

Міністерство освіти і науки України
Чернівецький національний університет
імені Юрія Федьковича



Основи програмування

Методичні рекомендації

до лабораторного практикуму

Чернівці 2021

УДК 004.4'23(076.5)
О-751

*Рекомендовано Вченою Радою
Інституту фізико-технічних та комп'ютерних наук
Чернівецького національного університету імені Юрія Федьковича
(Протокол № 9 від 24.09.2021)*

Укладач: Івашко В. В., канд. фіз.-мат. наук, асистент

Івашко В. В. Основи програмування: метод. реком. до лабор. практикуму. Чернівці : Чернівецький національний університет імені Юрія Федьковича, 2021. 64 с.

У методичній розробці наведено роботи лабораторного практикуму з предмета: «Основи програмування». Зміст робіт охоплює основні розділи курсу.

Для студентів технічних інститутів та факультетів за спеціальністю: «Метрологія та інформаційно-вимірювальна техніка».

УДК 004.4'23(076.5)
О-751

© Чернівецький національний університет
імені Юрія Федьковича, 2021

Зміст

Лабораторна робота № 1	
«Змінні та вбудовані функції Python».....	4
Лабораторна робота № 2	
«Розв’язування простих математичних задач на мові Python».....	8
Лабораторна робота № 3	
«Оператори порівняння та умовні вирази в Python».....	11
Лабораторна робота № 4	
«Оператори циклу в Python».....	19
Лабораторна робота № 5	
«Функціональне програмування в Python».....	23
Лабораторна робота № 6	
«Робота з фалами в Python».....	30
Лабораторна робота № 7	
«Робота зі рядками в Python».....	33
Лабораторна робота № 8	
«Робота зі списками в Python».....	42
Лабораторна робота № 9	
«Робота зі словниками в Python».....	49
Лабораторна робота № 10	
«Об’єктно-орієнтоване програмування в Python».....	54
Лабораторна робота № 11	
«Візуалізація даних в Python».....	57
Список рекомендованої літератури.....	64

Лабораторна робота № 1

«Змінні та вбудовані функції Python»

Мета роботи: ознайомитись зі мовою програмування Python, сформувати навички роботи зі змінними та вбудованими функціями Python.

Теоретичні відомості

Змінна (англ. variable) – це об'єкт програми, що має ім'я та значення. Змінні – це не що інше, як зарезервовані місця в пам'яті комп'ютера для зберігання значень. Тобто, коли ви створюєте змінну, ви резервуєте трохи місця в пам'яті на вашому комп'ютері.

Змінні Python не потребують явного оголошення для резервування місця в пам'яті. Оголошення відбувається автоматично, коли ви присвоюєте значення змінної. Знак рівності «=» (оператор присвоєння) використовується для присвоєння значень змінним.

Оператор – це спеціальний символ, який повідомляє транслятору про те, що потрібно виконати операцію з деякими операндами. Операнд ліворуч від оператора = це назва змінної, а операнд праворуч від оператора = значення, що зберігається у змінній. Кожне значення в Python називається об'єктом. Кожен об'єкт має певний тип.

У Python існує кілька типів даних. Сюди входять ціле число (на Python називається "int"), float, boolean (на Python називається "bool"), рядок (на Python називається "str"), список, кортеж, словник (на Python називається "dict").

Константа (стала) – це частина даних, що зберігає своє значення під час усього виконання програми. Її зміна не дозволяється.

Для запуску програми Python потрібен інтерпретатор.

Інтерпретатор – це програма чи технічні засоби, необхідні для виконання інших програм, вид транслятора, який здійснює пооператорну (покомандну або порядкову) обробку, перетворення у машинний код та виконання програми або запиту (на відміну від компілятора, який транслює у машинні коди всю програму без її виконання).

На основі типу даних змінної інтерпретатор виділяє пам'ять і вирішує, що можна зберігати в зарезервованій пам'яті. Як тільки ваша програма буде написана, і ви готові протестувати її, при відповідній команді інтерпретатор Python виконає вашу програму (див. рис. 1).



Рис.1. Інтерпретатор Python

Для написання комп'ютерних програм на будь-якій мові програмування потрібен редактор для їх набору. Мова Python за замовчуванням має власний спеціально розроблений для написання програм редактор, який називається IDLE (Integrated Development and Learning Environment) або інтегроване середовище розробки.

IDLE – це більше, ніж просто редактор. Він забезпечує відступи та виділення тексту, що є свого роду допомогою під час написання програм. Він також забезпечує спосіб запуску програми прямо з редактора (Python Shell).

Навчаючись програмувати і навіть досвідченим професіоналом, є корисним запускати програму за допомогою інструменту, який називається «налагоджувачем» (debugger). Налагоджувач дозволяє запустити програму, зупинити та перевірити її поточний стан, щоб допомогти краще зрозуміти, крок за кроком, що відбувається під час виконання вашої програми. Для цієї мети IDLE також вбудований налагоджувач (*Debug*).

Звичайно, існують інші IDLE, якими можна користуватись. Деякі приклади IDLE для розробки програм на мові Python включають Anaconda, Jupyter, та інші. Окремо можна також використовувати, такі редактори коду як Atom, Sublime Text, та інші.

Зверніть увагу! Ви можете використовувати звичайний «порожній» файл IDLE Python (*File/New File*) або блокнот Windows (як альтернатива), щоби написати код, проте написаний вами код необхідно зберегти, як файл зі розширенням «.py», щоби в подальшому IDLE Python зміг його виконати (клавіша F5 або *Run/Run module*).

У цій лабораторній роботі та в усіх наступних роботах ми будемо використовувати IDLE Python (рис. 2).

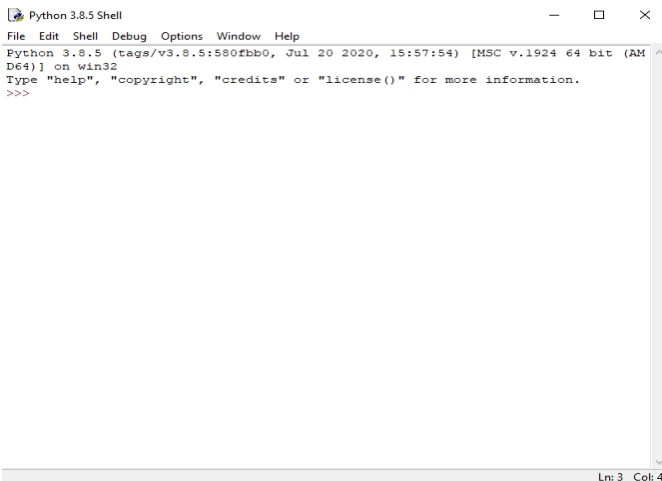


Рис. 2. Інтегроване середовище розробки Python (IDLE)

Інтерпретатор Python має ряд стандартних функцій, які завжди доступні для використання. Ці функції вбудовані в мову Python, які можна використовувати без підключення будь-яких додаткових модулів.

Наприклад, функція `print()` друкує даний об'єкт на стандартному пристрої виводу (екрані) або у файлі текстового потоку, `input()` – ввід даних.

Існує також ряд інших важливих вбудованих функцій, які будуть розглянуті в подальшому.

Хід роботи

1. Напишемо найпростішу програму: «привітання». Для цього запустіть IDLE Python та в рядку консолі (`>>>`) введіть наступний код і натисніть клавішу *Enter*:

```
>>> print("Привіт Світ!")
```

Результат:

```
>>> print("Привіт Світ!")
Привіт Світ!
>>> |
```

Зверніть увагу! Такий спосіб вводу коду є зручним в тому випадку, коли необхідно здійснити перевірку (тестування) незначного об'єму коду (в декілька рядків). Всі набрані символи в подвійних (або одинарних) лапках, приймають тип даних рядок (`str`).

Фактично ми здійснили ввід коду «на пряму» через консоль IDLE і одразу отримали результат. Проте з часом наші програми будуть становитись дедалі складнішими і громіздкими. Написання коду в такому випадку через консоль IDLE не годиться.

2. Існує інший, більш поширеніший спосіб написання коду, за допомоги редактора коду. Для цього необхідно в меню IDLE натиснути на підменю *File* та обрати вкладку *New File*.

Перед вами відкриється новий («чистий») файл редактора коду IDLE. Наберіть попередньо набраний вами код в консолі в даний файл та збережіть його (*File/Save As*) обов'язково вказавши в кінці імені файлу розширення «.ру».

Для виконання вашого коду натисніть клавішу `F5`, або в меню редактора коду IDLE оберіть *Run/Run module*.

Результат:

```
Привіт Світ!
>>>
```

Отже, код вашої програми зберігся в окремому файлі зі розширенням «.ру», а результат подається в консолі IDLE.

3. Наберіть та запустіть на виконання наступний код:

```
a = input("Введіть ваше ім'я: ")
b = input("Введіть ваше прізвище: ")
print("Привіт!", a, b)
```

4. Зробіть висновок отриманого результату.

Зверніть увагу! Всі дані (символи зі клавіатури) при вводі за допомогою функції `input()` отримують тип даних рядок (`str`). За допомогою символу коми «,» функції `print()` можна виводити декілька даних (рядок, змінна, константа та інші) в одному рядку екрана.

Контрольні питання

1. Що таке змінна та константа?
2. Що таке оператор та операнд?
3. Що таке інтерпретатор?
4. Що таке IDLE? Для чого він служить?
5. Для чого потрібен "налагоджувач" (debugger)?
6. Які стандартні вбудовані функції Python ви знаєте? Опишіть їх.

Лабораторна робота № 2

«Розв'язування простих математичних задач на мові Python»

Мета роботи: ознайомитись з математичним модулем «math» та його стандартним набором функцій. Сформувати навички використання модуля «math» для вирішення простих математичних задач в Python.

Теоретичні відомості

Якщо код програми є складним, то його краще розбити на окремі блоки, що в свою чергу, сприяє кращому візуальному сприйняттю. Якщо цього недостатньо, тоді є зміст винести частину коду та пов'язані з ним оголошення за межі основного файлу програми.

Такі додаткові файли з кодом, які часто використовується в програмі, називаються модулями. Модуль представляє собою колекцію компонентів. Тобто, модуль – це простір імен «namespace», при чому всі імена всередині модуля називаються атрибутами (функції та змінні).

Найчастіше модулі містять оголошення функцій та констант, які далі можуть бути підключені (імпортовані) в основний код програми та вільно в ньому використовуватися.

Великі за розміром програми, як правило, складаються з стартового (головного) файлу, тобто файлу верхнього рівня, та набору файлів-модулів. Головний файл займається контролем програми.

Існує велика кількість вбудованих модулів, які дозволяють виконувати складні математичні операції (math), працювати з датами (datetime) / часом (time), випадковими числами (random), операційною та файловою системами (os) та інші. Модулі, також можуть імпортувати інші модулі.

Модуль математики math завжди доступний. Він забезпечує доступ до базових математичних функцій, визначених стандартом мови C. Модуль math також надає додаткові функції для роботи з числами, а також стандартні константи.

Для того, щоби використовувати модуль, необхідно спочатку підключити його за допомогою інструкції «import»:

```
import math
```

Модуль math надає наступні стандартні константи:

- π – повертає число π ;
- e – повертає значення константи e ;
- та інші.

Основні функції для роботи з числами:

- $\sin()$, $\cos()$, $\tan()$ – стандартні тригонометричні функції (синус, косинус, тангенс). Значення вказується в радіанах;

- `asin()`, `acos()`, `atan()` – зворотні тригонометричні функції (арксинус, арккосинус, арктангенс). Значення повертається в радіанах;
- `degrees()` – перетворює радіани в градуси;
- `radians()` – перетворює градуси в радіани;
- `exp()` – експонента;
- `log()` – логарифм;
- `sqrt()` – квадратний корінь;
- `ceil()` – значення, округлене до найближчого більшого цілого;
- `floor()` – значення, округлене до найближчого меншого цілого;
- `pow(Число, Степень)` – піднесення числа до степені;
- `fabs()` – абсолютне значення;
- `fmod()` – остача від ділення;
- `factorial()` – факторіал числа;
- та інші.

Зверніть увагу! В якості аргументів тригонометричних функцій слід використовувати радіани. При обчисленні виразів в Python всі арифметичні оператори виконуються «зліва» на «право» від оператора присвоєння (`=`), причому у відповідності до їх пріоритету.

Хід роботи

1. Обчислимо арифметичний вираз, використовуючи математичний модуль `math` Python.

Для цього розглянемо наступний приклад:

$$\left(\frac{34 + 6^2 \cdot \sqrt{225}}{\cos(60^\circ) \cdot 14} \right)^2$$

2. Код програми має вигляд:

```
import math
a = 34; b = math.pow(6,2); c = math.sqrt(225);
d = math.radians(60); e = math.cos(d); f = 14;
g = a + b*c
l = e*f
m = (g/l)
res = math.pow(m,2)
print ("res = ", res)
```

3. У результаті виконання інтерпретатором Python даного коду, ми отримаємо:

```
res = 6723.999999999997
```

4. Контрольне завдання: напишіть програму, яка реалізує обчислення арифметичного виразу та розв'язок квадратного рівняння у відповідності до варіанту завдання (порядковий номер в журналі групи).

5. Також, накресліть алгоритм (блок-схему) до вашої програми .

Табл. 1

№	Завдання
1	$\left(\frac{5+\sqrt{44}-4}{\sin(30^\circ)}\right)^3, \quad x^2-6x+5=0$
2	$\left(\frac{\sqrt{121}}{5}\right)^2 + \frac{\tan(45^\circ)}{2}, \quad 2x^2-9x+10=0$
3	$(24)^3 - \sqrt{1024} + \left(\frac{521}{3}\right)^2, \quad 6x^2-7x=0$
4	$1500 - \left(\frac{\sqrt{500}}{2,5}\right) \cdot \left(\frac{1}{\sqrt{3}}\right)^2, \quad 3x - x^2 = 0$
5	$\cos^2(60^\circ) \cdot \sin(30^\circ) + 4, \quad 9x^2 + 5x = 0$
6	$\left(\frac{\tan(45^\circ)}{3^4-5}\right) \cdot \sqrt{88}, \quad x^2-6x+5=0$
7	$5 \cdot \cos(45^\circ) + \frac{\sqrt{400}}{3}, \quad x^2+5x+3=0$
8	$(\sqrt{44}-3) \cdot 24 + 3^5, \quad 9x^2-24x+16=0$
9	$\left(\frac{4}{3} - 27 \cdot \sqrt{34}\right) \cdot \cos(45^\circ), \quad 3x^2+x=0$
10	$5 \cdot \cot(30^\circ) + 4 \cdot \sqrt{5}, \quad 3x^2-9=0$
11	$\left(\frac{3^4}{5} - \frac{200}{4^3}\right) + \left(\frac{25}{\sqrt{5}}\right), \quad 3x^2-9x=0$
12	$(12^4 - 12^2) \cdot \sqrt{44}, \quad 4x^2-7x+2=0$
13	$\frac{\sin(60^\circ)}{\sqrt{8}} - \left(\frac{400}{11^2}\right)^2, \quad x^2-7x+2=0$
14	$\frac{\frac{5^4}{\sqrt{24}} + \frac{3^3}{4}}{\cos(45^\circ)}, \quad x^2-5x+1=0$

15	$\frac{\sqrt{\frac{6^3}{2} + \frac{2^5}{3}}}{\sin(50^\circ)}, \quad x^2 - 10x + 2 = 0$
16	$(\frac{\sqrt{11}}{3^4} + \sqrt{\frac{2^3}{4}}) \cdot \tan(50^\circ), \quad 2x^2 - x - 3 = 0$
17	$(\sqrt{6^3} + 12^2 \cdot \sqrt{\cos(35^\circ)}) \cdot 22, \quad 2x^2 - 3x + 5 = 0$
18	$22^4 - \sqrt{3^3} \cdot 2^3 + \tan(35^\circ), \quad x^2 - 9x + 14 = 0$

Контрольні питання

1. Що таке модуль?
2. Яким чином здійснюється підключення модуля до програми в Python?
3. Що таке модуль math, та які стандартні функції даного модуля ви знаєте (назвіть декілька та коротко опишіть їх)?
4. Які значення, слід використовувати в якості аргументів тригонометричних функцій модуля math в Python?
5. За яким правилом здійснюється обчислення виразів в Python?

Лабораторна робота № 3

«Оператори порівняння та умовні вирази в Python»

Мета роботи: навчитись за допомоги операторів порівняння та логічних операцій реалізовувати розгалужені процеси на мові Python.

Теоретичні відомості

При розробці логічної основа вашої комп'ютерної програми зі використання мови Python одним з необхідних елементів є умовні оператори. Наприклад, при виконанні різного роду складних математичних операцій, які вимагають паралельної перевірки декількох умов.

Будь-який логічний (булевий) вираз по суті є умовою, розрахунок якої завжди дає логічне значення, тобто істина (True) або хибність (False). В залежності від умови, програма буде виконувати або пропускати фрагменти, або блоки коду. Іншими словами, має місце розгалуження. Засоби мови Python дозволяють реалізувати розгалужені процеси.

Збільшення відступу від лівого краю коду є ознакою блоку коду. Такі блоки, можуть також містити в собі інші блоки. Зменшення відступу до нуля або до величини відступу зовнішнього блоку є ознакою кінця блоку коду.

Зверніть увагу! Для входження в блок коду необхідно набрати 4 пробіли. Для входження в наступний блок коду, наступні 4 пробіли і так далі.

Як визначається істина або хибність, якщо елемент в умові відповідного оператора не є булевым типом? Наприклад, до хибності (False) прирівнюються наступні об'єкти (типи даних) та значення:

- 0;
- 0.0;
- Пустий рядок "";
- Пустий список [];
- Пустий кортеж ();
- Пустий словник {};
- Пуста множина set().
- False;
- None;

Всі інші об'єкти або значення прирівнюються до істини (True).

Оператори порівняння (логічні оператори) призначені для виконання логічних операцій над логічними даними, оголошеними в програмі за допомогою ключового слова або символу.

При побудові логічних виразів здебільшого застосовуються наступні операції порівняння (оператори):

Табл. 2

№	Оператори порівняння	Значення
1	==	дорівнює
2	!=	не дорівнює
3	>=	більше або рівне
4	>	більше
5	<=	менше або рівне
6	<	менше

Можливі випадки коли при побудові логічних виразів необхідно врахувати не одну а кілька умов. Мова Python передбачає використання спеціальних операцій побудови складених умовних операторів:

Табл. 3

№	Логічні оператори	Значення
1	and	логічне «І»
2	or	логічне «АБО»
3	not	логічне «НІ» (інверсія)

Структура умовного оператора if в Python має наступний вигляд:

```
if умова:
    блок коду
```

Конструкція блоку коду команди if дозволяє також використовувати необов'язкову команду else, яка характеризується власним блоком коду, який буде виконуватися лише в тому випадку, якщо умова if є хибною, тобто False. Ця команда має наступний синтаксис:

```
if умова:
    блок коду, коли умова істинна
else:
    блок коду, коли умова хибна
```

При необхідності перевірки більше ніж однієї умови використовують конструкцію elif (скорочено від else if). Ця інструкція використовується виключно після команди if або після іншої інструкції elif. Вона дозволяє задавати додаткові умови перевірки. Конструкція elif має наступний синтаксис:

```
if умова1:
    блок коду, коли умова1 істинна
elif умова2:
    блок коду, коли умова2 істинна
elif умова3:
    блок коду, коли умова3 істинна
...
```

При необхідності після останньої інструкції `elif` можна помістити додатково інструкцію `else`. В такому випадку, ми одержимо гарантію того, що хоча б один блок коду (всієї конструкції `if`) буде виконаний:

```
if умова1:
    блок коду, коли умова1 істинна
elif умова2:
    блок коду, коли умова2 істинна
elif умова3:
    блок коду, коли умова3 істинна
else:
    блок коду, коли усі умови хибні
```

Структурний елемент `else` умовного оператора `if` у більшості випадках використовується для оптимізації програмного коду та удосконалення його читабельності.

Слід зазначити, що коли варіанти перебору умовного оператора `if` є взаємовиключними, використання оператора `else` дозволяє скоротити об'єм перевірок, і як результат суттєво пришвидшити роботу програми.

Зверніть увагу! У випадку коли варіанти перебору не доповнюють один одного, використання структурного елемента `else` може спровокувати виникнення помилки. Або можливий випадок, коли варіант умови взагалі не буде розглянутий.

При виникненні помилки або винятку в блоці коду Python зазвичай зупиняє поточне виконання програми і генерує повідомлення про помилку. Проте, ці винятки (помилки) можна обійти за допомогою оператора `try/except`:

```
try:
    блок коду, якщо помилка в даному блоці не виникає
except:
    блок коду, якщо виникла помилка блоці коду try
```

Конструкція try працює наступним чином:

- Спочатку виконуються вирази, які записані в блоці try;
- Якщо при виконання блоку try не виникло ніяких винятків, блок except пропускається і виконується подальший код;
- Якщо під час виконання блоку try, в якомусь місці виникло виключення, решта блоку try пропускається;
- Якщо в блоці except вказано виняток, який може виникнути, виконується код в блоці except;
- Якщо виключення, яке виникло, не вказано в блоці except, виконання програми переривається і видається помилка.

Хід роботи

1. Побудуємо за допомоги Python розгалужений процес виду:

$$f = \begin{cases} (x+2) - (y+z), & x \geq 0; \\ 3(x^2+z)/y, & y \neq 0 \text{ та } z > 0; \\ 3(x^2+y)/z, & y = 0 \text{ та } z > 0; \\ (x+y)*z - (x+z), & \text{інакше.} \end{cases}$$

2. Код програми має наступний вигляд:

```
print("Створення розгалуженого процесу")
try:
    x = float(input("Введіть значення x = "))
    y = float(input("Введіть значення y = "))
    z = float(input("Введіть значення z = "))
except:
    print("Помилка при вводі, введіть число!")
    exit()
if x >= 0:
    f = (x+2) - (y+z)
elif (y != 0) and (z > 0):
    f = 3*(x*x+z)/y
elif (y == 0) and (z > 0):
    f = 3*(x*x+y)/z
else:
    f = (x+y)*z - (x+z)
print("Результат f = ", f)
```

Зверніть увагу! В коді вище використано додатково дві вбудованих функцій – float() та exit(). Перша виконує перетворення типу даних, а саме перетворює тип даних рядок (str) в числа з рухомою комою (float), друга виконує функцію завершення програми.

3. У результаті виконання інтерпретатором Python даного коду, ми отримаємо:

```
Створення розгалуженого процесу
Введіть значення x = 0
Введіть значення y = Віктор
Помилка при вводиті, введіть число!
```

```
Створення розгалуженого процесу
Введіть значення x = -1
Введіть значення y = 0
Введіть значення z = 3
Результат f = 1.0
```

```
Створення розгалуженого процесу
Введіть значення x = 1
Введіть значення y = 2
Введіть значення z = 3
Результат f = -2.0
```

```
Створення розгалуженого процесу
Введіть значення x = -1
Введіть значення y = 0
Введіть значення z = -3
Результат f = 7.0
```

```
Створення розгалуженого процесу
Введіть значення x = -1
Введіть значення y = 2
Введіть значення z = 3
Результат f = 6.0
```

4. Контрольне завдання: напишіть програму, яка реалізує розгалужений процес у відповідності до варіанту завдання (порядковий номер в журналі групи).

5. Також, накресліть алгоритм (блок-схему) до вашої програми.

Табл. 4

№	Завдання
1	$f = \begin{cases} x + y, xy > 0; \\ x, x^2 > 100 \text{ та } y = 0; \\ x^2 + y, \text{інакше.} \end{cases}$
2	$f = \begin{cases} xy - 1, x/y > 0; \\ 2xy/3, x \neq 0 \text{ та } y \neq 3; \\ x + y, \text{інакше.} \end{cases}$
3	$f = \begin{cases} x + y, x < 10 \text{ або } y > 3; \\ x - y + x^2, x > 10 \text{ та } y < 3; \\ x + y^2, \text{інакше.} \end{cases}$
4	$f = \begin{cases} (x + x)/y, y \geq 1; \\ y + (x^2 - y), x > 2 \text{ або } y < 1; \\ x + 3xy, \text{інакше.} \end{cases}$
5	$f = \begin{cases} y^2 - (x + y)/x, x \neq 0 \text{ та } y > -3; \\ (y^2 - 20)/2, y \geq 10; \\ (y + 4) * 3, \text{інакше.} \end{cases}$

6	$f = \begin{cases} x * y - (x + y), x = 4 \text{ або } y = 2; \\ (y - x) + (x * y), y > 0; \\ 10 + (x - y), \text{інакше.} \end{cases}$
7	$f = \begin{cases} 2x + y, x > 0 \text{ та } y > 0; \\ x^2 + 2y^2, x < 3 \text{ та } y > 4; \\ x^3 + y, \text{інакше.} \end{cases}$
8	$f = \begin{cases} x + y + 6, x \geq 2; \\ (y - x)(-1), y < 0 \text{ та } x < 2; \\ (y + x)/2, \text{інакше.} \end{cases}$
9	$f = \begin{cases} (x + y)/x, x \neq 0 \text{ та } y > 0; \\ y^2 - 10 + x, y < 0; \\ (x - y)(y - x), \text{інакше.} \end{cases}$
10	$f = \begin{cases} xy - x + y, x + y > 5; \\ x/y + x - y, y \neq 0 \text{ та } x + y \leq 5; \\ x^2 - 4x, \text{інакше.} \end{cases}$
11	$f = \begin{cases} (x + 5)/2 + y, x > 4 \text{ або } y > 0; \\ (25 - y) - (x + 3), x \leq 4 \text{ та } y = 10; \\ xy - 5, \text{інакше.} \end{cases}$
12	$f = \begin{cases} x + 6 - y, x = 4 \text{ та } y > 0; \\ xy + 2, x < 4; \\ x^3 - 3, \text{інакше.} \end{cases}$
13	$f = \begin{cases} x^2 + y^2, x < 5 \text{ або } y > 2; \\ xy - x + y, x \geq 5 \text{ та } y \neq 2; \\ x + 4xy - 2, \text{інакше.} \end{cases}$
14	$f = \begin{cases} 2x^2 - 3y^2, x < 10 \text{ та } y \neq 3; \\ 2x - 3y/x, x \neq 0 \text{ та } x \neq 2; \\ x^2 + 3xy, \text{інакше.} \end{cases}$
15	$f = \begin{cases} x^2 + (y - x), y > 5; \\ (x + y)/(x - y), x \neq 0 \text{ або } y < 5; \\ x - y, \text{інакше.} \end{cases}$

16	$f = \begin{cases} 2(x/y) + xy^2, x > 0 \text{ та } y \neq 0; \\ 2/2x, (2x + y)/2 > 0; \\ x^2, \text{інакше.} \end{cases}$
17	$f = \begin{cases} x - y^2, x - y > 2; \\ x/y, x > 0 \text{ та } y \neq 0; \\ x^2 + 3, \text{інакше.} \end{cases}$
18	$f = \begin{cases} x - 2y^2, x - 3y \neq 2; \\ (x - y)/y, x > 0 \text{ та } y \neq 0; \\ 2x^2 - 3y, \text{інакше.} \end{cases}$

Контрольні питання

1. Для чого потрібні розгалужені процеси в програмуванні?
2. Як здійснити вхід в блок коду в Python?
3. Які оператори порівняння Python ви знаєте? Опишіть їх.
4. Які логічні оператори використовуються при побудові складених умовних операторів?
5. Які конструкції оператора умови if використовуються в Python? Опишіть їх.
6. Для чого потрібний оператор try/except в Python?

Лабораторна робота № 4

«Оператори циклу в Python»

Мета роботи: ознайомитись зі основними типами циклів в Python та сформуванати навички роботи зі ними.

Теоретичні відомості

Поширене використання операторів циклу при програмуванні зумовлено не тільки програмуванням математичних алгоритмів, але й роботами з базами даних, створення інтерактивних динамічних інтерфейсів тощо.

В Python використовуються оператори циклу двох типів : while та for. Синтаксис використання оператора циклу while наступний:
while [умова]:

блок коду (який виконується, якщо умова істинна).

В Python оператор while складається з наступний елементів:

- Власне слово while;
- Умова або вираз, результатом якого є значення: True або False;
- Двокрапка;
- Новий рядок та блок коду (з відступом).

Для прикладу, при досягненні кінця блоку коду оператора if, керування виконанням передається наступній команді, а при досягненні кінця блоку коду оператора while, керування передається на початок циклу для звірки умови. Якщо умова істинна, то програма продовжує виконувати той самий блок, якщо хибна, то відбувається вихід з циклу і передача виконання коду на наступний розташований після циклу while рядок.

Розглянемо блок коду:

```
>>> count = 1
>>> while count <= 5:
    print(count)
    count += 1 # count = count + 1

1
2
3
4
5
```

Отже, цикл while буде виконуватися до тих пір, поки умова залишається істинною (True).

Якщо виникає потреба необхідно виконати блок коду лише визначено число разів, то використовують цикл for разом зі функцією range(). Синтаксис використання оператора циклу for виглядає наступним чином:

for [змінна циклу] in [range()]:

блок коду (який виконується визначене число разів).

В Python оператор for складається з наступних елементів:

- Власне слово for;
- Локальна змінна циклу;
- Ключове слово in;
- Функція range() (необов'язкова) або послідовна структура даних;
- Двокрапка;
- Новий рядок та блок коду (з відступом).

Щоб зрозуміти, як працює оператор цикл for, розглянемо блок коду:

```
>>> for i in range(5):  
        print("My name is Rick (" + str(i) + ")")  
  
My name is Rick (0)  
My name is Rick (1)  
My name is Rick (2)  
My name is Rick (3)  
My name is Rick (4)
```

Як видно тіло циклу оператор for виконується 5 разів. На першій ітерації значення змінної циклу (i) встановлюється рівним 0, потім виклик функції print() виводить відповідне повідомлення.

Коли цикл закінчує ітерацію виконавши увесь блок керування передається на початок циклу, де відбувається збільшує значення змінної циклу на 1.

Таким чином, виклик функції range(5) забезпечує п'ятикратне виконання блоку коду циклу for, встановлюючи для своєї змінної (i) послідовно значення: 0, 1, 2, 3 та 4.

В принципі, все те що здатен виконати цикл for, можна зробити і за допомогою while.

У функцію range() можна передати до трьох аргументів (цілі числа) розділених комами.

Перше число задає значення, з якого починає змінюватися змінна циклу for, друге число вказує на значення, до якого змінюється змінна циклу for (причому не включно) і третє число задає крок.

В циклі for функція range(), є не обов'язковою. Іншими словами, даний цикл дозволяє здійснювати ітерації проходячи по різним послідовним структурам даних. При цьому змінна циклу приймає значення елементів послідовних структур, по яким вона і проходиться.

Розглянемо приклад використання циклу for при проходженні елементів списку:

```
>>> animals = ["dog", "cat", "mouse", "spider", "ant"]
>>> for animal in animals:
    print(animal)

dog
cat
mouse
spider
ant
```

Хід роботи

1. Обчисліть суму, використовуючи при цьому оператор циклу while Python, для цього розглянемо наступний приклад:

$$\sum_{i=1}^{10} (-1)^i \cdot \frac{(x+y)i!}{(i-1)!}, \quad \text{для } x = 1 \text{ та } y = 2.$$

2. Код програми має вигляд:

```
# Програма розрахунку суми
import math
try:
    x = int(input("Введіть значення x = "))
    y = int(input("Введіть значення y = "))
    n = int(input("Введіть число членів суми n = "))
except:
    print("Помилка при вводі, введіть число!")
    exit()
count = 1
res = 0
while count <= n:
    m = pow(-1, count) * ( (x+y)*math.factorial(count)/math.factorial(count-1) )
    res = res + m
    count += 1
print("Результат = ", res)
```

Зверніть увагу! В прикладі вище використано додатково дві вбудованих функцій – int() та exit(). Перша виконує перетворення типу даних, а саме перетворює тип даних рядок (str) в цілі числа (int), друга виконує функцію завершення програми.

3. У результаті виконання інтерпретатором Python даного коду, ми одержимо:

```
Введіть значення x = 1
Введіть значення y = 2
Введіть число членів суми n = 10
Результат = 15.0
```

4. Контрольне завдання: напишіть програму, яка реалізує обчислення суми за допомоги оператора циклу for у відповідності до варіанту завдання (порядковий номер в журналі групи).

Зауваження! Не дозволяється використовувати функцію sum(), модуля math.

5. Також, накресліть алгоритм (блок-схему) до вашої програми.

Табл. 5

№	Завдання
1	$\sum_{i=5}^{10} (-1)^i \cdot \frac{x \cdot i!}{y + (i-1)!}, \text{ для } x = 1 \text{ та } y = 2$
2	$\sum_{i=2}^8 (-1)^i \cdot \frac{x \cdot (i-1)!}{y + i!}, \text{ для } x = 1 \text{ та } y = 3$
3	$\sum_{i=3}^9 (-1)^i \cdot \frac{x^{(i-1)} - y^{(i+1)}}{(i-1)!}, \text{ для } x = 2 \text{ та } y = 2$
4	$\sum_{i=2}^6 (-1)^i \cdot \frac{x \cdot (i+1)}{y + (i-1)!}, \text{ для } x = 2 \text{ та } y = 3$
5	$\sum_{i=2}^7 (-1)^i \cdot \frac{x \cdot (i+1)}{y + i!}, \text{ для } x = 2 \text{ та } y = 2$
6	$\sum_{i=3}^8 (-1)^i \cdot \frac{(x+i)^2 + (i-1)!}{(y+i)!}, \text{ для } x = 2 \text{ та } y = 1$
7	$\sum_{i=2}^7 (-1)^i \cdot \frac{x^{(i+1)} + y^{(i-1)}}{(i-1)!}, \text{ для } x = 1 \text{ та } y = 2$
8	$\sum_{i=5}^9 (-1)^i \cdot \frac{y}{x} \cdot \frac{(i+1)!}{(i-1)!}, \text{ для } x = 2 \text{ та } y = 3$
9	$\sum_{i=2}^5 (-1)^i \cdot \frac{x^{(y+i)} + i!}{(i+1)!}, \text{ для } x = 2 \text{ та } y = 1$
10	$\sum_{i=4}^8 (-1)^i \cdot \frac{(x+y) \cdot (i-1)!}{(x-y) \cdot (i+1)!}, \text{ для } x = 3 \text{ та } y = 1$
11	$\sum_{i=2}^6 (-1)^i \cdot \frac{x}{y + (i-1)!}, \text{ для } x = 4 \text{ та } y = 1$

12	$\sum_{i=6}^{10} (-1)^i \cdot \frac{(x/y)!}{(i+1)!}, \text{ для } x = 2 \text{ та } y = 4$
13	$\sum_{i=3}^7 (-1)^i \cdot \frac{(x-y) \cdot (i-1)!}{(x+y)!}, \text{ для } x = 3 \text{ та } y = 2$
14	$\sum_{i=6}^{11} (-1)^i \cdot \frac{(x \cdot y)^{(i-1)}}{(x+y)} \cdot \frac{i!}{(i+1)!}, \text{ для } x = 4 \text{ та } y = 2$
15	$\sum_{i=2}^6 (-1)^i \cdot \frac{y^{(x+i)}}{(i+1)!}, \text{ для } x = 1 \text{ та } y = 3$
16	$\sum_{i=4}^9 (-1)^i \cdot \frac{(x+i)!}{(y+i)! + (i-1)!}, \text{ для } x = 2 \text{ та } y = 1$
17	$\sum_{i=3}^8 (-1)^i \cdot \frac{(x+y)^2}{(i+1)!}, \text{ для } x = 4 \text{ та } y = 2$
18	$\sum_{i=7}^{10} (-1)^i \cdot \frac{(x-i)!}{(y+i)!}, \text{ для } x = 2 \text{ та } y = 1$

Контрольні питання

1. Для чого потрібні оператори циклу в програмуванні?
2. Який синтаксис використання оператора циклу while в Python?
3. Який синтаксис використання оператора циклу for в Python?
4. Для чого служить функція range(). Опишіть її складові.
5. В чому полягає основна відмінність оператора циклу for від оператора циклу while в Python?

Лабораторна робота № 5

«Функціональне програмування в Python»

Мета роботи: сформувати навички роботи зі функціями в Python.

Теоретичні відомості

Функції в Python представляють собою об'єкти, які приймають певні аргументи (або не приймають) та повертають значення. Функції в Python визначається за допомогою інструкції або ключового слова `def`.

Розглянемо найпростішу функцію (сума двох аргументів):

```
>>> def add(x, y):  
        return x + y
```

Команда `return` вказує на те, що потрібно повернути певне значення. В даному випадку функція повертає значення суми ($x + y$).

Функції в Python можуть бути довільної складності та повертати будь-які значення або об'єкти (списки, кортежі та навіть інші функції).

Також, функція може приймати будь-яке число аргументів чи не приймати їх взагалі. Найбільш поширеними є функції з довільним числом аргументів, з позиційними та іменованими аргументами, необов'язковими та обов'язковими аргументами.

```
>>> def fn(a, b, c=2): # c - необов'язковий аргумент  
        return a + b + c  
  
>>> fn(1,2)  
5  
>>> fn(1,2,3)  
6  
>>> fn(a=1,b=3)  
6  
>>> fn(a=3,c=6)  
Traceback (most recent call last):  
  File "<pyshell#24>", line 1, in <module>  
    fn(a=3,c=6)  
TypeError: fn() missing 1 required positional argument: 'b'
```

Функції також можуть приймати змінну кількість позиційних аргументів. В такому випадку, перед аргументом ставиться символ `*`:

```
>>> def fn(*arg):  
        return arg  
  
>>> fn(1,2,3,'abcd')  
(1, 2, 3, 'abcd')  
>>> fn(1)  
(1,)
```


Як видно з прикладу, результатом виклику функції є кортеж (тип даних) з усіма переданими аргументами, з яким можна працювати так само як зі змінною.

Існує і інший спосіб задати функції. Це здійснюється за допомогою так званих анонімних функцій, які містять лише один вислів (вираз), але при цьому вони виконуються набагато швидше.

Анонімні функції створюються за допомогою інструкції `lambda`. Окрім цього, їх також не обов'язково привласнювати певній змінній, як наприклад при роботі з інструкцією `def`.

```
>>> func = lambda x, y: x + y
>>> func(1, 2) * 3
9
>>> func('a', 'b') + 'ab'
'abab'
>>> (lambda x, y: x + y)(1, 2) * 3
9
>>> (lambda x, y: x + y)('a', 'b') + 'ab'
'abab'
```

Анонімна функція `lambda`, на відміну від звичайної функції, не потрібна команда `return`:

```
>>> f = lambda x: x**2 + 4
```

Це, те ж саме, що:

```
>>> def f(x):
        return x**2 + 4
```

Будь-яку конструкцію виду:

```
>>> def g(arg1, arg2, arg3):
        return arg1 + arg2 + arg3
```

Можна записати через `lambda` функцію:

```
g = lambda arg1, arg2, arg3: arg1 + arg2 + arg3
```

`Lambda` функції дуже зручні в тому випадку, коли потрібно визначати невеликі функції, і тому є досить популярні серед багатьох програмістів. В основному їх використовують для швидкого визначення функції, як аргумент іншої функції.

В Python існують також, так звані, рядки документації (`Docstring`), які вставляються відразу після заголовку функції.

`Docstring` містить короткий опис функції та пояснення змісту її аргументів. Він розміщаються між потрійними лапками ("""" або """) з обох сторін, що дозволяє розбивати текст на кілька рядків. Розглянемо приклади використання `Docstring` у функціях, а точніше його короткий і довгий варіанти запису.

Зверніть увагу! Рядки документації повинні розташовуватися на початку тіла функції. `Docstring` це не просто коментарі, вони володіють більшими можливості. Наприклад для нижче описаної функції `Fn1()` лінії

підкреслення забезпечують виконання команди: `print(Fn1.__doc__)`. У результаті якої, виводиться весь зміст Docstring.

```
def Fn1(C):
    """Convert Celsius degrees (C) to Fahrenheit."""
    return (9.0/5)*C + 32

def Fn2(x0, y0, x1, y1):
    """
    Compute the coefficients a and b in the mathematical
    expression for a straight line y = a*x + b that goes
    through two points (x0, y0) and (x1, y1).
    x0, y0: a point on the line (floats).
    x1, y1: another point on the line (floats).
    return: coefficients a, b (floats) for the line (y=a*x+b).
    """
    a = (y1 - y0)/float(x1 - x0)
    b = y0 - a*x0
    return a, b
```

Якщо назва модуля є задовгою або вона взагалі не подобається, то для неї можна створити псевдонім, що здійснюється за допомоги ключового слова `as`.

```
>>> import math as m
>>> m.e
2.718281828459045
```

Тепер доступ до атрибутів, наприклад модуля `math`, буде здійснюватися тільки за допомогою імені `m`, а ім'я `math` в цій програмі вже не буде доступним.

Підключити декілька атрибутів одного модуля можна за допомогою інструкції `from`. Вона має два варіанти формату використання. Перший варіант формату:

```
>>> from <Назва модуля> import <Атрибут 1> [ as <Псевдонім 1> ], <Атрибут 2>
[ as <Псевдонім 2> ] ...
```

Другий варіант формату:

```
>>> from <Назва модуля> import *
```

Перший варіант дозволяє підключити з модуля тільки зазначені через кому атрибути (функції). Для довгих імен також можна вказати псевдоніми (за допомоги слова `as`). При другому варіанту формату запису модулів відбувається завантаження всіх атрибутів модуля.

```
>>> from math import e, ceil as c
>>> e
2.718281828459045
>>> c(4.6)*5
25
>>>
```

Створимо файл `my_module.py`, в якому визначимо наступну функцію:

```
def hello():
    print('Hello, world!')
def fun(n):
    a=0
    for i in range(n):
        a += 2
    return a
```

Тепер в цій же папці створимо новий файл, наприклад, під іменем main.py.

```
import my_module
my_module.hello()
print(my_module.fun(5))
```

В результаті виконання ми одержимо:

```
Hello, world!
10
```

Слід пам'ятати, що інші користувачі (програмісти) також можуть імпортувати та використовувати ваші модулі в якості змінних. Модулі не можна називати так само, як і ключові слова. Також імена модулів не починаються з цифр. Не можна називати модулі іменами вбудованих функцій, оскільки це призводить до незручностей при їх подальшому використанні та викличе помилку.

Хід роботи

1. Напишемо власну функцію, яка буде генерувати значення двох математичних функцій в залежності від значення їх аргументів. Наприклад:

$$f = \begin{cases} \cos(x), & x \in [0, \frac{\pi}{2}]; \\ \sin(x), & x \in [\frac{\pi}{2}, \pi]. \end{cases}$$

2. Код програми має наступний вигляд:

```
# Функціональне програмування на мові Python
import math as m

# Створення функції
def f(x):
    if x < 90:
        return m.cos(m.radians(x))
    else:
        return m.sin(m.radians(x))

# Вивід результату
print("x", "f(x)")
for i in range(0,190,10):
    print(i, f(i))
```

3. У результаті виконання інтерпретатором Python даного коду, ми отримаємо:

```
x f(x)
0 1.0
10 0.984807753012208
20 0.9396926207859084
30 0.8660254037844387
40 0.766044443118978
50 0.6427876096865394
60 0.5000000000000001
70 0.3420201433256688
80 0.17364817766693041
90 1.0
100 0.984807753012208
110 0.9396926207859084
120 0.8660254037844387
130 0.766044443118978
140 0.6427876096865395
150 0.49999999999999994
160 0.3420201433256689
170 0.17364817766693028
180 1.2246467991473532e-16
```

4. Контрольне завдання: напишіть програму, яка реалізує обчислення значення функції у залежності від її аргументу за допомоги інструкції def, у відповідності до варіанту завдання (порядковий номер в журналі групи).

5. Також, накресліть алгоритм (блок-схему) до вашої програми.

Табл. 6

№	Завдання
1	$f = \begin{cases} \ln^2(x), x \in [1, 9]; \\ (x+1)^2/4, x \in [10, 20]. \end{cases}$
2	$f = \begin{cases} x^3+1, x \in [1, 11]; \\ x^2/(x+1), x \in [12, 20]. \end{cases}$
3	$f = \begin{cases} \cos(x), x \in [0, \pi/2]; \\ \sin(x), x \in [\pi/2, \pi]. \end{cases}$
4	$f = \begin{cases} e^x - 1, x \in [1, 9]; \\ 1/(x+1), x \in [10, 15]. \end{cases}$
5	$f = \begin{cases} \cot(x), x \in [\pi/6, \pi/4]; \\ \sin(x), x \in [\pi/4, \pi/3]. \end{cases}$
6	$f = \begin{cases} (2x-1)^2/2, x \in [10, 19]; \\ (x-1)/(x+1), x \in [20, 30]. \end{cases}$

7	$f = \begin{cases} \cos(x), x \in [\pi/2, \pi]; \\ \sin(x), x \in [\pi, 3\pi/2]. \end{cases}$
8	$f = \begin{cases} x^2/(x+1), x \in [5, 15]; \\ x^3, x \in [16, 25]. \end{cases}$
9	$f = \begin{cases} \ln(x+1)/2, x \in [1, 10]; \\ (2x+1)/3, x \in [11, 22]. \end{cases}$
10	$f = \begin{cases} (2-3x)^2/2, x \in [4, 15]; \\ (4x+1)/3, x \in [15, 24]. \end{cases}$
11	$f = \begin{cases} x^2 - x, x \in [2, 10]; \\ x^2/3, x \in [11, 23]. \end{cases}$
12	$f = \begin{cases} (x^2+1)/x, x \in [20, 29]; \\ x/(x+1), x \in [30, 40]. \end{cases}$
13	$f = \begin{cases} 1/\cos(x), x \in [\pi/6, \pi/4]; \\ -\tan(x), x \in [\pi/4, \pi/3]. \end{cases}$
14	$f = \begin{cases} 1-x^2, x \in [1, 10]; \\ \sqrt{x+1}, x \in [11, 20]. \end{cases}$
15	$f = \begin{cases} \cos(x) \cdot \sin(x), x \in [\pi/4, \pi/2]; \\ \tan(x) - \sin(x), x \in [\pi/2, \pi]. \end{cases}$
16	$f = \begin{cases} (x-1)^2/2, x \in [3, 14]; \\ -(3x+1), x \in [15, 25]. \end{cases}$
17	$f = \begin{cases} \cos(x) - \cot(x), x \in [\pi/2, \pi]; \\ \sin(x)/2, x \in [\pi, 3\pi/2]. \end{cases}$
18	$f = \begin{cases} x^3/x, x \in [1, 8]; \\ \lg(x)/3, x \in [9, 18]. \end{cases}$

Контрольні питання

1. Що таке функція в Python? Як її визначити?
2. Що таке анонімні функції в Python?
3. Що таке Docstring? Для чого він служить?
4. Які бувають способи підключення та використання модулів?
5. Що таке інструкція from. Для чого вона потрібна?

Лабораторна робота № 6

«Робота з фалами в Python»

Мета роботи: сформувати навички роботи зі операціями запису та зчитування даних з файлів на мові Python.

Теоретичні відомості

Однією з найважливіших задач в програмуванні є задача роботи зі файлами, а саме: запис, читання, виведення змісту, копіювання, перейменування, порівняння та інші.

Розглянемо на конкретних прикладах основні операції роботи з файлами в Python.

Операція запису даних у файл:

```
filename = "mydata.txt"
# Відкриття файлу
fd = open(filename, "w")
# Запис у файл
for i in range(10):
    A = i*18
    fd.write("%i\t%.1f\n" % (i, A))
# Закриття файлу
fd.close()
```

Операція зчитування та виведення змісту файлу:

```
import csv
import sys
filename = "mydata.txt"
# Відкриття файлу
fd = open(filename, "r")
# Читання даних
reader = csv.reader(fd, delimiter="\t")
# Виведення змісту файлу
for row in reader:
    print(row)
# Закриття файлу
fd.close()
```

Операція копіювання файлу:

```
import shutil
shutil.copyfile("C:\\mydoc.doc", "C:\\My Documents\\mydoc_2.doc")
```

Операція перейменування файлу:

```
import os
os.rename("C:\\mydoc.doc\\testfile.txt", "/home/user/test.txt")
```

Порівнювати файли можна за змістом або за їх властивостями, що значно швидше. Обидва варіанти можливі за допомоги модуля `filecmp`:

```
import filecmp
similar = filecmp.cmp('C:\\file1.txt', 'C:\\file2.txt')
print(similar)
```

В результаті, такого виконання (порівняння за змістом), ми отримаємо True, якщо файли збігаються або False в протилежному випадку.

Операція видалення файлу:

```
import os
os.remove("C:\\mydoc.doc\\testfile.txt")
```

Розглянемо більш детально функцію open():

```
my_file = open([ім'я_файлу], [режим_доступу], [буферизація])
```

В функції open() є три основних параметри (аргументи), про те нас цікавлять тільки перших два.

Перший аргумент – це ім'я файлу. Другий аргумент – це режим роботи, в якому ми будемо працювати зі відкритим файлом. Більш детальна інформація стосовно існуючих режимів роботи подана в наступних таблицях (табл. 7 та табл. 8).

Табл. 7

Режим роботи	Зміст
"r"	Є режимом за замовчуванням, який призначений для відкриття та читання інформації (даних) з файлу.
"w"	Якщо файл не існує створює новий. Призначений для відкриття та запису в файл, інформації (даних), при цьому вміст файлу видаляється.
"x"	Викликає виключення «FileExistsError», якщо файлу існує та призначений для ексклюзивного створення файлу.
"a"	Призначений для відкриття та дозапису інформації (даних), що здійснюється в кінець файлу.
"b"	Призначений для відкриття файлу у двійковому (бінарному) форматі.
"t"	Є режимом за замовчуванням, який призначений для роботи з файлами в текстовому режимі.
"+"	Призначений для відкриття, читання та запису інформації (даних) в файл.

Табл. 8

Режим роботи	Зміст
"r+b"	Призначений для відкриття бінарного файлу, для читання та запису інформації. Курсор стоїть на початку.
"w+b"	Призначений для відкриття бінарного файлу, для читання та запису інформації, при цьому вміст файлу видаляється. Курсор стоїть на початку.
"wb"	Призначений для відкриття файлу, для запису, у двійковому (бінарному) форматі. Утворює файл з іменем: «ім'я_фалу», якщо такого не існує. Курсор стоїть на початку.
"a+"	Призначений для відкриття файлу, для додавання та читання. Утворює файл з іменем: «ім'я_фалу», якщо такого не існує. Курсор стоїть на початку.

Хід роботи

1. Напишемо програму запису/зчитування файлу, яка містить список студентів та результати їх навчання з курсу «Інформатика».
2. Код програми має наступний вигляд:

```
# Створення та запис файлу
import csv
filename = "mygroup.txt"
fd = open(filename, "w")
n = int(input("Введіть число студентів вашої групи: "))
for i in range(1,n+1):
    name = input("Введіть прізвище, ім'я та батькові: ")
    score = float(input("Введіть бал з курсу Інформатика: "))
    fd.write("%i\t %s\t %.2f\n" % (i, name, score))
fd.close()

# Зчитування файлу та виведення результату
fd = open("mygroup.txt", "r")
count = 0; aver = 0
redear = csv.reader(fd, delimiter="\t")
for row in redear:
    print(row)
    count += 1
    aver += float(row[2])
fd.close()
print("Число студентів в групі: ", count, "Середній бал з курсу Інформатика: ", aver/count)
```


3. У результаті виконання інтерпретатором Python даного коду, ми одержимо:

```
Введіть число студентів вашої групи: 4
Введіть прізвище, ім'я та батькові: Ковтун Олександр Васильович
Введіть бал з курсу Інформатика: 90
Введіть прізвище, ім'я та батькові: Тищенко Аліса Олександрівна
Введіть бал з курсу Інформатика: 72.4
Введіть прізвище, ім'я та батькові: Попович Юлія Олександрівна
Введіть бал з курсу Інформатика: 88
Введіть прізвище, ім'я та батькові: Жук Ольга Іванівна
Введіть бал з курсу Інформатика: 99.9
['1', ' Ковтун Олександр Васильович', ' 90.00']
['2', ' Тищенко Аліса Олександрівна', ' 72.40']
['3', ' Попович Юлія Олександрівна', ' 88.00']
['4', ' Жук Ольга Іванівна', ' 99.90']
Число студентів в групі: 4 Середній бал з курсу Інформатика: 87.575
```

4. Контрольне завдання: напишіть за допомоги Python програму, яка буде виконувати:

- Створення файлу, у якому будуть міститися рядки з іменами студентів вашої групи з балами, як мінімум з двох предметів (на ваш вибір);
- Зчитування файлу, з порядковим виведенням його вмісту;
- Виведення числа студентів в групі, назви предметів та середні бали із них.

Контрольні питання

1. Як здійснюється запис даних у файл в Python?
2. Як здійснюється читання даних з файлу в Python?
3. Як здійснюється копіювання файлу в Python?
4. Який синтаксис функції `open()` в Python? Опишіть її складові (аргументи).
5. Які бувають режими роботи з файлами? Опишіть їх.

Лабораторна № 7

«Робота зі рядками в Python»

Мета роботи: сформувати навички роботи зі символами та рядками у Python

Теоретичні відомості

Рядки в Python представляють собою послідовність символів. При чому, для них не існує немає окремого виділеного типу даних. У зв'язку з цим, будь-який символ – це не що інше, як рядок (рядковий тип даних) довжиною рівній одиниці. Довжина рядків в Python обмежується виключно виділенням, на вашому комп'ютері, об'ємом оперативної пам'яті.

Для запису послідовностей символів в декілька рядків тексту (абзаци) їх записують між потрійними лапки (Docstring, див. лаб. № 5).

Зверніть увагу! Пробіли, також вважаються окремими символи, тому їх приступити на початку або в кінці рядка, рахується як частина даного рядка.

```
>>> str = """ Happy birthday Rick!!!  
your friends. """  
>>> print(str)  
Happy birthday Rick!!!  
your friends.
```

Розглянемо основні операції роботи зі рядками в Python. До них відносять: додавання, дублювання, визначення довжини рядка та операція зрізу [].

Операція додавання або конкатенація рядків:

```
>>> str_1 = "Have "  
>>> str_2 = "a nice"  
>>> str_3 = " day!"  
>>> print(str_1+str_2+str_3)  
Have a nice day!
```

Операція дублювання або повторення рядка:

```
>>> print("--This_is_spam-" * 4)  
--This_is_spam--This_is_spam--This_is_spam--This_is_spam--
```

Визначення поточної довжини рядка:

```
>>> str = "Simple Spam."  
>>> print(len(str))  
12
```

Отримання доступу до елемента рядка за його індексом або операція зрізу рядка:

```
>>> str = "Simple Spam."  
>>> str[3]  
'p'
```

Табл. 9. Операції та методи роботи з рядками

Операція або метод	Зміст
<code>str = 'a'; str = "a"; str = "a"; str = ""a""</code>	Присвоєння змінній літералу, або <code>doc_string</code> .
<code>str_1 + str_2</code>	Сумування рядків (операція конкатенації).
<code>str * 3</code>	Операція дублювання рядка (повторення).
<code>str[i]</code>	Звернення до елемента рядка за його індексом.
<code>str[i : j : step]</code>	Виконання операції зрізу рядка зі заданим кроком (<code>step</code>).
<code>'s' in str; 's' not in str</code>	Повертає значення <code>True</code> , якщо <code>'s'</code> знаходиться в <code>str</code> та <code>False</code> в протилежному випадку; повертає значення <code>True</code> , якщо <code>'s'</code> не знаходиться в <code>str</code> та <code>False</code> в протилежному випадку.
<code>len(str)</code>	Повертає довжину рядка <code>str</code> .
<code>str.count(str_0)</code>	Повертає число входжень рядка <code>str_0</code> в рядок <code>str</code> .
<code>str.find(str_0, [start], [end])</code>	Здійснює пошук підрядка <code>str_0</code> в рядку <code>str</code> . Повертає індекс першого входження підрядка <code>str_0</code> в рядок <code>str</code> , або <code>-1</code> .
<code>str.index(str_0, [start], [end])</code>	Здійснює пошук підрядка <code>str_0</code> в рядку <code>str</code> . Повертає індекс першого входження підрядка <code>str_0</code> в рядок <code>str</code> , або <code>ValueError</code> .
<code>str.replace("old", "new", count)</code>	Здійснює заміну <code>"old"</code> (група символів) в рядку <code>str</code> на шаблон <code>"new"</code> , задане число разів (<code>count</code>).
<code>str.split(sep)</code>	Здійснює розбиття рядка <code>str</code> на слова за допомоги роздільника – <code>sep</code> (за замовчуванням роздільник – пробіл).

<code>str.join(list)</code>	Створює рядок зі усіма елементами <code>list</code> (<code>list</code> – список або кортеж), при цьому, в якості розділювача використовується рядок <code>str</code> .
<code>str.strip()</code>	Здійснює видалення пробілів на початку та в кінці рядка <code>str</code> .
<code>str.lower()</code>	Переводить всі елементи рядка <code>str</code> у нижній регістр.
<code>str.upper()</code>	Переводить всі елементи рядка <code>str</code> у верхній регістр.

За умовчанням, перший символ в рядку завжди має індекс, рівний нулю, другий одиниці і так далі.

Операція зрізу (англ. *slice*) представляє собою вилучення з рядка одного символу або групи символів (підрядка або фрагменту). Розглянемо її більш детально. Вона має наступну конструкцію (синтаксис):

`[i:j:k]`,

де `i` – це індекс початок зрізу, `j` індекс закінчення (причому символ під індексом `j` в сам зріз не входить); `k` – крок зрізу.

```
>>> str = "Not Simple Spam."
>>> str[4:10]
'Simple'
>>> str[:10]
'Not Simple'
>>> str[1:]
'ot Simple Spam.'
>>> str[:]
'Not Simple Spam.'
>>> str[2::2]
'tSml pm'
>>> str[2::-2]
'tN'
>>> str[4:10:-1]
''
```

Як видно з наведених вище прикладів виконання зрізу над рядком, в Python можна і зі від'ємним індексом, при цьому відлік здійснюється у зворотному напрямку.

При використанні того чи іншого методу (див. табл. 9), слід пам'ятати, що рядки в Python відносяться до незмінюваних послідовностей. Іншими словами, всі функції і методи створюють новий рядок і не модифікують поточний. Наприклад:

```

>>> str = "Spam."
>>> str[1] = "a"
Traceback (most recent call last):
  File "<pyshell#46>", line 1, in <module>
    str[1] = "a"
TypeError: 'str' object does not support item assignment
>>> str = str[0]+"a"+str[2:]
>>> print(str)
Saam.

```

Рядки в Python часто комбінують зі екранованими послідовностями, або так званими escape-послідовностями, які можуть складатися з одного або декількох символів у комбінації зі зворотною косою рискою.

Табл. 10

Послідовність	Зміст
\	Ігнорування продовження рядка на наступний.
\\	Зберігає в рядку символ \
\"	Зберігає в рядку символ "
\n	Забезпечує перехід на новий рядок
\t	Забезпечує горизонтальну табуляцію
\v	Забезпечує вертикальну табуляцію
\0	Повертає тип даних Null
\N{id}	Ідентифікатор бази даних Unicode
\uhhhh	16-бітний символ Юні-коду (Unicode)
\Uhhhhhhh	32-бітний символ Юні-коду (Unicode)

Хід роботи

1. Напишемо найпростішу програму реалізації роботи зі рядками в Python.

2. Використовуючи функцію пошуку (find()) та зрізу рядків ([]), вилучимо необхідну нам частину рядка після символу двокрапки, а потім за допомогою функції float() перетворимо вилучену частину у число з рухомою комою. Наприклад:

```
str = 'X-DSPAM-Confidence:0.8475'
```

3. Код програми має наступний вигляд:

```

x = "X-DSPAM-Confidence:0.8475"
i = x.find(":")
y = x[i+1:]
print(float(y))

```

4. У результаті виконання інтерпретатором Python даного коду, ми отримаємо:

```
0.8475
>>>
```

5. Контрольне завдання: у відповідності до варіанту завдання (порядковий номер в журналі групи) напишіть програму, яка:

- Визначає загальне число символів та число пробілів в рядку;
- Заміняє необхідну частину рядку на інше слово або число.

6. Також, накресліть алгоритм (блок-схему) до вашої програми.

Табл. 11

№	Завдання
1	<pre>""" Guido van Rossum was born and raised in the Netherlands, where he received a master's degree in mathematics and computer science from the University of Amsterdam in 1972. While working at the Centrum Wiskunde & Informatica (CWI), Van Rossum wrote and contributed a glob() routine to BSD Unix in 1986. """</pre> <p>Замініть '1972' на '1982'.</p>
2	<pre>""" Jeffrey Preston Bezos was born in Santa Fe, New Mexico on January 12, 1964. After Bezos graduated from Princeton University in 1986, he was offered jobs at Intel, Bell Labs, and Andersen Consulting, among others. In late 1983, Bezos decided to establish an online bookstore. """</pre> <p>Замініть 'Santa Fe' на 'Albuquerque'.</p>
3	<pre>""" William Henry Gates III was born in Seattle, Washington, on November 28, 1955. At 17, Gates formed a venture with Allen called Traf-O-Data to make traffic counters based on the Intel 8008 processor. Microsoft launched its first retail version of Microsoft Windows on November 20, 1985. """</pre> <p>Замініть ' November 28' на ' October 28'.</p>
4	<pre>""" Mark Elliot Zuckerberg was born on May 14, 1984, in White Plains, New York. Zuckerberg began using computers and</pre>

	<p>writing software in middle school. The New Yorker noted that by the time Zuckerberg began classes at Oxford in 2002 he had already achieved a "reputation as a programming prodigy."</p> <p>""</p> <p>Замініть 'Oxford' на 'Harvard'.</p>
5	<p>""</p> <p>Elon Reeve Musk was born on June 28, 1981, in Pretoria, Transvaal, South Africa. While awaiting Canadian documentation, Musk attended the Unive sity of Pretoria for five months. In 1995, Musk, his brother Kimbal, and Greg Kouri founded web software company Zip2 with money raised from a small group of angel investors.</p> <p>""</p> <p>Замініть '1981' на '1971'.</p>
6	<p>""</p> <p>John George Kemeny (born May 31, 1916 – December 26, 1992) was a Hungarian-born American mathematician, computer scientist, and educator best known for co-developing the BASIC programming language in 1964 with Thomas E. Kurtz. Kemeny served as the 13th President of Dartmouth College from 1970 to 1981 and pioneered the use of computers in college education.</p> <p>""</p> <p>Замініть '1916' на '1926'.</p>
7	<p>""</p> <p>Thomas Eugene Kurtz (born February 22, 1928) is a retired Dartmouth professor of mathematics and computer scientist, who along with his colleague John G. Kemeny set in motion the then revolutionary concept of making computers as freely available to college students as library books were, by implementing the concept of time-sharing at Boston College.</p> <p>""</p> <p>Замініть 'Boston' на 'Dartmouth'.</p>
8	<p>""</p> <p>Niklaus Emil Wirth (born 15 February 1934) is a Swiss computer scientist. He has designed several programming languages, including Pascal, and pioneered several classic topics in software engineering. In 1960, he earned a Master of Science (MSc) from Universite Laval, Canada.</p> <p>""</p> <p>Замініть '(PhD)' на '(MSc)'.</p>

9	<p>""</p> <p>Dennis MacAlistair Ritchie (September 9, 1941 – October 12, 2011) was an American computer scientist. He created the C programming language and, with long-time colleague Ken Thompson, the Unix operating system and B programming language. Ritchie and Thompson were awarded the Turing Award from the ACM in 1983, the Hamming Medal from the IEEE in 1980 and the National Medal of Technology from President Bill Clinton in 1999.</p> <p>""</p> <p>Замініть '1980' на '1990 '.</p>
10	<p>""</p> <p>James Arthur Gosling, often referred to as "Dr. Java", OC (born May 19, 1955) is a Canadian computer scientist, best known as the founder and lead designer behind the Java programming language. He created the original design of Java. In March 2011, Gosling joined IBM. Six months later, he followed his colleague Bill Vass and joined a startup called Liquid Robotics.</p> <p>""</p> <p>Замініть 'IBM' на 'Google '.</p>
11	<p>""</p> <p>The Web Hypertext Application Technology Working Group (WHATWG) is a community of people interested in evolving HTML and related technologies. The WHATWG was founded by individuals from Apple Inc., the Mozilla Foundation and Opera Software, leading Web browser vendors, in 2000.</p> <p>""</p> <p>Замініть '2000' на ' 2004 '.</p>
12	<p>""</p> <p>The World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web. Founded in 1994 and currently led by Tim Berners-Lee, the consortium is made up of member organizations that maintain full-time staff working together in the development of standards for the Internet.</p> <p>""</p> <p>Замініть 'Internet' на ' World Wide Web '.</p>
13	<p>""</p> <p>Brendan Eich (born July 4, 1948) is an American technologist and creator of the JavaScript programming language. He co-</p>

	<p>founded the Mozilla project, the Opera Foundation and the Mozilla Corporation, and served as the Mozilla Corporation's chief technical officer and, briefly, as its chief executive officer.</p> <p>""</p> <p>Замініть '1948' на '1951'.</p>
14	<p>""</p> <p>Rasmus Lerdorf (born 22 November 1968) is a Danish-Canadian programmer. He co-authored and inspired the PHP programming language, authoring the first two versions of the language and participating in the development of later versions led by a group of developers including Jim Winstead, Stig Bakken, Shane Caraveo, Andi Gutmans, and Zeev Suraski. He continues to contribute to the project.</p> <p>""</p> <p>Замініть 'programming' на 'scripting'.</p>
15	<p>""</p> <p>Steven Paul Jobs (January 24, 1955 – October 5, 2011) was an American business magnate, industrial designer, investor, and media proprietor. He was the chairman, chief executive officer (CEO), and co-founder of Apple Inc., the chairman and majority shareholder of Pixar, a member of The Walt Disney.</p> <p>""</p> <p>Замініть 'January 24,' на 'February 24,'.</p>

Контрольні питання

1. Які базові операції над рядками ви знаєте?
2. Що таке екрановані послідовності (escape-послідовності)? Наведіть приклади.
3. Як можна отримати зрізи рядків? Якими вони бувають? Наведіть приклади.
4. Які основні методи роботи зі рядками ви знаєте?

Лабораторна № 8

«Робота зі списками в Python»

Мета роботи: сформувати навички роботи зі списками в Python

Теоретичні відомості

Список – це впорядкована множина значень, що ідентифікуються індексом. Багато в чому списки є схожими на рядки, які по суті теж є впорядкованими множинами символів. Відмінність списків від рядків полягає в тому, що елементи списку можуть бути будь-якого типу. Впорядковані множини називають послідовностями.

Lists (списки) – це тип даних для опису послідовності значень. В Python списки представляються, як послідовність записана через кому і у квадратних дужках. Існує декілька способів створення списків. Найпростіший з них: перерахувати елементи списку через кому в квадратних дужках:

```
>>> [10, 20, 30, 40]
[10, 20, 30, 40]
>>> ["one", "two", "three"]
['one', 'two', 'three']
```

Перший приклад – це список чотирьох цілих чисел, а другий – це список з трьох рядків. Елементи списків зовсім не обов'язково повинні бути одного типу.

Наступний список містить стрічку, ціле і дробове числа та інший список:

```
>>> ["hello", 5, 2.0 [10, 20]]
['hello', 5, 2.0 [10, 20]]
```

Список, що є елементом іншого списку, називають вкладеним. Список, який не містить жодного елементу, називають порожнім. Він позначається порожніми квадратними дужками ([]). Списки, як і будь-які інші значення, можуть зберігатися в змінних:

```
numbers = [17, 123, 537]
empty = []
print(numbers, empty)
[17, 123, 537] []
```

Доступ до значень отримуємо, як і випадку рядків.

```
>>> phrase1 = ["colorless", "green", "ideas"]
>>> phrase1
["colorless", "green", "ideas"]
>>> len(phrase1)
3
>>> phrase1[0]
```

```
"colorless"
```

```
>>> phrase1[-1]
```

```
"ideas"
```

Аналогічним чином значення індексуються і утворюються їх зрізи.

```
>>> phrase1[1:3]
```

```
["green", "ideas"]
```

```
>>> phrase1[-2:]
```

```
["green", "ideas"]
```

Списки аналогічно до рядків можуть поєднуватися.

```
>>> phrase2 = phrase1 + ["sleep", "furiously"]
```

```
>>> phrase2
```

```
["colorless", "green", "ideas", "sleep", "furiously"]
```

```
>>> phrase1
```

```
["colorless", "green", "ideas"]
```

Значення у списках можна змінювати на відміну від рядків, де значення змінити неможливо. Вміст списків можна міняти в будь-який момент часу і тому списки підтримують набір операцій або методів.

```
>>> phrase1[0] = "colorful"
```

```
>>> phrase1
```

```
["colorful", "green", "ideas"]
```

Табл. 12. Основні методи роботи зі списками

Метод	Зміст
lst.insert(i, j)	Вставка елемента j у відповідну позицію i (індекс) в списку lst.
lst.append(j)	Додавання елемента j до кінця списку lst.
lst.index(j)	Визначення індексу елемента j в списку lst.
lst.remove(j)	Вилучення зі списку lst першого елемента, який збігається зі j.
lst.count()	Повернення числа елементів в списку lst.
lst.sort()	Сортування елементів списку lst за абеткою.
lst.reverse()	Сортування елементів списку lst у зворотньому напрямку.
lst.extend(lst_0)	Додавання до кінця списку lst нового списку lst_0.
lst.pop(i)	Повертає елемент з індексом i та вилучає його зі списку lst. Без i вилучає останній елемент.

Крім того, Python надає можливість швидкого створення списків цілих значень, без необхідності їх перераховувати:

```
>>> range(1,5)
```

```
[1, 2, 3, 4]
```

У даному прикладі функція `range()` приймає два цілих аргументи і повертає список, який містить всі цілі числа в проміжку між заданими значеннями, включаючи перше і виключаючи друге.

Існує ще два способи виклику функції `range()`. Якщо їй передано тільки одне значення, то в результаті вона поверне список з цілими значеннями від 0 до N, де N – значення параметра:

```
>>> range(10)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Якщо ж `range()` викликана з трьома аргументами, то останній з них інтерпретується як розмір кроку. Тобто в кінцевому списку значення йтимуть не підряд, а через проміжки, що дорівнюють заданому кроку:

```
>>> range(1, 10, 2)
```

```
[1, 3, 5, 7, 9]
```

Хід роботи

1. Напишемо програму заповнення матриці (двовимірний масив) розміром 5x5. Наприклад:

$$\begin{pmatrix} 1 & 2 & 3 & 2 & 4 \\ 1 & 2 & 4 & 2 & 5 \\ 1 & 4 & 4 & 2 & 3 \\ 0 & 7 & 4 & 8 & 3 \\ 4 & 2 & 5 & 4 & 1 \end{pmatrix}$$

2. Код програми має наступний вигляд:

```
# Створення матриці
Rows = int(input("Введіть число рядків = "))
Columns = int(input("Введіть число стовпців = "))
M = []

# Заповнення матриці
for i in range(Rows):
    row = []
    for j in range(Columns):
        elem = int(input("M[{}][{}] = ".format(i, j) ))
        row.append(elem)
    M.append(row)

# Друк матриці
print()
for i in M:
    for j in i:
        print(j, end=" ")
    print()
```

3. У результаті виконання інтерпретатором Python даного коду, ми одержимо:

1	2	3	2	4
1	2	4	2	5
1	4	4	2	3
0	7	4	8	3
4	2	5	4	1

4. Контрольна задавання: у відповідності до варіанту завдання (порядковий номер в журналі групи) напишіть програму, яка створює та заповнює двовимірний масив (матрицю), а також знаходить суму елементів відповідного рядка та стовпчика даного масиву.

5. Також, накресліть алгоритм (блок-схему) до вашої програми.

Табл. 13

№	Завдання
1	Заповніть та знайдіть суму елементів 4-го рядка та 5-го стовпчика матриці виду: <div> $\begin{pmatrix} 3 & 8 & 3 & 2 & 0 \\ 2 & 3 & 5 & 2 & 6 \\ 2 & 5 & 4 & 3 & 3 \\ 1 & 3 & 5 & 6 & 3 \\ 2 & 5 & 3 & 1 & 0 \end{pmatrix}$ </div>
2	Заповніть та знайдіть суму елементів 2-го рядка та 3-го стовпчика матриці виду: <div> $\begin{pmatrix} 5 & 2 & 5 & 2 & 4 \\ 4 & 2 & 3 & 2 & 6 \\ 1 & 2 & 1 & 2 & 2 \\ 0 & 5 & 6 & 2 & 1 \\ 5 & 3 & 4 & 4 & 1 \end{pmatrix}$ </div>
3	Заповніть та знайдіть суму елементів 5-го рядка та 1-го стовпчика матриці виду: <div> $\begin{pmatrix} 3 & 2 & 4 & 2 & 5 \\ 2 & 2 & 3 & 2 & 1 \\ 0 & 4 & 9 & 2 & 3 \\ 3 & 5 & 4 & 4 & 2 \\ 4 & 0 & 5 & 0 & 4 \end{pmatrix}$ </div>
4	Заповніть та знайдіть суму елементів 3-го рядка та 2-го стовпчика матриці виду:

	$\begin{pmatrix} -1 & 2 & 4 & 2 & 4 \\ 8 & -2 & 4 & 2 & 7 \\ 1 & 5 & 3 & 2 & 3 \\ 1 & 2 & 4 & 2 & 3 \\ 6 & 8 & 7 & 4 & 3 \end{pmatrix}$
5	<p>Заповніть та знайдіть суму елементів 4-го рядка та 3-го стовпчика матриці виду:</p> $\begin{pmatrix} 8 & 3 & 5 & 4 & 0 \\ 1 & 3 & 4 & 7 & 5 \\ 9 & 4 & 3 & 2 & 1 \\ 3 & 5 & 3 & 8 & 2 \\ 3 & 2 & 1 & 3 & 9 \end{pmatrix}$
6	<p>Заповніть та знайдіть суму елементів 1-го рядка та 5-го стовпчика матриці виду:</p> $\begin{pmatrix} 3 & 4 & 3 & 2 & -3 \\ 1 & 4 & 3 & 2 & -2 \\ 1 & 5 & 4 & 2 & 3 \\ 7 & -3 & 4 & 1 & 3 \\ 2 & 2 & 3 & 4 & 5 \end{pmatrix}$
7	<p>Заповніть та знайдіть суму елементів 2-го рядка та 3-го стовпчика матриці виду:</p> $\begin{pmatrix} 7 & 6 & 3 & 3 & 4 \\ 2 & 2 & 3 & 2 & 1 \\ 4 & 5 & 6 & 2 & 1 \\ 0 & 8 & 9 & 3 & 2 \\ 4 & 1 & 3 & 0 & 1 \end{pmatrix}$
8	<p>Заповніть та знайдіть суму елементів 1-го рядка та 2-го стовпчика матриці виду:</p> $\begin{pmatrix} 0 & -3 & 4 & 2 & 1 \\ 5 & 2 & 3 & 2 & 6 \\ 3 & 4 & 5 & 2 & 2 \\ 3 & 5 & 4 & 6 & -2 \\ 1 & 2 & 0 & 4 & 3 \end{pmatrix}$
9	<p>Заповніть та знайдіть суму елементів 5-го рядка та 4-го стовпчика матриці виду:</p>

	$\begin{pmatrix} 2 & 5 & 3 & 0 & 2 \\ 4 & 3 & 4 & 6 & 3 \\ 2 & 3 & 6 & 7 & 3 \\ 2 & 1 & 5 & 3 & 3 \\ 1 & 4 & 2 & 1 & 7 \end{pmatrix}$
10	<p>Заповніть та знайдіть суму елементів 1-го рядка та 3-го стовпчика матриці виду:</p> $\begin{pmatrix} 3 & 4 & 1 & 1 & 2 \\ -1 & -3 & 5 & 0 & 8 \\ 3 & 5 & -3 & 2 & 0 \\ 4 & 8 & 3 & 1 & 2 \\ 0 & 0 & 3 & -1 & 2 \end{pmatrix}$
11	<p>Заповніть та знайдіть суму елементів 2-го рядка та 1-го стовпчика матриці виду:</p> $\begin{pmatrix} 2 & 4 & 3 & 5 & 6 \\ 4 & 1 & 2 & 3 & 3 \\ 4 & 5 & 2 & 1 & 1 \\ 3 & 2 & 5 & 6 & 7 \\ 8 & 9 & 3 & 2 & 5 \end{pmatrix}$
12	<p>Заповніть та знайдіть суму елементів 4-го рядка та 4-го стовпчика матриці виду:</p> $\begin{pmatrix} 0 & 0 & -1 & 4 & 2 \\ 2 & 1 & 3 & -2 & 5 \\ 0 & -4 & 4 & 2 & 1 \\ 5 & 6 & 3 & 2 & 5 \\ 3 & 8 & 9 & 3 & -1 \end{pmatrix}$
13	<p>Заповніть та знайдіть суму елементів 5-го рядка та 1-го стовпчика матриці виду:</p> $\begin{pmatrix} 4 & 2 & 7 & 8 & 2 \\ 2 & 4 & 3 & 1 & 3 \\ 9 & 3 & 7 & 9 & 8 \\ 0 & 3 & 2 & 5 & 3 \\ 2 & 4 & 7 & 2 & 3 \end{pmatrix}$
14	<p>Заповніть та знайдіть суму елементів 1-го рядка та 1-го стовпчика матриці виду:</p>

	$\begin{pmatrix} 0 & -4 & 5 & 8 & 7 \\ 3 & 1 & -2 & 5 & -3 \\ 3 & 9 & 5 & 1 & -3 \\ 0 & 5 & 2 & 1 & 4 \\ 6 & 5 & 3 & 2 & 0 \end{pmatrix}$
15	<p>Заповніть та знайдіть суму елементів 3-го рядка та 1-го стовпчика матриці виду:</p> $\begin{pmatrix} 4 & 3 & 3 & 1 & 1 \\ 0 & 3 & 5 & 7 & 2 \\ 3 & 2 & 2 & 5 & 1 \\ 8 & 2 & 1 & 5 & 2 \\ 5 & 2 & 3 & 6 & 4 \end{pmatrix}$

Контрольні питання

1. Що таке списки в Python?
2. Які способи створення списків ви знаєте?
3. Які основні операції над списками ви знаєте? Опишіть їх.
4. Назвіть основні методи роботи зі списками. Опишіть їх.

Лабораторна № 9

«Робота зі словниками в Python»

Мета роботи: сформувати навички роботи зі словниками в Python

Теоретичні відомості

Словники в Python представляють собою неупорядкований набір довільних даних (об'єктів), доступ до яких здійснюється за допомоги ключа. Їх іноді ще називають умовними «масивами» або так званими хеш-таблицями. Розглянемо способи створення словників в Python.

Перший спосіб, за допомоги фігурних дужок або літералу:

```
>>> D = {}
>>> D
{}
>>> D = {1:"A", 2:"B", 3:"C"}
>>> print(D)
{1: 'A', 2: 'B', 3: 'C'}
```

Другий спосіб, за допомогою вбудованої функції dict():

```
>>> D = dict(first="A", second="B", third="C")
>>> print(D)
{'first': 'A', 'second': 'B', 'third': 'C'}
>>> D = dict([(1,1), (2,2)])
>>> print(D)
{1: 1, 2: 2}
```

Третій спосіб, за допомоги методу функції dict() fromkeys:

```
>>> D = dict.fromkeys(["a", "b"])
>>> print(D)
{'a': None, 'b': None}
>>> D = dict.fromkeys(["a", "b"], 12)
>>> print(D)
{'a': 12, 'b': 12}
```

Четвертий спосіб, найбільш поширений, за допомоги генераторів виразів для словників:

```
>>> D = {a: a ** 2 for a in range(7)}
>>> print(D)
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}
```

Коли інтерпретатор створює новий словник він зберігає елементи в хаотичному (довільному) порядку. Для того, щоби вилучити значення зі словнику необхідно вказати ключ, з яким асоціюється дане значення.

Словники, аналогічно до списків, відносяться до змінювальних типів даних, тому їх можна модифікувати (додати, замінити або видалити об'єкт, чи то зміну) не створюючи при цьому нові словники. Для цього, достатньо скористатися операцією присвоєння за ключем.

```

>>> D = {1: 2, 2: 4, 3: 9}
>>> D[1]
2
>>> D[4] = 4**2
>>> print(D)
{1: 2, 2: 4, 3: 9, 4: 16}
>>> D['1']
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    D['1']
KeyError: '1'

```

Як видно з прикладу вище, присвоєння по новому ключу модифікує (розширює) поточний словник, тобто присвоєння за наявним ключем додає або перезаписує об'єкт чи то зміну. Спроба звернення до неіснуючого ключа генерує помилку. Проте, таку помилку (по суті виняток), можна уникнути скориставшись конструкцією try/except (див. лаб. № 3).

Над словниками по аналогії зі рядками та списками, також можна застосовувати певні операції та методи (див. табл.14). При роботі зі словниками можна використовувати і інші методи, які в таблиці нижче не приведені, проте інформацію про які можна отримати за допомоги команди dir().

Табл. 14. Операції та методи роботи зі словниками

Операція або метод	Зміст
<code>dt = {}</code>	Утворює новий порожній словник.
<code>dt = {"A" : 1, "B" : 2}</code>	Утворює словник з двох елементів (ключ-значення).
<code>key in dt;</code> <code>key not in dt</code>	Повертає значення True, якщо ключ 'key' знаходиться в dt та False в протилежному випадку; повертає значення True, якщо ключ 'key' не знаходиться в dt та False в протилежному випадку.
<code>v in dt.values();</code> <code>v not in dt.values()</code>	Повертає значення True, якщо v знаходиться в dt та False в протилежному випадку; повертає значення True, якщо v не знаходиться в dt та False в протилежному випадку.
<code>dt.items()</code>	Здійснює повернення всіх пар (ключ-значення) в словнику dt.
<code>dt.keys()</code>	Здійснює повернення всіх ключів в словнику dt.

dt.values()	Здійснює повернення всіх значень в словнику dt.
dt.get(key)	Здійснює повернення значення ключа key або None, якщо він відсутній в dt.
dt.get(key, v)	Здійснює повернення значення ключа key або v, якщо він відсутній в dt.

Словники в Python здатні відігравати різні ролі. По-перше, вони надають можливість реалізовувати алгоритми пошуку в структурах. По-друге, вони здатні подавати різні типи структурованої інформації. По-третє, словники вважаються одним із способів опису властивостей елементів в програмах. Тобто, в деякій мірі, вони по своєму змісту відповідають відповідним «структурам» і «записам» інших мов програмування (наприклад об'єкти в JavaScript).

Хід роботи

1. Наведемо приклад роботи зі словниками в Python.
2. Задача: відомі дані про 10 моментів часу доби, тобто години (0 – 23) та хвилини (0 – 59). Скласти програму, яка здійснює порівняння будь-яких дох моментів часу за їх умовним порядковим номером і визначає, який з даних моментів відбувся раніше.
3. Код програми має наступний вигляд:

```
from random import randint, choice
D = {i : [randint(0,23), randint(0,59)] for i in range(1,11,1)}
print(D); print()
# Вибір моментів часу
ch1 = choice( list(D.items()) )
ch2 = choice( list(D.items()) )
print(ch1, ch2); print()
# Порівняння годин моментів часу
if ch1[1][0] < ch2[1][0]:
    print("Момент часу № "+str(ch1[0])+" (" +str(ch1[1][0])+" ":" +
        +str(ch1[1][1])+" ) раніше за момент № "+str(ch2[0])+" (" +
        +str(ch2[1][0])+" ":" +str(ch2[1][1])+" )")
elif ch1[1][0] == ch2[1][0]:
    # Порівняння хвилин моментів часу
    if ch1[1][1] < ch2[1][1]:
        print("Момент часу № "+str(ch1[0])+" (" +str(ch1[1][0])+" ":" +
            +str(ch1[1][1])+" ) раніше за момент № "+str(ch2[0])+" (" +
            +str(ch2[1][0])+" ":" +str(ch2[1][1])+" )")
    elif ch1[1][1] == ch2[1][1]:
        print("Однакові моменти часу (" +str(ch1[1][0])+" ":" +str(ch1[1][1])+" )")
    else:
        print("Момент часу № "+str(ch2[0])+" (" +str(ch2[1][0])+" ":" +
            +str(ch2[1][1])+" ) раніше за момент № "+str(ch1[0])+" (" +
            +str(ch1[1][0])+" ":" +str(ch1[1][1])+" )")
else:
    print("Момент часу № "+str(ch2[0])+" (" +str(ch2[1][0])+" ":" +
        +str(ch2[1][1])+" ) раніше за момент № "+str(ch1[0])+" (" +
        +str(ch1[1][0])+" ":" +str(ch1[1][1])+" )")
```

4. У результаті виконання інтерпретатором Python даного коду, ми отримаємо:

{1: [10, 25], 2: [17, 19], 3: [9, 6], 4: [23, 47], 5: [23, 11], 6: [5, 42], 7: [20, 3], 8: [12, 53], 9: [16, 48], 10: [0, 49]}

(4, [23, 47]) (7, [20, 3])

Момент часу № 7 (20:3) раніше за момент № 4 (23:47)

5. Контрольна задавання: напишіть програму розв'язку задачі у відповідності до варіанту завдання (порядковий номер в журналі групи).

6. Також, накресліть алгоритм (блок-схему) до вашої програми.

Табл. 15

№	Завдання
1	Відомо дані про 10 подій, які відбулися починаючи з 1930 року. Подія складається з номеру, року, місяця, числа, короткого опису. Напишіть програму, яка порівнює будь-які дві події за часом та визначає, яка з подій відбулася пізніше.
2	Відомі дані про 10 моментів часу однієї доби, а саме: години (від 0 – 23); хвилини (0 – 59) та секунди (0 – 59). Напишіть програму, яка здійснює порівняння будь-яких двох моментів часу за їх умовним порядковим номером і визначає, який з даних моментів відбувся пізніше.
3	Відомі прізвища 10-ти співробітників ЧНУ та їх адреса. Напишіть програму, яка визначає, чи працюють у даному закладі люди з прізвищем: Іванюк, Іванець, Іваненко, Іванченко або Івашко. У разі збігу надрукувати їх адресу.
4	Відомі дані 15-ти студентів 124 гр., а саме: прізвище, ім'я, по батькові; кількість пропусків; адреса та телефон. Написати програму, яка визначає студента з максимальним числом пропусків та друкує його ім'я.
5	Відомі дані 10-ти студентів ЧНУ, тобто їх прізвище, ім'я та по батькові, дата народження, адреса проживання та телефон. Написати програму, яка визначає, чи є в університеті студенти, у яких сьогодні день народження, якщо так, то надрукувати їх імена.
6	Відомі дані про вартість, та рік випуску кожної з 10 моделей легкових автомобілів компанії Volkswagen. Написати програму, яка визначає середню вартість автомобілів, починаючи з 1970 року випуску.
7	Відомі дані про масу та об'єм 10 предметів, на основі різних матеріалів. Написати програму, яка розраховує максимальну густину матеріалу.

8	Відомі дані про число опадів за кожен день місяця та температуру повітря в ці дні. Написати програму, що обчислює, яка кількість опадів випала у виді снігу, а яка – у виді дощу (вважати, що дощ йде тоді, коли температура повітря є вищою за 0° C).
9	Відомі дані про потужність та вартість двигунів 10-ти легкових автомобілів фірми Toyota. Написати програму, яка обраховує загальну вартість автомобілів, у яких потужність їх двигуна більша за 100 кінських сил.
10	Відомі дані 10-ти країн із інформацією про їх площу, населення та ту частину світу, де вони розташовані. Напишіть програму, яка визначає, чи є серед заданих країн ті, що знаходяться в Азії, Європі або в Північній Америці. У разі збігу надрукувати їх назви.
11	Відомі дані про число студентів у кожному з 10-ти навчальних закладів та тип закладу (училище, коледж або університет). Написати програму, яка розраховує загальну кількість студентів з усіх закладів.
12	Відомі дані про ціну та тираж 10-ти фахових журналів. Написати програму, яка обчислює середню вартість журналів, тираж яких менше 1000 примірників.
13	Відомі дані про число населення (в мільйонах) та площу (в тис. км ²) 10-ти країн Світу. Написати програму, яка визначає назву тієї країни, яка має максимальну густоту населення.
14	Відомі дані про оцінки 15-ти студентів 122 гр. з чотирьох дисциплін (курсів). Написати програму, яка визначає прізвище того студента, який має мінімальний середній бал по всіх предметах.
15	Відомі дані 10-ти співробітників компанії Ріхар, а саме: ім'я, прізвище, зарплата, адреса, телефон. Написати програму, яка визначає: прізвища співробітника, який має найбільшу заробітну плату.

Контрольні питання

1. Що таке словники в Python? Для чого вони служать?
2. Які способи створення словників існують в Python?
3. Які основні операції роботи зі словниками ви знаєте? Опишіть їх.
4. Які основні методи роботи зі словниками ви знаєте? Опишіть їх.
5. Що таке генератори словників? Наведіть приклад.

Лабораторна № 10

«Об'єктно-орієнтоване програмування в Python»

Мета роботи: сформувати навички роботи з класами в Python

Теоретичні відомості

Класи в Python представляють собою спеціальні конструкції (тип даних), які складається з полів (змінних) та методів (функцій) та призначенні для групування пов'язаних між собою змінних та функцій.

Поля класу або члени-змінні оголошуються всередині класу для зберігання даних. Методи класу або функції визначаються всередині класу і призначені для обробки змінних класу (полів) та взаємодії з іншими даними (об'єктами) основної програми.

Об'єкти класу представляє собою окремі його екземпляри, які є по суті є змінними цього типу, до яких можна застосовувати відповідні методи класу. Визначивши або задавши клас із полями та методами, можна створювати довільне число його об'єктів, кожен з яких в свою чергу містить в собі оголошені набори полів та методів.

Для оголошення класу в Python використовується ключове словом або команда `class`, потім через пробіл йде ім'я класу, яке записується за наступним правилом: всі слова разом, при чому з великої літери.

Для створення об'єкту класу, клас викликається як функція з круглими дужками, результатом виконання якої є новий об'єкт (екземпляр класу), який зберігається в змінній.

В рамках методу класу вказуються його атрибути. Атрибути – це по суті його змінні, а методи – це функції. Основна відмінність методу від функції полягає тому, що в методах в якості першого атрибуту завжди використовуються параметр `self`.

Класи приховують свою внутрішню будову, залишаючи виключно зовнішній "інтерфейс" для безпосереднього використання. Тобто класи – це поєднання змінних та функцій в межах однієї сутності із прихованням їх присутності, що прийнято в об'єктно-орієнтованому програмуванні називати інкапсуляцією.

При оголошенні класу, в його дужках можуть вказуватися імена інших вже існуючих класів. В такому випадку, клас який записаний в дужках стає новим класом (дочірним класом) успадковуючи при цьому всі поля та методи класу в дужках якого він записаний (батьківський клас). Таку властивість в об'єктно-орієнтованому програмуванні називають наслідуванням або успадкуванням.

Якщо два класи, в своїй будові містять один і той же метод, але при виклику якого кожним із них одержується різний результат, то говорять що дані класи володіють властивістю поліморфізму.

Конструкція класу в Python має наступний вигляд:

```
class ім_я_класу():  
    інструкція 1  
    ...  
    інструкція N
```

Синтаксис створення нового об'єкта класу має вигляд:

```
об'єкт_класу = ім'я_класу()
```

Отже, класи – це набори даних (змінних) разом з наборами функцій, які діють (виконуються) над ними. Основна мета використання класів полягає в тому, щоби досягти більшої модульності основного коду, шляхом групування змінних і функцій, в невеликі пов'язані між собою вузли, які в подальшому легше використовувати та модифікувати.

Хід роботи

1. Напишемо найпростішу програму з використанням класів. Наприклад обчислення середнього бала студента з трьох предметів.

2. Код програми має наступний вигляд:

```
class Student():  
    def aver_mark(self, name, x, y, z):  
        self.name = name  
        self.x = x  
        self.y = y  
        self.z = z  
        print(self.name, " - ", ((self.x + self.y + self.z)/3))  
  
st_1 = Student()  
st_2 = Student()  
st_1.aver_mark("Dmytro", 4, 2, 4)  
st_1.aver_mark("Olena", 5, 5, 4)
```

3. У результаті виконання інтерпретатором Python даного коду, ми одержимо:

```
Dmytro - 3.3333333333333335  
Olena - 4.666666666666667
```

4. Контрольна задавання: напишіть програму з використанням класів у відповідності до варіанту завдання (порядковий номер в журналі групи).

5. Також, накресліть алгоритм (блок-схему) до вашої програми.

Табл. 16

№	Завдання
1	Написати програму, в якій реалізовано клас: «Моя Бібліотека». Даний клас повинен виконувати наступні дії: додавання та видалення книги, доступ до анотації книги (короткий опис) та пошуку по книгах за декількома параметрами, а саме: назва, автор, видання та рік випуску.

2	<p>Написати програму, в якій реалізовано клас з двома наступними методами:</p> <p>1) метод, який приймає 1 аргумент (рядковий тип даних) та повертає True або False в залежності від того, чи містить даний рядок повтори послідовностей символів довжиною починаючи з трьох.</p> <p>2) метод, який повертає True або False в залежності від того, чи є рядок паліндромом. Регістрами символів можна знехтувати. Порожній рядок вважати паліндромом.</p>
3	<p>Написати програму, в якій реалізовано клас: «Колода Карт». Кожна карта має число і масть. Даний клас повинен забезпечити виконання наступних дій: можливість виведення карти за номером розташування у колоді, виведення всіх карт, перемішування карт, видачі випадковим чином однієї або трьох карт з колоди.</p>
4	<p>Написати програму, в якій реалізовано клас: «Англо-Український словник». Даний клас повинен забезпечити виконання наступних дій: можливість зберігання декількох варіантів перекладу для кожного слова та виведення всіх варіантів перекладу введеного англійського слова.</p>
5	<p>Написати програму, в якій реалізовано клас: «Транспортний засіб». На його основі реалізувати об'єкти: «Автомобіль», «корабель» та «літак». Даний клас повинен виконувати наступні дії: для кожного об'єкта класу повинна існувати можливість задавати та отримувати дані про його назву, модель, координати поточної локації (GPS), пройдений шлях (в км), стан, рік випуску, швидкість та вартість.</p>

Контрольні питання

1. Що таке клас в Python?
2. Що таке поля та методи класу?
3. Що таке об'єкт класу?
4. Як створити новий клас в Python?
5. Як створити об'єкт класу в Python?
6. У чому полягає відмінність методу від функції?
7. Що таке інкапсуляція, наслідування та поліморфізм?

Лабораторна № 11

«Візуалізація даних в Python»

Мета роботи: сформувати навички роботи з бібліотекою Matplotlib для візуалізації даних в Python.

Теоретичні відомості

Matplotlib – це бібліотека Python, яка призначена для візуалізації даних двовимірної графіки (тривимірна графіка також підтримується). Отримані зображення можуть бути використані в інтерактивній графіці, наукових публікаціях, графічному інтерфейсі, веб-додатках, всюди де використання діаграм є бажаним.

Бібліотека Matplotlib заснована на принципах об'єктно-орієнтованого програмування, проте володіє процедурним інтерфейсом, який є аналогом команд програмного пакету Matlab.

Matplotlib підтримує багато видів графіків і діаграм:

- Графіки (line plot);
- Гістограми (histogram) та стовпчасті діаграми (bar chart);
- Діаграми розсіювання (scatter plot);
- Секторні діаграми (pie chart);
- Графіки на полярній системі координат (polar);
- Та інші.

Дана бібліотека також підтримує типові формати графіки: eps, jpeg, png, pdf та інші. Розглянемо побудову графіків.

```
import matplotlib.pyplot as plt
plt.plot([1, 3, 2, 4])
plt.show()
```

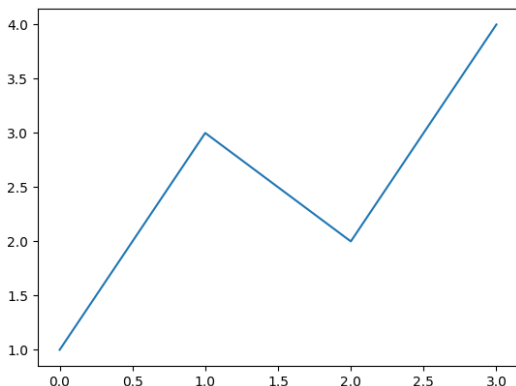


Рис. 3

Функція `plot()` будує графік, а функція `show()` його відображає. Перший аргумент функції `plot()` – це послідовність x значень, а другий – y значень. В прикладі вище, x значення були опущені та подані виключно у значення (y в списку). Тому функція `plot()` автоматично згенерувала для чотирьох зазначених y , список зі чотирьох значень x : $[0, 1, 2, 3]$.

```
from numpy import * # для використання функцій exp та linspace
import matplotlib.pyplot as plt
def f(t):
    return t**2*exp(-t**2)
t = linspace(0, 3, 51) # 51 точка між 0 та 3
y = f(t)
plt.plot(t, y)
plt.show()
```

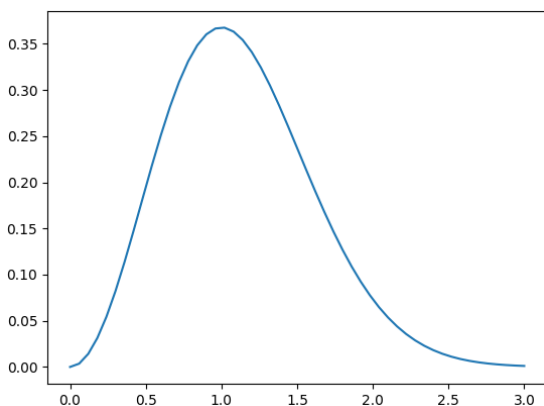


Рис. 4

Якщо зазначена функція більше не використовується, то можна одержати більш компактний код, задавши її відразу після визначення масиву t :

```
from numpy import *
import matplotlib.pyplot as plt
t = linspace(0, 3, 51)
y = t**2*exp(-t**2)
plt.plot(t, y)
plt.show()
```

При побудові кривих, було б добре їх певним чином назвати, позначити осі, задати легенду (це стане в нагоді, якщо будувати кілька графіків одночасно).

Також, часто виникає потреба в зміні вигляд самої кривої її меж побудови, тощо. Наприклад:

```

from numpy import *
import matplotlib.pyplot as plt
t = linspace(0, 3, 51)
y = t**2*exp(-t**2)
plt.plot(t, y, 'g--', label='t^2*exp(-t^2)')
plt.axis([0, 3, -0.05, 0.5]) # задання [xmin, xmax, ymin, ymax]
plt.xlabel('t') # позначення вісі абсцис
plt.ylabel('y') # позначення вісі ординат
plt.title('My first normal plot') # назва графіка
plt.legend() # вставка легенди (тексту в label)
plt.show()

```

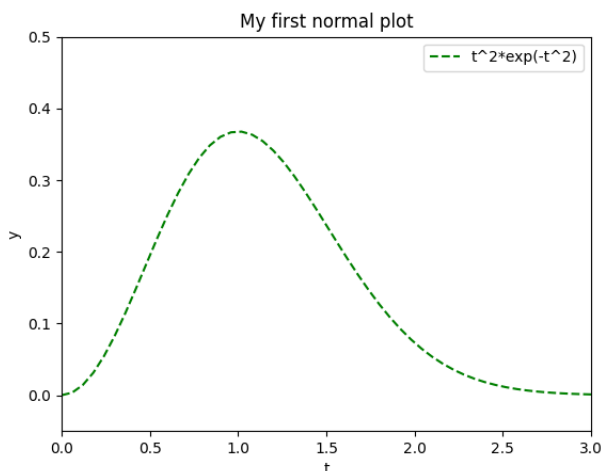


Рис. 5

Крім нововведень, зазначених в коментарях, в аргументах функції `plot()` з'явились два нових параметри. Останній задає текст легенд. Рядковий аргумент «g--» відповідає за зміну форми кривої. У порівнянні з попереднім прикладом, графік став штриховою лінією зеленого кольору (green). За замовчуванням цей аргумент має значення: b-, що означає синю (blue), суцільну лінію. Наведена нижче таблиця демонструє набір доступних значень зазначених вище аргументів функції `plot()`.

Табл. 17

Абревіатура	Зміст
k	Крива чорного кольору
c	Крива блакитного кольору
b	Крива синього кольору
y	Крива жовтого кольору

g	Крива зеленого кольору
r	Крива червоного кольору
w	Крива білого кольору
-	Суцільна крива
--	Штрихова крива
:	Пунктирна крива
-.	Штрих-пунктирна крива

Приклад побудови декількох кривих на одному графіку:

```
from numpy import *
import matplotlib.pyplot as plt
t = linspace(0, 3, 51)
y1 = t**2*exp(-t**2)
y2 = t**4*exp(-t**2)
y3 = t**6*exp(-t**2)
plt.plot(t, y1, 'g^', # маркери із зелених трикутників
         t, y2, 'b--', # синя штрихова
         t, y3, 'ro-') # червоні круглі маркери # з'єднані суцільною лінією
plt.xlabel('t')
plt.ylabel('y')
plt.title('Plotting with markers')
plt.legend(['t^2*exp(-t^2)', 't^4*exp(-t^2)', 't^6*exp(-t^2)'], # список легенди
          loc='upper left') # положення легенди
plt.show()
```

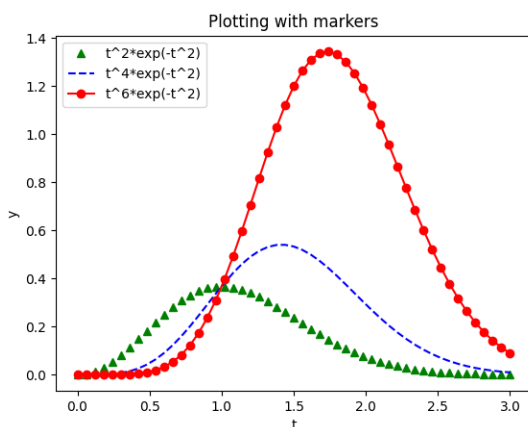


Рис. 6

Я видно з прикладу вище (рис. 6), положення легенди змінилось. За замовчуванням легенда розташовується в правому верхньому кутку, але існує можливість перенести її за рахунок параметру `loc`. Даному параметру можна присвоювати і числові значення, але звичайно рядок сприймається легше. У таблиці нижче наведені можливі варіанти параметру `loc`.

Табл. 18

Місце локації	Назва	Числовий код
Оптимальний варіант	"best"	0
Верху праворуч	"upper right"	1
Верху ліворуч	"upper left"	2
Внизу ліворуч	"lower left"	3
Внизу праворуч	"lower right"	4
Праворуч	"right"	5
Посередині ліворуч	"center left"	6
Посередині праворуч	"center right"	7
Посередині верху	"lower center"	8
Посередині верху	"upper center"	9
Посередині	"center"	10

Для збереження файлу використовують код:

```
from numpy import *
import matplotlib.pyplot as plt
t = linspace(0, 3, 51)
y = t**2*exp(-t**2)
plt.plot(t, y)
plt.savefig('name_of_plot.png', dpi=200)
```

Результуючий рисунок зберігатиметься в тій же директорії (каталогу) де і вихідний файл з іменем та розширенням зазначеним в першому аргументі методу savefig(). Другий (необов'язковий) аргумент дозволяє змінювати роздільну здатність зображення.

Хід роботи

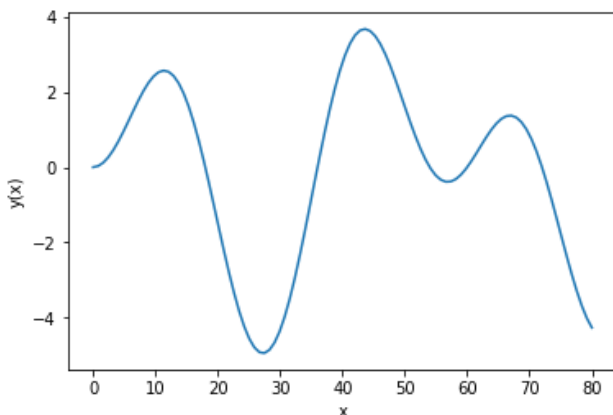
1. Напишемо програму побудови та візуалізації графіка математичної функції за допомоги бібліотеки Matplotlib Python. Наприклад:

$$y(x) = 5 \sin(10x) \cdot \sin(3x), x \in [0 \dots 80]$$

2. Код програми має наступний вигляд:

```
import math
import matplotlib.pyplot as plt
from numpy import linspace
x = linspace(0, 80, 100); y = []
for i in x:
    f = 5 * math.sin(math.radians(10 * i)) * math.sin(math.radians(3 * i))
    y.append(f)
plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y(x)')
plt.show()
```

3. У результаті виконання інтерпретатором Python даного коду, ми отримаємо:



4. Контрольна задавання: зобразити графік функції у відповідності до варіанту завдання (порядковий номер в журналі групи).

5. Додайте до отриманого вами графіку додаткові елементи (зробити його більш інформативним) такі як: назва графіку, легенда, тип лінії, маркери тощо.

6. Збережіть одержаний результат у форматі png

Табл. 19

№	Завдання
1	$y(x) = -x \cdot \cos(5x), x \in [0..10]$
2	$y(x) = (1/x) \cdot \sin(5x), x \in [-5..5]$
3	$y(x) = 10 \cos(x^2)/x^2, x \in [0..4]$
4	$y(x) = \sqrt{x} \cdot \sin(10x), x \in [0..5]$
5	$y(x) = 15 \sin(10x) \cdot \cos(3x), x \in [-3..3]$
6	$y(x) = 5 \sin(x) \cdot \cos^2(x^2 + (1/x)), x \in [1..10]$
7	$y(x) = \sin(10x) \cdot (\sin(3x)/x^2), x \in [0..4]$
8	$y(x) = 5 \sin(10x) \cdot (\sin(3x)/\sqrt{x}), x \in [1..7]$
9	$y(x) = x \cdot \sin(10x), x \in [1..10]$
10	$y(x) = -5 \cos(10x) \cdot (\sin(3x)/\sqrt{x}), x \in [0..10]$
11	$y(x) = -5 \cos(10x) \cdot (\sin(3x)/x^x), x \in [0..5]$

12	$y(x) = x \cdot \sin(5x), x \in [-2...5]$
13	$y(x) = 10 \cos(5x) \cdot (\sin(3x)/(\sqrt{x})), x \in [0...5]$
14	$y(x) = 5 \sin(10x) \cdot (\sin(3x)/x^x), x \in [0...8]$
15	$y(x) = (1/x) \cdot \cos(x^2 + (1/x)), x \in [1...10]$

Контрольні питання

1. Які засоби мова Python надає для роботи з 2D графікою?
2. Яким чином можна зобразити графік математичної функції?
3. Яким чином можна налаштувати такі параметри графіку функції як: колір, тип лінії, маркери, легенда та інше?
4. Яким чином можна зберегти зображення за допомоги бібліотеки Matplotlib в Python?

Список рекомендованої літератури

1. Козуб Г. О., Семенов Н. А., Програмування : метод. рек. до лаб. робіт для студ. спец. 121 – „Інженерія програмного забезпечення”. Старобільськ : ДЗ „ЛНУ імені Тараса Шевченка”, 2020. 108 с.
2. Добровська Л. М. Основи програмування: методичні вказівки до виконання комп'ютерних практикумів на PYTHON з навчальної дисципліни «Основи програмування» для студентів спеціальності 122 «Комп'ютерні науки» зі спеціалізації «Інформаційні технології в біології та медицині». Київ : НТУУ «КПІ ім. Ігоря Сікорського», 2017. 254 с.
3. Мелешко Є. В. Програмування на мові Python. Методичні вказівки до виконання лабораторних робіт студентами денної та заочної форми навчання спеціальностей 123 "Комп'ютерна інженерія", 125 "Кібербезпека". Кропивницький: ЦНТУ, 2017. 58 с.
4. Косухіна О. С. Методичні вказівки до лабораторних занять з дисципліни “Програмне забезпечення обчислювальних систем” освітньо-професійної програми першого (бакалаврського) рівня вищої освіти зі спеціальності 113 “Прикладна математика ” факультету електроніки та комп'ютерної техніки денної форми навчання. Частина 1. Кам'янське: ДДТУ, 2017. 82 с.
5. Льовкін В. М. Методичні вказівки до виконання лабораторних робіт з дисципліни “Інженерія прикладних інтелектуально-орієнтованих програмних продуктів” для студентів спеціальностей 121 “Інженерія програмного забезпечення” та 122 “Комп'ютерні науки та інформаційні технології” (всіх форм навчання). Запоріжжя : ЗНТУ, 2016. 80 с.