Контрольна лабораторна робота № 2

Загальна постановка задачі та математичні деталі

Напишіть програму для обчислення значення функції ***S*** у заданій користувачем точці із заданою точністю.

Приклад умови:

$$S(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!} + \ldots,\ D_S = [a; b],$$

$a = 0.9$, $b = 0.9$.

Гарантовано, що функція $S(x)$ визначена на відрізку $[a; b]$.

Обчислення S(x)

$S(x) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!}$ є сумою степеневого ряду.

Введемо позначки

$$x_n = \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!},$$

$S_k(x) = \sum_{n=1}^{k} \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!} = \sum_{n=1}^{k} x_n$ – часткова сума.

Формально $S(x) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!} = \lim_{k \to \infty} S_k(x)$.

Під час обчислення за значення $S(x)$ взяти перше значення $S_k(x)$, для якого $|x_k| < \varepsilon$. Де значення ε ($\varepsilon > 0$) задається наперед. Назвемо ε «точністю» обчислення; насправді реальне значення $|S(x) - S_k(x)|$ може бути більше.

Про те, як обчислювати $S(x)$ за наведеною схемою, більш детально можна подивитися у файлі **13 Задача наближеного обчислення суми степеневого ряду.pdf**

Уточнення загальної постановки задачі:

Напишіть програму, яка запитує в користувача значення дійсної змінної, бажану «точність» обчислення та у відповідь друкує введене значення змінної та значення функції S для цього значення.

На початку роботи програми має виводитися інформація щодо виконавця, стисла умова задачі та номер варіанту (друкувати формули НЕ треба).

Для некоректних вхідних даних (у тому числі, якщо виникнуть помилки введення) має виводитися відповідне діагностичне повідомлення про суть проблеми та неможливість обчислень. Значення x за межами $[a; b]$ вважати некоректними.

Варіанти завдань та розподіл варіантів (такий самий, як для ЛР1) – див. файли **lab2.pdf** та **Variants.xlsx** відповідно.

Жодних друкованих/рукописних папірців здавати **НЕ ТРЕБА**.

Рекурентні співвідношення, які використовувалися для програмування S, бажано мати десь у зошиті, щоб полегшити собі процес наладки.

Вимоги

Наведені далі вимоги описують уточнення постановки задачі та деталі оформлення лабораторної роботи. Рекомендовано не починати кодування до тих пір, поки цей документ не переглянуто до кінця хоча б по діагоналі.

Позначка (**inf**) означає, що даний пункт є інформативним та на оцінювання не впливає. Позначка (**F**) позначає, що виконання цієї вимоги впливатиме на оцінювання якості коду. Якщо позначок нема, то вимога є **критичною** і її варто виконати.

Якщо не зазначено інше, під наявністю функції/методу/класу/… мається на увазі наявність належно реалізованої функції/…, що відповідає умові лабораторної та вимогам до неї.

Алгоритм роботи програми

А1. Виконання програми за коректних вхідних даних відповідає такому алгоритму (зі збереженням хронології):

1) повідомити призначення програми (номер варіанту та стисло умову задачі, формулу не виводити);

2) повідомити виконавця;

3) ввести вхідні дані (спочатку точку, в якій слід обчислити функцію, потім бажану точність);

4) з нового рядка вивести

```
***** do calculations ...
```

5) провести обчислення;

6) вивести

```
done
```

У результаті на екрані маємо бачити рядок

```
***** do calculations ... done
```

7) з нового рядка вивести введені дані (**x** – у форматі з фіксованою крапкою, 5 знаків після крапки; **eps** – в експоненційному форматі, 4 знаки після крапки)

```
for x =  11.22209
for eps =  1.0000E-06
```

8) на наступному рядку вивести результати обчислення (у форматі з фіксованою крапкою, 9 знаків після крапки, число взято як приклад)

```
result =  13.131313000
```

А2. У випадку некоректних вхідних даних після п.3) алгоритму з нового рядка має виводитися повідомлення про помилку з діагностикою проблеми, після чого програма має завершувати свою роботу без жодних обчислень.

На першому рядку повідомлення про помилку має зазначатися

```
***** Error
```

Решта рядків повідомлення мають містити розгорнуту діагностику. Їх вигляд визначайте на власний розсуд. Проте їх відсутність не найкращим чином вплине на оцінювання якості коду.

Виділені частини вихідного тексту мають виводитися з 100%-ою точністю.

Примітка. Не слід програмувати введення точки та точності в одному рядку.

А3. Значення змінної та значення точності вводяться користувачем. На кожне введення користувачу дається рівно одна спроба! (Повторні запити на введення категорично забороняються. Крім введення значення змінної жодних інших запитів на введення або натискання клавіш у програмі нема!)

A4. (**F**) Наявні зрозумілі підказки на введення даних, читаючи які, можна з першого разу ввести коректні значення

A5. Програма правильно виконує математичні обчислення відповідно до варіанту, якщо введено коректні вхідні значення. Наприкінці у зрозумілому вигляді виводить введені дані та результат обчислення для них.

A6. Програма правильно визначає неможливість обчислення, якщо задані вхідні дані не належать області визначення виразу або некоректні, і друкує повідомлення про помилку.

A7. Програма самостійно завершує свою роботу.

A8. (**F**) Розгорнута діагностика причини неможливості обчислення є необхідною.

A9. Інструкції, що відповідають за виведення (крім запрошень на введення) та обчислення, розташовуються послідовно (спочатку обчислення, потім виведення) й **не** перетинаються в тексті.

A10. Аварійне завершення програми є неприпустимим.

Контроль помилок

E1. Мають контролюватися і помилки введення (введення нечислових значень), і помилки пов'язані із введенням некоректних числових значень. Допускається, що після першої помилки решта вхідних даних вже не зчитується.

Структура програми

S1. Програма записана рівно в одному текстовому файлі з розширенням **`py`**, в імені файлу використовуються виключно символи US-ASCII. У ній відсутні синтаксичні помилки та вона може бути виконана інтерпретатором Python версії 3.8.

S2. (**F**) Розташування коду програми має бути таким:
документація на програмну одиницю
означення глобальних змінних **A**, **B**, що задають область визначення
означення функції **s** та за необхідністю інших функцій
решта коду (жодних означень)

S3. Після встановлення початкового значення змінні **A**, **B** значення не змінюють.

S4. Наявна функція з іменем **`s`**, яка приймає два дійсних аргументи (перший – значення, в якому слід обчислити функцію; другий – точність обчислення). Жодних перевірок аргументів на допустимість не виконує. Винятків не генерує. Повертає значення типу **`float`** – результат обчислення заданої у варіанті функції *S*.

Обчислення є її єдиною відповідальністю. Викликається рівно один раз. Крім як в цій функції, більше ніде обчислення функції *S* чи її частин не виконуються.

Під час обчислення функції **s** **заборонено** використовувати піднесення до степеня, рекурсію, вкладені цикли та будь-які функції. Узагалі програма в цілому має містити рівно одну інструкцію циклу, що записана всередині функції **s**.

Обчислення функції **s** має проектуватися на основі рекурентних співвідношень та вимагати кількість операцій, пропорційну значенню *k*, для якого часткова сума S_k приймається за результат обчислення.

S5. (**F**) Обчислення функції **s** є якомога більше економним за кількістю виконуваних арифметичних дій (та за часом їх виконання).

S6. Структури даних для збереження проміжних числових даних не використовуються.

S7. Інструкції **`global`**, **`nonlocal`**, **`break`**, **`continue`** заборонені до використання.

S8. Завершення функції за інструкцією **`return`** з середини циклу заборонено.

S9. Крім глобальних змінних **A**, **B**, що задають область визначення функції *S* і використовуються саме в цьому значенні, інші власні глобальні змінні використовувати заборонено.

S10. У функціях в якості глобальних змінних допускається використовувати тільки **A**, **B** і тільки там, де це доречно і цього неможливо уникнути.

S11. Усі функції означені на рівні модуля.

S12. Інструкції **`try`** не повинні бути вкладеними (за текстом) в інші інструкції **`try`**.

S13. Інструкції імпортування відсутні.

Інше

М1. Усі надіслані версії лабораторної не порушують принципи академічної доброчесності.

М2. Усі ідентифікатори записані символами US-ASCII.

М3. (**F**) Код не повинен містити дублювання та невикористовувані фрагменти (навіть закоментовані).

М4. (**F**) Коментарі не повинні бути надлишковими.

М5. Використання у власному коді **exit** або інших засобів дострокового завершення програми заборонено.

М6. Використання **async** не дозволяється. (Хоча це неймовірно, щоб в цій лабці воно комусь дійсно було потрібно.)

М7. Код має відповідати рекомендаціям підрозділів 7.1.6 та 7.4 (див. файл **7 Керування порядком обчислень.pdf**).

М8. (**F**) Використовувані в програмі імена є змістовними, їх призначення зрозуміле з їх назви. Довгих імен слід уникати.

М9. (**F**) Назви суто допоміжних функцій мають починатися з підкреслення. Усі функції належно задокументовані.

М10. (**F**) Під час проектування слід дотримуватись принципів функціонального проектування, уникати довгих функцій.

М11. (**F**) Код не має бути заплутаним. Код не має бути захаращений перевірками. Використання механізму винятків має бути доречним та адекватним.

М12. (**F**) Власних функцій, що приймають або повертають рядки, бути не повинно.

М13. Рядки в операторах порівняння не порівнюються. (Це насправді більше підказка. Якщо в цій задачі десь виникає порівняння рядків, то явно щось не так спроектовано.)

М14. Змінні, що використовуються в тілі функції виключно як локальні змінні функції для збереження проміжних результатів, не повинні бути її параметрами.

М15. Кожен фізичний рядок файлу з текстом програми цілком вміщується в одну ширину екрана ноутбука. (Текст програми можна прочитати в текстовому редакторі без горизонтальної прокрутки.)

Оформлення коду

Документацію та коментарі краще писати англійською мовою (PEP8). Якщо це ну дуже важко, то за використання інших мов бали знижуватися не будуть.

Для файлів з текстом програми використовується виключно кодування **UTF-8** (це основне використовуване Python кодування).

Принципи академічної доброчесності (дуже важливо!)

Студент вільно орієнтується в коді лабораторної роботи, яку він здає, розуміє усі використані синтаксичні елементи мови та бібліотечні засоби, зміст та призначення частин коду, вміє самостійно запустити програму на виконання, здатен самостійно внести неглобальні виправлення в код. У коді відсутні коментарі, що перекладають зміст інструкцій на слов'янські мови.

Принципи академічної доброчесності передбачають, що ані брати чужий код, ані давати комусь свій не можна. Сумісна розробка лабораторних також заборонена.

Якщо листи з кодом та скріншотами лабораторної відправлено не з власної поштової адреси (а з адреси товариша!), або якщо з однієї адреси відправлено лабораторні різних студентів, або якщо студент помилився в номері варіанту, або виконав не свій варіант, або неправильно зазначив виконавця, або виконавцем зазначений хтось інший, то все це також вважається порушенням академічної доброчесності.

Звітність

Звітністю з цієї лабораторної є: розташована в коді документація та надісланий код.

Жодних друкованих/рукописних папірців окремо оформлювати та здавати **НЕ ТРЕБА**.

Організаційні моменти

Не пізніше 25 листопада 2020 року на адресу LabAssignment2@i.ua (далі «адреса для лабораторних робіт») надійшов лист з повним кодом лабораторної роботи (усі суттєві для проекту ру-файли і нічого іншого, як вкладення). У темі листа зазначено, що це лабораторна робота 2 та номер варіанту, у форматі **Lab2, <номер варіанту>**. Наприклад, **Lab2, 66**

У самому листі зазначено виконавця та середовище, що використовувалося для виконання лабораторної роботи. Листи з архівами та посиланнями на інтернет-ресурси не припустимі. Один лист – одна лабораторна робота, повністю.

Дозволяється відправляти код лабораторної кілька разів (наприклад, якщо було усунуто якісь недоліки). У випадку, коли код лабораторної надходив кілька разів, розглядатиметься та оцінюється тільки остання версія (навіть, якщо передостання працювала, а остання "зламалася"). Дата/час версії визначається за датою надходження на адресу для лабораторних робіт. Кількість спроб на оцінювання не впливає.

Попередня програмна перевірка (inf)

Щоб мінімізувати кількість зовсім поганих балів, програмно перевіряється виконання дуже невеликої частини критичних вимог (переважно арифметика), протокол перевірки повідомляється. Ця попередня перевірка буде розпочата майже відразу, як почнуть надходити лабораторні роботи.

Більшість критичних вимог перевірятись програмно не буде (нема часу програмувати ці перевірки)**!** Іноді тестер може давати підказки щодо потенційних порушень (можливо, що порушення є, можливо, що нема). Їх варто прочитати.

Менше двох функцій точно бути не може.

Звертаю увагу, що є частина вимог, пов'язана з оформленням і через порушення яких тестер не зможе обробити суть та буде писати на перший погляд дивні речі. Тут 1) **тестер правий, що відхиляє лабораторну, бо він працює згідно загальних вимог, які зазначені в цьому файлі**; 2) варто перевіряти, чи в тому вигляді, послідовності, форматі виводиться те, що має виводитись.

Щодо критичних вимог, які тестер перевіряти не буде, та якості коду, то є викладачі, що ведуть лабораторні заняття. Не найгірша думка інколи з ними консультуватись.

Якщо раптом виявляється, що код стає заплутаним, у ньому з'являються різноманітні "затички", збільшується кількість перевірок та розгалужень, то це явна ознака того, що щось явно не так.

Перевірка лабораторних робіт

Перевірка лабораторних робіт розпочнеться ПІСЛЯ завершення терміну приймання.

Максимально можлива кількість балів за лабораторну становить 20 балів (з 100 за семестр).

Розв’язувана задача має відповідати умові та варіанту.

Отримана сукупність лабораторних робіт ПРОГРАМНО перевіряється на дотримання принципів академічної доброчесності (запозичення коду). Чим більше буде збіг, тим більш ретельною буде співбесіда. Якщо збіг буде зашкалювати, то всі такі лабораторні будуть оцінені в 0 балів, незалежно від того, хто був клієнтом, а хто сервером.

Не таємниця, що умови майже однакові і на лабораторних заняттях схожі задачі розв'язуються. Але власноруч набраний код, навіть за мотивами розібраної схожої задачі, буде набагато більш різноманітним, ніж код, отриманий заміною деяких символів у набраному кимось файлі. І програма це чудово вміє враховувати, щоб визначити код, який був переважно просто скопійований.

Лабораторна робота передбачає захист у формі співбесіди.

Співбесіда за лабораторною (у zoom) може відбуватися не тотально, а тільки в тих випадках, коли наявна інформація (інші результати оцінювання, надісланий код та скріншоти, відгуки викладачів, що ведуть лабораторні заняття; результати програмної перевірки на запозичення) або її відсутність ставить під сумнів дотримання принципів академічної доброчесності під час виконання лабораторної роботи.

Захист лабораторної роботи вважається неуспішним, якщо під час захисту виявляється, що студент не до кінця розуміє код або погано в ньому орієнтується чи не розуміє використані синтаксичні елементи мови, зміст та призначення частин коду, а також якщо захист не відбувся з ініціативи студента.

IF (тестер відхилив лабораторну,
або хоча б одна версія порушує принципи академічної доброчесності,
або лабораторна має критичну кількість запозичень,
або порушено S4,
або є три чи більше порушень критичних вимог,
або захист був неуспішним) :
лабораторна отримує 0 балів
ELSEIF є 1-2 порушення критичних вимог:
лабораторна отримує 12 балів
ELSEIF наявні недоліки, пов'язані з якістю коду та порушеннями вимог категорії **F**:
лабораторна отримує 15 балів
ELSE:
лабораторна отримує 20 балів

Бажаю всім потрапити в останню гілку обчислень (і без співбесіди)!

Порада. Перед тим, як надсилати код лабораторної, варто її ретельно самостійно протестувати, а також з олівцем перевірити виконання вимог (методом: перевірили, якщо виконується, поставили галочку). За необхідністю внести зміни і ще раз протестувати. І якщо все гаразд, то лабораторну можна здавати ☺.