Велика підказка до структури коду лабораторної роботи № 2

Умова лабораторної роботи вимагає за всіх можливих помилок вхідних даних ( і введення нечислових значень, і введення некоректних числових значень) виводити повідомлення про помилку і НЕ виконувати обчислення та НЕ виводити результат.

Найпростіше та найпрозоріше це реалізується з використанням механізму винятків. Інші очевидні способи будуть призводити до жахливих розгалужень та "сходинок" операторів if-elif-else.

Починається все з того, що уважно дивимось на алгоритм роботи програми:

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

2) повідомити виконавця;

3) ввести вхідні дані;

4) з нового рядка вивести

**\*\*\*\*\* do calculations ...**

5) провести обчислення;

6) вивести

**done**

7) з нового рядка вивести введені дані (x та eps )

8) на наступному рядку вивести результати обчислення

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.

Оскільки глобальних змінних, крім кінців ОДЗ A, B, бути не повинно, то весь алгоритм має бути втілений всередині функції, що виконує всю необхідну роботу. Назва: main, demo, тощо, має відображати її призначення.

Також має бути наявна функція s, що виконує обчислення.

Зараз розглядаємо структуру функції main. Щодо повідомлення призначення та виконавця, то це можна зробити або у функції, або перед її ВИКЛИКОМ. Це не найважливіше в цій лабці.

## Варіант структури коду

ввести x

ввести eps

# якщо знаходимось тут, то дані коректні

вивести **\*\*\*\*\* do calculations ...**

провести обчислення

вивести **done**

вивести x

вивести eps

вивести результат обчислення

Де знаходиться перевірка коректності вхідних даних?

У ЛР1 результатом обчислень насправді були наявність результату та сам результат, тому там перевірка x була частиною обчислень.

У ЛР2 результат є суто числовим, за будь-якої помилки у вхідних даних вважається, що результату нема і має виводитись виключно повідомлення про помилку. Тому в ЛР2 перевірка коректності числових значень має виконуватись перед початком обчислень. Більш того, введення x можна розуміти як введення x, що задовольняє умову. Те саме стосується і eps.

Варіант перший

ввести x

ввести eps

перевірити x

перевірити eps

Варіант другий

ввести та перевірити x

ввести та перевірити eps

Природнє питання, а як робити діагностику помилок? – За допомогою винятків.

Коли рядок перетворюється на число і виникає помилка, то перехоплювати виняток і кидати власний з потрібним повідомленням.

Якщо числове значення не задовольняє умову, то кидати виняток з потрібним повідомленням.

Головна ідея: основний код повинен бути вільним від обробки помилок.

Зрозуміло, що його треба охороняти (помістити в середину try-секції). А ось в пастках слід виводити повідомлення про помилки.

## Один з варіантів можливої реалізації

Функція, що вводить дійсне з дуже ретельною діагностикою помилок. Щодо аргументу where, то його призначення стане більш зрозумілим після перегляду усієї програми в цілому.

def input\_float(prompt=None, where=**''**):  
 try:  
 x = input(prompt)  
 except KeyboardInterrupt:  
 raise Exception(where, **'Input was aborted'**)  
 except:  
 raise Exception(where, **'There are no input to convert to float.'**)  
  
 try:  
 x = float(x)  
 except ValueError:  
 raise Exception(where, **'Input [{x}] could not be converted to float'**)  
  
 return x

Функція, що вводить дійсне, не піклуючись про дуже ретельну діагностику.

def input\_float2(prompt=None):  
 try:  
 x = float(input(prompt))  
 except KeyboardInterrupt:  
 raise Exception(**'Input was aborted'**)  
 except:  
 raise Exception(**'There are no input or it could not be converted to float.'**)

### Функція, що вводить дійсне та перевіряє чи задовольняє воно умові.

Умова є функцією check, що за дійсним повертає істиннісне значення. У прикладі в якості значення умови за умовчанням підставлено анонімну функцію, що завжди повертає істину. (Так, цей синтаксис не розбирався. Але його використання в ЛР2 не вимагається. У прикладі ж просто зручно.)

def input\_with\_check(prompt=None, check=lambda x: True, \*, where=**''**):  
 x = input\_float(prompt, where)  
 if not check(x):  
 raise ValueError(where, **'Incorrect value'**)  
 return x

За належних аргументів виклику ця функція здатна забезпечити введення з перевіркою і x, і eps.

### Невеликий приклад використання

Пишемо функції, що виконують перевірки.

def \_check\_1(x):  
 return x > 2  
  
  
def \_chech\_2(x):  
 return x > 6

Сама демо-програма.

print(**'This program wants user to input two reals, first >2 and second >6'**)  
print(**'Its code has "magic" constants which should be eliminated'**)  
try:  
 x1 = input\_with\_check(**'Input real >2\n'**, \_check\_1, where=**"In first value input."**)  
 x2 = input\_with\_check(**'Input real >6\n'**, \_check\_2, where=**"In second value input."**)

# решта дій ЛР2 має бути тут  
except Exception as e:  
 print(**'\*\*\*\*\* error'**)  
 print(e)  
except BaseException as e:  
 print(**'Something went wrong'**)

**Зауваження 1**. Програма містить "магічні" змінні, від яких у ЛР2 слід позбутись. (Можливо, запрошення на введення x утворити з використанням значень глобальних змінних A та B…)

**Зауваження 2**. Якщо хтось вміє використати анонімні функції у викликах input\_with\_check, то це не забороняється. Але й не вимагається.

**Зауваження 3**. Це не єдиний спосіб забезпечити введення з перевіркою коректності введених значень.

**Зауваження 4**. Мети вивести діагностичне повідомлення красиво тут не було. За бажання це можна зробити, якщо вручну вивести значення e.args (це кортеж). У реальності тут більш доречно будувати власний клас винятків (класи – це про другий семестр), який буде займатись форматуванням свого повідомлення. Тому сенсу налаштовувати "красивість" виведення діагностичного повідомлення у цьому прикладі нуль. І в лабораторній роботі також жодних вимог щодо оформлення змісту діагностичного повідомлення нема. Сказано тільки, що повідомлення про помилку має починатись з рядка **\*\*\*\*\* error**

**Зауваження 5**. У лабораторній роботі буде достатньо розрізняти три випадки:

* помилки введення (коли нічого не введено або введено нечислове значення),
* невідповідність дійсного значення x,
* невідповідність дійсного значення eps.

**Зауваження 6**. Украй не рекомендовано використовувати наведені тут фрагменти, не розібравшись у них від і до.