jeffrey Terry	Kathleen Ferrell	Andrea Bowman	Brent Sharp
4	Nathan Orr	Gabrielle Ewing	Michael Glass	Stephen Jones	Kelsey Evans
5	Ryan Jones	Melanie Kane	Kim Bailey	Shannon Chavez	Tasha Bell
6	Elizabeth Cohen	Brandon Harrell	Julie Ayers	Amanda Jacobs	David Velasquez
7	Sarah Vargas	Laura Carpenter	Patrick Palmer	Olivia Wong	Brandon Watts
8	Suzanne Hall	Matthew Paul	John Branch	Marcus Morgan	Shane West
9	Nicholas Lawrence	Alexandria Allen	Christina Miller	Brandon Schmitt	Frederick Sanchez
10	Alex Morgan	Elizabeth Thomas	Steven Rosales	Teresa Baker	Nicholas Hart
11	Angel Johnson	Melissa Adams	Gerald Duke	Dr. Tonya Mccullough	Donna Wallace
12	Elizabeth Gutierrez	Cory Russell	Erica Collins	Henry Jackson	Rebecca Molina
13	Kevin Romero	Taylor Wilson	Judy Hickman	Brittany Smith	Devin Frazier
14	John Fisher	Timothy Tucker	Kristi Acosta	Victoria Thompson	Brad Wagner
15	Joseph Rivers	Cynthia Bates	Samantha Barnes	Andrew Atkins	Christopher Hanson Jr.

Рисунок 6.3 – Ім'я та прізвища на англійській мові

	A	B	C	D	E
1	№ 01	№ 02	№ 03	№ 04	№ 05
2	Слідчий	Паблік рилейшнз	Бізнесмен	Модель	Фермер
3	Водій	Детектив	Бухгалтер	Дипломат	Професор
4	Співак	Лікар	Хімік	Провізор	Фрілансер
5	Бібліограф	Соціолог	Дантист	Етнограф	Епідеміолог
6	Археолог	Електрик	Агроном	Акушер	Візажист
7	Літературознавець	Драматург	Астроном	Скульптор	Ґрунтознавець
8	Промисловець	Ботанік	Доцент	Головний меркшейдер	Драматург
9	Дерун	Фермер	Композитор	Редактор	Фізик
10	Візажист	Перекладач	Астрофізик	Водій	Державний службовець
11	Кінорежисер	Педагог	Фермер	Гицель	Співак
12	Лікар	Хореограф	Журналіст	Етнограф	Провізор
13	Паблік рилейшнз	Палеонтолог	Режисер	Геолог	Клоун
14	Кіноактор	Мистецтвознавець	Гірник	М'ясник	Художник
15	Прокурор	Дантист	Дерун	Письменник	Вчитель

Рисунок 6.4 – Назви провесій на українській мові

	A	B	C	D	E
1	№ 01	№ 02	№ 03	№ 04	№ 05
2	+38 036 031-76-43	022 311-05-40	+38 062 315-15-87	228 50 89	+38 036 801-11-84
3	245 27 22	+38 085 301-36-56	076 259-39-61	087 109 04 57	976-73-58
4	080 379-37-03	039 209-05-25	+38 066 097-23-30	222-01-87	458 51 14
5	079 187-22-38	019 158-10-89	+38 013 745 26 07	+38 016 439-44-78	+38 048 984-51-32
6	+38 063 880-79-98	+38 091 293-10-55	+38 014 647-39-09	022 987 55 67	+38 054 424-21-39
7	+38 (089) 291-13-54	+38 012 527 36 39	+38 020 017-49-16	+38 041 107-92-73	087 346 07 99
8	084 314 07 80	+38 030 888-44-11	022 143 97 49	067 574 76 06	045 646 67 75
9	196 10 78	680 91 05	+38 085 893-06-09	529-59-93	+38 076 606-88-82
10	011 456-42-14	+38 045 128-17-67	+38 077 633-53-07	+38 054 260-64-85	000 470 31 52
11	523-52-35	+38 006 462 26 15	+38 078 556 09 75	+38 (013) 699-72-88	+38 034 756-75-91
12	978 56 20	821-48-01	017 692 81 81	+38 057 276-57-85	+38 011 614 23 04
13	143-76-93	+38 (093) 262-45-73	+38 098 297-57-46	017 379-77-93	+38 (042) 354-63-78
14	600-82-39	435 60 30	+38 083 886-58-07	041 506-01-16	000 704-05-42
15	+38 012 665-64-46	906 40 77	011 043 30 20	060 953 00 28	+38 036 184-73-69

Рисунок 6.5 – Номера телефонів

Файл «write_data_M.py» (Додаток В)

Скрипт генерує задану функцію у електронну таблицю Excell. У коді можна змінювати кількість стовбців та рядків. По рядкам дані заповнюються завдяки скрипту et.write_rnd_data_to_xls або et.write_faker_data_to_xls.

В залежності від кількості заповнених комірок змінюється час опрацювання коду. Наприклад, для заповнення 100 комірок час на опрацювання коду дорівнює 2,127с, а для 100 000 комірок – 7.188с (Рис. 6.6, Рис. 6.7).

```
файл try.xls збережено  
лист 'sheet': записано 100 комірок  
час роботи 2.127 сек.  
розмір файлу: 5 КБ  
  
Process finished with exit code 0
```

Рисунок 6.6 – Результати заповнення та опрацювання 100 комірок електронної таблиці Excell

```
файл try.xls збережено  
лист 'sheet': записано 100000 комірок  
час роботи 7.188 сек.  
розмір файлу: 626 КБ  
  
Process finished with exit code 0
```

Рисунок 6.7 – Результати заповнення та опрацювання 100 комірок електронної таблиці Excell

Усі коди працюють без помилок. Після опрацювання кожного запиту у терміналі виводиться:

- назву створеного або збереженого файлу;
- кількість записаних комірок;
- час роботи програми;
- розмір створених файлів

ВИСНОВКИ

Широке застосування і використання інформаційних технологій є глобальною тенденцією світового розвитку та науково–технічної революції останніх десятиліть. На сьогоднішній день сучасні організації, що надаються будь-які послуги, вже неможливо уявити без застосування інформаційних технологій. Автоматизація процесу управління і контролю стала закономірним етапом розвитку управлінських структур. Створення та впровадження інформаційних технологій, а саме баз даних орієнтоване, перш за все, на автоматизацію процесу обробки інформації, отже, на підвищення ефективності роботи співробітників організації.

Python – це універсальна мова, що широко використовується в усьому світі для самих різних цілей – бази даних і оброблення текстів, вбудовування інтерпретатора в ігри, програмування GUI, швидке створення прототипів (RAD) програмування Internet і Web додатків – серверних (CGI), клієнтських (роботи), Webсерверів і серверів додатків. Серед переваг мови Python можна виділити переносимість написаних програм, на комп'ютери різної архітектури та з різними операційними системами, лаконічність запису алгоритмів, можливість отримати ефективний код програм за швидкістю виконання. Зручність мови Python ґрунтується на тому, що вона є мовою високого рівня, має набір конструкцій структурного програмування та підтримує модульність. Гнучкість та універсальність мови Python забезпечує її широке розповсюдження.

Сучасні технології обробки інформації часто приводять до того, що виникає потреба подання даних у вигляді таблиць. Електронні таблиці (табличні процесори) – це прикладне програмне забезпечення спільного призначення, яке використовується для обробки даних, представлених у табличній формі. Електронна таблиця (ЕТ) дозволяє зберігати в табличній

формі велику кількість вхідних даних, результатів, а також зв'язків (алгебраїчних і логічних співвідношень) між ними.

База даних — це певний набір даних, які пов'язані між собою спільною ознакою або властивістю, та впорядковані, наприклад, за алфавітом.. Об'єднання великої кількості даних в єдину базу дає змогу для формування безлічі варіації групування інформації — особисті дані клієнта, історія замовлень, каталог товарів та будь-що інше.

Головною перевагою БД є швидкість внесення та використання потрібної інформації. Завдяки спеціальним алгоритмам, які використовуються для баз даних, можна легко знаходити необхідні дані всього за декілька секунд. Також в базі даних існує певний взаємозв'язок інформації: зміна в одному рядку може спричинити зміни в інших рядках — це допомагає працювати з інформацією простіше і швидше.

Метою даної роботи було створення модуля Python, що містить набір функцій для аналізу даних, що зберігаються в електронних таблицях, наприклад Microsoft Excel або Google Sheets, передачі цих даних в бази даних SQLite, що передбачає автоматичне формування і виконання запитів на мові програмування SQL, контроль за ходом виконання і збір статистичної інформації, такої як обсяг сформованих і переданих даних, час виконання, розмір отриманих файлів і т.д.

Також серед функціоналу модуля є можливості формування запитів на вибірку даних з баз SQLite, створення і видалення таблиць і даних з них. Передбачена можливість генерації електронних таблиць будь-якого розміру з випадковими числами для аналізу можливостей масштабування баз даних SQLite

ПЕРЕЛІК ПОСИЛАНЬ

1. **Бизли Д.** Python. Подробный справочник [Текст]/ Д. Бизли. – СПб. : Символ-Плюс, 2010. – 864 с.
2. **Федоров Д. Ю.** Основы программирования на примере языка Python : учеб.пособие [Текст]/ Д. Ю. Федоров. – СПб. : Питер, 2016. – 176 с.
3. **Лутц М.** Изучаем Python, 4-е издание [Текст]/ М. Лутц. – СПб. : Символ-Плюс, 2011. – 1280 с.
4. **Доусон М.** Програмуємо на Python [Текст]/ М Доусон. – СПб. : Питер, 2014. – 416 с.
5. **Берко А. Ю.** Системи баз даних та знань. Книга 2. Системи управління базами даних та знань : навч. посіб. [Текст] / А. Ю. Берко, О. М. Верес, В. В. Пасічник – Львів : «Магнолія-2006», 2012. – 584 с.
6. **Любанович Б.** Python. Простой Python. Современный стиль программирования [Текст] / Б. Любанович. – СПб. : Питер, 2016. – 480 с.
7. **Фуфаев Э. В.** Базы данных : учебн. пособ. для студ. [Текст] / Э. В. Фуфаев, Д. Э. Фуфаев. – [7-е изд.] – М. : Издательский центр "Академия", 2012. – 320 с.
8. **Львов М.С., Співаковський О.В.** Основы алгоритмізації та програмування. Навч. посібник. [Текст] / М.С. Львов, О.В Співаковський. – Херсон: Айлант. 2000. – 214 с
9. **Зарицька О. Л.** Бази даних та інформаційні системи : метод. посіб. [Текст] / О. Л. Зарицька – Житомир : Вид-во ЖДУ ім. І. Франка, 2009. – 132 с..
10. **Хомоненко А. Д.** Базы данных : учебник [для высших учебных заведений] [Текст] / А. Д. Хомоненко, В. М. Мальцев М. Г. Цыганков,; под ред. проф. А. Д. Хомоненко. – [4-е изд., доп. и перераб.] – СПб. : КОРОНА принт, 2004. – 736 с.
11. **Фісун М. Т., Ніколенко С. Г.** Створення та ведення баз даних засобами мови Jet SQL. Методичні вказівки до виконання блоку лабораторних

робіт з дисципліни «Організація баз даних». [Текст]/ М. Т Фісун, М. Т Ніколенко ч. – Миколаїв. : ЧДУ ім. Петра Могили, 2009. – 83 с

12. **Хомоненко А. Д.** Базы данных : учебник [для высших учебных заведений] [Текст] / А. Д. Хомоненко, В. М. Мальцев М. Г. Цыганков,; под ред. проф. А. Д. Хомоненко. – [4-е изд., доп. и перераб.] – СПб. : КОРОНА принт, 2004. – 736 с.

13. **Бадд Т.** Объектно-ориентированное программирование в действии пер. с англ. [Текст] / Бадд. Т. – СПб.:Питер, 1997.- 464 с. ил.

14. **Фісун М. Т.** Організація баз даних : методичні вказівки до виконання курсових робіт. [Текст]/ Фісун М. Т – Миколаїв : ЧДУ ім. Петра Могили, 2012. – 126 с.

Додаток А

Код програми «start.py»

```
import excel_tools as et
et.write_rnd_data_to_xls('test.xls', 'sheet', [c for c in ["january", "february",
"march", "april", "may", "june"]], 5)
et.show_sheet('test.xls', 'sheet')
print(et.generate_sql_queries('test.xls', 'sheet')[1])
et.transfer_data_excel_to_sqlite('test.xls', 'sheet', 'test.db')
```

Додаток Б

Код програми «excel_tools.py»

```

import xlrd
import xlwt
import sqlite3
import time
import random
import sys
import os
import faker

def get_sheet_data(name_book, name_sheet):
    wb = xlrd.open_workbook(name_book)
    data = []
    for s in wb.sheets():
        if s.name == name_sheet:
            for row in range(s.nrows):
                values = []
                for col in range(s.ncols):
                    values.append(s.cell(row, col).value)
                data.append(values)
            break
    return data

def show_sheet(name_book, name_sheet):
    wb = xlrd.open_workbook(name_book)
    for s in wb.sheets():
        if s.name == name_sheet:
            i = 1
            for row in range(s.nrows):
                values = []
                for col in range(s.ncols):
                    values.append(s.cell(row, col).value)
                print('{}|'.format(i) + '|'.join(['{: ^15}'.format(e) for e in values]) + '|')
                i += 1
            break
    else:
        print('no such sheet_name')

def generate_sql_queries(name_book, name_sheet):
    data = get_sheet_data(name_book, name_sheet)
    col_names = []

```

```

table_name = name_sheet
types = {'str': 'CHAR(30)', 'int': 'INTEGER'}
query_create = 'CREATE TABLE {}'.format(table_name)
for i in range(len(data[0])):
    cell1, cell2 = data[0][i], data[1][i]
    col_names.append(cell1)
    line = '{} {},\n'.format(cell1, get_sql_type(cell2))
    query_create += line
query_create = query_create[:-2] + '\n);'
query_insert = 'INSERT INTO {} ({} VALUES\n'.format(table_name,
    ', '.join(data[0]))
for row in data[1:]:
    line = '({}),\n'.format(', '.join(['{}'.format(e) for e in row]))
    query_insert += line
query_insert = query_insert[:-2] + ';'
query_select_all = 'SELECT * FROM {};\n'.format(table_name)
return (query_create, query_insert, query_select_all, len(data))
def get_sql_type(value):
    if isinstance(value, str): return 'CHAR(30)'
    if isinstance(value, int): return 'INTEGER'
    if isinstance(value, float): return 'NUMERIC'
def tranfer_data_excel_to_sqlite(name_book, name_sheet, name_db):
    start = time.time()
    query = generate_sql_queries(name_book, name_sheet)
    rows = query[3]
    conn = sqlite3.connect(name_db)
    if conn is not None:
        for i in range(2):
            try:
                c = conn.cursor()
                c.execute(query[i])
                conn.commit()
            except Exception as e:
                print(e)
        end = time.time()
        print('RESULTS:\nrows: {} \ntime: {:.3f} sec.'.format(rows, (end-start)))
    else:
        print("Error! cannot create the database connection.")
def write_rnd_data_to_xls(filename, sheetname, cols, rows):
    start = time.time()
    wb = xlwt.Workbook()
    ws = wb.add_sheet(sheetname)
    for i in range(len(cols)):
        ws.write(0, i, cols[i])

```

```

    for j in range(1, rows):
        ws.write(j, i, random.randint(0, 100000) / 100)
    wb.save(filename)
    end = time.time()
    print('файл {} збережено\нлист {}: записано {} комірок'.format(filename,
sheetname, len(cols)*rows))
    print('час роботи {:.3f} сек.'.format(end-start))
    print('розмір файлу: {} КБ'.format(os.path.getsize(filename) // 1024))
def write_faker_data_to_xls(filename, sheetname, cols, rows):
    start = time.time()
    wb = xlwt.Workbook()
    ws = wb.add_sheet(sheetname)
    fake = faker.Faker(['uk-UA'])
    for i in range(len(cols)):
        ws.write(0, i, cols[i])
        for j in range(1, rows):
            ws.write(j, i, fake.phone_number())
    wb.save(filename)
    end = time.time()
    print('файл {} збережено\нлист {}: записано {} комірок'.format(filename,
sheetname, len(cols)*rows))
    print('час роботи {:.3f} сек.'.format(end-start))
    print('розмір файлу: {} КБ'.format(os.path.getsize(filename) // 1024))
if __name__ == '__main__':
    print(generate_sql_quaries('try.xls', 'sheet')[1])

```

Додаток В

Код програми «write_data_M.py»

```
import excel_tools as et
n = 50
cols = ['№ {:0>2}'.format(i) for i in range(1, n+1)]
et.write_faker_data_to_xls('try.xls', 'sheet', cols, 100)
```