МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

**Практические работы**

по дисциплине

**«Проектный практикум»**

Выполнил: студент гр. БЭИ2202

Садиев А. А.

Вариант №17

Проверил: к.п.н., доцент кафедры «Информатика» Гуриков С. Р.

Москва 2024

Практическая работа №1

**«Создание пользовательских функций в языке Python. Обработка исключений. Запись информации в текстовый файл»**

1 Индивидуальное задание на разработку проекта

Вычислить значение функции, приведённой в формуле 1.

Назначение функции: вычисление гиперболического косинуса по формуле 2.

Кроме того, организовать обработку исключений, которая заключается в обработке ошибки ValueError, указывающей на некорректный ввод. Затем, используя методы записи в файл, реализовать запись результатов программы в файл. Выполнить сравнительный анализ результатов работы программ на C++ и текущей программы на Python.

Для выполнения задания следует разработать две функции:

- cosh(x) — возвращает значение гиперболического косинуса;

- expression(x) — возвращает значение функции, приведённой в формуле 1.

В основной части программы осуществить ввод данных, вызов подпрограммы вычисления функции и вывод результата.

2 Схемы алгоритмов

Схема алгоритма вычисления значения гиперболического косинуса изображена на рисунке 1.

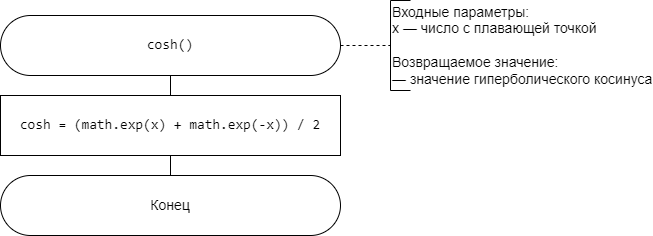


Рисунок 1 — Схема алгоритма вычисления значения гиперболического косинуса

Схема алгоритма вычисления значения функции, приведённой в формуле 1, изображена на рисунке 2.

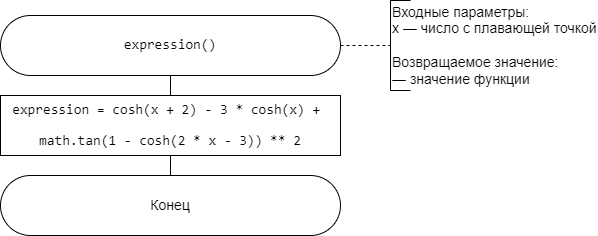


Рисунок 2 — Схема алгоритма вычисления значения функции

Схема алгоритма основной части программы изображена на рисунке 3.

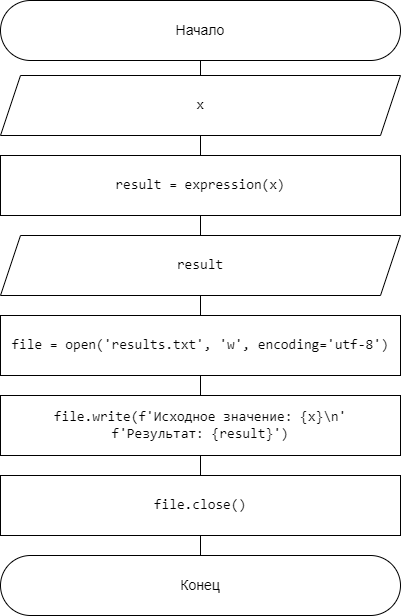


Рисунок 3 — Схема алгоритма основной части программы

3 Программный код

import math  
  
  
def cosh(x):  
 *"""  
 Вычисляет значение гиперболического косинуса.  
  
 :param x: аргумент функции  
 :return: значение гиперболического косинуса для заданного аргумента x.  
 """* return (math.exp(x) + math.exp(-x)) / 2  
  
  
def expression(x):  
 *"""  
 Вычисляет значение выражения cosh(x + 2) - 3 \* cosh(x) + tan(1 - cosh(2 \* x - 3))^2.  
  
 :param x: аргумент функции  
 :return: значение выражения  
 """* return cosh(x + 2) - 3 \* cosh(x) + math.tan(1 - cosh(2 \* x - 3)) \*\* 2  
  
  
try:  
 x = float(input('Введите число: '))  
 result = round(expression(x), 5)  
 print('Результат:', result)  
 with open('task1\_result.txt', 'w', encoding='utf-8') as file:  
 file.write(f'Исходное значение: {x}\n'  
 f'Результат: {result}')  
except ValueError:  
 with open('task1\_result.txt', 'w', encoding='utf-8') as file:  
 file.write('Ошибка выполнения программы.')  
 print('Ошибка ввода: ожидалось число.')

4 Результаты работы программы

Программа успешно работает. Результаты изображены на рисунках 4-6.

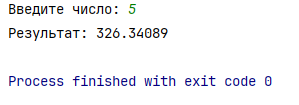


Рисунок 4 — Результат выполнения программы

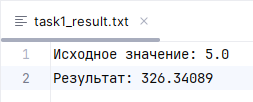


Рисунок 5 — Запись результатов в файл

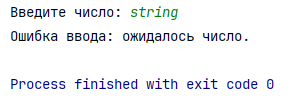


Рисунок 6 — Обработка программных исключений

5 Сравнительный анализ результатов работы программ

Обе программы разработаны с целью получения значения функции и обеспечения взаимодействия с пользователем.

Программа на C++ использует графический интерфейс пользователя, основанный на Windows Forms. Пользователь вводит значение переменной в текстовое поле и после нажатия кнопки результат выводится в том же текстовом поле (рисунок 7).

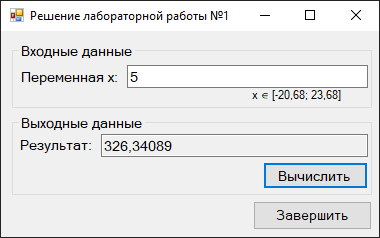


Рисунок 7 — Пример работы программы на C++

Программа на Python предоставляет интерфейс в виде командной строки. Пользователь вводит значение переменной в консоль, и результат выводится в консоль (рисунок 4).

В обеих программах предусмотрена обработка возможных ошибок ввода данных. В программе на C++ при попытке ввода некорректного значения пользователю выводится сообщение об ошибке в диалоговом окне (рисунок 8).

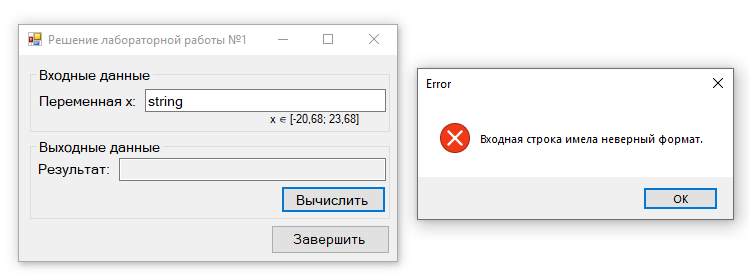


Рисунок 8 — Сообщение об ошибке в программе на C++

В программе на Python при вводе некорректного значения выводится сообщение об ошибке в консоль (рисунок 6).

Обе программы обеспечивают аналогичный функционал по вычислению значения функции, однако имеют различия в среде разработки, способе ввода/вывода данных и обработке ошибок. В случае программы на C++ предусмотрен графический интерфейс, в то время как программа на Python обладает более простым интерфейсом командной строки и записью результатов в текстовый документ (рисунок 5).

Список использованных источников

1 Гуриков С. Р. Основы алгоритмизации и программирования на Python : учебное пособие / С. Р. Гуриков. — Москва : ИНФРА-М, 2022. — 343 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-017142-5. - Текст : электронный. - URL: https://znanium.com/catalog/product/1356003 (дата обращения: 12.02.2024). – Режим доступа: по подписке.

2 ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) — М., Стандартинформ, 2017.

Практическая работа №2

**«Программирование алгоритмов разветвляющихся структур»**

1 Индивидуальное задание на разработку проекта

Вычислить значение кусочно-заданной функции, приведённой в формуле 1.

Для выполнения задания следует разработать одну пользовательскую функцию branching\_structure(x, y, a, b, c), которая должна возвращать значение кусочно-заданной функции, приведённой в формуле 1.

Встроенные функции max и min не использовать. В написанной функции должен быть только один оператор return. Пользовательскую функцию разместить в модуле. Округлить конечный результат до трёх знаков. Использовать функции ceil(), floor(), trunc(), round() относительно полученного результата.

Полученные результаты следует записать в текстовый файл, используя методы записи в файл. Предусмотреть запись в лог-файл и реализовать обработку двух видов ошибок:

* InputError — ошибка ввода, возникающая при некорректном вводе данных;
* ArgumentEqualsZeroError — ошибка вычисления (деление на ноль), возникающая при определённом выборе аргументов.

В основной части программы осуществить ввод данных, вызов подпрограммы вычисления функции, вывод результата, обработку ошибок, запись результатов в файл и запись в лог-файл.

Также создать пакет, включающий в себя первую и вторую практические работы. Продемонстрировать вызов пакета и полученные результаты.

2 Программный код модуля

import math  
  
  
def branching\_structure(x, y, a, b, c):  
 *"""Вычисляет значение ветвящейся структуры на основе входных параметров."""* if x > 0 and y > 2:  
 print('Ветка: "x > 0 и y > 2"')  
  
 inner\_max = math.log10(x)  
 if inner\_max < x + y:  
 inner\_max = x + y  
  
 inner\_min = math.cos(1 - (c \* x \*\* 2) / b)  
 if inner\_min > a \* x:  
 inner\_min = a \* x  
  
 if inner\_max < inner\_min:  
 result = inner\_max  
 else:  
 result = inner\_min  
 else:  
 if x < 0 < y:  
 print('Ветка: "x < 0, y > 0"')  
  
 if x < y:  
 result = 1 + math.cos(x \*\* x)  
 else:  
 result = 1 + math.cos(x \*\* y)  
 else:  
 print('Ветка: "в противном случае"')  
  
 result = 1 + x \*\* 2  
  
 return result

3 Программный код

from module import branching\_structure

from math import floor, ceil, trunc  
import logging  
  
logging.basicConfig(level=logging.INFO, filename="task2\_log.txt", encoding='utf8',  
 format="%(asctime)s %(levelname)s %(message)s")

class ArgumentEqualsZeroError(Exception):  
 pass  
  
  
class InputError(Exception):  
 pass  
  
  
try:  
  
 try:  
 x, y, a, b, c = map(float, input('Введите значения x, y, a, b, c: ').split())  
  
 except Exception:  
 raise InputError('Ошибка ввода: ожидалась строка из пяти чисел, разделённых пробелом.')  
  
 if x > 0 and y > 2 and b == 0:

:  
 raise ArgumentEqualsZeroError('Ошибка вычисления: деление на ноль (b).')  
  
 result = round(branching\_structure(x, y, a, b, c), 3)  
 print(f'Результат: {result}\n'  
 f'Округленный результат (round): {round(result)}\n'  
 f'Результат, округленный в меньшую сторону (floor): {floor(result)}\n'  
 f'Результат, округленный в большую сторону (ceil): {ceil(result)}\n'  
 f'Результат без дробной части (trunc): {trunc(result)}')  
  
 with open('task2\_result.txt', 'w', encoding='utf-8') as file:  
 file.write(f'Исходные значения: x = {x}, y = {y}, a = {a}, b = {b}, c = {c}\n'  
 f'Результат: {result}\n'  
 f'Округленный результат (round): {round(result)}\n'  
 f'Результат, округленный в меньшую сторону (floor): {floor(result)}\n'  
 f'Результат, округленный в большую сторону (ceil): {ceil(result)}\n'  
 f'Результат без дробной части (trunc): {trunc(result)}')  
  
except (InputError, ArgumentEqualsZeroError) as e:  
 with open('task2\_result.txt', 'w', encoding='utf-8') as file:  
 file.write('Ошибка выполнения программы.')  
 print(str(e))  
 logging.error(str(e))

4 Создание пакета

Программный код practice1\\_\_init\_\_.py:

from . import task

Программный код practice2\\_\_init\_\_.py:

from . import task

Программный код practice\\_\_init\_\_.py:

print('├── Практическая работа №1')  
from . import practice1  
print()  
print('├── Практическая работа №2')  
from . import practice2

Программный код main.py:

import practice

Итоговая структура проекта с созданными пакетами изображена на рисунке 9.

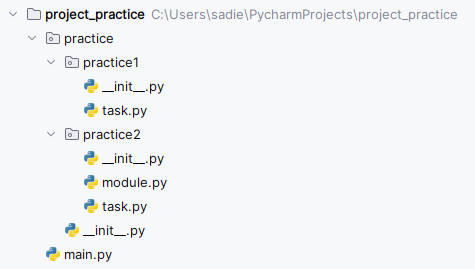


Рисунок 9 — Структура проекта с созданными пакетами

5 Результаты выполнения программы

Результат выполнения программы по ветке «x > 0 и y > 2» изображен на рисунке 10.

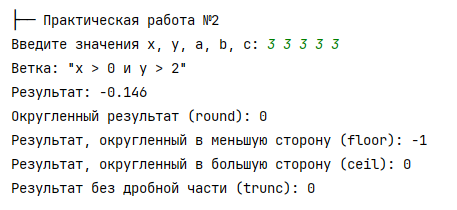


Рисунок 10 — Результат выполнения программы по ветке «x > 0 и y > 2»

Результат выполнения программы по ветке «x < 0, y > 0» изображен на рисунке 11.

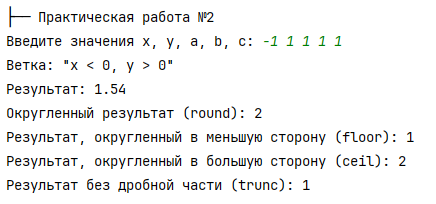


Рисунок 11 — Результат выполнения программы по ветке «x > 0 и y > 2»

Результат выполнения программы по ветке «в противном случае» изображен на рисунке 12.

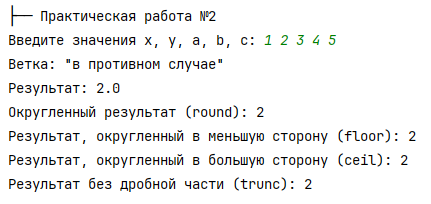


Рисунок 12 — Результат выполнения программы по ветке «в противном случае»

Запись результата выполнения программы в файл изображена на рисунке 13.

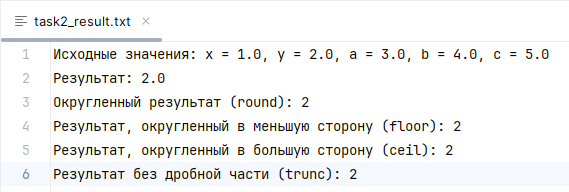


Рисунок 13 — Запись результата выполнения программы в файл

Результат выполнения программы с перехватом исключения изображен на рисунках 14-15.

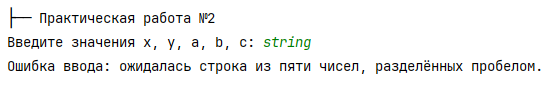


Рисунок 14 — Перехват исключения ошибки ввода

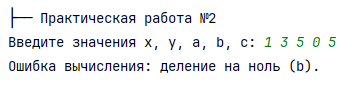


Рисунок 15 — Перехват исключения ошибки вычисления

Запись в файл в сценарии с ошибкой изображена на рисунке 16.

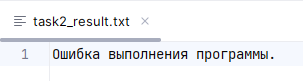


Рисунок 16 — Запись в файл в сценарии с ошибкой

Содержимое лог-файла изображено на рисунке 17.

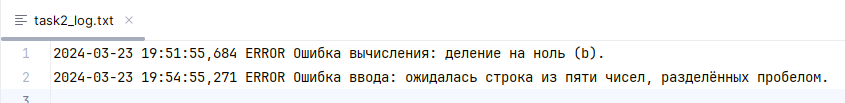


Рисунок 17 — Содержимое лог-файла

Результат выполнения созданного пакета изображен на рисунке 18.

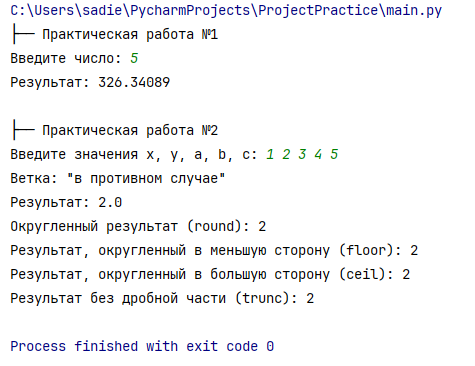


Рисунок 18 — Результат выполнения созданного пакета

Список использованных источников

1 Гуриков С. Р. Основы алгоритмизации и программирования на Python : учебное пособие / С. Р. Гуриков. — Москва : ИНФРА-М, 2022. — 343 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-017142-5. - Текст : электронный. - URL: https://znanium.com/catalog/product/1356003 (дата обращения: 21.03.2024). – Режим доступа: по подписке.

2 Дроботун Н. В. Алгоритмизация и программирование. Язык Python : учебное пособие / Н. В. Дроботун, Е. О. Рудков, Н. А. Баев. — Санкт-Петербург : Санкт-Петербургский государственный университет промышленных технологий и дизайна, 2020. — 119 c. — ISBN 978-5-7937-1829-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/102400.html (дата обращения: 21.03.2024). — Режим доступа: для авторизир. пользователей.

3 ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) — М., Стандартинформ, 2017.

Практическая работа №3

**«Создание классов. Создание экземпляров классов. Создание конструкторов»**

1 Первая часть

1.1 Первое задание

1.1.1 Задание на разработку программы

Разработать класс Dist, который содержит два атрибута meters и centimeters. Класс должен иметь следующие методы:

* set\_dist(self, mt, ct) — принимает значение метров и сантиметров, устанавливает соответствующие значения внутренним переменным объекта;
* get\_dist(self) — запрашивает у пользователя ввод количества метров и сантиметров, устанавливает их соответствующие значения внутренним переменным объекта;
* show\_dist(self) — выводит на экран расстояние в формате «метры и сантиметры».

Создать три объекта класса Dist с названиями dist1, dist2, dist3. Установить расстояние для объекта dist2 методом set\_dist() с аргументами и . Для объекта dist3 запросить расстояние у пользователя методом get\_dist(). Вывести на экран расстояния всех трех объектов методом show\_dist().

1.1.2 Программный код

class Dist:  
 meters, centimeters = 0, 0.0  
  
 def set\_dist(self, mt, ct):  
 self.meters = mt  
 self.centimeters = ct  
  
 def get\_dist(self):  
 self.meters = int(input('Введите число метров: '))  
 self.centimeters = float(input('Введите число сантиметров: '))  
  
 def show\_dist(self):  
 print(f'{self.meters} м {self.centimeters} см')  
  
  
dist1 = Dist()  
dist2 = Dist()  
dist3 = Dist()

dist2.set\_dist(14, 25.)  
dist3.get\_dist()  
  
print('dist1 = ', end='')  
dist1.show\_dist()

print('dist2 = ', end='')  
dist2.show\_dist()

print('dist3 = ', end='')  
dist3.show\_dist()

1.1.3 Результат выполнения программы

Результат выполнения программы приведён на рисунке 19.

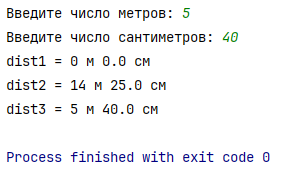


Рисунок 19 — Результат выполнения программы первого задания

1.2 Второе задание

1.2.1 Задание на разработку программы

Разработать пустой класс MyClass. В основной части программы создать атрибут x объекта класса MyClass. Создать два экземпляра класса MyClass с именами obj1 и obj2. Добавить обоим экземплярам атрибут y с помощью динамического создания атрибутов. Вывести на экран значения атрибутов x и y для обоих экземпляров класса MyClass.

1.2.2 Программный код

class MyClass:  
 pass  
  
  
MyClass.x = 100  
obj1, obj2 = MyClass(), MyClass()  
obj1.y = 10  
obj2.y = 20

print(f'obj1.x = {obj1.x} obj1.y = {obj1.y}')  
print(f'obj2.x = {obj2.x} obj2.y = {obj2.y}')

1.2.3 Результат выполнения программы

Результат выполнения программы приведён на рисунке 20.

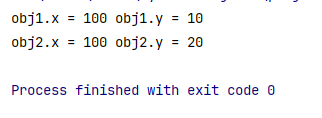


Рисунок 20 — Результат выполнения программы второго задания

1.3 Третье задание

1.3.1 Задание на разработку программы

Разработать класс MyClass со статическим атрибутом x, по умолчанию равным , и динамическим атрибутом y, инициализируемым значением в конструкторе класса. Создать два объекта класса MyClass с именами obj1 и obj2. Вывести на экран значения атрибута x для обоих объектов. Затем изменить значение статического атрибута x класса MyClass на и вывести значения атрибута x для обоих объектов. Затем изменить значения атрибута y объекта obj1 на и вывести значения атрибутов y для обоих объектов. Установить значение атрибута x объекта obj2 равным и значение статического атрибута x класса MyClass равным . Вывести на экран значения атрибута x для обоих объектов и для класса MyClass.

1.3.2 Программный код

class MyClass:  
 x = 10  
  
 def \_\_init\_\_(self):  
 self.y = 20  
  
  
obj1, obj2 = MyClass(), MyClass()  
print(f'obj1.x = {obj1.x} obj2.x = {obj2.x}')  
MyClass.x = 50  
print(f'obj1.x = {obj1.x} obj2.x = {obj2.x}')  
print(f'obj1.y = {obj1.y} obj2.y = {obj2.y}')  
obj1.y = 90  
print(f'obj1.y = {obj1.y} obj2.y = {obj2.y}')  
obj2.x = 60  
MyClass.x = 80  
print(f'obj1.x = {obj1.x} obj2.x = {obj2.x} MyClass.x = {MyClass.x}')

1.3.3 Результат выполнения программы

Результат выполнения программы приведён на рисунке 21.

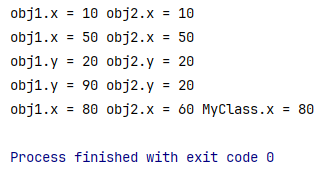


Рисунок 21 — Результат выполнения программы третьего задания

1.4 Четвертое задание

1.4.1 Задание на разработку программы

Разработать класс Dist. Определить метод инициализации, который принимает два аргумента: метры (mt) и сантиметры (ct). В методе инициализации установить начальные значения атрибутов meters и centimeters объекта класса. Класс должен иметь следующие методы:

* get\_dist(self) — запрашивает у пользователя ввод количества метров и сантиметров, устанавливает их соответствующие значения внутренним переменным объекта;
* show\_dist(self) — выводит на экран расстояние в формате «метры и сантиметры».

Создать три объекта класса Dist с названиями dist1, dist2, dist3, используя метод инициализации. Для dist1 и dist2 установить аргументы, равные нулю. Для объекта dist2 установить аргументы равные и . Для объекта dist3 запросить расстояние у пользователя методом get\_dist(). Вывести на экран расстояния всех трех объектов методом show\_dist().

1.4.2 Программный код

class Dist:  
 def \_\_init\_\_(self, mt, ct):  
 self.meters = mt  
 self.centimeters = ct  
 print('Работает конструктор')  
  
 def get\_dist(self):  
 self.meters = int(input('Введите число метров: '))  
 self.centimeters = float(input('Введите число сантиметров: '))  
  
 def show\_dist(self):  
 print(f'{self.meters} м {self.centimeters} см')  
  
  
dist1 = Dist(0, 0.0)  
dist2 = Dist(14, 25.)  
dist3 = Dist(0, 0.0)

dist3.get\_dist()

print('dist1 = ', end='')  
dist1.show\_dist()

print('dist2 = ', end='')  
dist2.show\_dist()

print('dist3 = ', end='')  
dist3.show\_dist()

1.4.3 Результат выполнения программы

Результат выполнения программы приведён на рисунке 22.

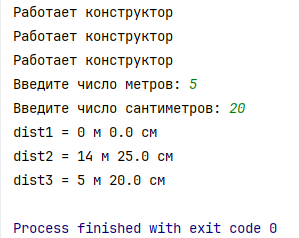


Рисунок 22 — Результат выполнения программы четвёртого задания

2 Вторая часть

2.1 Индивидуальное задание на разработку программы

Вычислить значение кусочно-заданной функции, приведённой в формуле 1.

Для вычисления значения кусочно-заданной функции написать функции для каждой ветви системы. Для каждой из них создать свой модуль.

Для выполнения задания следует разработать класс BranchingStructure с методом-конструктором, который должен выводить строку «Работает конструктор». Внутри класса должны быть реализованы следующие методы:

* set\_input\_values(self) — реализует ручной ввод данных для вычисления значения кусочно-заданной функции;
* calculate\_first\_branch(self) — обращается к модулю, вызывая функцию вычисления ветки «»;
* calculate\_second\_branch(self) — обращается к модулю, вызывая функцию вычисления ветки «»;
* calculate\_third\_branch(self) — обращается к модулю, вызывая функцию вычисления ветки «».

В основной части программы, кроме ввода данных, осуществить проверку условий для каждой ветви программы с вызовом соответствующего метода и вывести полученный результат.

2.2 Программный код модулей

Программный код модуля branch\_first:

from math import log10, cos  
  
  
def expression(x, y, a, b, c):  
 print('"Ветка: x > 0 и y > 2')  
  
 return min(max(log10(x), x + y), min(cos(1 - (c \* x \*\* 2) / b), a \* x))

Программный код модуля branch\_second:

from math import cos  
  
  
def expression(x, y):  
 print('"Ветка: x < 0, y > 0')  
  
 return 1 + cos(x \*\* min(x, y))

Программный код модуля branch\_third:

def expression(x):  
 print('Ветка: в противном случае')  
  
 return 1 + x \*\* 2

2.3 Программный код

import branch\_first  
import branch\_second  
import branch\_third  
  
  
class BranchingStructure:  
 def \_\_init\_\_(self, x, y, a, b, c):  
 self.x = x  
 self.y = y  
 self.a = a  
 self.b = b  
 self.c = c  
 print('Работает конструктор')  
  
 def set\_input\_values(self):  
 values = map(float, input('Введите значения (x, y, a, b, c), разделённые пробелом: ').split())  
 self.x, self.y, self.a, self.b, self.c = values  
  
 def calculate\_first\_branch(self):  
 return branch\_first.expression(self.x, self.y, self.a, self.b, self.c)  
  
 def calculate\_second\_branch(self):  
 return branch\_second.expression(self.x, self.y)  
  
 def calculate\_third\_branch(self):  
 return branch\_third.expression(self.x)  
  
  
branching\_structure = BranchingStructure(0, 0, 0, 0, 0)  
branching\_structure.set\_input\_values()  
x = branching\_structure.x  
y = branching\_structure.y  
  
if x > 0 and y > 2:  
 result = branching\_structure.calculate\_first\_branch()  
elif x < 0 < y:  
 result = branching\_structure.calculate\_second\_branch()  
else:  
 result = branching\_structure.calculate\_third\_branch()  
  
print(f'Результат: {round(result, 3)}')

2.4 Результат выполнения программы

Результат выполнения программы приведён на рисунке 23. Полученный результат совпадает с результатом из предыдущей работы, приведённом на рисунке 10, с тем же набором значений.

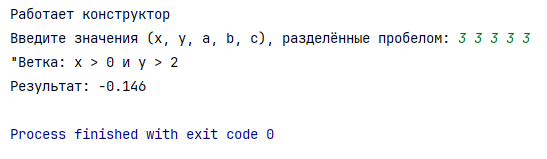


Рисунок 23 — Результат выполнения программы

Результат выполнения программы приведён на рисунке 24. Полученный результат совпадает с результатом из предыдущей работы, приведённом на рисунке 11, с тем же набором значений.

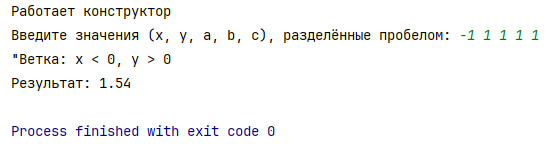


Рисунок 24 — Результат выполнения программы

Результат выполнения программы приведён на рисунке 25. Полученный результат совпадает с результатом из предыдущей работы, приведённом на рисунке 12, с тем же набором значений.

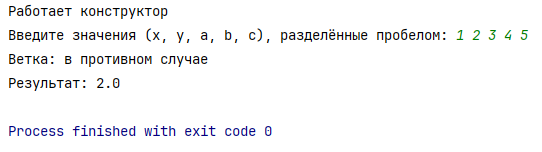


Рисунок 25 — Результат выполнения программы

Список использованных источников

1 Гуриков С. Р. Основы алгоритмизации и программирования на Python : учебное пособие / С. Р. Гуриков. — Москва : ИНФРА-М, 2022. — 343 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-017142-5. - Текст : электронный. - URL: https://znanium.com/catalog/product/1356003 (дата обращения: 12.02.2024). – Режим доступа: по подписке.

2 Букунов, С. В. Объектно ориентированное программирование на языке Python : учебное пособие / С. В. Букунов, О. В. Букунова. — Санкт-Петербург : Санкт-Петербургский государственный архитектурно-строительный университет, ЭБС АСВ, 2020. — 119 c. — ISBN 978-5-9227-1128-9. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/117194.html

3 ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) — М., Стандартинформ, 2017.