Міністерство освіти та науки України
Національний університет водного господарства та природокористування
Кафедра комп'ютерних наук та прикладної математики

04-01-59М

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт
з навчальної дисципліни

«Сучасні та спеціалізовані мови програмування»

для здобувачів вищої освіти першого (бакалаврського) рівня
за освітньо-професійною програмою «Прикладна математика»
спеціальності 113 «Прикладна математика»
денної та заочної форм навчання
Частина 1

Рекомендовано науково-методичною
радою з якості ННІ АКОТ
Протокол № 1 від 11.11.2021 р.

Рівне – 2021

Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Сучасні та спеціалізовані мови програмування» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Прикладна математика» спеціальності 113 «Прикладна математика» денної та заочної форм навчання. Частина 1 [Електронне видання] / Грицюк І. М. – Рівне: НУВГП, 2021. – 44 с.

Укладач:
Грицюк І. М., асистент кафедри комп'ютерних наук та прикладної математики.

Відповідальний за випуск:
Турбал Ю. В., д.т.н., професор, завідувач кафедри комп'ютерних наук та прикладної математики

Керівник групи забезпечення спеціальності 113 «Прикладна математика»: Прищепа О. В., к.ф.-м.н., доцент кафедри комп'ютерних наук та прикладної математики.

Зміст

Лабораторна робота №1

Основи роботи з Python. Встановлення, налаштування середовища розробки

1.1. Мета роботи

Ознайомитися з середовищем програмування Python. Навчитися встановлювати, налаштовувати середовище для комфортної та ефективної роботи. Ознайомитися з інтерфейсом та основними функціями. Створити першу, ознайомчу програму.

1.2. Теоретичні відомості

1.2.1. Загальні відомості

На даний час існує велике розмаїття мов програмування, але найбільш передовою і розвиненою вважається Python.

Мова програмування Python була створена приблизно в 1991 році голландцем Гвідо ван Россумом. Своє ім'я — Python (Пайтон) — отримав від назви телесеріалу ("Monty Python"), а не плазуна.

Python – це високорівнева мова програмування загального значення, що орієнтована на підвищення продуктивності розробника та читання коду. Також *Python* – це інтерпретована мова програмування – вихідний код частинами перетворюється в машинний в процесі його читання спеціальною програмою - інтерпретатором. *Python* – це повноцінна та універсальна мова програмування, що використовується в різних сферах. Основна, але не єдина, підтримувана їм парадигма – об'єктно-орієнтоване програмування.

Python характеризується ясним синтаксисом. Читати даний код легше, ніж на інших мовах програмування. В Python мало використовуються такі допоміжні синтаксичні елементи як: дужки, крапки з комою. З іншого боку, правила мови змушують програмістів робити відступи для позначення вкладених конструкцій. Зрозуміло, що добре оформлений текст з малою кількістю відволікаючих елементів читати і розуміти легше.

З вище сказаного можемо сформулювати *плюси*:

1. Низький поріг входу в порівнянні з іншими мовами;
2. Зрозумілий та легкий для читання навіть новачкам;
3. Короткий (стислий) синтаксис;
4. Має велику кількість бібліотек;
5. Одна із найбільших общин розробників світу;

6. Швидкий у розвитку з великою кількістю середовищ розробки;
7. Затребуваний на ринку.

Що до *мінусів*:

1. Не самий швидкий серед мов програмування. Швидкість виконання програм може бути нижча;
2. Не самий зручний для мобільних додатків.

Сфери використання:

1. WEB-розробка;
2. Штучний інтелект та машинне навчання;
3. Аналіз даних;
4. Автоматизація задач;
5. Комп'ютерні додатки;
6. Ігри та інше.

Не зважаючи на певні мінуси, переваги Python роблять його самою перспективною мовою програмування для вивчення та роботи, а розмаїття сфер використання взагалі «закриває нам очі» на описані мінуси.

1.3. Програма роботи

1. Встановити програмний інтерпретатор Python 3.9.
2. Встановити текстовий редактор Sublime Text.
3. Перевірити працездатність встановленого інтерпретатора.
4. Розробити та виконати пробний запуск тестової програми. Додати коментарі до коду.
5. Оформити звіт.

1.4. Порядок виконання роботи

1. Ознайомитися з теоретичними відомостями та програмою роботи.
2. Встановлення інтерпретатора.

Щоб встановити інтерпретатор перейдіть за адресою офіційного сайту Python (1) - https://www.python.org/downloads/ та натисніть (2) Download Python X.X.X (де X.X.X версія інтерпретатора).

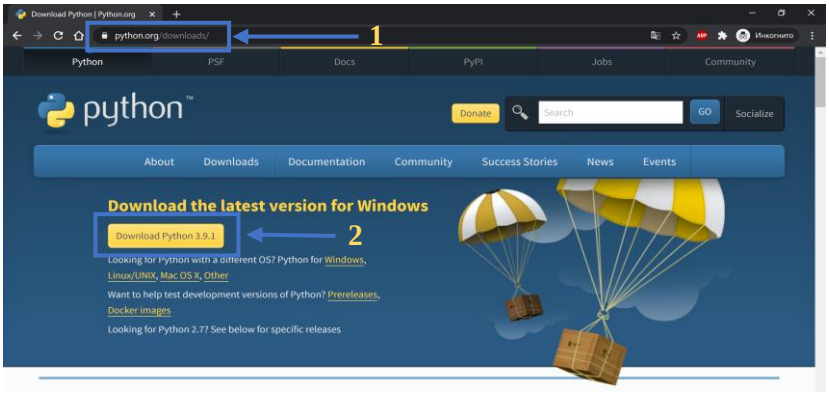


Рис. 1.1. Адреса офіційного сайту Python

3. Після цього натискаємо на завантажений файл. В вікні, що з'явиться, позначаємо поле (3) "Add Python X.X to PATH". Далі натискаємо (4) "Install Now" для встановлення по замовчуванні або ж "Customize installation" для вибіркового встановлення та встановлення за обраним вами шляхом. Очікуємо встановлення та натискаємо (5) "Close".

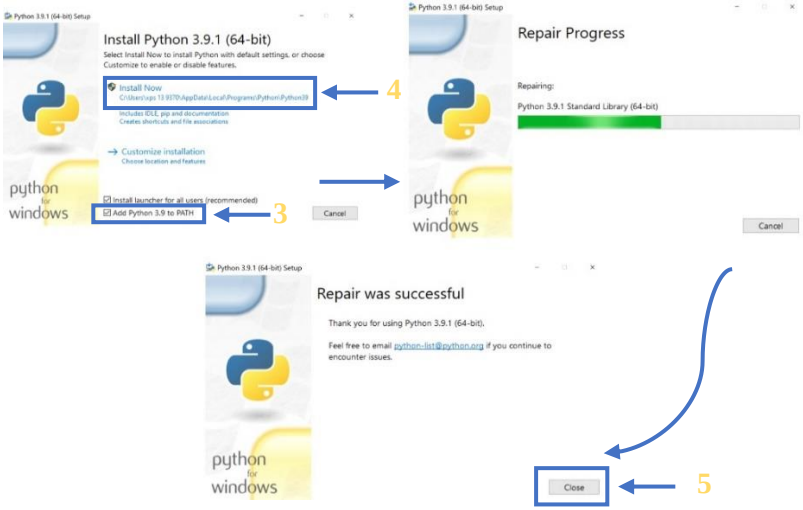


Рис. 1.2. Встановлення програмного продукту

4. Також для зручності написання програм нам знадобиться текстовий редактор.

Для прикладу використаємо Sublime Text. Щоб встановити редактор перейдемо на офіційний сайт (6) https://www.sublimetext.com/ та натиснемо (7) “DOWNLOAD FOR WINDOWS”.

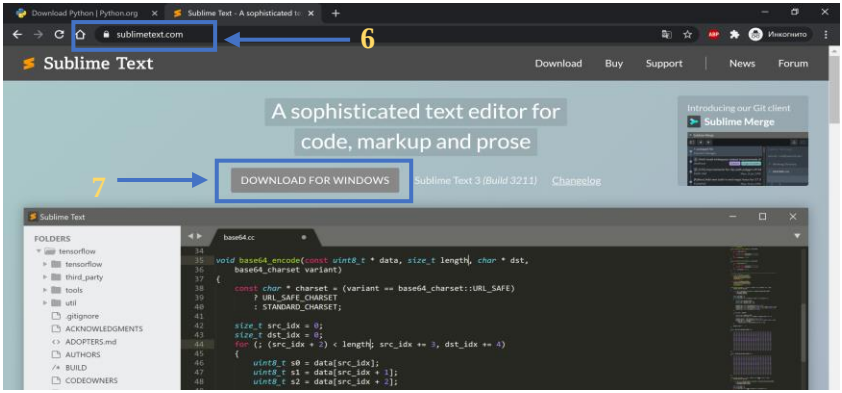


Рис. 1.3. Офіційний сайт Sublime Text

5. Натискаємо на завантажений файл, після чого на пункт Next → Next → Install → Finish.

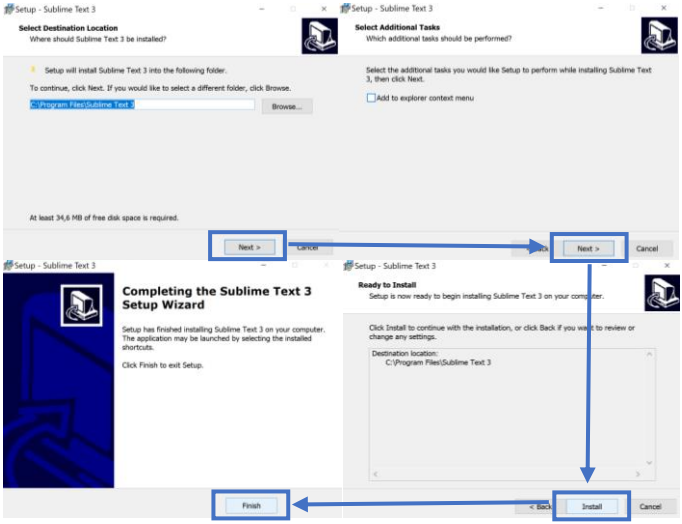


Рис. 1.4. Встановлення програмного продукту

6. Перевіримо працездатність та коректність роботи Python.

Натискаємо комбінацію клавіш WIN+R, щоб відкрити команду «Виконати». В рядку «Відкрити» вписуємо команду (8) “cmd” для відкриття командного рядка та натискаємо (9) «ОК».

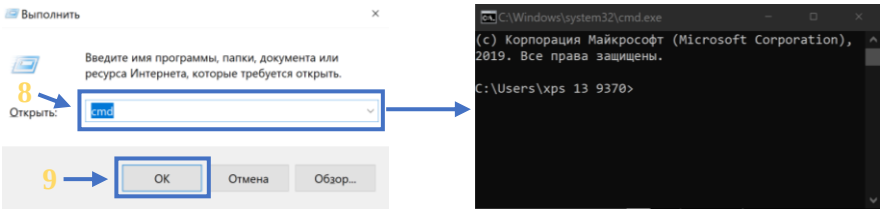


Рис. 1.5. Перевірка працездатності роботи

7. В командному рядку вводимо (10) “python”, на що отримуємо вхід в режим вводу команд та короткі відомості.

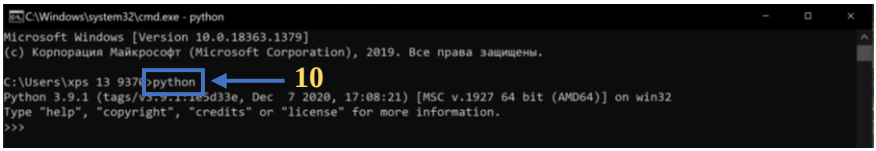


Рис. 1.6. Робота в командному рядку

8. Проводимо елементарні розрахунки.

Сумуємо будь-які цифри. Якщо результати вірні, робимо висновок, що Python працює коректно. Для виходу з режиму прописуємо команду (11) “exit ()”.

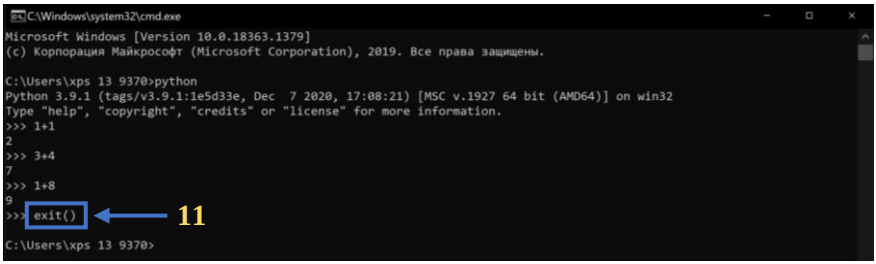


Рис. 1.7. Робота в командному рядку

9. Перейдемо до написання нашої першої програми.

Відкрийте текстовий редактор Sublime Text. Для зручності роботи вмикнемо підказки та прогнозування, натискаємо View → Syntax → Python.

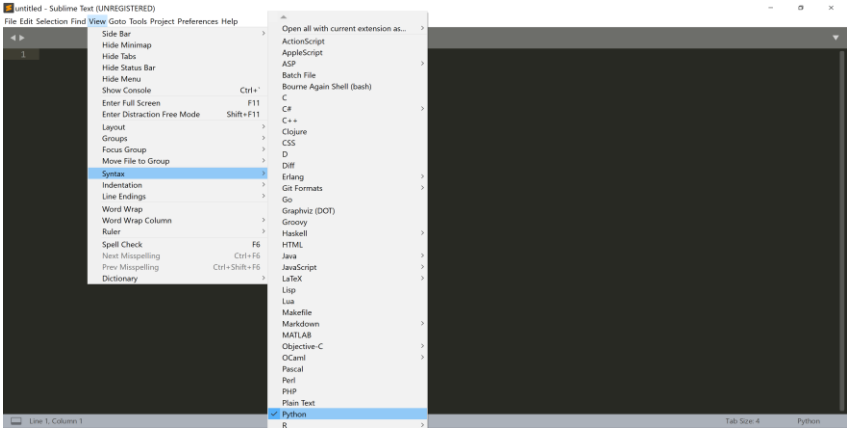


Рис. 1.8. Текстовий редактор Sublime Text

10.Пропишемо команду “print ()”, де в круглих дужка, взявши в лапки, вписується текст для виведення. Додавши потрібний текст натискаємо File → Save, обираємо місце (диск D, папка Python) зберігання файлу та ім’я згідно номеру роботи (Lab_1).

Рис. 1.9. Написання першої програми

11.Перевіряємо роботу програми.

Відкривши командний рядок переходимо до папки з розміщенням файлу (12). Після входимо в режим вводу команд та звертаємося до файлу (13) створеного раніше.

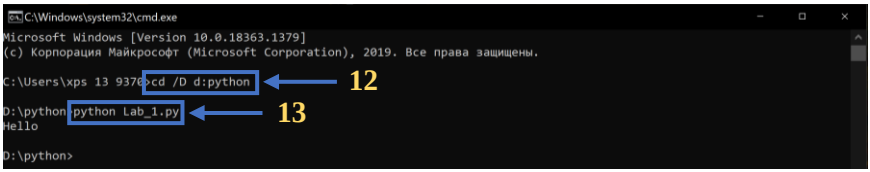


Рис. 1.10. Перевірка роботи програми

12.Якщо була допущена помилка, виведеного результату не буде, а командний рядок вкаже на місце допущення.

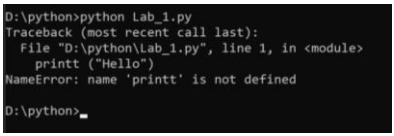


Рис. 1.11. Помилка результатів

13.Також програмування можна виконувати в середовищі розробки IDLE Shell, що встановлюється разом з Python та знаходиться Пуск → Python.

Рис. 1.12. Створення програми за допопомгою IDLE Shell

14.Виведемо на екран «Привіт» та «Мене звати _____» та запустимо програму за допомогою Run → Rub Module або ж за допомогою гарячої клавіші F5.

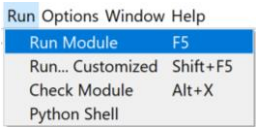


Рис. 1.13. Запуск програми

15.Перед нами відкривається вікно з вивединими результатами.

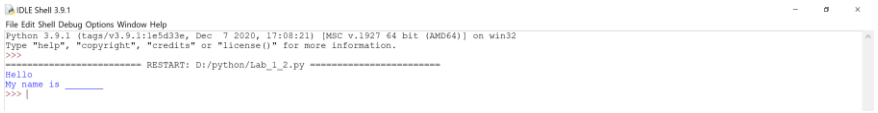
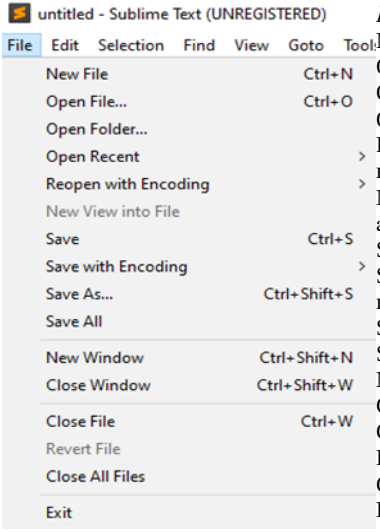


Рис. 1.14. Вивід результату програми

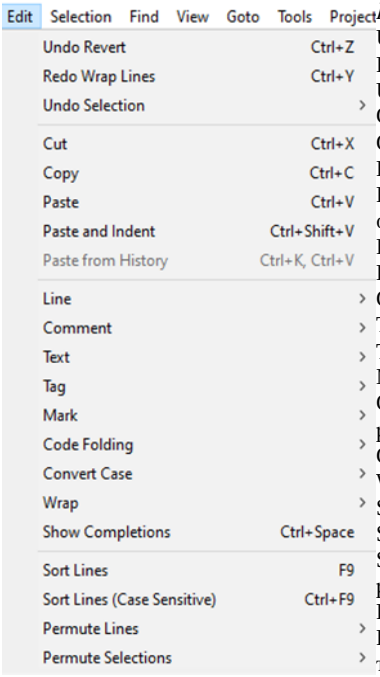
1.5. Інтерфейс та основні функції

1.5.1. Sublime Text 3



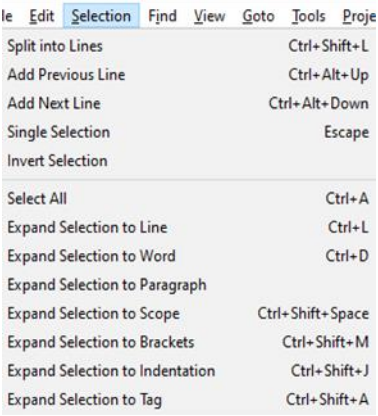
Вкладка File:

New File – створення нового файлу;
Open File – відкриває вже існуючий файл;
Open Folder – відкриває існуючу папку;
Open recent – історія раніше відкритих файлів;
Reopen with Encoding – відкриває файл з обраним кодуванням;
New view into File – відкриває новий екземпляр активного файлу;
Save – зберігає активний файл;
Save with Encoding – зберігає активний файл в обраному кодуванні;
Save as – зберігає файл в іншій папці;
Save All - зберігає всі відриті файли;
New Window – відкриває нове вікно редактора;
Close Window – закриває вікно редактора;
Close File – закриває файл;
Revert File – відміняє не збережені зміни у файлі;
Close All Files – закриває всі відкриті файли;
Exit – закриває текстовий редактор.

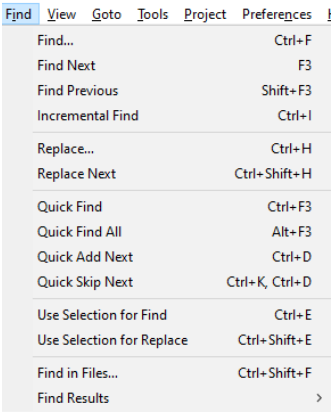


Вкладка Edit:

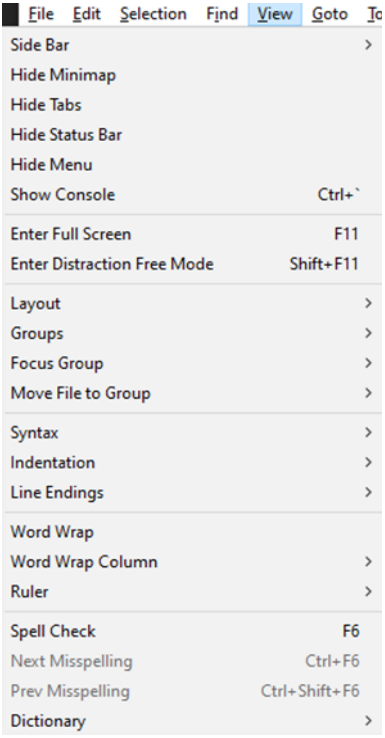
Undo Revert – відміняє останню зміну;
Redo Wrap Lines – повторює останню дію;
Undo Selection – повернутися до обраної дії;
Cut – вирізає виділений текст або рядок;
Copy – копіює виділений текст або рядок;
Paste – вставляє текст з буфера;
Paste and Indent – вставляє текст з буфера із правильним форматуванням;
Paste from History - вставляє текст із історії;
Line – набір команд для переміщення рядків;
Comment – набір команд для коментарів коду;
Text – набір команд для роботи з текстом;
Tag – набір команд для роботи з тегами;
Mark – набір інструментів для позначок в коді;
Code Folding – швидке згортання блоків коду різних рівнів;
Convert Case – зміна регістру символів в рядку;
Wrap – зміна к-сті символів в рядку;
Show Completions – увімкнення підказок коду;
Sort Lines – сортує рядки;
Sort Lines (Case Sensitive) – сортує рядки зважаючи на регістр символів;
Permute Lines – зміна порядку рядків у файлі;
Permute Selections – змін порядку рядків для виділеного тексту.



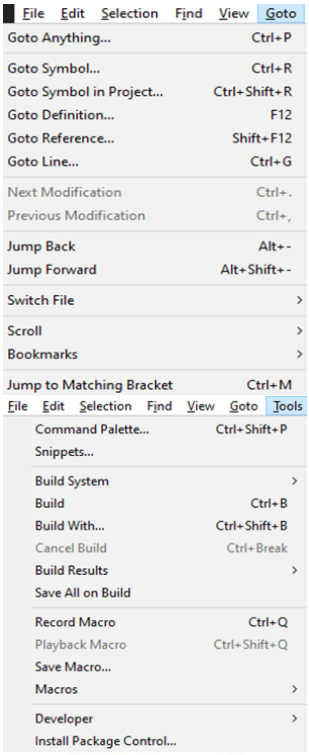
Вкладка Selection
містить набір інструментів та команд для виділення різних частин тексту



Вкладка Find
містить набір інструментів та команд для пошуку по документу

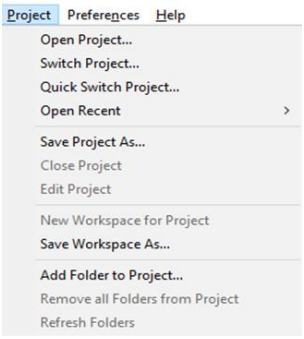


Вкладка View:
Side bar – команди для зміни сайдбару;
Hide Minimap – для вкл/викл відображення мінімапи файлу;
Hide Tabs – вкл/викл відображення вкладки із файлами;
Hide Status Bar – вкл/викл відображення рядка стану;
Hide Menu – вкл/викл відображення меню;
Show Console - вкл/викл відображення консолі;
Enter Full Screen – вкл. повноекранного режиму;
Enter Distraction Free Mode – «відволікаючий» режим;
Layout – розділення робочої області редактора;
Groups – групує робочі файли;
Focus Group – перехід між групами;
Move File to Group – переміщає файл в іншу групу;
Syntax – вибір підсвітки синтаксису файлу;
Indentation – конфігурація відображення символу tab в документі;
Line Endings – конфігурація символу завершення рядка;
Word Wrap – вкл. режиму друку в декілька рядків визначеної довжини;
Word Wrap Column – встановлює максимальну довжину для відображення рядка;
Spell Check – перевірка орфографії файла;
Dictionary – словники перевірки орфографії.

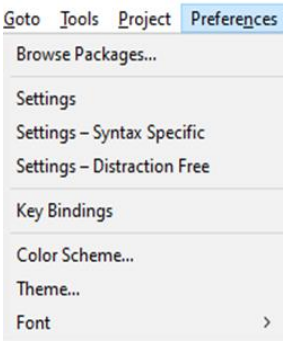


Вкладка Goto:
містить набір інструментів для переходу по вмісту файлу, символах, виразах, рядках та інших елементах.

Вкладка Tools:
Command Palette – відкриття панелі команд;
Build System, Build with … та інші – набір команд та інструментів для конфігурування компіляції та запуску проекту, а також пункти для запуску цих подій.
Record Macro, Save Macro… та інші – інструменти для запису та роботи із макросами для Sublime Text 3
Developer – набір інструментів для розробки тем, плагінів та нових варіантів підсвітки синтаксису у пункті.

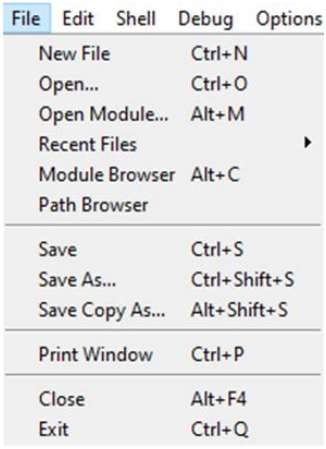


Вкладка Project
містить інструменти для роботи із проектами, є можливість відкривати, змінювати, відкривати нещодавно переглянутий проект, зберігати проект або експортувати робочий простір у файл.

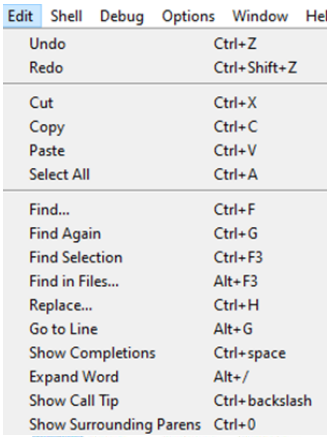


Вкладка Preferences
мстить налаштування програми, є можливість відкриття глобальних налаштувань програми, налаштування перевірки синтакисису, вигляд теми та кольорової схеми програми

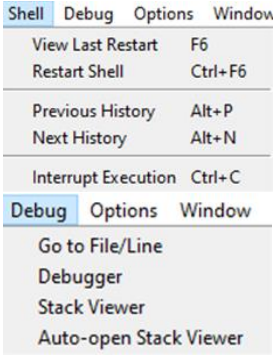
1.5.2. IDLE



Вкладка File:
New File – створення нового файлу;
Open – відкриває вже існуючий файл;
Open Module – відкриває модуль;
Recent File – відкриває вже існуючий файл;
Module Browser – огляд існуючих модулів;
Path Browser – огляд шляхів, де знаходяться модулі;
Save – зберігає активний файл;
Save As – зберігає активний файл із можливістю зміни назви та вибору шляху розміщення;
Save Copy As - зберігає копію файлу;
Print Window – надсилає контент вікна до друку;
Close – закриває файл;
Exit – вихід з програми.

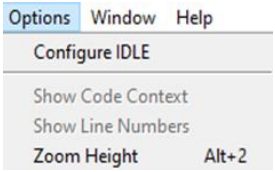


Вкладка Edit:
Undo – відміняє останню зміну;
Redo – повторює останню зміну;
Cut – вирізає виділений текст;
Copy – копіює виділений текст;
Paste – вставляє скопійований текст;
Select All – виділяє весь текст;
Find - пошук по тексту;
Find Again – повторний пошук;
Find Selection – пошук виділеного тексту;
Find in Files – пошук по файлах;
Replace – заміна тексту;
Go to Line – переміщення до рядку;
Show Completions – перехід до кінця файлу;
Expand Word – розгортає згорнутий блок;
Show Call Tip – показує підказку, що до виклику функції.



Вкладка Shell:
View Last Restart – перегляд останнього запуску;
Restart Shell – перезапуск консолі;
Preview History – попередній перегляд історії;
Next History – перехід до наступного елементу історії;
Interrupt Exception – виклик виключення під назвою «перервано з клавіатури».

Вкладка Debug:
Go to File/Line – перехід до файлу або рядку у файлі;
Debugger – запуск дебагеру;
Stack View – перегляд стеку;
Auto-open Stack View – автоматичне увімкнення перегляду стеку.



Вкладка Option:
Configure IDLE – відкриття налаштувань IDLE;
Show Code Context – показує область дії функції, класу, циклу або іншої конструкції;
Show Line Number – показує номер рядка;
Zoom Height – збільшення висоти рядків.

1.6. Контрольні запитання

1. Що таке Python?
2. Перелічіть плюси мови програмування Python.
3. Опишіть основні мінуси мови програмування Python.
4. Назвіть сфери використання мови програмування Python.
5. Які переваги має мова програмування Python?
6. Які основні функції має текстовий редактор?
7. Назвіть методи виведення результатів розробленої програми.

Лабораторна робота №2

Змінні та типи даних. Операції з числами. Умовні вирази

2.1. Мета роботи

Ознайомитися з поняттям змінні та з основними типами даних мови програмування Python. Освоїти математичні операції з змінними в Python. Навчитися працювати з змінними та їх елементами.

2.2. Теоретичні відомості

2.2.1. Загальні відомості

Всі дані та інформація в комп'ютерах зберігаються в пам'яті. Для зберігання тої чи іншої інформації потрібно присвоювати місце її зберігання серед комірок з яких складається пам'ять. Це місце і називають змінними.

Змінна – це назва для зарезервованого місця пам'яті комп'ютера. Коли ви створюєте змінну і присвоюєте їй значення, це означає, що ви резервуєте місце. Можна уявити, що змінна – це корзина з визначеним розміром, в даному випадку розмір – це і є тип. В Python вам не потрібно оголошувати тип змінної вручну, як наприклад в мові програмування C++ або ж Java, оголошення проходить автоматично коли ви присвоюєте значення змінній і це називається – *динамічна типізація*.

Якщо коротко, то змінні нам потрібні для зберігання інформації, подальшої її обробки та організації даних.

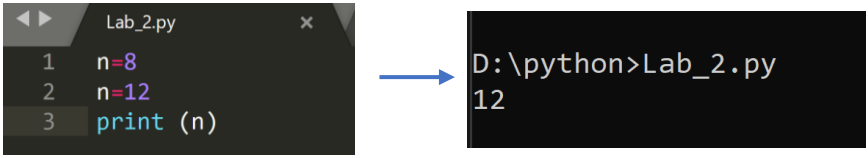
2.2.2. Змінні, типи змінних та їх використання

Щоб надати даним певне ім'я або ж змінну використовують операцію *присвоєння.* В Python дана операція позначається знаком «=». Ліворуч вказуєте змінну (ім'я), праворуч – дані, що належать даній змінній. Повторне присвоєння даних стирає минулі дані та присвоює нове значення змінної.

Наприклад:

Присвоїмо змінній «*n*» значення 8 за допомогою операції присвоєння «=». Змінимо присвоєне значення змінній 8 на 12 та виведемо результат на екран за допомогою «print (*ім'я змінної*)».

Результат (Sublime + командний рядок):



Увага:

Ім’я може містити: латинські букви, цифри, знаки підкреслення.

Заборонено: починати змінну з цифри, використовувати пробіли та співпадіння з спеціальними словами, що вже закладені в мову програмування Python.

Намагайтеся давати змінним значення (імена), що несуть інформацію про дані, що вони містять.

Наприклад:

Виведемо три змінні використовуючи правила створення змінних.

Результат (IDLE Shell):

```
name = "My name is Hrytsiuk Ivanna"
year_now = 2021
rate_1_dolar = 27.84
print (name, year_now, rate_1_dolar)
```

```
========== RESTART: D:\python\Lab_2_3.py
My name is Hrytsiuk Ivanna 2021 27.84
>>>
```

Типи даних в Python не потребують оголошення, тому таку типізацію називають ще *неявною.* Серед великої кількості типів даних найважливішими в Python вважають:

INT – цілі числа (додатні та від’ємні числа без дробової частини)

FLOAT – дробові числа (числа з плаваючою крапкою)

STR – рядковий тип (набір символів або ж текстової частини)

BOOL – логічний тип (тип, що приймає значення true або false)

LIST – список (масив упорядкованої структури, що містить об’єкти різних типів)

Наприклад:

Виведемо за допомогою «print (type(*ім’я змінної*))» на екран типи даних в залежності від введених в змінні даних типу INT, FLOAT, STR

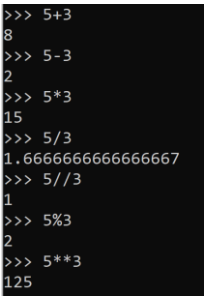
(дані можна вводити в одинарних або ж в подвійних лапках), BOOL і LIST.

Результат (Sublime + командний рядок):

2.2.3. Математичні операції з числами та умовні вирази

В Python можна використовувати *звичайні математичні операції* (приклад використання (командний рядок)):

A+B – додавання (сума);
A-B – віднімання (різниця);
A*B – множення (добуток);
A/B – ділення (частка);
A//B – ціла частина від ділення чисел;
A%B – залишок від ділення чисел;
A**B – степінь числа;



Умовні вирази:

> – більше ;
< – менше;
>= – більше або рівне;
<= – менше або рівне;
== – рівне;
!= – не рівне;

2.2.4. Робота з змінними типу рядок

Змінні рядкового типу можуть бути записані за допомогою одинарних ‘*текст*’ або подвійних “*текст*” лапок. До елементів рядка можна звертатися (виводити) по індексу за допомогою квадратних

дужок [*номер елементу*] після оголошеної змінної. Лічба елементів в рядку розпочинається з 0.

Для отримання певного проміжку з рядка використовується оператор зрізу «:» та записується наступним виглядом [*початковий елемент*:*кінечний елемент*].

Також існує зворотна індексація, що дає змогу виводити елементи з кінця рядку використовуючи знак «-» перед номером елемента.

Рядки можна сумувати між собою та множити на число для дублювання.

Наприклад:

Задаймо змінній значення власного прізвища та імені. Виведемо на екран елемент під номером 0 та 2. Виведемо прізвище використовуючи оператор зрізу. Виведемо 1 і 6 елемент з кінця заданого рядка в змінну. Додаймо до змінної з прізвищем та іменем власне по-батькові. Продублюємо отриманий рядок.

Результат (командний рядок):

```
>>> my_name = "Hrytsiuk Ivanna"
>>> my_name[0]
'H'
>>> my_name[2]
'y'
>>> my_name[0:8]
'Hrytsiuk'
>>> my_name[-1]
'a'
>>> my_name[-6]
'I'
>>> my_name + " Mykhaylivna"
'Hrytsiuk Ivanna Mykhaylivna'

> " Hrytsiuk Ivanna Mykhaylivna" *3
Hrytsiuk Ivanna Mykhaylivna Hrytsiuk Ivanna Mykhaylivna Hrytsiuk Ivanna Mykhaylivna'
```

2.3. Програма роботи

1. Ознайомитися з теоретичними відомостями, що до змінних, типів і методів роботи з ними.
2. Виконати ознайомчі завдання теоретичних відомостей використовуючи Sublime Text 3 або IDLE Shell.

3. Виконати завдання згідно вашого варіанту. Додати коментарі до коду.
4. Оформити звіт.

2.4. Порядок виконання роботи

Завдання 1.

1. Створити три змінні: перша – цілочисельного типу, в якості значення використати порядковий номер в списку групи, друга і третя зміна – строкового типу, в якості значення використати свої ПІБ та дату народження.
2. Вивести значення змінних в один рядок використовуючи Sublime Text 3 + командний рядок (зберегти програму під назвою Lab_2_1 в папці Python на диску D).

Завдання 2.

1. Створити змінну та присвоїти їй значення вашого ПІБ.
2. Використовуючи цю змінну вивести літери, які знаходяться на позиціях X та Y, де: X – десятки числа, яке отримане в результаті множення порядкового номера в списку групи на ваш вік, а Y – одиниці числа, яке отримане в результаті множення порядкового номеру групи на сьогоднішній день в році (всі розрахунки виконати мовою програмування Python).
3. Використовуючи змінну з вашим ПІБ вивести ім'я N разів за допомогою зрізів, де N – ваш порядковий номер в списку групи.
4. Зберегти програму під назвою Lab_2_2 в папці Python на диску D та вивести результати.

2.5. Контрольні запитання

1. Дайте визначення терміну змінна.
2. Які є вимоги до іменування змінних?
3. Для чого використовується операція присвоєння?
4. Назвіть основні типи даних мови програмування Python та дайте їх коротку характеристику.
5. Який тип типізації використовується в Python?
6. Назвіть математичні операції мови програмування Python.
7. Перелічить умовні вирази мови програмування Python.
8. Яким чином можна звертатись до елементів рядка? Навести приклади синтаксису.

Лабораторна робота №3

Умовні конструкції. Цикли. Функції. Область видимості змінних. Модулі. Обробка виключень

3.1 Мета роботи

Ознайомитися з поняттям умовних конструкцій, циклів, функцій, області видимості змінних, модулів та обробки виключень мови програмування Python. Освоїти роботу з умовними конструкціями та методами застосування. Навчитися розрізняти види циклів та напрямки застосування. Навчитися створювати функції, працювати з ними та використовувати для створення нових модулів. Вивчити методи використання модулів та обробки виключень.

3.2. Теоретичні відомості

3.2.1. Загальні відомості

Для реалізації логіки роботи програми в мові програмування Python існують оператори розгалуження або їх ще називають умовні оператори. Ці оператори слугують для зміни поведінки роботи програми в залежності від певних умов. До прикладу необхідно відправити електронний лист користувачу, але перед відправкою необхідно виконати перевірку чи вказав користувач свою електронну адресу, для цього можна використати умовний оператор.

Також бувають ситуації коли потрібно виконати певні дії декілька разів, для цього в мові програмування Python існують цикли.

Цикл – це керуюча конструкція, яка призначена для багаторазового повторення інструкції. Прикладом може слугувати ситуація із попереднього прикладу про відправку повідомлень, у випадку, якщо повідомлення необхідно відправити декільком користувачам.

Під час написання програмного коду часто виникають ситуації коли якась логіка виконується в різних частинах програми і для того щоб не копіювати логіку з одного місця в інше в Python існують функції. Функції дають можливість позбавити код від дублювання логіки, що зменшує загальний об’єм кодової бази та подальшої зміни цієї логіки в майбутньому, тобто змини в роботі програми потрібно буде виконати в одному місці, що значно швидше ніж шукати дану логіку у всьому проекті та змінювати код.

Для кращої організації коду в Python існують модулі. Модулі дають можливість розділити код на логічні частини, це спрощує розуміння та пошук потрібної ділянки коду в проекті. Також цей інструмент дає можливість групувати модулі в пакети та бібліотеки, а в подальшому використовувати їх в різних проектах.

Під час виконання програм можливе виникнення помилок які впливають на роботу програми. У великій кількості випадків програма після виникнення такої помилки може завершити свою роботу достроково, тому такі ситуації називаються винятковими. Виняткові ситуації це до прикладу, повертаючись до відправки електронних листів, це може бути відсутність підключення до почтового сервера, що унеможливлює відправку повідомлення і призводить до завершення роботи програми з помилкою, яку не бажано показувати кінцевому користувачу програмою, тому для вирішення цієї проблеми в Python існують інструменти для перехоплювання таких помилок та коректного, зрозумілого для кінцевого користувача відображення помилок.

3.2.2. Умовні конструкції та ввід користувача

Будь-якому розробнику часто потрібно перевіряти різні умови і в залежності від вибору приймати відповідні рішення. Оператор *IF* в мові програмування Python перевіряє умову та обирає подальші дії в залежності від результатів свого порівняння умов. Умова може бути або істино – *«так»* (умова вірна, умова виконується) або хибною – *«ні»* (умова не виконана). Якщо в нас умова істина, то ми виконуємо будь-яку дію або групу дій, що вказана далі.

Команда IF:

if *умова:*
    дія 1
    дія 2
    …
    дія n

Наприклад:

Виведемо умову – якщо вік більше 18, то вхід відкрито.

```
age = 18
if (age >= 18):
    print ("Вхід відкрито")
```

Результат:

```
====================
Вхід відкрито
>>>
```

В прикладі було оголошено, що якщо вам більше 18, то вхід відкрито, а як повідомити, що вхід зачинено, якщо вік менший? Для цього в дію вводиться повна вітка команди *IF* – конструкція команд *IF – ELSE*.

Якщо умова *IF* виконана програма виконує «*дію 1*», якщо умова *IF* не виконалась програма переходить до «*дії 2*» через умову *ELSE*.

Команда IF-ELSE:

if *умова:*
    дія 1
else:
    дія 2

Наприклад:

Доповнимо умову – якщо вік більше 18, то вхід відкрито, якщо менше, то вхід зачинено.

```
age = 16
if (age >= 18):
    print ("Вхід відкрито")
else:
    print ("Вхід зачинено")
```

Результат:

```
===============
Вхід зачинено
>>>
```

Припустимо, що в нас умова, якщо вік менше 18 – вхід заборонено, вік від 18 до 25 – вхід тільки з мамою, а якщо вік від 25 – вхід відчинено. Перед нами з'являється ще одна умова, ще одна перевірка. В даному випадку вступає в дію конструкція команд *IF – ELIF – ELSE*. Для роботи з оператором *ELIF* використовуємо логічні операції, щоб ввести декілька умов.

Основні логічні операції:

AND (і) – логічне І (*True*, якщо обидва *x* і *y* дорівнюють *True, False* в інших випадках).

OR (або) – логічне АБО (*True,* якщо хоча б один з *x* і *y* дорівнює *True, False* в інших випадках).

NOT (ні) – логічне НІ (якщо *x* є *True*, то поверне *False*, якщо *x* є *False*, то поверне *True).*

Якщо умова *IF* виконана програма виконує «*дію 1*», якщо умова *IF* не виконалась програма переходить до умови *ELIF* та виконує «*дії 2*». Це триває доки не виконається умова одного з усіх блоків *ELIF*. Після чого програма переходить до *«дії n»* через умову *ELSE.*

Команда
IF – ELIF – ELSE:

if *умова 1:*
    дія 1
elif *умова 2:*
    дія 2
else:
    дія 3

Наприклад:

Виправимо умову – якщо вік менше 18 – вхід заборонено, вік від 18 до 25 – вхід тільки з мамою, а якщо вік від 25 – вхід відчинено.

```
age = 22
if (age >= 25):
    print ("Вхід відкрито")
elif ((age >= 18) and (age < 25)):
    print ("Вхід тільки з мамою")
else:
    print ("Вхід зачинено")
```

Результат:

Вводимо вік – 16, 46, 22

```
====================
Вхід зачинено
>>>
====================
Вхід відкрито
>>>
====================
Вхід тільки з мамою
>>>
```

Для того, щоб ви змогли самі ввести свій вік та перевірити результат не змінюючи код програми використовується функція вводу користувача за допомогою *input ()*. Але так як ми працюємо з числами, а ввід інформації виконується тільки рядками використаємо один із перетворювачів: *int(), float(), str()*.

Наприклад:

```
age = int(input ("Ваш вік:"))
if (age >= 25):
    print ("Вхід відкрито")
elif ((age >= 18) and (age < 25)):
    print ("Вхід тільки з мамою")
else:
    print ("Вхід зачинено")
```

Результат:

```
Ваш вік:22
Вхід тільки з мамою
>>>
```

3.2.3. Функції

Функції – це створювана інструкція, яка може приймати дані вводу, виконувати вказівки чи дії з ними і повертати дані виводу. Використовуються для розрахунків, виконання певних операцій, але головною особливістю і плюсом використання функції є можливість повторного використання певної ділянки коду не повторюючи її, що дуже спрощує роботу програміста.

Функцію можна описати математично:

$$f(x) = 20 + 5x$$

де f – назва функції, що приймає в себе певні вихідні дані від x і після цього з даними x виконує певні операції $20 + 5x$. Якщо ми

використаємо дану функцію ще раз вона нам виведе вже певну відповідь що міститиме змінна.

За цим принципом працює і функція в мові програмування Python.

Для цього оголошуємо функцію з назвою та параметром в круглих дужках, виводимо потрібне значення в наступному рядку та запускаємо функцію за допомогою її назви та параметра.

Тепер ми можемо за допомогою імені і параметра функції викликати її довільну кількість разів в довільному місці коду, що спрощує задачі, пришвидшує роботу та дозволяє краще зрозуміти той чи інший код.

Функція *DEF:*
DEF *ім'я_функції* (*параметри*):
визначення_функції

Наприклад:

Оголосимо функцію з назвою нашої лабораторної роботи та визвемо її.

```
def lab_3():
    print ("Умовні конструкції. Цикли...")
lab_3()
```

Результат:

```
Умовні конструкції. Цикли. Функції.
Область видимості змінних. Модулі.
Обробка виключень
>>>
```

Використаємо функцію для сумування двох чисел, що вводить користувач.

Для цього задаємо змінним можливість вводу двох цілочисельних чисел. Використаємо функцію, що приймає два параметри для сумування. Оголошуємо «*return*» з значенням нашого розрахунку для повернення отриманих значень.

Далі викликаємо нашу функцію, що буде використовувати значення введених *x* і *y* та записувати значення в змінну z, а в подальшому виводити значення на екран.

Наприклад:

```
x= int(input("Введіть перше число: "))
y= int(input("Введіть друге число: "))

def sum (a, b):
    return a+b
z= sum (x,y)
print (z)
```

Результат:

```
Введіть перше число: 18
Введіть друге число: 12
30
>>>
```

На прикладі вище згаданої формули розглянемо обов'язкові і не обов'язкові параметри.

Якщо в даному випадку при виведенні значення параметр не буде вказаний, то результату функції не буде виведено, якщо параметр вказати в оголошенні функції – результат буде обраховано без помилки, якщо параметр буде вказаний і в оголошенні функції і в виведенні результату – функція буде обрахована за другим параметром.

Якщо функція задана в середині функції – це локальна змінна і вона буде використовуватися тільки в самій функції. Всім змінні, що знаходяться за межами функцій є глобальними змінними і для використання в середині функцій викликаються за допомогою оператора *global.*

Наприклад:

```
def f(x):
    return 20+5*x

print(f(4))
```

Результат (без параметру):

```
Traceback (most recent call last):
  File "D:/python/Lab_3_4.py", line
4, in <module>
    print(f())
TypeError: f() missing 1 required p
ositional argument: 'x'
>>>
```

Результат (з параметром):

```
40
>>>
```

Наприклад:

```
def f(x=4):
    return 20+5*x

print(f())
```

Результат:

```
40
>>>
```

Наприклад:

```
def f(x=42):
    return 20+5*x

print(f(4))
```

Результат:

```
40
>>>
```

3.2.4. Цикли

Цикл – фрагмент коду, який виконується багаторазово. В мові програмування Python є два основних цикли: *FOR* та *WHILE*.

FOR – це цикл, що перебирає ітераційний об'єкт.

Ітераційний об'єкт – це об'єкт (рядок, список, кортеж чи інше), що можна розбити на малі об'єкти та присвоїти порядкові номера.

Ітерація – процес перебору ітераційних об'єктів.

Для наглядного роз'яснення створимо цикл з використанням свого імені. Задаємо для початку змінну з власним ім'ям. Використовуючи

цикл *FOR* виведемо кожен елемент ітераційного об'єкту (символ) на екран.

Для виводу результатів декілька разів в мові програмування Python використовується функція *range*, що дає послідовність цілих чисел та приймає в себе два параметри: те з якого числа вона взяла своє значення і те, до якого вона буде брати свої значення послідовності.

Цикл FOR:

for *ім'я_змінної* in *ім'я_ітераційного_об'єкта:*
    інструкції

Наприклад:

Виведемо ім'я за допомогою цикла та ітерації рядка.

```
name = "Hrytsiuk Ivanna"
for i in name:
    print(i)
```

Результат:

```
==
H
r
y
t
s
i
u
k

I
v
a
n
n
a
>>
```

Для того, щоб вивести 10 разів один і той же рядок, ми повинні: оголосити сам рядок, запустити цикл з назвою та кількістю повторів (останнє значення береться за формулою *n-1*) та вивести інформацію на екран.

Наприклад:

```
name = "Hrytsiuk Ivanna"
for i in range (1,11):
    print(name)
```

Результат:

```
Hrytsiuk Ivanna
Hrytsiuk Ivanna
Hrytsiuk Ivanna
Hrytsiuk Ivanna
Hrytsiuk Ivanna
Hrytsiuk Ivanna
Hrytsiuk Ivanna
Hrytsiuk Ivanna
Hrytsiuk Ivanna
Hrytsiuk Ivanna
```

WHILE – цикл з передумовою. В даному циклі для початку проводиться перевірка умови, подібно як в умовній конструкції *IF*, а після того виконується завдання циклу.

Виведемо за допомогою циклу *WHILE* вибірку чисел від 1 до 10. Оголосимо змінну, напишемо в циклі, що змінна не повинна бути більшою за 10 та виведемо значення змінної з кроком в 1.

Для зупинки циклу використовується ключове слово – *BREAK*, яке не залежно на умову циклу виконає вихід в тому місці, де буде знаходитися.

Для переривання кола циклу, що йде в даний момент та переходу відразу до наступного використовується ключове слово – *CONTINUE*.

Цикл WHILE:

while *умова:*

фрагмент коду

Наприклад:

```
i = 1
while i<=10:
    print (i)
    i=i+1
```

Результат:

```
1
2
3
4
5
6
7
8
9
10
>>>
```

Наприклад:

```
i = 1
while i<=10:
    print (i)
    i=i+1
    break
```

Результат:

```
1
>>>
```

Наприклад:

```
i = 1
while i<=10:
    if i != 5:
        print (i)
    i=i+1
    continue
```

Результат:

```
1
2
3
4
6
7
8
9
10
>>>
```

3.2.5. Модулі та пакети

Модулі і пакети значно спрощують роботу програміста. Класи, об’єкти, функції, константи, якими потрібно часто користуватися можна запакувати в модуль і в подальшому загружати його в свої програми при необхідності.

Пакети дозволяють формувати простір імен для роботи з модулями. Можна умовно поділити на модулі і програми. Програми призначені для прямого запуску, а модулі в свою чергу для імпортування їх в інші

програми. Можна відмітити, що модулі можуть бути написані не тільки мовою програмування Python, а й іншими мовами, наприклад C.

Як імпортувати модуль? Самий простий варіант – це використати конструкцію *import*.

Наприклад:

```
import math, datetime
print(math. factorial(5))
print(datetime.datetime.now())
```

Результат:

```
120
2021-02-21 00:45:50.239209
```

Відкриємо модуль створений раніше в завданнях з функціями та використаємо його для виводу результата.

Наприклад:

```
import Lab_3_2
print(Lab_3_2.lab_3())
```

Модуль:

```
def lab_3():
    print ("Умовні
```

Результат:

```
Умовні конструкції. Цикли...
```

3.2.6. Обробка виключень

Виключення – це події, що стаються через помилки при виконанні скріпта.

Основна проблема з виключеннями в тому, що коли стається дана помилка, хід програми «грубо» переривається. Це може привести до втрати раніше оброблених даних, зупинки виконання важливої задачі або ж втрати зв'язку та працездатності сайту.

Тобто, в тій ситуації, якщо під час виконання скріпта з'являється деяка помилка і викликається виключення в нас повинна бути можливість перемикнутися на той блок коду, який зробить під час даного виключення те що потрібно зробити нам, а не завершення роботи.

Для цього в мові програвання Python з'являється конструкція *TRY – EXCEPT*.

Синтаксис дуже простий та схожий до *IF – ELSE*, тобто ми повинні той рядок, що викликає підозру на виключення внести в блок *TRY*.

Якщо блок *TRY* призвів до виключення, Python припиняє всі дії та умови, що були надані блоку та автоматично починає виконуватися умова блоку *EXCEPT* (кількість блоків довільна).

При розширенні конструкції додається умова *FINALLY*, що виконує обов'язкову заключну дію для будь-якого розвитку подій (при наявності *return* виконується перед ним).

Конструкція TRY – EXCEPT:

основна умова

try:

дія 1

except:

другорядна умова

дія 2

Наприклад:

Створити розклад вільних годин, вивести час та перевірити його. Якщо година зайнята перейти до іншої умови

```
def main():
    time = ["12:25" "13:10" "14:40"]
    try:
        print (time["15:40"])
    except:
        sum = 1 + 2
        print(sum)

main()
```

Результат:

```
3
>>>
```

3.3. Програма роботи

1. Ознайомитися з теоретичними відомостями, що до умовних конструкцій, функцій, циклів, модулів та виключень.
2. Виконати ознайомчі завдання теоретичних відомостей.
3. Виконати завдання згідно вашого варіанту. Додати коментарі до коду.
4. Оформити звіт.

3.4. Порядок виконання роботи

Завдання 1

Написати програму з використанням умовних конструкцій, яка буде відображати ознаку тіла згідно формули «індексу маси тіла».

Умови:

- ввід ваги в (кг) і зросту в (см) в ручну з консолі;
- вивід результату у вигляді «*ознака, показник* = »
- заокруглити значення показника до десятих.

Індекс визначається за формулою:

$$I = m / h^2$$

де I – показник ІМТ, m – маса тіла в кілограмах, h – зріст в метрах.

Умови для відображуваних ознак.

Показник ІМТ, кг/м^2	Ознака
Менше 18.5	свідчить про недостатню вагу
18,5-24,9	еквівалент нормальної маси тіла
25,0-29,9	вказує на наявність зайвої ваги
Понад 30	є ознакою ожиріння

Завдання 2

Написати програму використовуючи функції, яка буде розраховувати значення за формулою, що обирається відповідно до варіанту з використанням аргументів, які програма отримує від користувача з консолі. Використати обробку виключень, яка повідомлятиме про помилку обчислення – «значення знаменника дорівнює 0». За допомогою модулю вивести дату та час виконання роботи.

Варіанти:

1	$result = \frac{a}{b^2} + \frac{1}{a^2 + b^2}$	6	$result = \frac{-b + (b^2 - 4 \cdot a \cdot c)^{0.5}}{2a}$
2	$result = \frac{a^4}{b} + \frac{\frac{c}{78}}{a + b^5}$	7	$result = \frac{2a^2 b^2}{a^2 - b^2} e^3$
3	$result = \frac{a}{1.7} + \frac{1}{\frac{a^5}{19} + b^2}$	8	$result = \frac{\frac{a^3}{2e^{12}}}{6b}$
4	$result = \frac{\frac{1}{12} + b^3}{a^4 + c \cdot (1 + a)^3}$	9	$result = \frac{\frac{a}{b^2}}{a^2 + b^2} e$
5	$result = \frac{e}{b^2} - \frac{a^{1.2}}{5 + e^4}$	10	$result = \frac{a}{b^2} + a^2 + b^{0.2}$

Завдання 3

Вивести за допомогою циклу *for* лише парні результати множення чисел від 1 до 10 на ваш номер варіанту.

Завдання 4

За допомогою циклу *while* вивести числа Фібоначчі до N (введення з консолі), зробити перевірку введених даних користувачем (N > 1000, якщо введені дані не відповідають умові запропонувати ввести число ще раз). Виключити з вибірки число M, що вводиться із клавіатури (>1 та належить вибірці).

3.5. Контрольні запитання

1. Напишіть та опишіть синтаксис умовних конструкцій.
2. Що таке функція, як використовується в програмування?
3. Опишіть різницю між циклами *FOR* та *WHILE.* Охарактеризуйте кожен із цих циклів.
4. Що таке *BREAK* та *CONTINUE?* Використання в програмах.
5. Опишіть, що називають областю видимості змінних. Які вони бувають?
6. Що таке модуль? Сенс використання модулю.
7. Як підключити модуль до програми? Яким чином створюються модулі власноруч?
8. Що таке обробка виключень? Застосування обробки виключень.
9. Синтаксис використання виключень.
10. Приклад використання модулю.
11. Приклад використання функції.
12. Приклад використання виключень.

Лабораторна робота №4

Списки. Кортежі. Словники. Множини

4.1. Мета роботи

Вивчити теоретичні відомості, що до списків, кортежів, словників та множин. Навчитися розрізняти та використовувати їх функції та особливості. Вивчити основні методи роботи над списками, кортежами, словниками та множинами. Навчити використовувати та модифікувати.

4.2. Теоретичні відомості

4.2.1. Загальні відомості

Щоб удосконалити програмний код, додайте до нього нові важливі функції, що використовують різноманітні методи. Python містить важливі типи даних, які мають високу ймовірність, використовуватись кожного дня у програміста. До них відносяться: створення списків, кортежів або словників та множин.

4.2.2. Списки

Що таке списки? Багато хто говорить, що *списки* – це ітераційний об'єкт – своєрідний тип даних, який дозволяє також зберігати якісь певні значення. На відміну від тих типів даних, що були розглянути раніше (*INT, FLOAT, BOOL, STR*), список дозволяє зберігати в собі багато значень відразу, не залежно від того, якого вони типу. Тобто список – це свого роду контейнер, що зберігає об'єкти в певному порядку (в кожного об'єкта присвоєний свій порядковий номер і через нього проходить звернення до даного об'єкту)

Списки в собі можу містити: цифри цілі та дробові, текст і навіть самі списки. Також можна використовувати пості списки.

Наприклад:

Виведемо пустий, чисельний, змішаний списки та список в списку

```
spisok = []
spisok_num = [1, 3, 5, 7]
print(spisok_num)
spisok_oll = [3, 5, 13.4, 'text']
print(spisok_oll)
spisok_spisok = [2, 5, 31.6, 'text', [1, 2, 3, 4]]
print(spisok_spisok)
```

Результат:

```
[1, 3, 5, 7]
[3, 5, 13.4, 'text']
[2, 5, 31.6, 'text', [1, 2, 3, 4]]
>>>
```

В чому переваги списків? Наведемо приклад для очного розуміння і освоєння.

Для прикладу введемо ім'я домашніх улюбленців, в нашому випадку котів, які вам прийдуть на розум. Присвоїмо кожному котику свою змінну і порядковий номер. Виведем всіх котів на екран.

Повторимо цю ж дію за допомогою списку. Кожному значенню в списку присвоюється свій порядковий номер починаючи від 0, що дає змогу викликати елементи по одному.

Наприклад:

```
name_cat1 = 'Мурчик'
name_cat2 = 'Пушок'
name_cat3 = 'Васька'
name_cat4 = 'Семен Семенович'
print(name_cat1)
print(name_cat2)
print(name_cat3)
print(name_cat4)


print('     ')


name_cats = ['Мурчик', 'Пушок', 'Васька','Семен Семенович']
print(name_cats)
print(name_cats[0])
print(name_cats[-1])
```

Результат:

```
Мурчик
Пушок
Васька
Семен Семенович

['Мурчик', 'Пушок', 'Васька', 'Семен Семенович']
Мурчик
Семен Семенович
```

В минулих лабораторних ми познайомилися з циклами. Спробуємо використати цикл для списку. Створимо цикл використовуючи наш список котиків та виведемо його на екран.

Наприклад:

```
name_cats = ['Мурчик', 'Пушок', 'Васька','Семен Семенович']

for name in name_cats:
    print(name)
```

Результат:

```
Мурчик
Пушок
Васька
Семен Семенович
>>>
```

Для роботи з списками найчастіше використовуються методи та їх варіації. Розглянемо деякі із них на прикладі наших котів.

Метод *APPEND* додає в кінець списку вказаний елемент в (). Додамо в список ще одного домашнього улюбленця.

Метод *POP* видаляє останній елемент з визначеного списку.

Метод *INDEX* визначає індекс обраного елементу. На прикладі присвоїмо змінній індекс елемента *МУРЧИК*, а *LEN* допоможе визначити кількість об'єктів (довжину списку).

Метод *SORT* дасть змогу відсортувати дані в списку чи то чисельні чи то рядкові. Метод сортує дані лише одного типу даних, при вмісту двох і більше типів – операція не дасть результату. Для сортування в зворотному порядку використовується метод *SORT* з уточненням *REVERSE=TRUE*. Відсортуємо наших домашніх улюбленців та переглянемо результат.

Також списки мають змогу змінювати якийсь певний елемент вказавши його індекс та присвоївши нове значення. Замінимо ім'я останнього улюбленця.

Наприклад:

```
name_cats = ['Мурчик', 'Пушок', 'Васька','Семен Семенович']
print(name_cats)

name_cats.append('Котик')
print(name_cats)

name_cats.pop()
print(name_cats)

n = name_cats.index('Мурчик')
print(n)

print(len(name_cats))

name_cats.sort()
print(name_cats)

name_cats.sort(reverse=True)
print(name_cats)

name_cats[3]='Семен'
print(name_cats)
```

Результат:

```
['Мурчик', 'Пушок', 'Васька', 'Семен Семенович']
['Мурчик', 'Пушок', 'Васька', 'Семен Семенович', 'Котик']
['Мурчик', 'Пушок', 'Васька', 'Семен Семенович']
0
4
['Васька', 'Мурчик', 'Пушок', 'Семен Семенович']
['Семен Семенович', 'Пушок', 'Мурчик', 'Васька']
['Семен Семенович', 'Пушок', 'Мурчик', 'Семен']
```

4.2.3. Кортежі та словники

Кортеж (*tuple*) – це контейнер, що зберігає в собі упорядковані об'єкти. Схожий до списку, але має основну відмінність від списку – це незмінність. Для чого ж вони потрібні, а для того, щоб ви були впевнені, що сталі і не змінні дані в вашій програмі (особливо, якщо вона масштабна) були точно не змінені.

Розглянемо на прикладі звичайний кортеж. Відмінність в синтаксисі є дві: запис в круглих дужках () та вкінці елементів ставиться кома (для позначення, що це і справді є кортеж).

Наприклад:

```
tuple = ('dog', 3, 12, 24.8,)
print (tuple)
```

Результат:

```
('dog', 3, 12, 24.8)
```

Дані в кортежах є незмінними, а для того, щоб виправити потрібно створювати новий та записувати в нього нові дані.

Будь-який кортеж можна переформатувати в список (*LIST + ( )*), а список в кортеж (*TUPLE + [ ]*).

Наприклад:

```
print(tuple([45, 11, 12, 2,]))
print(list((45, 11, 12, 2)))
```

Результат:

```
(45, 11, 12, 2)
[45, 11, 12, 2]
```

Словник (*dict*) – це колекція багатьох значень. Грубо кажучи – це такий самий контейнер для зберігання значень, як і список та кортеж, але якщо в списку чи кортежі в якості індексу в нас використовувались порядкові номера, то тут в якості індексу використовується певний ключ. Даний ключ може бути будь-якого типу даних. Кожен ключ має своє певне значення.

Розглянемо роботу з словниками на прикладі. Створимо словник з чотирьох ключів та значень (створюється за допомогою знаку «:» в фігурних дужках*{}*).

Наприклад:

```
dict = {"apple":"red", "car":"fast", "cat":"nice", "boy":"strong"}
print(dict)
```

Результат:

```
{'apple': 'red', 'car': 'fast', 'cat': 'nice', 'boy': 'strong'}
```

Для отримання та виведення всіх ключів з словника використовується метод *KEYS*, а метод *VALUES* навпаки для отримання всіх значень списку. Використаємо дані методи до нашого словника.

Наприклад:

```
for K in dict.keys():
    print(K)
```

Результат:

```
apple
car
cat
boy
```

Наприклад:

```
for V in dict.values():
    print(V)
```

```
red
fast
nice
strong
```

Для отримання та виведення попарно ключів та значень зі списку використовується метод *ITEMS*. Використаємо даний метод до нашого словника.

Наприклад:

```
for I in dict.items():
    print(I)
```

Результат:

```
('apple', 'red')
('car', 'fast')
('cat', 'nice')
('boy', 'strong')
```

Для виводу значення через оголошення ключа використовується вивід на екран з зазначенням назви словника та ключового слова потрібного значення в квадратних дужках.

Для зміни значення ключа оголошується сам ключ через назву словника та присвоюється нове значення для нього. Для видалення значення та ключа з словника використовується команда *DEL* з назвою словника та потрібного елемента. Скористаємося даними функціями.

Наприклад:

```
print(dict["car"])
```

Наприклад:

```
dict["car"] = "near"
print(dict)
```

Наприклад:

```
del(dict["car"])
print(dict)
```

Результат:

```
fast
{'apple': 'red', 'car': 'near', 'cat': 'nice', 'boy': 'strong'}
{'apple': 'red', 'cat': 'nice', 'boy': 'strong'}
```

4.2.4. Множини

Множини (set) в мові програмування Python – це структура даних, що містить в собі не впорядковані елементи. *Невпорядкований елемент* – це ті елементи, що не проіндексовані та не мають свого порядкового номеру та розташовані в випадковому порядку.

Множини мають ряд властивостей, що відділяють їх від інших структур даних:

1. Невпорядкованість елементів;
2. Всі елементи унікальні та не мають дублікатів;
3. Елементи множини є незмінними, але саму множину ми можемо змінювати.

Розглянемо основні функції та методи для роботи з множинами. Для створення множини запишемо значення змінної в фігурних дужках. Для створення «замороженої» множини без права зміни даних використовується функція *FROZENSET*. Для виведення пустої множини не достатньо залишити пусті фігурні дужки, в такому випадку створиться пустий словник, нам ж потрібно оголосити, що це саме множина за допомогою *SET*. Розглянемо на прикладах.

Наприклад:
```
numb = {1, 2, 3, 77, 5}
print(numb)
```
Результат:
```
{1, 2, 3, 5, 77}
```

Наприклад:
```
numb = frozenset({1, 2, 3, 77, 5})
print(numb)
```
```
frozenset({1, 2, 3, 5, 77})
```

Наприклад:
```
numb2 = set()
print(type(numb2))
```
```
<class 'set'>
```

Як і раніше, до множин також використовують цикли. Розглянемо їх роботу.

Наприклад:

```
numb4 = {1, 2, 3, 4, 5, 6, 7}
for i in numb4:
    print(i)
```

Результат:

```
1
2
3
4
5
6
7
```

Для визначення наявності елементу в множині використовують наступний синтаксис.

Наприклад:

```
print(3 in numb4)
print(8 in numb4)
```

Результат:

```
True
False
```

Для додавання елемента до множини використовують метод *ADD*, який додає вказаний елемент в дужках в кінець множини. Для видалення елементів з множин існує декілька методів:

1. *DISCARD* – видалення елемента без повідомлення при його відсутності, тобто, якщо вказаного елементу немає в множині метод спрацює та не виведе помилку.
2. *REMOVE* – видаляє елемент з повідомленням, якщо він відсутній виводить помилку.
3. *POP* – виведення останнього елементу не відсортованої множини.
4. *CLEAR* – очищення множини.

Розглянемо на прикладі.

Наприклад:

```
numb4.discard(59)
print(numb4)
```

Результат:

```
{1, 2, 3, 4, 5, 6, 7, 58}
```

Наприклад:

```
numb4.remove(58)
print(numb4)
```

Результат:

```
{1, 2, 3, 4, 5, 6, 7}
```

Наприклад:

```
numb4.pop()
print(numb4)
```

Результат:

```
{2, 3, 4, 5, 6, 7}
```

Наприклад:

```
numb4.clear()
print(numb4)
```

Результат:

```
set()
```

З множинами також можна проводити сумування, віднімання, переріз.

Для сумування використовується модуль *UNION* або ж знак |. Об'єднання відбувається не простим шляхом, адже елементи в множині не можуть повторюватися, а отже до першої множини додаються всі елементи другої, які не тотожні елементам першої множини. Розглянемо на прикладі.

Наприклад:

```
numb = {1, 2, 3, 77, 5}
numb4 = {1, 2, 3, 4, 5, 6, 7}
numb5 = numb.union((numb4))
print(numb5)
```

Результат:

```
{1, 2, 3, 4, 5, 6, 7, 77}
```

Наприклад:

```
numb6 = numb | numb4
print(numb6)
```

```
{1, 2, 3, 4, 5, 6, 7, 77}
```

Для перерізу множини використовуємо метод *INTERSECTION* або &. Це означає, що результатом будуть тільки ті значення, що є спільними для обох множин. Розглянемо на прикладі.

Наприклад:

```
numb7 = numb.intersection(numb4)
print(numb7)
```

Результат:

```
{1, 2, 3, 5}
```

Наприклад:

```
numb8 = numb & numb4
print(numb8)
```

```
{1, 2, 3, 5}
```

Множини можна віднімати одна від одної. В результаті отримаємо елементи ті елементи першої множини, що відмінні від другої.

Також значення однієї множини можна копіювати в другу за допомогою метода *COPY*, або ж раніше вивченим чином дізнаватися кількість елементів в множині. Розглянемо на прикладі.

Наприклад:

```
numb9 = numb - numb4
print(numb9)
```

Результат:

Наприклад:

```
numb10 = numb.copy()
print(numb10)
```

```
{77}
{1, 2, 3, 5, 77}
```

Наприклад:

```
print(len(numb10))
```

```
5
```

4.3. Програма роботи

1. Ознайомитися з теоретичними відомостями, що до списків, кортежів, словників та множин.
2. Виконати ознайомчі завдання теоретичних відомостей.
3. Виконати завдання згідно вашого варіанту. Додати коментарі до коду.
4. Оформити звіт.

4.4. Порядок виконання роботи

Завдання 1

Зі списку згідно варіанту вивести всі числа без повторів.

Варіанти:

1	9, 5, 2, 13, 8, 14, 5, 10, 13, 13, 4, 2, 3, 5, 9	6	12, 3, 6, 13, 6, 6, 10, 10, 11, 11, 3, 6, 12, 4, 8
2	9, 7, 14, 5, 12, 6, 13, 7, 14, 3, 10, 13, 9, 8, 1	7	6, 9, 10, 14, 13, 13, 7, 3, 11, 7, 4, 11, 3, 9, 2
3	2, 6, 3, 2, 3, 9, 1, 14, 1, 6, 11, 4, 12, 3, 8	8	13, 14, 12, 9, 2, 6, 10, 8, 8, 9, 11, 9, 12, 3, 8
4	14, 3, 2, 1, 13, 1, 2, 12, 10, 7, 2, 10, 1, 4, 9	9	3, 6, 7, 1, 4, 1, 13, 3, 1, 11, 4, 6, 2, 12, 8
5	8, 4, 13, 11, 10, 11, 6, 1, 3, 4, 8, 2, 12, 9, 11	10	3, 3, 3, 9, 8, 8, 2, 11, 14, 4, 7, 14, 7, 2, 6

Завдання 2

Знайти всі числа менші N та замінити їх значеннями K після чого вивести отриманий відсортований список.

Умови:

- список повинен вводитись із клавіатури (згідно варіантів завдання 1) та після кожного введеного числа запитувати про продовження введення;
- N та K водиться із клавіатури.

Завдання 3

Знайти максимальне та мінімальне значення елемента в кортежі, обраного відповідно до варіанту, та вивести його довжину.

Варіанти:

1	12, 7, 8, 9, 11, 5, 4, 3, 7, 10, 12, 13, 13, 4, 10	6	9, 10, 11, 8, 7, 6, 12, 13, 7, 13, 1, 13, 12, 2, 12
2	11, 8, 4, 4, 5, 5, 10, 11, 5, 14, 4, 3, 3, 5, 2	7	6, 12, 1, 6, 12, 7, 13, 4, 8, 11, 4, 9, 10, 8, 2
3	8, 12, 13, 11, 13, 3, 7, 3, 11, 3, 1, 5, 3, 4, 4	8	14, 6, 11, 14, 12, 13, 5, 6, 5, 7, 4, 1, 4, 2, 12
4	10, 14, 4, 7, 10, 13, 5, 13, 10, 8, 3, 11, 14, 2, 7	9	9, 8, 13, 5, 11, 5, 13, 1, 6, 7, 7, 3, 3, 6, 5
5	9, 14, 4, 11, 7, 1, 6, 11, 10, 4, 13, 3, 2, 7, 12	10	10, 8, 9, 12, 12, 11, 14, 11, 14, 6, 12, 4, 1, 8, 9

Завдання 4

Створити словник в якому ключ – це ім’я студента, а значення – це словник, в якому ключем виступає назва предмету, а значення – це семестровий бал студента з певного предмету.

Визначити:

- Рейтингові бали студентів, які розраховуються як середнє арифметичне балів за всі предмети, що помножено на 0.8
- Визначити найгіршого та найкращого студента із кожного предмету та вивести їх на екран.

4.5. Контрольні запитання

1. Що таке список?
2. В чому переваги списків?
3. Назвіть основні методи які використовуються для роботи з списками.
4. Наведіть приклади списків та роботу з ними.
5. Що таке кортеж?
6. Яка різниця між кортежами та списками?
7. Перетворення кортежів.
8. Наведіть приклади кортежів та роботу з ними.
9. Що таке словник?
10. Що таке ключ?
11. Наведіть приклади словників та роботу з ними.
12. Наведіть приклади роботи з ключами.
13. Що таке множина?
14. Назвіть основні властивості множин.
15. Назвіть методи роботи з множинами.
16. Назвіть методи видалення елементів з множини. В чому їх особливості?
17. Наведіть приклади словників та роботи з ними.

Список літератури

1. Welcome to Python : веб сайт. URL: https://www.python.org/
2. Документація Python : веб сайт. URL: https://docs.python.org/2/
3. The Python Tutorial : веб сайт. URL: https://docs.python.org/3/tutorial/
4. Т. Гедис. Починаємо програмувати на Python: підручник. БХВ-Петербург, 4-е видання, 2019. 768 с. URL: https://fileskachat.com/getfile/69426_d9e0b302eb1cdcc520c14037fef1682b
5. М. Лутц. Вивчаємо Python, том 1: підручник. Діалектика, 5-е вид, 2020. 832 с. URL: https://balka-book.com/files/2019/09_02/10_41/u_files_store_3_1985456.zip
6. Е. Метиз. Пришвидшений курс Python: підручник. Видавництво Старого Лева, 2021. 600 с.
7. О. Швець. Занурення в патерни проектування : підручник. 2021. 393 с. URL: https://refactoring.guru/files/design-patterns-uk-demo.pdf
8. Присяжнюк, О. В. Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Програмування. Частина 2. Програмування мовою Python» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Автоматизація та комп'ютерно-інтегровані технології» спеціальностей 151 «Автоматизація та комп'ютерноінтегровані технології», 141 «Електроенергетика, електротехніка та електромеханіка» денної та заочної форм навчання: методичне забезпечення. Рівне : НУВГП. 2020, 165 с. URL: http://ep3.nuwm.edu.ua/17989/