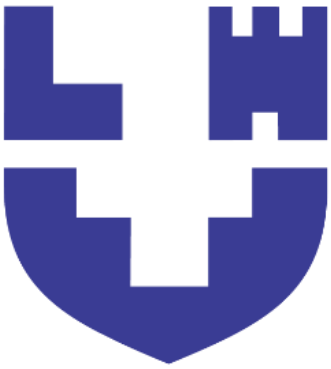
Міністерство освіти і науки України
Луцький національний технічний університет



ПРОГРАМУВАННЯ НА PYTHON

Методичні вказівки до практичних занять
для здобувачів першого (бакалаврського) рівня вищої освіти
галузі знань 12 (F) Інформаційні технології
денної та заочної форм навчання

Луцьк 2026

УДК 004.432 (07)
П 78

Рекомендовано до видання вченою радою факультету КІТ ЛНТУ,
протокол № _____ від « _____ » _____ 20 26 року.

Голова вченої ради факультету КІТ _____ Інна КОНДІУС

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ

Директор бібліотеки _____ Наталія ПОЛІЩУК

Розглянуто і схвалено на засіданні кафедри комп'ютерної інженерії та безпеки
ЛНТУ, протокол № 9 від « _____ » 04 20 26 року.

Укладачі: _____ Сергій КОСТЮЧКО, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та безпеки ЛНТУ

_____ Людмила КОНКЕВИЧ, асистент кафедри комп'ютерної інженерії та безпеки, ЛНТУ

Рецензент: _____ Микола ПОЛІЩУК, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та безпеки ЛНТУ

Відповідальний за випуск: _____ Тарас ТЕРЛЕЦЬКИЙ, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та безпеки ЛНТУ

П 78 Програмування на Python. Методичні вказівки до практичних занять для здобувачів першого (бакалаврського) рівня вищої освіти галузі знань 12 (F) Інформаційні технології денної та заочної форм навчання / уклад. С.М. Костючко, Л.М. Конкевич. Луцьк: ЛНТУ, 2026. 60 с.

Методичне видання до практичних занять з дисципліни «Програмування на Python» складене відповідно до діючої програми курсу.

Призначене для здобувачів першого (бакалаврського) рівня вищої освіти галузі знань 12 (F) Інформаційні технології.

ЗМІСТ

Практичне заняття №1

Початок роботи

Мета: ознайомитися з середовищем розробки Spyder та основами роботи в оболонці IPython; навчитися виконувати найпростіші математичні операції та запускати Python-код; сформувати практичні навички створення, редагування, документування та перевірки власних функцій у файлі програми; засвоїти принципи обчислення арифметичного середнього, відстані між числами, геометричного середнього та об’єму піраміди засобами Python.

Підготовка:

1. Запустіть середовище Spyder.

2. Скиньте простір імен за допомогою команди % reset .

Це забезпечує нам «чистий» сеанс інтерпретатора IPython (у межах Spyder). Ви повинні виконувати ці дії перед кожною лабораторною.

Тепер у вас доступний інтерпретатор IPython, і вам пропонується виконувати основні математичні операції в підказці iPython (у [?] :).

Передумови

Ви повинні знати, як обчислити прості вирази, такі як

3 + 4

(9 + 3) / 6

11 * 12

Вам знадобляться деякі з цих виразів нижче, і ви повинні бути знайомими з використанням оболонки Python для оцінки виразів Python.

Ви повинні знати, як користуватися середовищем розробки Python. Ви повинні знати, як виконати код Python в Spyder і викликати функції із запиту Python.

Для наступних лабораторних сесій вам також може бути корисно знати, як змінювати та оновлювати визначення функцій та викликати оновлену функцію з підказки Python.

Завантажте Training1.py (https://www.southampton.ac.uk/~fangohr/training/python/labs/lab1/training1.py) та збережіть як Training1.py (1 показує, що цей файл є частиною лабораторії 1).

Відкрийте файл за допомогою Spyder і спробуйте зрозуміти його зміст.

Переконайтесь, що середнє значення визначення функції відомо Python (натисканням клавіші F5, наприклад), а потім переконайтеся, що ви можете використовувати середню функцію із запиту Python:

```
In [ ]: average(10, 20)
Out[ ]: 15.0
In [ ]: average(10, 4)
Out[ ]: 7.0
In [ ]: help(average)
Help on function average in module __main__:
average(a, b)
```

За заданими параметрами a і b обчислити і повернути середнє арифметичне a і b.

Запит підказки Python

Хоча ми очікуємо, що ви використовуєте IPython для цих вправ, спочатку ми також надамо приклади взаємодії з підказкою Python з >>>, як відображається звичайною консоллю Python замість In [3]: як відображається на консолі IPython: номер 3 тут безглуздий, оскільки враховує лише кількість входів саме для цього сеансу. Тому ми пишемо приклад вище як:

```
>>> average(10, 20)
15.0
>>> average(10, 4)
7.0
>>> help(average)
Довідка щодо середнього значення функції в модулі __main__:
average(a, b)
```

Задані параметри a і b, обчислення та повернення середнє арифметичне a і b.

Еквівалентна взаємодія IPython:

```
In [ ]: average(10, 20)
Out[ ]: 15.0
In [ ]: average(10, 4)
Out[ ]: 7.0
In [ ]: help(average)
```

Довідка про середнє значення функції в модулі __main__:

```
average(a, b)
```

За заданими параметрами a і b обчислити і повернути середнє арифметичне a і b.

Консоль Python та IPython

Ми також зазначаємо, що ми будемо називати консоль IPython програшним як оболонку Python або консоль Python. З точки зору мови програмування, оболонка IPython та Python може обробляти один і той же набір команд. Оболонка IPython є лише потужнішою у забезпеченні інструментів, які допомагають нам ефективно писати та налагоджувати код.

Завдання:

Визначте наступні функції у файлі training1.py та переконайтесь, що вони поводяться належним чином. Ви також повинні задокументувати їх способом, подібним до середньої функції.

Функційна відстань (a, b), яка повертає відстань між числами a і b .

Приклади:

```
>>> distance(3, 4)
1
>>> distance(3, 1)
2
```

Приклади (IPython):

```
In [ ]: distance(3, 4)
Out[ ]: 1
```

In [ ]: distance(3, 1)

Out[ ]: 2

Функція geometric_mean (a, b), яка повертає середнє геометричне двох чисел – тобто довжину ребра, яку повинен мати квадрат, щоб його площа дорівнювала площі прямокутника зі сторонами a і b .

Приклади:

```
>>> geometric_mean(2, 2)
2.0
>>> geometric_mean(2, 8)
4.0
>>> geometric_mean(2, 1)
1.4142135623730951
```

Приклади (IPython):

In [ ]: geometric_mean(2, 2)

Out[ ]: 2.0

In [ ]: geometric_mean(2, 8)

Out[ ]: 4.0

In [ ]: geometric_mean(2, 1)

Out[ ]: 1.4142135623730951

Функція pyramid_volume (A, h), яка обчислює і повертає обсяг піраміди з базовою площею A та висотою h .

Приклад:

```
>>> pyramid_volume(1, 2)
0.6666666666666666
```

Example (IPython):

In [ ]: pyramid_volume(1, 2)

Out[ ]: 0.6666666666666666

Для кожної функції намагайтеся ретельно перевіряти правильність вашої реалізації.

Контрольні питання:

1. Що таке середовище Spyder і для чого воно використовується під час програмування мовою Python?

2. Для чого виконується команда «%reset» перед початком роботи?

3. У чому полягає відмінність між консоллю Python та оболонкою IPython?

4. Як відкрити, запустити та перевірити Python-файл у середовищі Spyder?

5. Що таке функція в Python?

6. Яке призначення параметрів функції?

7. Для чого використовується команда «help()»?

8. Що означає документування функції та навіщо воно потрібне?

9. Як працює функція «average(a, b)»?

10. Як обчислюється відстань між двома числами у функції «distance(a, b)»?

11. Що таке геометричне середнє та як його обчислити?

12. За якою формулою обчислюється об’єм піраміди у функції «pyramid_volume(A, h)»?

13. Чому після зміни коду функції потрібно повторно запускати файл?

14. Яким чином можна перевірити правильність роботи створеної функції?

15. Які результати повинні повертати функції «distance», «geometric_mean» та «pyramid_volume» для наведених у завданні прикладів?

Практичне заняття №2

Прості функції та умовне розгалуження (if)

Мета: ознайомитися з принципами створення простих функцій мовою Python та використанням умовного розгалуження if; сформувати практичні навички обчислення об’єму, площі поверхні, часу падіння, швидкості удару, координати точки в інтервалі, перетворення секунд у дні та визначення знаку числа; навчитися реалізовувати математичні формули у вигляді функцій, тестувати їх роботу та аналізувати отримані результати.

Будь ласка, визначте такі функції у файлі training2.py:
функція box_volume (a, b, c), яка обчислює і повертає обсяг кубоїда з довжинами ребер a , b , c .

Приклади:

```
>>> print(box_volume(1, 1, 1))
1
>>> print(box_volume(1, 2, 3.5))
7.0
>>> print(box_volume(1, 1, 0))
0
```

Приклади (IPYthon):

```
In [ ]: box_volume(1, 1, 1)
Out[ ]: 1
In [ ]: box_volume(1, 2, 3.5)
Out[ ]: 7.0
In [ ]: box_volume(1, 1, 0)
Out[ ]: 0
```

Функція fall_time (h), яка повертає час (у секундах), необхідний об’єкту, що падає з вежі висотою h (у метрах), щоб впасти на землю (ігноруючи тертя повітря). Використовуйте формулу 2.1:

$$h(t) = \frac{1}{2}gt^2 \quad \text{with} \quad g = 9.81\,\frac{\text{m}}{\text{s}^2} \tag{2.1}$$

де, h – відстань падіння через час t.

Приклади:

```
>>> print(fall_time(10))
1.4278431229270645
>>> print(fall_time(1))
0.4515236409857309
```

Приклади (IPython):

```
In [ ]: fall_time(10)
Out[ ]: 1.4278431229270645
In [ ]: fall_time(1)
Out[ ]: 0.4515236409857309
```

Функція interval_point (a, b, x), яка приймає три числа і інтерпретує a і b як початкову і кінцеву точки інтервалу, а x як частку від 0 до 1, що визначає, як далеко рухатися до b, починаючи з a .

Приклади:

```
>>> print(interval_point(100, 200, 0.5))
150.0
>>> print(interval_point(100, 200, 0.2))
120.0
```

Приклади (IPython):

```
In [ ]: interval_point(100, 200, 0.5)
Out[ ]: 150.0
In [ ]: interval_point(100, 200, 0.2)
Out[ ]: 120.0
```

Функція impact_velocity (h), яка повертає швидкість (у метрах за секунду), з якою предмет, що падає з висоти h метрів, впаде на землю. Використовуйте v (t) = g * t, при v (t) швидкість у момент часу t, і g = 9,81 м / с ^ 2.

Функція signum (x), яка:

- returns 1 if x > 0
- returns 0 if x = 0
- returns -1 if x < 0

Приклад:

```
>>> print(signum(2012))
1
```

Приклад (IPython):

```
In [ ]: signum(2012)
Out[ ]: 1
```

Завдання

Створіть файл lab2.py, який містить такі функції:

Функція seconds2days (n), яка приймає кількість секунд (як int, так і float) і перетворює кількість секунд у відповідну кількість днів. Функція повинна повертати кількість днів як змінну з плаваючою комою.

Приклад:

```
>>> seconds2days(43200)
0.5
```

Приклад (IPython):

```
In [ ]: seconds2days(43200)
Out[ ]: 0.5
```

Для вашої розваги: скільки днів 10! секунд? (10! = 3628800).

Функція box_surface (a, b, c), яка обчислює та повертає площу поверхні коробки (тобто кубоподібної) з цими довжинами ребер a , b і c . Уявіть, що нам потрібно пофарбувати поверхню коробки і, отже, нам потрібно знати її площу, щоб придбати потрібну кількість фарби.

Приклад:

```
>>> print(box_surface(1, 1, 1))
6
>>> print(box_surface(2, 2, 0))
8
```

Приклад (IPython):

```
In [ ]: box_surface(1, 1, 1)
Out[ ]: 6

In [ ]: box_surface(2, 2, 0)
```

Out[ ]: 8

Функція трикутник_площа (a, b, c), яка обчислює та повертає площу A трикутника з довжинами ребер a, b та c. Ви можете скористатися рівнянням (2.2):

$$A = \sqrt{s(s-a)(s-b)(s-c)} \quad \text{with} \quad s = \frac{a+b+c}{2} \tag{2.2}$$

Контрольні питання:

1. Що таке функція в Python і для чого вона використовується під час розв’язання обчислювальних задач?

2. Яке призначення умовного оператора if та в яких випадках його доцільно застосовувати?

3. За якою формулою обчислюється об’єм прямокутного паралелепіпеда та площа його поверхні?

4. Як визначити час вільного падіння тіла з висоти h та швидкість його удару об землю без урахування опору повітря?

5. Як працює функція signum(x) і які значення вона повинна повертати для додатного, нульового та від’ємного числа?

Практичне заняття №3
Робота зі списками

Мета: ознайомитися з реалізацією функцій у мові Python для виконання математичних обчислень, роботи зі списками та кортежами; сформувати практичні навички перетворення величин з радіанів у градуси, знаходження мінімального і максимального значень у послідовності, обчислення середнього геометричного, визначення часу коливання маятника, побудови списку квадратів чисел та підрахунку кількості входжень елемента в послідовність; навчитися створювати, тестувати та аналізувати роботу власних функцій у середовищі Python.

Реалізуйте наступні функції у файлі, який називається training3.py :
ступінь функції (x), яка приймає аргумент x в радіанах і повертає відповідне значення в градусах. Тобто, отримавши значення x , функція повинна повернути (3.1)

$$x \frac{360}{2\pi} \tag{3.1}$$

Приклад:

```
In [ ]: degree(math.pi)
Out[ ]: 180.0
```

Функція min_max (xs), яка обчислює мінімальне значення xmin елементів у списку xs та максимальне значення xmax елементів у списку, і повертає кортеж (xmin, xmax) .

Приклад:

```
In [ ]: min_max([0, 1, 2, 10, -5, 3])
Out[ ]: (-5, 10)
```

Функція geometric_mean (xs), яка обчислює середнє геометричне чисел, наведених у списку xs . Підказка: Пам'ятайте, що a ** b обчислює b (тобто приймає a-to-the-bth-потужність).

Приклад:

```
In [ ]: geometric_mean([1, 2])
```

Out[ ]: 1.4142135623730951

Завдання

Створіть файл lab3.py, який містить такі функції:

Функція swing_time (L), яка обчислює і повертає час T [у секундах], необхідний ідеалізованому маятнику довжиною L [у метрах] для завершення одного коливання, використовуючи рівняння (3.2)

$$T = 2\pi\sqrt{\frac{L}{g}} \quad \text{with} \quad g = 9.81\,\mathrm{m/s^2} \tag{3.2}$$

Приклад:

In [ ]: swing_time(1)

Out[ ]: 2.0060666807106475

Функція range_squared (n), яка приймає ціле невід'ємне значення n і повертає список [0, 1, 4, 9, 16, 25, ..., (n-1) ^ 2] . Якщо n дорівнює нулю, функція повинна повернути порожній список.

Приклад:

In [ ]: range_squared(3)

Out[ ]: [0, 1, 4]

Функція count (element, seq), яка підраховує, як часто даний елемент елемент зустрічається в заданій послідовності seq , і повертає це ціле значення. Наприклад, count (2, list (range (5))) повинен повернути 1 .

Приклад:

In [ ]: count('dog',['dog', 'cat', 'mouse', 'dog'])

Out[ ]: 2

In [ ]: count(2, list(range(5)))

Out[ ]: 1

Контрольні питання:

1. Яке призначення функції degree(x) та за якою формулою виконується перетворення радіанів у градуси?

2. Що таке кортеж у Python і чому функція min_max(xs) повертає саме кортеж?

3. Як обчислюється середнє геометричне елементів списку у функції geometric_mean(xs)?

4. За якою формулою визначається період коливання маятника у функції swing_time(L)?

5. Яким чином функції range_squared(n) і count(element, seq) працюють зі списками та послідовностями в Python?

Практичне заняття №4
Введення / виведення файлів

Мета: ознайомитися з основами роботи з файлами в Python, навчитися відкривати, зчитувати й обробляти текстові та csv-файли, формувати навички опрацювання рядкових даних, списків і числових послідовностей, а також засвоїти принципи створення функцій для обчислення середнього значення, підрахунку слів у файлі, знаходження квадратних коренів елементів списку та виділення потрібної інформації з текстових даних.

Створіть файл training4.py, який забезпечує:
функція line_average (ім'я файлу), яка приймає рядок імені файлу, що містить ім'я файлу, який підлягає обробці. Функція повинна відкрити та прочитати цей файл. Очікується, що файл буде містити числа, розділені комами (формат відомий як файл із розділеними комами значення, короткий csv). Функція повинна обчислювати середнє значення для кожного рядка та повертати середні значення у списку. Наприклад, якщо ми викликаємо line_average ("data.csv"), а файл data.csv читає:

```
1,2
1,1,1,1
-1,0,1
42,17
```

тоді функція повинна повернути список [1,5, 1,0, 0,0, 29,5] .

Національне управління океанічних і атмосферних досліджень США (NOAA) забезпечує спостереження поточних погодних умов по всьому світу. Використовуючи спеціальний 4-літерний код станції для Саутгемптона – це EGHI – ми можемо знайти кількісну інформацію про поточні погодні умови, ввівши веб-браузер на: http://tgftp.nws.noaa.gov/data/observations/metar/decoded/EGHI.TXT.

Ми пропонуємо таку функцію (яка буде включена в Training4.py), яка завантажує цю веб-сторінку та повертає її у вигляді рядка:

```
def noaa_string():
```

```
url = "http://tgftp.nws.noaa.gov/data/observations/metar/decoded/EGHI.TXT"
noaa_data_string = urllib.request.urlopen(url).read()
return noaa_data_string.decode("utf-8")
```

Бібліотека urllib.request дозволяє отримати доступ до веб-сторінки, як файл, за допомогою функції urlopen (), і ви повинні включити імпорт urllib.request на початку вашого файлу training4.py .

Викличте функцію noaa_string із запиту Python і перевірте повернене значення.

Ваше завдання – написати функцію noaa_temperature (s), яка повинна взяти рядок s як повернутий з noaa_string () як вхідний аргумент, витягнути з рядка температуру в градусах Цельсія і повернути цю температуру як ціле число:

```
In [ ]: noaa_temperature(noaa_string())
Out[ ]: 10
```

NOAA може часом змінювати кількість і порядок рядків у даних, але можна припустити, що формат рядка, що містить дані про температуру, не змінюється.

Завдання

Створіть файл lab4.py, який містить:

функція seq_sqrt (xs), яка бере список невід’ємних чисел xs з елементами [x0, x1, x2, ..., xn] і повертає список [sqrt (x0), sqrt (x1), sqrt (x2), ..., sqrt (xn)] . Іншими словами, функція бере список чисел і повертає список однакової довжини, що містить квадратний корінь для кожного числа у списку.

Середнє значення функції (xs), яке приймає послідовність xs чисел і повертає (арифметичне) середнє (тобто середнє значення).

Приклад:

```
In [ ]: mean([0, 1, 2])
Out[ ]: 1.0
```

Функція wc (ім'я файлу), яка повертає кількість слів у назві файлу. Назва wc розшифровується як Word Count. Щоб розділити рядок s на слова, використовуйте s.split () для цієї вправи (тобто поведінка методу split () тут використовується для визначення того, що таке слово).

Приклад 1: Для файлу data.txt із вмістом:

One Two

виклик функції wc ('data.txt') повинен повернути 2.

Приклад 2 : Для файлу data.txt із вмістом:

One Two

Three Four Five

виклик функції wc ('data.txt') повинен повернути 5.

Контрольні питання:

1. Які основні операції введення та виведення файлів підтримує мова Python?

2. Для чого використовується функція відкриття файлу та які режими роботи з файлами існують?

3. Як із текстового рядка, що містить числа, розділені комами, отримати окремі значення для подальшої обробки?

4. Яким чином обчислюється середнє значення елементів у рядку файлу або у списку чисел?

5. Як у Python можна підрахувати кількість слів у текстовому файлі та виділити потрібні дані з текстового рядка?

Практичне заняття №5
Списки, файли, винятки, рядки

Мета: ознайомитися з основами роботи зі списками, текстовими файлами, рядками та винятками в мові Python; сформувати практичні навички створення функцій для пошуку підрядків у файлах, підрахунку голосних літер у рядках, обробки помилок відкриття файлів, виконання операцій над послідовностями чисел, обчислення векторного добутку, множення послідовності на скаляр, формування списку степенів числа та реалізації умовного вибору результату залежно від значення параметра.

Надайте таку функціональність у файлі training5.py:
напишіть функцію count_sub_in_file (ім'я файлу), яка приймає два аргументи: підрядок s (типу рядка) та ім'я файлу (рядка типу). Функція повинна повертати кількість випадків s у файлі, заданому через ім'я файлу. (Зверніть увагу, що кожен рядковий об’єкт має кількість методів, яку слід використовувати тут.)

Ви можете перевірити свою функцію count_sub_in_file на «Пригоди Аліси в країні чудес», яку ви можете завантажити з http://www.gutenberg.org/files/28885/28885-8.txt

Як часто трапляється підрядок «Аліса»? Очікуйте кілька сотень.

Як часто трапляється підрядок «аліса»? Очікуйте дуже невелику кількість.

Змініть функцію count_sub_in_file(filename, s), так що , якщо файл з ім'ям файлу не може бути відкритий, значення -1 повертається (замість числа підстрічок s).

Напишіть функцію count_vowel (s), яка повертає кількість букв 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' у заданому рядку s (повертане значення має ціле число).

Приклади :

```
In [ ]: count_vowels('This is a test')
```

```
Out[ ]: 4
In [ ]: count_vowels('aoeui')
Out[ ]: 5
In [ ]: count_vowels('aoeuiAOEUI')
Out[ ]: 10
In [ ]: count_vowels('N0 v0w3ls @t @ll ln thls strlng')
Out[ ]: 0
```

Завдання

Надайте таку функціональність у файлі lab5.py:

напишіть функцію vector_product3 (a, b), яка приймає дві послідовності чисел. І послідовність a, і послідовність b мають три елементи. З входами a = [ax, ay, az] і b = [bx, by, bz] , функція повинна повернути список, що містить векторний добуток 3d-векторів a і b , тобто значенням повернення є список:

[ay * bz - az * by, az * bx - ax * bz, ax * by - ay * bx].

Приклади:

```
In [ ]: vector_product3([1, 0, 0], [0, 1, 0])
Out[ ]: [0, 0, 1]
In [ ]: vector_product3([1, 2, 4], [3, 5, 6])
Out[ ]: [-8, 6, -1]
```

Напишіть функцію seq_mult_scalar (a, s), яка бере список чисел a та скаляр (тобто число) s . Для входу a = [a0, a1, a2, .., an] функція повинна повертати [s * a0, s * a1, s * a2, ..., s * an] .

Приклад:

```
In [ ]: seq_mult_scalar([-4, 9, 1], 10)
Out[ ]: [-40, 90, 10]
```

Функція повноважень (n, k), яка повертає список [1, n, n^2, n^3, ..., n^k], де k - ціле число. Зверніть увагу, що у списку повинно бути k + 1 елементів.

Приклад:

```
In [ ]: powers(2, 3)
Out[ ]: [1, 2, 4, 8]
In [ ]: powers(0.5, 2)
Out[ ]: [1.0, 0.5, 0.25]
```

Функція Traffic_light (навантаження), яка приймає навантаження з числом із плаваючою точкою . Функція повинна повернути рядок:

– «green» для значень навантаження нижче 0,7.

– «amber» для значень навантаження, що дорівнює або перевищує 0,7, але менше 0,9

– «red» для значень навантаження, що дорівнює 0,9 або більше 0,9

Приклад:

```
In [ ]: traffic_light(0.5)
Out[ ]: 'green'
```

Контрольні питання:

1. Що таке список у Python і які основні операції над списками використовуються під час розв’язання обчислювальних задач?

2. Як у Python виконується робота з текстовими файлами та які помилки можуть виникати під час їх відкриття?

3. Для чого використовуються винятки в Python і як вони допомагають обробляти помилки в програмі?

4. Яким чином можна підрахувати кількість входжень підрядка у тексті та кількість голосних символів у рядку?

5. Як реалізується умовне розгалуження для повернення різних текстових результатів у функції traffic_light(load)?

Практичне заняття №6
Функції вищого порядку

Мета: ознайомитися з поняттям функцій вищого порядку в мові Python та особливостями передавання функції як аргументу до іншої функції; сформувати практичні навички застосування функцій до елементів списку, обчислення значень функції в заданих точках, підсумовування результатів обчислення, а також використання параметрів за замовчуванням під час створення власних функцій; навчитися реалізовувати універсальні обчислювальні засоби для роботи з числовими послідовностями та аналізувати результати їх виконання.

Створіть файл training6.py і заповніть такими функціями:
функція positive_places(f, xs), який приймає в якості аргументів деякої функції F і список чисел хза і повертає список тих, і тільки ті елементи – x з xs, для яких f(x) строго більше нуля.

Приклад 1 :

```
In [ ]: def my_f(x):
  ...:     return x ** 3
  ...:
In [ ]: positive_places(my_f, [1, 2, -1, -2, 3, 42, -9])
Out[ ]: [1, 2, 3, 42]
In [ ]: positive_places(my_f, [1, 2, 3, 4, 5])
Out[ ]: [1, 2, 3, 4, 5]
In [ ]: positive_places(my_f, [])
Out[ ]: []
In [ ]: positive_places(my_f, [-1, -2, -3, -4, -5])
Out[ ]: []
```

Приклад 2 :

```
In [ ]: def f(x):
  ...:     return 2 * x + 4
  ...:
In [ ]: positive_places(f, [10, 1, -3, -1.5, 0, 0.5])
```

```
Out[ ]: [10, 1, -1.5, 0, 0.5]
```

Напишіть функцію eval_f_0123 (f), яка обчислює функцію f = f (x) у положеннях x = 0, x = 1, x = 2 та x = 3 . Функція повинна повернути список [f (0), f (1), f (2), f (3)] .

Приклад 1 :

```
In [ ]: def square(x):
  ...:     return x * x
  ...:
In [ ]: eval_f_0123(square)
Out[ ]: [0, 1, 4, 9]
```

Приклад 2 :

```
In [ ]: def cubic(x):
  ...:     return x ** 3
  ...:
In [ ]: eval_f_0123(cubic)
Out[ ]: [0, 1, 8, 27]
```

Приклад 3 :

```
In [ ]: def stars(x):
  ...:     return "*" * x
  ...:
In [ ]: eval_f_0123(stars)
Out[ ]: ['', '*', '**', '***']
```

Завдання

Створіть файл lab6.py та заповніть такими функціями:

функція eval_f (f, xs), яка приймає функцію f = f (x) та список xs значень, які слід використовувати як аргументи для f. Функція eval_f повинна застосувати функцію f згодом до кожного значення x у xs і повернути список fs значень функції. Тобто для вхідного аргументу xs = [x0, x1, x2, ..., xn] функція eval_f (f, xs) повинна повертати [f (x0), f (x1), f (x2), ..., f (xn)] .

Приклад 1 :

```
In [ ]: def square(x):
  ...:     return x * x
```

```
   ...:
In [ ]: eval_f(square, [-1, 10, 20, 42])
Out[ ]: [1, 100, 400, 1764]
```

Приклад 2 :

```
In [ ]: import math
In [ ]: eval_f(math.sqrt, [1, 2, 4, 9])
Out[ ]: [1.0, 1.4142135623730951, 2.0, 3.0]
```

Приклад 3 :

```
In [ ]: def sign(x):
   ...:     if x > 0:
   ...:         return 1
   ...:     elif x < 0:
   ...:         return -1
   ...:     else:
   ...:         return 0
   ...:
In [ ]: sign(-1.1)
Out[ ]: -1
In [ ]: sign(0.1)
Out[ ]: 1
In [ ]: sign(0.0)
Out[ ]: 0
In [ ]: eval_f(sign, [-0.2, -0.1, 0, 0.1, 0.2, 0.3])
Out[ ]: [-1, -1, 0, 1, 1, 1]
```

Функція sum_f (f, xs), яка повертає суму значень функції f, обчислених за значеннями x0, x1, x2, ..., xn, де xs = [x0, x1, x2, ..., xn] .

Приклад 1 :

```
In [ ]: def f(x):
   ...:     return x
   ...:
In [ ]: sum_f(f, [1, 2, 3, 10])
Out[ ]: 16
```

Приклад 2 :

```
In [ ]: def square(x):
```

```
...:     return x * x
...:
```

```
In [ ]: sum_f(square, [1, 2, 3, 10])
Out[ ]: 114
```

Функція box_volume_UPS(a, b, c), яка повертає обсяг коробки з довжинами країв a , b і c . Вхідні дані повинні бути вказані в дюймах, а вихідні дані - у дюймах ^ 3.

Стандартні розміри експрес-коробки складають a = 13 дюймів, b = 11 дюймів та c = 2 дюймів.

Ваша функція повинна використовувати ці значення для a , b та c, якщо не передбачено інших.

Приклади:

```
In [ ]: box_volume_UPS()
Out[ ]: 286
In [ ]: box_volume_UPS(a=10, b=10, c=10)
Out[ ]: 1000
In [ ]: box_volume_UPS(c=5)
Out[ ]: 715
```

Контрольні питання:

1. Що таке функція вищого порядку в Python?

2. Що означає передати функцію як аргумент до іншої функції?

3. Яким чином функція eval_f(f, xs) обробляє список значень xs?

4. У чому полягає відмінність між функціями eval_f(f, xs) та sum_f(f, xs)?

5. Для чого використовується функція positive_places(f, xs) і за якою умовою елементи списку потрапляють до результату?

6. Що таке параметри за замовчуванням у функції та як вони використовуються у box_volume_UPS(a, b, c)?

7. Які переваги дає використання функцій вищого порядку під час розв’язання обчислювальних задач у Python?

8. Як працює функція eval_f_0123(f) і які значення вона повинна повертати?

9. Чим відрізняється повернення списку значень функції від повернення одного підсумкового числового результату?

10. У яких практичних задачах доцільно використовувати функції, що приймають інші функції як аргументи?

Практичне заняття №7
Словники, рекурсія, пошук коренів (метод Ньютона)

Мета: ознайомитися з використанням словників у мові Python для підрахунку частоти появи символів у рядку, засвоїти принципи чисельного наближення похідної функції методом центральних різниць, сформувати практичні навички реалізації методу Ньютона-Рафсона для пошуку коренів функції, навчитися застосовувати рекурсію для розв'язання задач аналізу рядків, зокрема для перевірки паліндромів, а також відпрацювати обробку виняткових ситуацій у разі перевищення допустимої кількості ітерацій.

Створіть файл training7.py, який надає такі функції:
функція count_chars (s), яка приймає рядок s і повертає словник. Ключі словника – це набір символів, що зустрічаються в рядках s. Значення кожного ключа – це кількість випадків, коли цей символ зустрічається в рядку s. Приклади:

```
In [ ]: count_chars('x')
Out[ ]: {'x': 1}
In [ ]: count_chars('xxx')
Out[ ]: {'x': 3}
In [ ]: count_chars('xxxyz')
Out[ ]: {'x': 3, 'y': 1, 'z': 1}
In [ ]: count_chars('Hello World')
Out[ ]: {' ': 1, 'H': 1, 'W': 1, 'd': 1, 'e': 1, 'l': 3, 'o': 2, 'r': 1}
```

Зверніть увагу, що порядок, у якому пари ключ-значення перераховані у вихідному словнику, не важливий.

Функція derivative(f, x), яка обчислює числове наближення першої похідної функції f (x), використовуючи центральні різниці. Значення, яке повертає функція, є (7.1):

$$\frac{f\left(x+\frac{\epsilon}{2}\right)-f\left(x-\frac{\epsilon}{2}\right)}{\epsilon} \quad \text{where} \quad \epsilon = 10^{-6} \tag{7.1}$$

Приклад :

```
In [ ]: def f(x):
  ...:     return x * x
In [ ]: derivative(f, 0)
Out[ ]: 0.0
In [ ]: derivative(f, 1)
Out[ ]: 2.0000000000575113
In [ ]: derivative(f, 2)
Out[ ]: 4.000000000115023
```

Зверніть увагу, що ви можете отримати не зовсім такі цифри, як показано вище.

Змініть похідну функцію таким чином, щоб додаткові треті параметри eps представляли грецьку букву epsilon у наведеному вище рівнянні. Параметр eps повинен за замовчуванням мати значення 1e-6.

Приклади :

```
In [ ]: import math
In [ ]: derivative(math.exp, 0, eps=0.1)
Out[ ]: 1.000416718753101
In [ ]: derivative(math.exp, 0)
Out[ ]: 1.0000000000287557
In [ ]: derivative(math.exp, 0, eps=1e-6)
Out[ ]: 1.0000000000287557
```

Завдання

Збережіть файл training7.py під новою назвою lab7.py (оскільки вам потрібно буде використовувати функцію похідної із навчальної частини в оцінюваній частині) та додайте такі функції:

функція newton(f, x, feps, maxit), яка приймає функцію f(x) та початкову здогадку x для кореня функції f(x), допустимий допуск feps та максимальну кількість ітерацій, дозволених maxit. Функція Ньютона повинна використовувати наступний алгоритм Ньютона-Рафсона:

```
while |f(x)| > feps, do
  x = x - f(x) / fprime(x)
```

де fprime (x) – наближення першої похідної (df(x)/dx) у положенні x. Вам слід використовувати похідну функцію з навчальної частини цієї лабораторії.

Переконайтеся, що ви скопіювали визначення похідної функції з training7.py у lab7.py (є більш елегантні способи зробити це, але для цілей оцінки це найпростіший спосіб, який ми рекомендуємо).

Якщо максимум або менше ітерацій необхідні для |f(x)| щоб стати меншим за feps , тоді слід повернути значення x:

```
In [ ]: def f(x):
   ....:     return x ** 2 - 2
   ....:
In [ ]: newton(f, 1.0, 0.2, 15)
Out[ ]: 1.4166666666783148
In [ ]: newton(f, 1.0, 0.2, 15) - math.sqrt(2)
Out[ ]: 0.002453104305219611
In [ ]: newton(f, 1.0, 0.001, 15)
Out[ ]: 1.4142156862748523
In [ ]: newton(f, 1.0, 0.001, 15) - math.sqrt(2)
Out[ ]: 2.1239017571339502e-06
In [ ]: newton(f, 1.0, 0.000001, 15) - math.sqrt(2)
Out[ ]: 1.5949463971764999e-12
```

Якщо для функції ньютон потрібно більше ніж максимум ітерацій , тоді функція ньютон повинна викликати виняток RuntimeError із повідомленням: Помилка після ітерацій X, де X слід замінити кількістю ітерацій:

```
In [23]: def g(x):
   ....:     return math.sin(x) + 1.1  # has no root!
   ....:
In [24]: newton(g, 1.0, 0.02, 15)
Traceback (most recent call last):
  File "<ipython-input-6-0a9db3f67256>", line 1, in <module>
    newton(g, 1.0, 0.02, 15)
  File "..lab7.py", line 16, in newton
    raise RuntimeError("Failed after %d iterations" % maxit)
RuntimeError: Failed after 15 iterations
```

The relevant line of Python to be executed if the number of maxit iterations is reached, is

raise RuntimeError("Failed after %d iterations" % maxit)

Функція is_palindrome (s), яка приймає рядок s і повертає значення True, якщо s є паліндром, а в іншому випадку повертає False . (Зверніть увагу, що повернене значення – це не рядок "True", а спеціальне значення Python True. Те саме стосується False .)

Паліндром – це слово, яке читається так само назад, як і вперед, наприклад, мадам, байдарка, радар та ротатор .

Підказки щодо запропонованого алгоритму:

– якщо s – порожній рядок, то це паліндром.

– якщо s – це рядок з одним символом, то це паліндром.

– якщо перша буква s однакова з останньою буквою s, то s є паліндром, якщо інші букви s (тобто, починаючи з другої літери, за винятком останньої букви), є паліндромом.

Приклади:

In [ ]: is_palindrome('rotator')

Out[ ]: True

In [ ]: is_palindrome('radiator')

Out[ ]: False

In [ ]: is_palindrome('ABBA')

Out[ ]: True

Ми розглядаємо маленькі літери (наприклад, a) та великі літери (наприклад, A) як різні літери для цієї вправи: рядок ABba , таким чином, не є паліндромом.

Порада: якщо ви боретеся з концепцією рекурсії, знайдіть трохи часу, щоб вивчити результати цього рекурсивного факторіального обчислення.

Додаток

True і False є спеціальними булевими значеннями (типу bool Python) і відрізняються від рядків. Ось кілька демонстрацій цього:

```
In [ ]: a = True
In [ ]: b = "True"
In [ ]: type(a)
Out[ ]: bool
In [ ]: type(b)
Out[ ]: str
```

У наведеній вище функції is_palindrome () потрібно повернути значення bool True або False , але не рядок "True" або рядок "False".

Контрольні питання:

1. Що таке словник у Python і для чого він використовується під час підрахунку символів у рядку?

2. Як працює функція count_chars(s) і який вигляд має результат її виконання?

3. У чому полягає ідея чисельного обчислення похідної функції методом центральних різниць?

4. Яке призначення параметра eps у функції derivative(f, x, eps) і як його значення впливає на точність обчислення?

5. У чому полягає сутність методу Ньютона-Рафсона для знаходження коренів функції?

Практичне заняття №8
Різне застосування, побудова графіків

Мета: ознайомитися з практичним застосуванням мови Python для обчислення значень математичних функцій, формування даних для побудови графіків, візуалізації результатів за допомогою бібліотек matplotlib або pylab, а також зберігання графічних результатів у файли; сформувати практичні навички побудови графіків кількох функцій на одній координатній площині, аналізу точок їх перетину, використання чисельних методів пошуку коренів за допомогою scipy.optimize.brentq та роботи зі словниками, зокрема створення оберненого словника.

Створіть файл training8.py з такими функціями:
напишіть функцію f1 (x), яка приймає число x як вхід і обчислює та повертає (8.1):

$$f_1(x) = \cos(2\pi x)\exp(-x^2) \quad (8.1).$$

Напишіть функцію f2 (x), яка приймає число x як вхідне та обчислює та повертає (8.2):

$$f_2(x) = \log(x + 2.1) \quad (8.2),$$

де log відноситься до природного логарифму (ім'я функції Python – math.log).

А функція positive_places(f,xs), який приймає в якості аргументів деякої функції f і список чисел xs і повертає список тих, і тільки ті елементи – x з xs, для яких f(x) строго більше нуля. Це завдання було дано вже як навчальну вправу в лабораторній 6.

У цій лабораторній, ми запитуємо, що ви пишете ту ж функцію без використання для або в той час як цикл в списку практики розуміння або застосування фільтра.

Завдання

Створіть файл lab8.py із такими функціями. Як вам потрібно буде використовувати f1 і f2 у вправах нижче, ви можете скопіювати файл

training8.py на нове ім'я lab8.py, а потім додати додаткові функції lab8.py .

Напишіть функцію create_plot_data (f, xmin, xmax, n), яка повертає кортеж (xs, ys), де xs і ys – дві послідовності, кожна з яких містить n чисел (8.3):

$$\mathtt{xs} = [x_0, x_1, \ldots, x_{n-1}] \quad \text{with} \quad x_i = x_{\min} + i\frac{x_{\max} - x_{\min}}{n-1}$$
$$\mathtt{ys} = [f(x_0), f(x_1), \ldots, f(x_{n-1})] \tag{8.3}$$

Очікується, що функція буде працювати при будь-якому n> = 2.

Приклади (тут кортеж повернутих послідовностей являє собою кортеж списків):

```
In [ ]: def f(x):
...:     return x * 10
...:
In [ ]: create_plot_data(f, -1, 1, 2)
Out[ ]: ([-1.0, 1.0], [-10.0, 10.0])
In [ ]: create_plot_data(f, 0, 2, 5)
Out[ ]: ([0.0, 0.5, 1.0, 1.5, 2.0], [0.0, 5.0, 10.0, 15.0, 20.0])
In [ ]: def f(x):
...:     return 0
...:
In [ ]: create_plot_data(f, 0, 1.5, 4)
Out[ ]: ([0.0, 0.5, 1.0, 1.5], [0, 0, 0, 0])
```

Використовуючи pylab або matplotlib, напишіть функцію myplot (), яка обчислює f1 (x) [звичайно, викликаючи функцію f1, або – ще краще – викликаючи create_plot_data] та графіки f1(x), використовуючи 1001 бал для x, починаючи з - Від 2 до +2. Функція повинна повернути None .

Потім розширіть цю функцію myplot, щоб також побудувати графік f2(x) на тому ж графіку.

Щоб побудувати дві або більше кривих на одній і тій же фігурі, просто двічі використовуйте команду plot ().

Позначте вісь x та надайте легенду, що відображає назву функції (тобто f1 та f2) для двох кривих.

Розширте функцію myplot () так, щоб вона зберігала файл png графіка (з назвою plot.png) та pdf файл графіка (з назвою plot.pdf) через команди pylab / matplotlib.

(Примітка: Як правило, краще використовувати pdf-файли, а не png-файли, оскільки pdf-файли базуються на векторній графіці та забезпечують більш високу якість друку, і їх можна збільшувати, не маючи піксельних зображень. З іншого боку, файли png є гарним вибором для наприклад, включити у веб-сторінки.)

Використовуйте панель навігації pylab (внизу малюнка), щоб збільшити зображення. При x> 0, яке значення x має функція, де f1(x) = f2(x)?

(Якщо ви використовуєте консоль IPython і за замовчуванням ваші графіки відображаються вбудованими, тобто на консолі IPYthon, ви не можете збільшити малюнок. Якщо так, то слід використовувати команду % matplotlib qt у консолі. Після цього це, наступна фігура сюжету повинна з’являтися у власному спливаючому вікні та дозволяти збільшувати та зменшувати малюнок.

Щоб повернутися до цифр, що відображаються на консолі IPython, використовуйте % matplotlib inline).

Використовуючи цю графічну інформацію, напишіть функцію find_cross(), яка використовує scipy.optimize.brentq, щоб знайти значення x (приблизно), для якого f1 (x) = cos (2 * pi * x) * exp (-x * x) і f2 (x) = log (x + 2.1) мають однакове значення. Нас цікавить лише рішення, де x> 0 .

Ваша функція find_cross () повинна повернути апроксимацію кореня, який повертає scipy.optimize.brentq (налаштування допуску за замовчуванням нормальні).

Напишіть функцію reverse_dic (d), яка приймає словник d як вхідний аргумент і повертає словник r . Якщо словник d має ключ k і пов'язане з ним значення v , то словник r повинен мати ключ v і значення k . (Очікується, що це буде працювати лише для словників, які мають унікальний набір значень, хоча перевіряти це не потрібно.)

Контрольні питання:

1. Для чого використовується функція create_plot_data(f, xmin, xmax, n) і які дані вона повинна повертати?

2. Що таке кортеж у Python і чим він відрізняється від списку?

3. Які можливості надають бібліотеки matplotlib, pylab і scipy.optimize під час розв’язання прикладних задач?

4. Як побудувати графіки двох функцій на одному рисунку та для чого використовуються підписи осей і легенда?

5. Для чого зберігати побудований графік у форматах PNG і PDF та в чому полягає різниця між цими форматами?

6. Яким чином за графіком і чисельним методом можна знайти точку перетину двох функцій?

7. Як працює функція reverse_dic(d) і за якої умови вона коректно формує обернений словник?

Практичне заняття №9

Чисельне інтегрування, numpy

Мета: ознайомитися з основами чисельного інтегрування в мові Python, зокрема з реалізацією складеного трапецієподібного правила для наближеного обчислення визначених інтегралів; сформувати практичні навички оцінювання похибки чисельного інтегрування, використання бібліотеки scipy для точнішого обчислення інтегралів, опрацювання числових послідовностей та обчислення стандартного відхилення; навчитися застосовувати словники для побудови простих криптографічних відображень, реалізовувати функції кодування та декодування повідомлень, а також використовувати засоби Python і бібліотеки numpy/scipy для розв'язання прикладних обчислювальних задач

Завантажте файл training9.py (https://www.southampton.ac.uk/~fangohr/training/python/labs/lab9/training9.py) і відкрийте його у своєму редакторі. Натисніть F5, щоб усі визначення у файлі були видимими в підказці Python. Додайте до цього файлу всі функції, які вам запропоновано записати.

Напишіть функцію trapez (f, a, b, n).

Передумови: рівняння нижче показує, що інтеграл I функції f (x) від x = a до x = b може бути апроксимований A через так зване складене правило трапецієподібного інтегрування (9.1):

$$I = \int_a^b f(x)\mathrm{d}x \approx A = \frac{h}{2}\left(f(a) + f(b) + 2\sum_{i=1}^{n-1} f(x_i)\right) \quad (9.1),$$

де h = (ba) / n, x i = a + i * h, a n - кількість підрозділів.

Приклад:

Рисунок 9.1 демонструє ідею для $f(x) = x^2$:

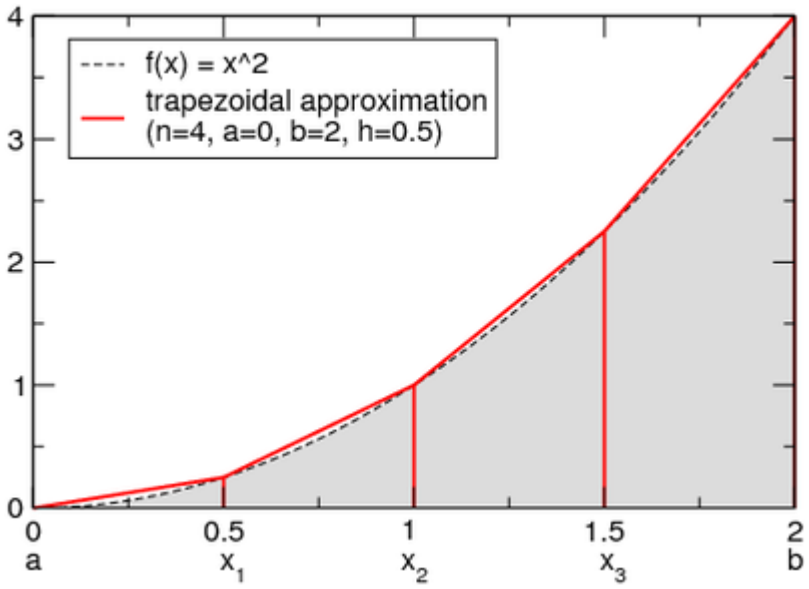


Рисунок 9.1 – $f(x) = x^2$

Складене трапецієподібне правило точно обчислює область під червоною лінією, де червона лінія є наближенням фактичної функції, що цікавить (тут $f(x) = x^2$), показана чорною пунктирною лінією.

Вправа.

Напишіть функцію trapez (f, a, b, n), яка задана:

– функція f, яка залежить від одного параметра (тобто f (x));

– нижня (a) та верхня (б) межа інтеграції;

– та кількість підрозділів (n), які будуть використані.

Функція повинна використовувати складене трапецієподібне правило для обчислення A і повернення цього значення.

Приклади.

```
In [ ]: def f(x):
   ...:     return x
   ...:
In [ ]: trapez(f, 0, 1, 1)
Out[ ]: 0.5
In [ ]: trapez(f, 0, 1, 5)
Out[ ]: 0.5
In [ ]: def f2(x):
   ...:     return x * x
```

...:

In [ ]: trapez(f2, 0, 1, 1)

Out[ ]: 0.5

In [ ]: trapez(f2, 0, 1, 10)

Out[ ]: 0.3350000000000001

In [ ]: trapez(f2, 0, 1, 100)

Out[ ]: 0.33335000000000004

In [ ]: trapez(f2, 0, 1, 1000)

Out[ ]: 0.33333349999999995

Передумови. У цій частині практичної роботи ми працюємо з простими криптографічними кодами. Уявіть, що Аліса хоче надіслати секретне повідомлення Бобу і не хоче, щоб треті сторони могли зрозуміти його повідомлення. Тут ми уявляємо, що Аліса та Боб можуть домовитись про певний код, перш ніж їм потрібно буде обмінюватися повідомленнями.

Аліса та Боб використовуватимуть один і той же ключ для (i) кодування повідомлення (тобто переходу з простого тексту на щось менш читабельне), а пізніше (ii) декодування кодованого повідомлення назад у звичайний текст. Це відоме як криптографія симетричного ключа, де один і той же ключ використовується як для шифрування, так і для дешифрування . На рисунку 9.2 нижче це схематично показано. «Зашифроване» повідомлення на зображенні називається «кіфертекстом», а код, що використовується для кодування та декодування, – «загальний секретний ключ».

Код у цій лабораторії – це відображення, яке приймає (вхідну) літеру алфавіту та відображає її на іншу (вихідну) букву. Ми можемо представити такий код на Python через словник, де ключі словника - це вхідні символи, а значення – вихідні символи.

Наприклад, давайте розглянемо тривіальний код, який замінює букву «e» у вхідних даних буквою «x» на виході, і одночасно замінює букву «x» у вхідних даних буквою «e» у вихідних. Це можна записати як e-> x та x-> e.

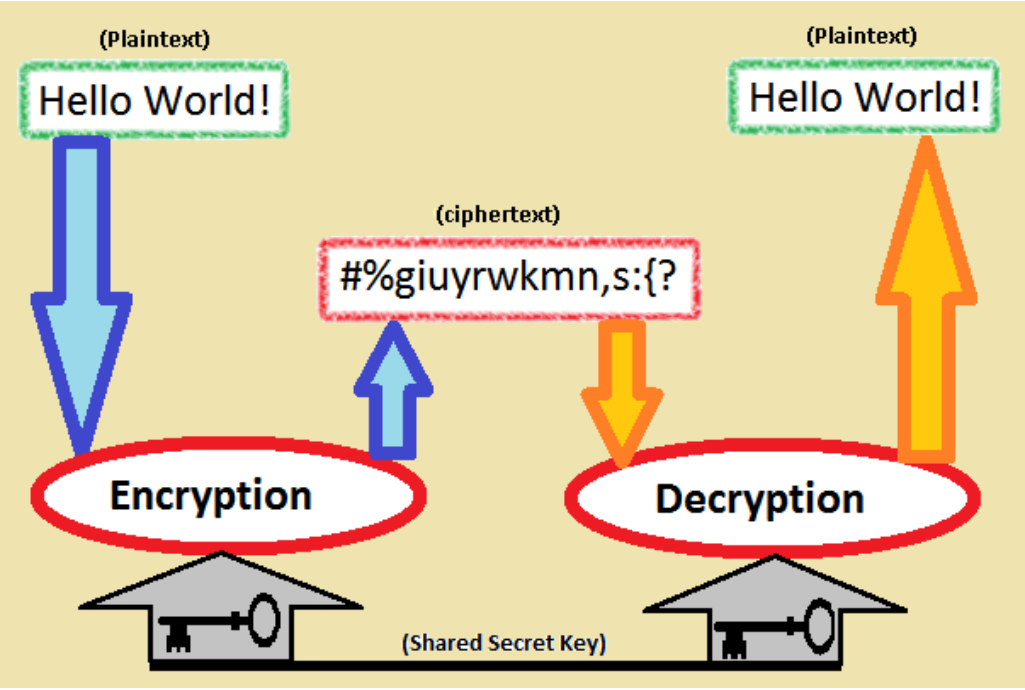


Рисунок 9.2 – Обмін ключами

Це може бути представлено словником {'x': 'e', 'e': 'x'} . Функція code1 забезпечує саме цей словник:

```
In [ ]: mycode = code1()
In [ ]: print(mycode)
{'x': 'e', 'e': 'x'}
```

Ми використовуємо домовленість, що якщо символ не знайдений у ключах словника, символ не повинен змінюватися кодом.

Тепер ми кодуємо рядок Hello World, використовуючи відображення, описане mycode , і отримуємо рядок Hxllo World .

(Якщо ми хочемо кодувати інформацію без втрати даних, нам потрібно переконатися, що жоден з двох ключів не відповідає одному значенню, тобто відображення повинно бути ін’єктивним. Пізніше ми хочемо змінити відображення – щоб декодувати кодоване повідомлення – і потрібно, щоб відображення мало бути біективним, тобто між наборами вхідних та вихідних даних повинна існувати відповідність один до одного.)

Вправа.

Напишіть кодування функції (код, повідомлення), яке приймає

аргументи:

code: це словник, що описує код

msg: це рядок

Функція повинна застосувати відображення до кожного символу рядка, як описано вище, і повернути закодований вихідний рядок.

Приклади

```
In [  ]: code = code1()
In [  ]: print(code)
{'x': 'e', 'e': 'x'}
In [  ]: encode(code, "Hello World")
Out[  ]: 'Hxllo World'
In [  ]: encode(code, "Jimi Hendrix")
Out[  ]: 'Jimi Hxndrie'
In [  ]: encode(code, "The name xenon (Xe) is derived from greek for strange.")
Out[  ]: 'Thx namx exnon (Xx) is dxrivxd from grxxk for strangx.'
```

Речення "швидка коричнева лисиця стрибає над ледачою собакою" містить усі літери англійського алфавіту і може бути тут корисним як приклад:

```
In [ ]: msg = "the quick brown fox jumps over the lazy dog"
In [ ]: encode(code1(), msg)
Out[ ]: 'thx quick brown foe jumps ovxr thx lazy dog'
```

Файл training9.py містить інші коди, наприклад code2, який відображає i-> s , s-> g та g-> i :

```
In [ ]: code2()
Out[ ]: {'g': 'i', 'i': 's', 's': 'g'}
In [ ]: encode(code2(), msg)
Out[ ]: 'the qusck brown fox jumpg over the lazy doi'
```

Код, наданий функцією code3, робить закодоване повідомлення досить нечитабельним:

```
У [22]: print (code3 ())
 {'!': '?', '': '$', '#': '.', '$': '', '.': '#', '?': '!', 'A' : 'B', 'C': 'D',
  'B': 'C', 'E': 'F', 'D': 'E', 'G': 'H', 'F': 'G', 'I': 'J', 'H ':' Я ',' К ':' Л ',
  'J': 'K', 'M': 'N', 'L': 'M', 'O': 'P', 'N': 'O', 'Q': 'R', 'P ':' Q ',' S ':' T ',
```

```
'R': 'S', 'U': 'V', 'T': 'U', 'W': 'X', 'V': 'W', 'Y': 'Z', 'X ':' Y ',' Z ':' A ',
'a': 'b', 'c': 'd', 'b': 'c', 'e': 'f', 'd': 'e', 'g': 'h', 'f ':' g ',' i ':' j ',
'h': 'i', 'k': 'l', 'j': 'k', 'm': 'n', 'l': 'm', 'o': 'p', 'n ':' o ',' q ':' r ',
'p': 'q', 's': 't', 'r': 's', 'u': 'v', 't': 'u', 'w': 'x', 'v ':' w ',' y ':' z ',
'x': 'y', 'z': 'a'}
In [23]: msg
Out[23]: 'the quick brown fox jumps over the lazy dog'
In [24]: encode(code3(), msg)
Out[24]: 'uif$rvjdl$cspxo$gpy$kvnqt$pwfs$uif$mbaz$eph'
```

Напишіть функцію reverse_dic (d), яка приймає словник d як вхідний аргумент і повертає словник r . Якщо словник d має ключ k і пов'язане з ним значення v, то словник r повинен мати ключ v і значення k. (Очікується, що це буде працювати лише для словників, які описують бієктивне відображення, хоча вам не потрібно перевіряти це.)

Приклади:

```
In [ ]: reverse_dic({"dog": 10, "cat": 20})
Out[ ]: {10: 'dog', 20: 'cat'}
In [ ]: code2()
Out[ ]: {'g': 'i', 'i': 's', 's': 'g'}
In [ ]: reverse_dic(code2())
Out[ ]: {'g': 's', 'i': 'g', 's': 'i'}
```

Надсилайте свій файл training9.py так часто, як це потрібно, щоб ваші функції працювали правильно. Вам потрібно буде використовувати trapez () у наведених нижче оцінюваних завданнях, тому переконайтесь, що це працює правильно. За необхідності зверніться за допомогою до демонстрантів.

Завдання

Створіть файл lab9.py і заповніть наступними функціями. Як вам потрібно буде використовувати Trapez () у файлі lab9.py, ми пропонуємо вам почати з копіювання training9.py на нове ім'я файлу lab9.py, а потім додати функції нижче lab9.py.

Напишіть функцію пошуку помилки (n), яка використовує функцію trapez () для пошуку помилки трапецеїдального інтегрального наближення. Функція повинна обчислювати інтеграл f (x) = x*x з межами інтегрування a = 0 і b = 2 чисельно. Потім функція повинна відняти цей числовий результат A з точного інтегрального значення I і повернути різницю. Використовуйте аналітичні методи, щоб знайти точне інтегральне значення I.

Приклад :

```
In [ ]: finderror(5)
Out[ ]: -0.05333333333333412
```

(Ви повинні виявити, що помилка зменшується в 4 рази, якщо подвоїти n (оскільки метод trapez має помилку порядку h 2 і h пропорційна 1/n. Ви можете побудувати помилку як функцію h, якщо хочете і обговорити криву з демонстрантом, хоча це не оцінюється.)

Напишіть функцію за допомогою_quad (), яка обчислює інтеграл x 2 від a = 0 до b = 2, використовуючи scipy.integrate.quad . Функція using_quad повинна повертати значення , що чотирьохядерний () повертає функцію, тобто кортеж (y, abserr) , де y є чотирьохядерний «s найкращим наближенням істинного інтеграла і abserr абсолютної оцінки похибки.

Ви повинні виявити, що цей метод (із налаштуваннями за замовчуванням) (набагато) точніший за нашу функцію трапеції .

Функція std_dev (x), яка бере список x чисел з плаваючою комою, обчислює та повертає стандартне відхилення вибірки чисел з плаваючою комою у списку x . Для наочності, якщо список x з N елементами визначено як (9.2)

$$x = [x_1, x_2, x_3, \ldots, x_N] \tag{9.2}$$

тоді стандартне відхилення визначається як (9.3):

$$\sigma = \sqrt{\frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \cdots + (x_N - \mu)^2}{N}}, \tag{9.3}$$

де mu – середнє (арифметичне) (9.4):

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{9.4}$$

Приклади:

In [ ]: std_dev([1.0, 1.0])

Out[ ]: 0.0

In [ ]: std_dev([1.0, 2.0])

Out[ ]: 0.5

Напишіть функцію декодування (код, encoded_msg), яка приймає словниковий код, що містить словник відображення, за допомогою якого був кодований рядок encoded_msg. Функція декодування повинна повернути декодоване повідомлення.

Приклад:

In [ ]: msg = "the quick brown fox jumps over the lazy dog"

In [ ]: code2()

Out[ ]: {'g': 'i', 'i': 's', 's': 'g'}

In [ ]: x = encode(code2(), msg)

In [ ]: x

Out[ ]: 'the qusck brown fox jumpg over the lazy doi'

In [ ]: decode(code2(), x)

Out[ ]: 'the quick brown fox jumps over the lazy dog'

In [ ]: y = encode(code3(), msg)

In [ ]: y

Out[ ]: 'uif$rvjdl$cspxogpykvnqt$pwfs$uif$mbaz$eph'

In [ ]: decode(code3(), y)

Out[ ]: 'the quick brown fox jumps over the lazy dog'

Підказки: Після того, як функція reverse_dic та encode визначена у навчальній частині, що працює, ви можете декодувати кодоване повідомлення, використовуючи зворотний словник коду у функції кодування:

In [ ]: msg = "the quick brown fox jumps over the lazy dog"

In [ ]: encoded = encode(code2(), msg)

In [ ]: encoded

Out[ ]: 'the qusck brown fox jumpg over the lazy doi'

```
In [ ]: code2_reversed = reverse_dic(code2())
In [ ]: encode(code2_reversed, encoded)
Out[ ]: 'the quick brown fox jumps over the lazy dog'
Can you now decode the message:
Zpv$ibwf$tvddfttgvmmz$efdpefe$uijt$tfdsfu$nfttbhf#$Dpohsbuvmbujpot?
```

який був закодований із відображенням, наданим функцією code3? Закодована рядок також доступна під ім'ям змінної secretmessage в training9.py.

Контрольні питання:

1. У чому полягає суть складеного трапецієподібного правила чисельного інтегрування?
2. Яке призначення параметрів a, b та n у функції trapez(f, a, b, n)?
3. Що таке похибка чисельного інтегрування і як її можна визначити для методу трапецій?
4. Які переваги має використання scipy.integrate.quad порівняно з самостійно реалізованим методом трапецій?
5. Що таке стандартне відхилення числової вибірки і для чого воно використовується?
6. Як у Python за допомогою словника можна реалізувати простий симетричний код для шифрування повідомлень?
7. Яким чином функції reverse_dic(d), encode(code, msg) та decode(code, encoded_msg) пов’язані між собою під час кодування та декодування тексту?
8. Чому при збільшенні кількості підрозділів n точність методу трапецій зростає?
9. Яке точне значення інтеграла функції x^2 на відрізку [0, 2] і як його можна знайти аналітично?
10. Яке призначення функції finderror(n) і що саме показує її результат?
11. Що повертає функція scipy.integrate.quad і який зміст мають обидва елементи цього результату?

12. Як обчислюється середнє арифметичне значення вибірки під час знаходження стандартного відхилення?

13. У чому полягає різниця між кодуванням і декодуванням повідомлення в задачах симетричного шифрування?

14. За якої умови функція reverse_dic(d) коректно створює обернений словник для подальшого декодування повідомлення?

Практичне заняття №10
Експоненціальна функція

Мета: ознайомитися з використанням бібліотеки scipy для апроксимації експериментальних даних методом підбору параметрів моделі; сформувати практичні навички роботи з масивами numpy, зчитування числових даних із текстового файлу, побудови математичної моделі процесу охолодження рідини та визначення її параметрів за експериментальними вимірюваннями; навчитися застосовувати функцію scipy.optimize.curve_fit для знаходження початкової температури напою, температури навколишнього середовища та сталої часу охолодження, а також визначати момент часу, коли температура напою знижується до безпечного рівня 60 °C.

У цій практичній ми будемо використовувати функцію curve_fit від scipy, яка знаходиться за адресою:

scipy.optimize.curve_fit

Насолоджуючись гарною чашкою чаю чи кави є важливою частиною нашої академічної діяльності. Тут ми вивчаємо, як гаряча чашка чаю або кави охолоджується після того, як рідина налита в чашку.

Спеціальний дослідницький підрозділ у маловідомому університеті в центральній Яві, Індонезія, проводив вимірювання температури чашки чаю та надавав дані наведені на рисунку 10.1.

Спеціалізоване обладнання приймає одне значення температури (у градусах Цельсія) кожні 10 секунд, хоча вимірювання дещо галасливі. На жаль, на початку вимірювання щось пішло не так, тому дані за першу хвилину відсутні.

Питання, на які ми прагнемо відповісти, такі:

– якою була початкова температура чаю в чашці в момент часу t = 0s?

– як швидко чай охолоджується? Зокрема: через який час безпечно пити його (ми припускаємо, що 60C - це безпечна температура).

– якою буде остаточна температура чаю, якщо ми чекатимемо нескінченно довго (мабуть, це буде кімнатна температура в цій конкретній лабораторії на Яві).

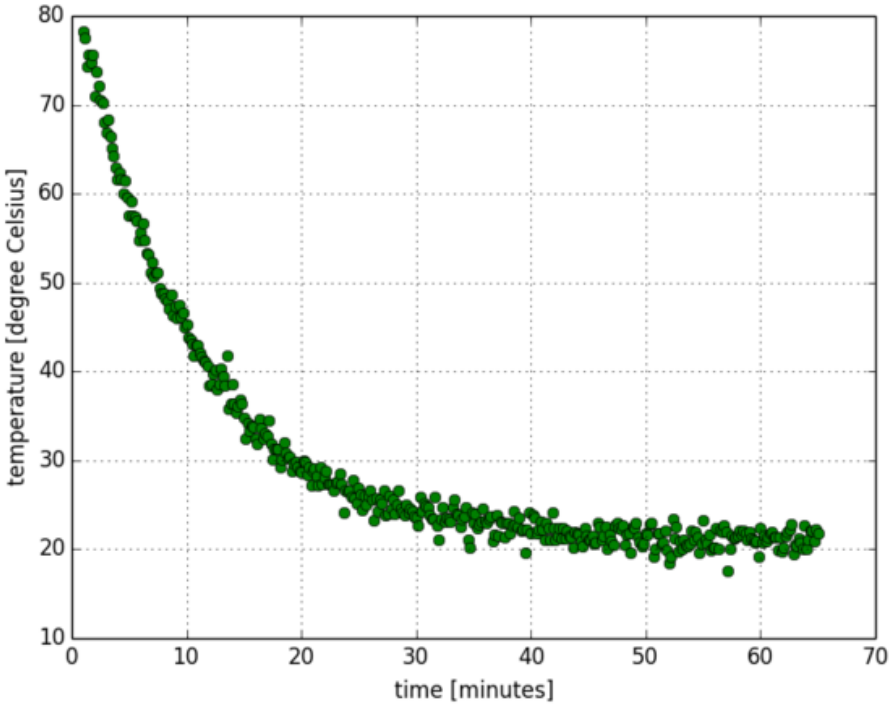


Рисунок 10.1 – Вимірювання температури чашки чаю

Стратегія

Для досягнення прогресу у виконанні цього завдання ми визначаємо деякі імена змінних та маємо зробити кілька спрощувальних припущень.

Ми припускаємо , що початкова температура, Ті, температура, з якою чай вилили в чашку.

Якщо ми чекаємо нескінченно довго, кінцева температура досягне значення температури навколишнього середовища, Та, яка буде температурою навколишнього середовища в лабораторії, де проводились вимірювання.

Далі ми припускаємо, що чашка не має значної теплоємності, щоб спростити проблему.

Ми припускаємо, що процес охолодження слідує певній моделі. Зокрема, ми припускаємо, що швидкість, з якою температура T змінюється як функція часу t, пропорційна різниці між поточною температурою T (t) і температурою середовища Ta, тобто (10.1):

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{1}{c}(T_\mathrm{a} - T) \tag{10.1}$$

Ми можемо вирішити це диференціальне рівняння аналітично і отримати модельне рівняння (10.2):

$$T(t) = (T_\mathrm{i} - T_\mathrm{a}) \exp\left(\frac{-t}{c}\right) + T_\mathrm{a} \tag{10.2}$$

де c – постійна часу для процесу охолодження (що виражається в секундах). Чим більше c , тим довше потрібно гарячому напою охолонути. Протягом c секунд температура напою знизиться приблизно на 2/3.

Навчання

Створіть файл training10.py, який ви заповнюєте такими функціями:

Функція model(t, Ti, Ta, c), яка реалізує рівняння (1).

Приклади

```
In [ ]: model(0, 100, 0, 10)
Out[ ]: 100.0
In [ ]: model(10, 100, 0, 10)
Out[ ]: 36.787944117144235
In [ ]: import math
In [ ]: math.exp(-1) * 100
Out[ ]: 36.787944117144235
In [ ]: model(10, 100, 100, 10)
Out[ ]: 100.0
In [ ]: model(1000000, 100, 25, 10)
Out[ ]: 25.0
```

Функція model(t, Ti, Ta, c) повинна повертати одне значення, якщо t – число з плаваючою комою, і масив, якщо t – масив. Наприклад:

```
In [ ]: model(0, 100, 20, 500)
Out[ ]: 100.0
```

```
In [ ]: from numpy import linspace
In [ ]: ts = linspace(0, 3600, 4)
In [ ]: ts
Out[ ]: array([    0.,  1200.,  2400.,  3600.])
In [ ]: model(ts, 100, 20, 500)
Out[ ]: array([ 100.        ,   27.25743626,   20.65837976,   20.05972686])
```

Ви досягаєте такої поведінки, використовуючи експоненційну функцію від numpy (тобто numpy.exp, а не експоненціальну функцію з математичного модуля), коли реалізуєте рівняння (1) у функції моделі .

Надішліть свій файл training10.py із навчальним рядком теми10, щоб перевірити код та отримати відгук.

Завдання

Створіть файл lab10.py та включіть копію визначення функції model () із навчальної частини (не використовувати з моделі імпорту Training10 – хоча це загалом елегантне рішення, воно не буде працювати, коли ми автоматично перевіряємо вашу заявку).

Потім додайте наступні функції до lab10.py:

read2coldata (ім'я файлу), який відкриває текстовий файл із двома стовпцями даних. Стовпці повинні бути розділені пробілами. Функція повинна повернути кортеж із двох масивів numpy, де перший містить усі дані з першого стовпця у файлі, а другий - усі дані у другому стовпці.

Приклад: для файлу даних testdata.txt, який містить

```
1,5 4
8 5
16 6
17 6.2
```

ми очікуємо такої поведінки:

```
In [ ]: read2coldata('testdata.txt')
Out[ ]: (array([  1.5,   8. ,  16. ,  17. ]), array([ 4. ,  5. ,  6. ,  6.2]))
In [ ]: a, b = read2coldata('testdata.txt')
In [ ]: a
Out[ ]: array([  1.5,   8. ,  16. ,  17. ])
```

```
In [ ]: b
Out[ ]: array([ 4. ,  5. ,  6. ,  6.2])
```

Функція extract_parameters (ts, Ts), яка очікує numpy-масив ts зі значеннями часу та numpy-масив Ts тієї ж довжини, що ts з відповідними значеннями температури. Функція повинна оцінювати і повертати кортеж з трьох параметрів моделі Ti , Ta і c (у такому порядку), пристосовуючи функцію моделі, як у рівнянні (1), до даних ts і Ts .

Підказки:

Функція curve_fit може потребувати деяких початкових здогадок (через необов'язковий параметр функції p0) для того, щоб параметри моделі могли знайти відповідність.

Вам настійно рекомендується побудувати вашу вмонтовану криву разом із вихідними даними, щоб перевірити, чи відповідає обгрунтування.

Ви можете використовувати таку функцію:

```
def plot(ts, Ts, Ti, Ta, c):
    """Input Parameters:
      ts : Data for time (ts)
              (numpy array)
      Ts : data for temperature (Ts)
              (numpy arrays)
      Ti : model parameter Ti for Initial Temperature
              (number)
      Ta : model parameter Ta for Ambient Temperature
              (number)
      c  : model parameter c for the time constant
              (number)
    This function will create plot that shows the model fit together
    with the data.
    Function returns None.
    """

    pylab.plot(ts, Ts, 'o', label='data')
    fTs = model(ts, Ti, Ta, c)
```

```
    pylab.plot(ts, fTs, label='fitted')
    pylab.legend()
    pylab.savefig('testcompare.pdf')  # or pylab.show()
```

Функція sixty_degree_time (Ti, Ta, c), яка очікує параметри моделі Ti (початкова температура), Ta (температура навколишнього середовища) і c постійна часу швидкості охолодження. Функція повинна повертати оцінку кількості секунд, після яких температура напою охолола до 60 градусів Цельсія (60 градусів – це температура, яка зазвичай вважається досить низькою, щоб не пошкодити тканини).

У вас є принаймні два різні можливі способи отримання такої кількості секунд для заданого набору параметрів моделі (Ti, Ta, c). Один включає алгоритм пошуку коренів. Можна припустити, що Ti > 60 градусів Цельсія, а Ti> Ta .

Щоб перевірити свої функції, ви можете зібрати їх, як це в ipython або в окремий файл:

```
from lab10 import read2coldata, extract_parameters, sixty_degree_time
ts, Ts = read2coldata('time_temp.txt')
Ti, Ta, c = extract_parameters(ts, Ts)
print("Model parameters are Ti={} C, Tf={}C,".format(Ti, Ta))
print("                    time constant={}s".format(c))
waittime = sixty_degree_time(Ti, Ta, c)
print("The drink reaches 60 degrees after {:.2f} seconds = {:.2f} minutes"
    .format(waittime, waittime / 60.))
```

Якщо ви хочете включити тестування у свій файл lab10.py , вам потрібно помістити його всередину оператора if __name__ == "___main__" , наприклад

```
if __name__ == "__main__":
    ts, Ts = read2coldata('time_temp.txt')
    Ti, Ta, c = extract_parameters(ts, Ts)
    print("Model parameters are Ti={} C, Tf={}C,".format(Ti, Ta))
    print("                    time constant={}s".format(c))
    waittime = sixty_degree_time(Ti, Ta, c)
    print("The drink reaches 60 degrees after {:.2f} seconds = {:.2f} minutes"
```

```
.format(waittime, waittime / 60.))
```

Написаний так, тест буде виконаний лише в тому випадку, якщо ви виконуєте lab10.py як основний файл (наприклад, натискаючи клавішу F5 під час редагування lab10.py у Spyder), але не якщо файл імпортується з іншого місця (саме це тестування система робить).

Чи можете ви зараз відповісти на три дослідницькі питання, тобто (Не потрібно подавати ці відповіді.):

– якою була початкова температура чаю в чашці в момент часу t = 0s?

– як швидко чай охолоджується? Зокрема: через який час безпечно пити його (ми припускаємо, що 60C - це безпечна температура).

– якою буде остаточна температура чаю, якщо ми чекатимемо нескінченно довго (мабуть, це буде кімнатна температура в цій конкретній лабораторії на Яві).

Контрольні питання:

1. Для чого використовується функція scipy.optimize.curve_fit і які задачі вона дозволяє розв’язувати?
2. Який фізичний зміст мають параметри Ti, Ta і c у моделі охолодження напою?
3. Чому для реалізації функції model(t, Ti, Ta, c) доцільно використовувати numpy.exp, а не math.exp?
4. Яке призначення функції read2coldata(filename) і які дані вона повинна повертати?
5. Як за експериментальними даними часу та температури визначити параметри моделі охолодження?
6. Що показує стала часу c і як її значення впливає на швидкість охолодження напою?
7. Яким способом можна визначити момент часу, коли температура напою досягає 60 °C?

8. У чому полягає різниця між експериментальними даними та математичною моделлю процесу охолодження?

9. Для чого під час апроксимації можуть знадобитися початкові наближення параметрів p0?

10. Чому доцільно будувати графік експериментальних даних разом із графіком апроксимуючої функції?

11. Який вигляд має кортеж, що повертається функцією extract_parameters(ts, Ts), і що означає кожен його елемент?

12. Які практичні висновки можна зробити на основі знайдених параметрів моделі охолодження напою?

Практичне заняття №11
Різні програми SciPy

Мета: ознайомитися з практичним використанням бібліотек SciPy та NumPy для розв'язання прикладних обчислювальних задач у Python; сформувати практичні навички чисельного інтегрування функції, пошуку максимуму, знаходження коренів рівнянь, побудови лінійної інтерполяції та створення функцій-генераторів; навчитися використовувати засоби SciPy для аналізу математичних моделей, а також реалізовувати функції, що повертають інші функції залежно від заданих параметрів.

Створіть файл training11.py і заповніть такими функціями:
функція f (x), яка обчислює (11.1):

$$f(x) = \frac{\exp(-x^2)}{1+x^2} + \frac{2\cos(x)^2}{1+(x-4)^2} \qquad (11.1)$$

Приклади

```
In [ ]: f(0)
Out[ ]: 1.1176470588235294
In [ ]: f(2)
Out[ ]: 0.07293440360502447
```

Хоча вам не потрібно робити наступне, можливо реалізувати функцію f, щоб вона могла обчислити функцію для аргументів вводу масиву. Якщо ви це зробите, наведені вище приклади повинні працювати, як показано, а крім того, ви зможете зробити:

```
In [ ]: import numpy
In [ ]: x = numpy.array([0, 2])
In [ ]: f(x)
Out[ ]: array([ 1.11764706,  0.0729344 ])
```

Для наступних завдань немає необхідності підтримувати аргументи масиву numpy x, як показано тут, але це може бути корисним для зручнішого побудови f (x) .

Напишіть функцію make_multiplier(factor), яка повертає функцію, яка приймає аргумент x і яка повинна повертати коефіцієнт * x .

Приклади

```
In [ ]: g = make_multiplier(10)
In [ ]: g(1)
Out[ ]: 10
In [ ]: g(2)
Out[ ]: 20
In [ ]: g = make_multiplier(0.5)
In [ ]: g(10)
Out[ ]: 5.0
In [ ]: g(100)
Out[ ]: 50.0
```

Надішліть свій файл training11.py із темою Training11, щоб перевірити код та отримати відгук.

Створіть файл lab11.py. Можливо, ви захочете зберегти training11.py під новою назвою lab11.py, оскільки вам потрібно буде використовувати функцію f з training11.py у цьому файлі. Populate lab11.py з наступними додатковими функціями і можливостями:

Функція integrate_f_from0 (b), яка обчислює (наближення) (11.2):

$$\int_0^b f(x)\mathrm{d}x \tag{11.2}$$

де функція f (x) є такою, як визначена в навчальній частині цієї лабораторії. Функція integrate_f_from0 (b) повинна повертати число з плаваючою комою.

Функція find_max_f (), яка повертає (наближення) x, для якого f (x) приймає максимальне значення. Ваша функція find_max_f повинна повертати число з плаваючою комою.

Функція find_f_equals_1 (), яка повертає плаваючу величину, яка є (наближенням) x, для якого f (x) = 1. Нас цікавить рішення, де x від’ємне.

Функція lin_int (xs, ys), яка приймає послідовності xs та ys даних як вхідні дані та повертає об'єкт f (x), що викликається, який повертає y (x) за допомогою лінійної інтерполяції між точками, наданими даними xs та ys .

Ви можете написати власну функцію інтерполяції або використати для цього функцію бібліотеки.

Приклад

```
In [ ]: xs = [1, 2, 3]
In [ ]: ys = [10, 20, 15]

In [ ]: f = lin_int(xs, ys)
In [ ]: f(1)
Out[ ]: array(10.0)
In [ ]: f(2)
Out[ ]: array(20.0)
In [ ]: f(1.5)
Out[ ]: array(15.0)
In [ ]: f(2.5)
Out[ ]: array(17.5)
```

Напишіть функцію make_oscillator(frequency), яка повертає функцію, яка приймає значення з плаваючою точкою t, щоб представляти час, і повертає sin(t * frequency).

Приклади

```
In [ ]: osc1 = make_oscillator(1)
In [ ]: osc1(0 * math.pi)
Out[ ]: 0.0
In [ ]: osc1(0.25 * math.pi)
Out[ ]: 0.7071067811865475
In [ ]: osc1(0.5 * math.pi)
Out[ ]: 1.0
In [ ]: osc2 = make_oscillator(2)
In [ ]: osc2(0 * math.pi)
Out[ ]: 0.0
In [ ]: osc2(0.25 * math.pi)
```

Out[ ]: 1.0

Контрольні питання:

1. Які основні можливості бібліотеки SciPy використовуються для чисельного інтегрування, оптимізації та інтерполяції?

2. Для чого призначена функція integrate_f_from0(b) і який результат вона повинна повертати?

3. Яким чином можна чисельно знайти точку максимуму функції f(x)?

4. Що означає знайти розв’язок рівняння f(x) = 1 і чому в умові окремо зазначено, що нас цікавить від’ємне значення x?

5. У чому полягає принцип лінійної інтерполяції між заданими точками xs та ys?

ПЕРЕЛІК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Костюченко А.О. Основи програмування мовою Python: навч. посіб. Чернігів: ФОП Баликіна С. М., 2020. 180 с.

2. Каштан В.Ю. Програмування комп'ютерних систем мовою Python. Частина 1: навч. наоч. посіб. / В.Ю. Каштан, В.В. Гнатушенко, Д.В. Сущевський, Є.О. Обиденний; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». Дніпро: НТУ «ДП», 2024. 189 с.

3. Пол Беррі. Head First, Python. Фабула, Ганна Якубовська, 2-ге вид., 2021. 624с.

4. Luciano Ramalho. Fluent Python, 2nd ed. Published by O'Reilly Media, Inc., April 2022. 1011p.

5. Reuven M. Lerner, Python Workout: 50 ten-minute exercises, 1st ed. Manning Publications., Juli 2020. 248p.

6. Michael Inden. Python Challenges: 100 Proven Programming Tasks Designed to Prepare You for Anything, 1st ed. Apress, April 2022. 691p.

7. Ramalho L. Fluent Python: Clear, Concise, and Effective Programming. 2nd Edition. O'Reilly Media, 2022. 1016 p

8. Eric Matthes. Python Crash Course : A Hands-On, Project-Based Introduction to Programming, 3nd ed. No Starch Press, May 2023. 552p.

9. Patrick Viafore. Robust Python: Write Clean and Maintainable Code 1st ed. Published by O'Reilly Media, Inc., August 2021. 378p.

10. Alice Zhao. SQL Pocket Guide: A Guide to SQL Usage 4th ed. Published by O'Reilly Media, Inc., October 2021. 354p.

Інформаційні ресурси

11. Real Python. Онлайн-курси, навчальні статті та підручники з Python. URL: https://realpython.com/ (дата звернення 02.02.2026)

12. GeeksforGeeks – Python. Практичні приклади, алгоритми та задачі для самостійного опрацювання. URL: https://www.geeksforgeeks.org/python-programming-language/ (дата звернення 03.02.2026)

13. W3Schools – Python Tutorial. Інтерактивні приклади синтаксису та основ Python для початківців. URL: https://www.w3schools.com/python/ (дата звернення 04.02.2026)

14. Kaggle. Практичні курси з Python для аналізу даних та машинного навчання. URL: https://www.kaggle.com/learn/python (дата звернення 05.02.2026)

15. Coursera. Python Courses. Практичні курси з Python для аналізу даних та машинного навчання. URL: https://www.coursera.org/courses?query=python (дата звернення 05.02.2026)

16. GitHub. Репозиторії прикладних проєктів, бібліотек та навчальних матеріалів для Python. URL: https://github.com/topics/python (дата звернення 06.02.2026)

17. Онлайн-курси Prometheus. URL: https://prometheus.org.ua/ (дата звернення 07.02.2026)

18. Онлайн-курси Coursera. URL: https://www.coursera.org (дата звернення 07.02.2026)

19. Академія Cisco. URL: https://www.netacad.com (дата звернення 07.02.2026)

П 78 Програмування на Python. Методичні вказівки до практичних занять для здобувачів першого (бакалаврського) рівня вищої освіти галузі знань 12 (F) Інформаційні технології денної та заочної форм навчання / уклад. С.М. Костючко, Л.М. Конкевич. Луцьк: ЛНТУ, 2026. 60 с.

Методичне видання до практичних занять з дисципліни «Програмування на Python» складене відповідно до діючої програми курсу.

Призначене для здобувачів першого (бакалаврського) рівня вищої освіти галузі знань 12 (F) Інформаційні технології.

Комп’ютерний набір С.М. Костючко, Л.М. Конкевич

Редактор С.М. Костючко, Л.М. Конкевич

Підп. до друку «___» _____ 2026р.
Формат 60х84/16. Папір офс. Гарнітура Таймс.
Ум. друк. арк. ____. Тираж 10 прим. Зам. ____

Відділ іміджу та промоцій
Луцького національного технічного університету
43018, м. Луцьк, вул. Львівська, 75
ВІП ЛНТУ