Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №1
«Основные конструкции языка Python»

Выполнил:
студент группы ИУ5-33Б
Малышева Екатерина
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2024 г.

Постановка задачи

Разработать программу для решения биквадратного уравнения.

1. Программа должна быть разработана в виде консольного приложения на языке Python.
2. Программа осуществляет ввод с клавиатуры коэффициентов A, B, C, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).
3. Коэффициенты A, B, C могут быть заданы в виде параметров командной строки (вариант задания параметров приведен в конце файла с примером кода). Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2. Описание работы с параметрами командной строки.
4. Если коэффициент A, B, C введен или задан в командной строке некорректно, то необходимо проигнорировать некорректное значение и вводить коэффициент повторно пока коэффициент не будет введен корректно. Корректно заданный коэффициент - это коэффициент, значение которого может быть без ошибок преобразовано в действительное число.
5. Дополнительное задание 1 (*). Разработайте две программы на языке Python - одну с применением процедурной парадигмы, а другую с применением объектно-ориентированной парадигмы.
6. Дополнительное задание 2 (*). Разработайте две программы - одну на языке Python, а другую на любом другом языке программирования (кроме C++).

Текст программы

```
import sys
import math
import random
def Discriminate(a:int, b:int, c:int)->float:
    return b**2 - 4*a*c

def Calc_solutions(discr:float,a:int,b:int) -> tuple:
    roots=[]
    if discr==0.0:
        roots.append(-b/(2.0*a))
    if discr>0.0:
        roots.append((-b+(discr)**(0.5))/(2*a))
        roots.append((-b-(discr)**(0.5))/(2*a))
    else:
        roots=[]
    return roots
'''
def Calc_solutions_be(solut:tuple)->tuple:
    return (solut[0]**(0.5), -(solut[0]**(0.5)), solut[1]**(0.5), -(solut[1]**(0.5)))
'''
```

```
print("Do you want to set the coefficients yourself?")
print("1-yes, 2-no")
koef=int(input())
if koef==1:
   print("Input coef")
   print("A: ")
   a = float(input())
   while a==0:
      print("A is incorrect, it should be >0, pleas enter a new one")
      print("A: ")
      a=float(input())
   print("B: ")
   b = float(input())
   print("C: ")
   c = float(input())

if koef==2:
   a=a.randint(1,100)
   a=float(a)
   b=b.randint(1,100)
   b=float(b)
   c=c.randint(1,100)
   c=float(c)

if len(Calc_solutions(Discriminate(a,b,c),a,b))==0:
   print("No roots. Discriminate < 0")
elif len(Calc_solutions(Discriminate(a,b,c),a,b))==1:
   print("1 Solutions:")
   print(Calc_solutions(Discriminate(a, b, c), a, b))
else:
   print("2 Solutions:")
   print(Calc_solutions(Discriminate(a, b, c), a, b))
'''
print("Solutions:")
print(Calc_solutions(Discriminate(a,b,c),a,b))
print("Solutions_be")
print(Calc_solutions_be(Calc_solutions(Discriminate(a,b,c),a,b)))
```

Полученный результат

```
Do you want to set the coefficients yourself?
1-yes, 2-no
1
Input coef
A:
3
B:
7
C:
2
2 Solutions:
[-0.3333333333333333, -2.0]
```