Министерство науки и высшего образования РФ
ФГБОУ ВО «Ульяновский государственный университет»
Факультет математики, информационных и авиационных технологий

Перцева И.А., Савинов Ю.Г., Санников И.А.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ ПО ДИСЦИПЛИНЕ

«Основы программирования на языке Python»

для студентов всех направлений и специальностей не ИТ профиля

Ульяновск

Методические указания для самостоятельной работы студентов по дисциплине «Основы программирования на языке Python» для студентов всех направлений и специальностей не ИТ профиля / составители: Перцева И.А., Савинов Ю.Г., Санников И.А. - Ульяновск: УлГУ, 2022

Настоящие методические указания предназначены для студентов всех направлений и специальностей, не относящихся к ИТ профилю. В работе приведены литература по дисциплине, методические указания для самостоятельной работы студентов.

Они будут полезны при выполнении и сдаче лабораторных работ и подготовке к зачету по данной дисциплине.

Рекомендованы к введению в образовательный процесс Ученым советом Факультета математики, информационных и авиационных технологий УлГУ (протокол № 3/22 от 19 апреля 2022 г.).

СОДЕРЖАНИЕ:

1. ЛИТЕРАТУРА ДЛЯ ИЗУЧЕНИЯ ДИСЦИПЛИНЫ

1. Златопольский Д.М., Основы программирования на языке Python [Электронный ресурс]: учебник / Златопольский Д. М. - М. : ДМК Пресс, 2017. - 284 с. - ISBN 978-5-97060-552-3 - Режим доступа: http://www.studentlibrary.ru/book/ISBN9785970605523.html
2. Федоров, Д. Ю. Программирование на языке высокого уровня python : учебное пособие для прикладного бакалавриата / Д. Ю. Федоров. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2019. — 161 с. — (Бакалавр. Прикладной курс). — ISBN 978-5-534-10971-9. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://www.biblio-online.ru/bcode/437489
3. Лучано Рамальо, Python. К вершинам мастерства [Электронный ресурс] / Лучано Рамальо - М. : ДМК Пресс, 2016. - 768 с. - ISBN 978-5-97060-384-0 - Режим доступа: http://www.studentlibrary.ru/book/ISBN9785970603840.html
4. Маккинли У., Python и анализ данных [Электронный ресурс] / Уэс Маккинли - М. : ДМК Пресс, 2015. - 482 с. - ISBN 978-5-97060-315-4 - Режим доступа: http://www.studentlibrary.ru/book/ISBN9785970603154.html
5. Ян Эрик Солем, Программирование компьютерного зрения на языке Python [Электронный ресурс] / Ян Эрик Солем - М. : ДМК Пресс, 2016. - 312 с. - ISBN 978-5-97060-200-3 - Режим доступа: http://www.studentlibrary.ru/book/ISBN9785970602003.html
6. Рацеев, Сергей Михайлович. Задачи по программированию и основные алгоритмы : учеб.-метод. пособие по курсу "Информатика". Ч. 2 : / Рацеев Сергей Михайлович. - Ульяновск : УлГУ, 2009.- URL^ ftp://10.2.96.134/Text/razeev1.pdf
7. Угаров, Владимир Васильевич. Технология программирования : учеб.-метод. пособие: в 2 ч. Ч. 1 : / Угаров Владимир Васильевич ; УлГУ, ФМиИТ. - Ульяновск : УлГУ, 2011.- URL^ ftp://10.2.96.134/Text/ugarov1.pdf
8. Жаркова, Галина Алексеевна. Методы программирования и прикладные алгоритмы : учеб.-метод. пособие / Жаркова Галина Алексеевна, А. В. Жарков ; УлГУ, ФМИиАТ. - Ульяновск : УлГУ, 2018.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Ниже приведены методические указания по выполнению лабораторных работ и варианты заданий. Номер варианта соответствует номеру студента в списке группы.

2.1. Лабораторная работа №1: «Линейные программы»

Цель работы: научиться вводить и выводить данные, создавать переменные и выполнять арифметические операторы, использовать операторы сравнения.

Постановка задачи

Напишите программу для расчета по заданным формулам. Предварительно подготовьте тестовые примеры с помощью калькулятора или электронной таблицы Excel.

$$\textbf{1)}\ y = tg^2\left(\frac{x^2}{2}-1\right)+\frac{2\cos(x-\pi/6)}{1/2+\sin^2\alpha};\quad \textbf{2)}\ y = 2^{\frac{\log_{(3+\sin(x))}(3-\cos(\pi/4+2x))}{1+tg^2(2x/\pi)}}.$$

Для математических вычислений в Python имеются как встроенные, так и дополнительные функции и методы. Применить дополнительные математические функций можно после подключения модуля `math`:

```
import math
```

либо

```
from math import *
```

В первом случае функции рассматриваются как методы объекта `math` и должны записываться так:

```
import math
print(math.sin(math.pi/4)) print(math.sqrt(2)/2)
```

Во втором случае вызов функции может быть сделан в более привычной для нас форме:

```
from math import *
print(sin(pi/4))
print(sqrt(2)/2)
```

Вместе с тем, такой способ импорта может нарушить пространство имен программы, поскольку может возникнуть конфликт между именами переменных, которые использует программист и именами импортируемых функций. При импорте можно ограничиться только необходимыми функциями, например:

```
from math import (pi, sin, cos,tan, log)
```

Ввод данных

Ввод данных можно выполнить с клавиатуры функцией `input()`:

```
m = input([str])
```

При этом на экран будет выведена строка `str`, а переменная `m` получит значение строкового типа, введённое пользователем. Строковый тип может быть преобразован, например, к типу `int` или `float`, если введённое значение – число.

Вывод данных

Вывод данных на экран монитора может быть выполнен функцией `print()`. Эта функция позволяет выполнять форматированный вывод, как с использованием Си-подобного форматирования, так и с использованием форматной строки Python.

Следующие строки демонстрируют, как можно форматировать вывод.

```
for x in range(1,11):
print('%2d %3d %7.2f' % (x, x*x, x*x*x))
print("{0:.2f} {1:.2f} {2:.4f}".format(a, x, y))
```

Буква в формате числа определяет тип выводимого числа. Так, `d` – это целый тип, `f` – вещественное число. Число в формате означает то число позиций, которое будет использовано для вывода числа. Для вещественного числа указывается, после точки, количество выводимых десятичных знаков.

Во второй строке использован Си-подобный формат, в котором формат числа начинается с процента "`%`". В этом формате аргументы отделяются от форматной части строки так же символом `%` – процент.

В третьей строке используется форматная строка Python, в которой в форматной строке позиции для значений аргументов выделяются фигурными скобками.

Обратите внимание на то, что сами форматные строки начинаются и завершаются одиночной или двойной кавычкой. В Python допускаются оба вида кавычек для выделения строки. Важно только что бы начало и конец были одинаковыми.

Так же следует понимать, что в промежутках между символами форматирования могут находиться и другие символы или слова:

```
print('x=%2d x^2=%3d x^3=%7.2f'% (x, x*x, x*x*x))
print("a={0:.2f} x={1:.2f} y={2:.4f}".format(a, x, y))
```

Использование форматных строк делает вывод данных более внятным.

Решение задания

Вернемся к нашим примерам и запишем их, используя правила языка Python. Первое выражение примет вид:

```
1. y = tan(x**2/2-1)**2+2*cos(x-pi/6)/(1/2+sin(a)**2)
```

Второе выражение представим в виде двух:

```
2. tmp=log(3-cos(pi/4+2*x),3+sin(x))/(1+tan(2*x/pi)**2)
   y = pow(2, tmp)
```

Описание алгоритма

Для вычислений необходимо обеспечить ввод двух переменных `x` и `a`. Поскольку по условиям задачи их тип и точность представления не заданы, выберем для них вещественный тип (`float`). Для оптимизации записи выражения используем промежуточную переменную `tmp`.

1. Ввести значения `a` и `x`, преобразовать к типу `float`.
2. Вычислить выражение 1.
3. Вывести результат вычисления.
4. Вычислить значение переменной `tmp`;
5. Вычислить выражение 2.
6. Вывести результат вычисления.

Листинг программы

```
from math import *
a = float(input('Введите параметр a: '))
x = float(input('Введите значение x: '))
y=tan(x**2/2-1)**2+(2*cos(x-pi/6))/(1/2+sin(a)**2)
print("{0:.2f} {1:.2f} {2:.4f}".format(a, x, y))
tmp=log(3-cos(pi/4+2*x),3+sin(x))/(1+tan(2*x/pi)**2)
y=pow(2,tmp)
print("{0:.2f} {1:.4f}".format(x, y))
```

Результаты тестирования программы

a	x	Первое выражение		Второе выражение
		Калькулятор	Программа	
-2	-2	1.196954	1.1970	1.1184
0	-2	-0.834654	-0.8347	1.1184
0	0	5.889618	5.8896	1.6880
2	0	3.730931	3.7309	1.6880
1.5	0.5	2.771242	2.7712	1.7955
4	3	-1.326566	-1.3266	1.0517

Задания к лабораторной работе №1
«Линейные программы»

Напишите программу для расчета по двум формулам. Подготовьте не менее пяти тестовых примеров. Предварительно выполните вычисления с использованием калькулятора или офисного приложения, например Excel или Calc. Результаты вычисления по обеим формулам должны совпадать. Отсутствующие в языке функции выразите через имеющиеся.

1. $z_1 = 2\sin^2(3\pi - 2\alpha)\cdot\cos^2(5\pi + 2\alpha)$; $z_2 = \frac{1}{4} - \frac{1}{4}\sin(\frac{5}{2}\pi - 8\alpha)$.

2. $z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha$; $z_2 = 2\sqrt{2}\cos\alpha\cdot\sin(\frac{\pi}{4} + 2\alpha)$.

3. $z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha}$; $z_2 = 2\sin\alpha$.

4. $z_1 = \frac{2\cdot\cos\alpha\cdot\sin 2\alpha - \sin\alpha}{\cos\alpha - 2\cdot\sin\alpha\cdot\sin 2\alpha}$; $z_2 = \mathrm{tg}3\alpha$.

5. $z_1 = 1 - \frac{1}{4}\sin^2 2\alpha + \cos 2\alpha$; $z_2 = \cos^2\alpha + \cos^4\alpha$.

6. $z_1 = \cos\alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha$;

$z_2 = 4\cos\frac{\alpha}{2}\cdot\cos\frac{5}{2}\alpha\cdot\cos 4\alpha$.

7. $z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\alpha}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\alpha}{4}\right)$; $z_2 = \frac{\sqrt{2}}{2}\sin\frac{\alpha}{2}$.

8. $z_1 = \cos^4 x + \sin^2 y + \frac{1}{4}\sin^2 2x - 1$; $z_2 = \sin(y+x)\cdot\sin(y-x)$.

9. $z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2$;

$z_2 = -4\sin^2\frac{\alpha - \beta}{2}\cdot\cos(\alpha + \beta)$.

10. $z_1 = \frac{\sin\left(\frac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)}$; $z_2 = \mathrm{ctg}\left(\frac{5}{4}\pi + \frac{3}{2}\alpha\right)$.

11. $z_1 = \frac{1 - 2\sin^2\alpha}{1 + \sin 2\alpha}$; $z_2 = \frac{1 - tg\alpha}{1 + tg\alpha}$.

12. $z_1 = \frac{\sin 4\alpha}{1 + \cos 4\alpha}\cdot\frac{\cos 2\alpha}{1 + \cos 2\alpha}$; $z_2 = \mathrm{ctg}\left(\frac{3}{2}\pi - \alpha\right)$.

13. $z_1 = \frac{\sin\alpha + \cos(2\beta - \alpha)}{\cos\alpha - \sin(2\beta - \alpha)}$; $z_2 = \frac{1 + \sin 2\beta}{\cos 2\beta}$.

14. $z_1 = \dfrac{\cos\alpha + \sin\alpha}{\cos\alpha - \sin\alpha}$; $z_2 = \operatorname{tg}2\alpha + \sec 2\alpha$.

15. $z_1 = \dfrac{\sqrt{2b + 2\sqrt{b^2-4}}}{\sqrt{b^2-4}+b+2}$; $z_2 = \dfrac{1}{\sqrt{b+2}}$.

16. $z_1 = \dfrac{x^2+2x-3+(x+1)\cdot\sqrt{x^2-9}}{x^2-2x-3+(x-1)\cdot\sqrt{x^2-9}}$; $z_2 = \sqrt{\dfrac{x+3}{x-3}}$.

17. $z_1 = \dfrac{\sqrt{(3m+2)^2-24m}}{3\sqrt{m}-\dfrac{2}{\sqrt{m}}}$; $z_2 = \sqrt{m}$.

18. $z_2 = \left(\dfrac{a+2}{\sqrt{2a}} - \dfrac{a}{\sqrt{2a}+2} + \dfrac{2}{a-\sqrt{2a}}\right)\cdot\dfrac{\sqrt{a}-\sqrt{2}}{a+2}$; $z_2 = \dfrac{1}{\sqrt{a}+\sqrt{2}}$.

19. $z_1 = \left(\dfrac{1+a+a^2}{2a+a^2} + 2 - \dfrac{1-a+a^2}{2a-a^2}\right)\cdot(5-2a^2)$; $z_2 = \dfrac{4-a^2}{2}$.

20. $z_1 = \dfrac{(m-1)\sqrt{m}-(n-1)\sqrt{n}}{\sqrt{m^3n}+nm+m^2-m}$; $z_2 = \dfrac{\sqrt{m}-\sqrt{n}}{m}$.

21. $z_1 = \dfrac{1+\sin^4(-a)-\cos^4(-a)}{\cos^2 a}$; $z_2 = 2tg^2 a$.

22. $z_1 = \sqrt{\dfrac{1-\sin(\frac{\pi}{2}-a)}{2}} + \sqrt{\dfrac{1+\cos(2\pi-a)}{2}}$; $\pi < a < \dfrac{3\pi}{2}$.

$z_2 = \sin\dfrac{a}{2} - \cos\dfrac{a}{2}$; $\pi < a < \dfrac{3\pi}{2}$.

23. $z_1 = \left(\dfrac{1+6ac}{a^3-8c^3} - \dfrac{1}{a-2c}\right) : \left(\dfrac{1}{a^3-8c^3} - \dfrac{1}{a^2+2ac+4c^2}\right)$;

$z_2 = 1-2c+a$.

24. $z_1 = \dfrac{\dfrac{1}{a}-\dfrac{1}{b+c}}{\dfrac{1}{a}+\dfrac{1}{b+c}}\cdot\left(1+\dfrac{b^2+c^2-a^2}{2bc}\right):\dfrac{a-b-c}{abc}$; $z_2 = \dfrac{a-b-c}{2}\cdot a$.

25. $z_1 = \dfrac{\sin^2(\pi+a)+\sin^2(\frac{\pi}{2}+a)}{\cos(\frac{3\pi}{2}+a)}\operatorname{ctg}(1.5\pi-a)$; $z_2 = \dfrac{1}{\cos a}$.

26. $z_1 = \dfrac{\mathrm{tg}(x-\frac{\pi}{2})\cos(\frac{3\pi}{2}+x)-\sin^3(3.5\pi-x)}{\cos(x-0.5\pi)\mathrm{tg}(1.5\pi+x)}$; $z_2 = \sin^2 x$.

27. $z_1 = a^{\sqrt{\log_a b}} - b^{\sqrt{\log_b a}} + \mathrm{tg}(ab+3\pi/2)$; $z_2 = \mathrm{tg}(ab+\frac{3}{2}\pi)$;

28. $z_1 = \dfrac{\sin^2 a - \mathrm{tg}^2 a}{\cos^2 a - \mathrm{ctg}^2 a}$; $z_2 = \mathrm{tg}^6 a$;

29. $z_1 = \dfrac{1-2\sin^2 a}{2\mathrm{ctg}(\frac{\pi}{4}+a)\cos^2(\frac{\pi}{4}-a)} + e^a$; $z_2 = 1+e^a$;

30. $z_1 = \dfrac{\sin^4 a + 2\sin a\cos a - \cos^4}{\mathrm{tg}2a - 1}$; $z_2 = \cos 2a$;

Константы (Math)	**Описание и пример использования**
pi	Возвращает число π. `math.pi => 3.141592653589793`
e	Возвращает число e. `math.e => 2.718281828459045`
Функции (Sys)	

int([<Объект>[,<Система счисления>]])	Преобразует объект в целое число. Второй параметр указывает систему счисления преобразуемого числа. По умолчанию используется десятичная система счисления.
	`>>>int(7.5), int('71',10), int('0o71',8)`*)
	`(7, 71, 57)`
	`>>>int("0xA",16)`
	`10`
	`>>>int(), int("0b11111111",2)`
	`(0, 255)`

float([<Число или строка>])	Преобразует целое число или строку в вещественное число.
	`>>>float(7), float("7.1"), float("12.")`
	`(7.0, 7.1, 12.0)`
	`>>> float("inf"), float("-inf"), float("nan")`
	`(inf, inf, nan)`

bin(<Число>)	Преобразует десятичное число в двоичное. Возвращает строковое представление числа.
	`>>>bin(255), bin(1), bin(-45)`
	`('0b11111111', '0b1', '-0b101101')`

oct(<Число>)	Преобразует десятичное число в восьмеричное. Возвращает строковое представление числа.
	`>>>oct(7), oct(8), ovt(64)`
	`('0o7', '0o10', '0o100')`

hex(<Число>)	Преобразует десятичное число в шестнадцатеричное. Возвращает строковое представление числа.
	`>>>hex(10), hex(16), hex(255)`
	`('0xa', '0x10', '0xff')`

round(<Число>[, Кол-во знаков после точки])	Вещественное число округляется до целого по следующему правилу: дробная часть
	меньше 0.5 – округление вниз;
	равна 0.5 – округление до четного числа;
	больше 0.5 – округление вверх.
	`>>>round(1.49), round(1.50), round(1.51)`
	`(1, 2, 2)`
	`>>>round(2.49), round(2.50), round(2.51)`
	`(2, 2, 3)`
	Второй параметр определяет число цифр после запятой. В этом случае округление выполняется по правилу: отбрасываемая часть

	меньше 0.5 – округление вниз;
	больше или равно 0.5 – округление вверх.
	`>>>round(2.449,1),round(2.450,1),round(2.451,1)`
	`(2.49, 2.5, 2.5)`

abs(<Число>)	Возвращает абсолютное значение.
pow(<Число>, <Степень>, [,<Делитель>])	Возвращает число, возведенное в степень. Степень может быть вещественного типа. Если указан третий параметр, то возвращается остаток от деления результата на значение этого параметра. `>>>pow(3, 3), pow(3, 3, 2), pow(3, 3.25)` (27, 1, 35.533998349717294)
max(<Список чисел через запятую>)	Возвращается максимальное число из списка.
min(<Список чисел через запятую>)	Возвращается минимальное число из списка.

sum(<Последовательность> [, <Начальное значение>)	Возвращает сумму значений элементов последовательности (списка, кортежа) плюс начальное значение.
	`>>>sum([1, 2, 3], 7), sum((1, 2, 3))`
	`(13, 6)`

divmod(x, y)	Возвращает кортеж из двух значений: целая часть и остаток от деления: `x // y` и `x % y`.
Функции (Math)	
sin(x), cos(x), tan(x)	Стандартные тригонометрические функции. Угол задается в радианах.
asin(x), acos(x), atan(x)	Обратные тригонометрические функции
degrees(x)	Преобразование радиан в градусы
radians(x)	Преобразование градусов в радианы
exp(x)	Экспонента
log(<Число>[, <База>])	Логарифм числа по основанию <База>. Если база не указана, то вычисляется натуральный логарифм.
log10(x), log2(x)	Десятичный и двоичный логарифмы (по базе 10 и 2 соответственно)
sqrt(x)	Квадратный корень
ceil(x)	Округление до ближайшего большего целого
floor(x)	Округление до ближайшего меньшего целого
fabs(x)	Возвращает абсолютное значение. тоже, что и `abs(x)`
fmod(x, y)	Остаток от деления. Тоже, что и x % y.
factorial(n)	Факториал числа.
fsum(<список чисел>)	Возвращает точную сумму чисел из заданного списка. >>>sum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1]) 0.9999999999999999 >>>fsum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1]) 1.0

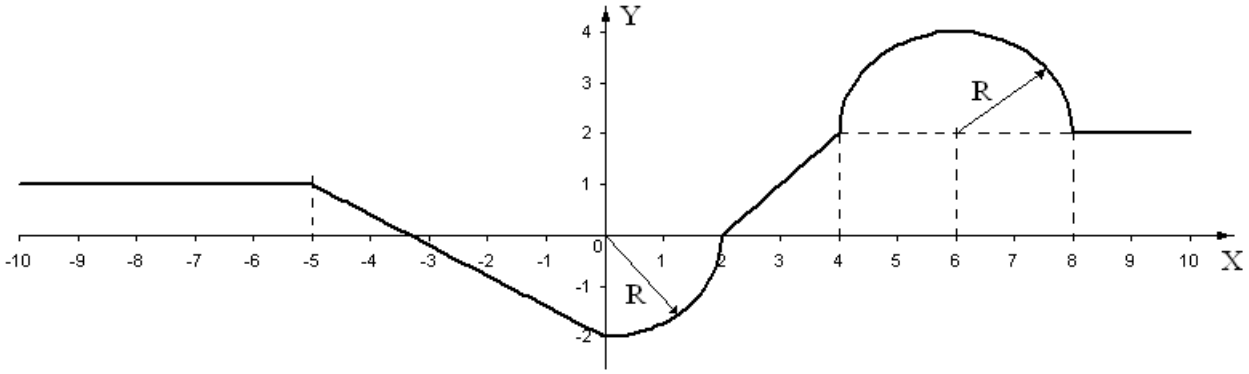
2.2. Лабораторная работа №2: «Разветвляющиеся процессы»

Цель работы: научиться работать с условными операторами на языке Python.

Задание 1

Постановка задачи

Написать программу, которая по введённому значению аргумента вычисляет значение функции, заданной в виде графика.



Теоретическая часть

Для решения задачи использован оператор ветвления, который в языке Python имеет следующий вид:

```
if <Логическое выражение>:
    <Блок – выполняется, если условие истинно>
[elif <Логическое выражение>:
    <Блок – выполняется, если условие истинно>
]
[else:
    <Блок – выполняется, если все условия ложны>
]
```

<Блок> – это набор вложенных инструкций, которые выделяются одинаковым количеством пробелов (обычно четырьмя).

Для ввода данных используется инструкция `input()`, которая возвращает строку. Введённые значения, перед использованием в арифметических выражениях, должны быть преобразованы к числовому формату.

Вывод данных выполняется инструкцией `print()`, в которой использован форматированный вывод данных.

График функции представлен фрагментами прямых линий, описываемых уравнением $y=kx+b$ и дугами кругов. В общем случае уравнение круга может быть представлено так: $(x-a)^2+(y-b)^2=R^2$

Неизвестные параметры, угол наклона и смещение прямой, а так же координаты центра дуг, определим, используя данные из графика.

Для прямой на интервале (-5, 0) можем записать следующую систему уравнений:

$$\begin{cases} 1 = k \cdot (-5) + b \\ -2 = k \cdot 0 + b \end{cases}$$

Из второго уравнения следует, что b = -2, а из первого – k = -3/5. Для полукруга с центром (6, 2) уравнение круга примет вид: $(x\text{-}6)^2+(y\text{-}2)^2=2^2$

Перепишем уравнение так: $(x\text{-}6)^2=4\text{-}(y\text{-}2)^2$. Отсюда следует:

$$y = 2 + \sqrt{4 - (x-6)^2}$$

Знак перед корнем выбран для случая, когда рассматривается верхняя часть полукруга.

Выполнив необходимые вычисления для всех фрагментов функции, мы получим систему уравнений, которую запишем в следующем виде:

$$y = \begin{cases} 1 & x < -5 \\ -\frac{3}{5}x - 2 & -5 <= x < 0 \\ -\sqrt{4 - x^2} & 0 <= x < 2 \\ x - 2 & 2 <= x < 4 \\ 2 + \sqrt{4 - (x-6)^2} & 4 <= x < 8 \\ 2 & x >= 8 \end{cases}$$

Функция определена на всём диапазоне . При этом, особых точек у неё нет.

Описание алгоритма

1. Ввести значение аргумента `x` и преобразовать его к типу `float`.
2. Определить, к какому интервалу из области определения функции оно принадлежит, и вычислить значение функции `y` по соответствующей формуле.
3. Вывести значение `x` и `y`.

Описание входных и выходных данных

Входные данные поступают с клавиатуры, а выходные - выводятся на монитор для просмотра. Входные и выходные данные имеют тип `float`.

Листинг программы (вариант 1)

```
from math import * # теперь можно так:
                   # print(sin(pi/4))
x = float(input('Введите значение x='))
if x < -5: y = 1
if x >=-5 and x<0: y = -(3/5)*x-2
if x >= 0 and x<2: y = -sqrt(4-x**2)
```

```
if x >= 2 and x<4: y = x-2
if x >= 4 and x<8: y = 2+sqrt(4-(x-6)**2)
if x >= 8: y = 2
print("X={0:.2f} Y={1:.2f}".format(x, y))
```

Следует отметить, что в такой записи алгоритма проверка выполняется для всех условных операторов, в том числе и тех, которые следуют за вычисленным. Так, например, если `x` равно -3, то выполнится второй оператор, но и во всех последующих операторах операция сравнения будет проведена. Число проверок можно сократить, если написать программу с использованием вложенных условных операторов.

Листинг программы (вариант 2)

```
from math import * # теперь можно так:
                   # print sin(pi/4)
x = float(input('Введите значение x='))
if x < -5:
    y = 1
elif x >=-5 and x<0:
    y = -(3/5)*x-2
elif x >= 0 and x<2:
    y = -sqrt(4-x**2)
elif x >= 2 and x<4:
    y = x-2
elif x >= 4 and x<8:
    y = 2+sqrt(4-(x-6)**2)
else: y = 2
print("X={0:.2f} Y={1:.2f}".format(x, y))
```

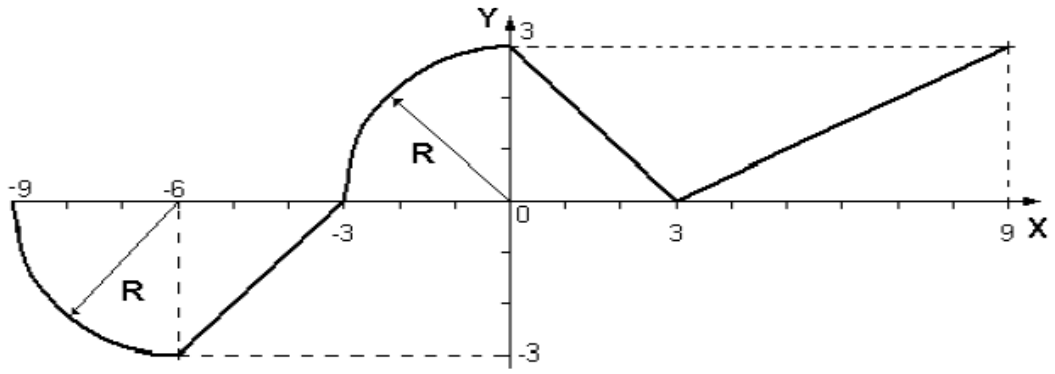
Результат работы программы

```
Введите значение аргумента: -6
X= -6.00 Y= 1
Введите значение аргумента: -3.33
X= -3.33 Y= -0.00
Введите значение аргумента: 6
X= 6.00 Y= 4.00
```

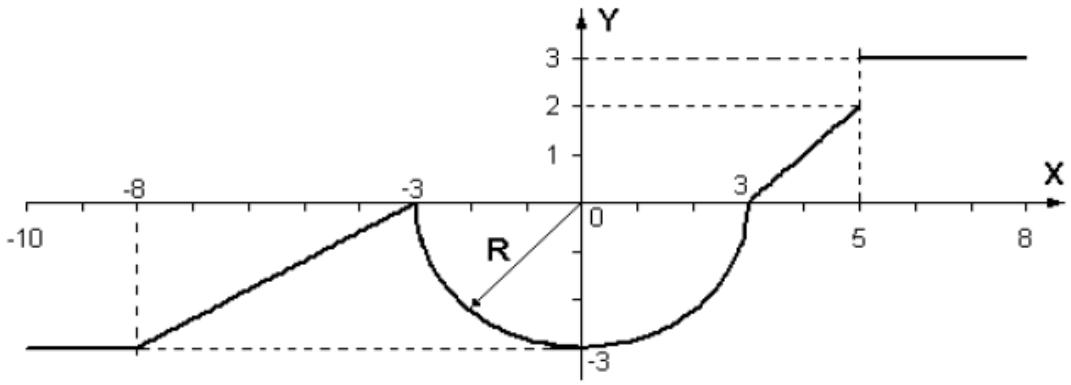
Задания к лабораторной работе №2
«Разветвляющиеся вычислительные процессы»

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметры, необходимые для решения задания следует получить из графика и определить в программе.

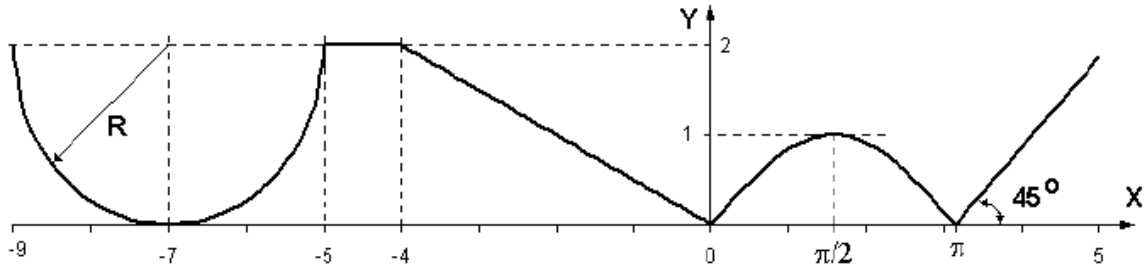
1)



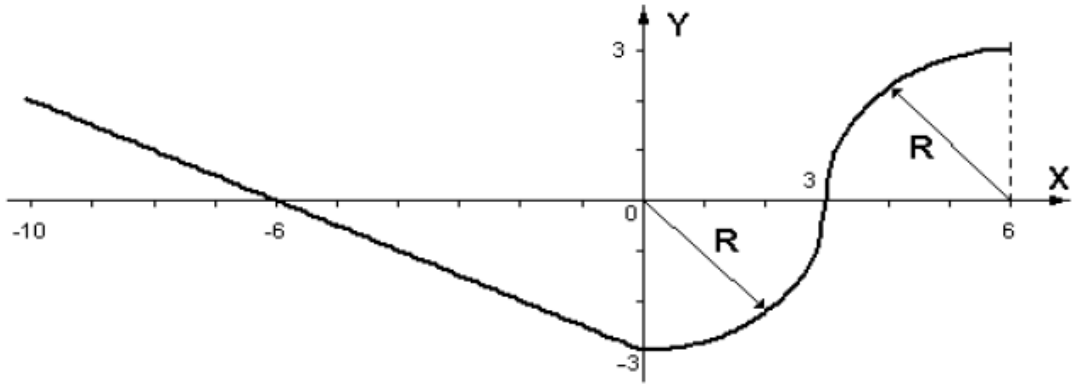
2)



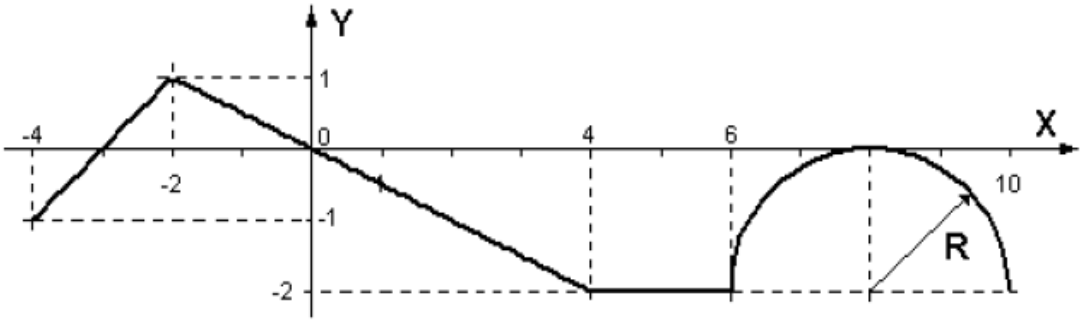
3)



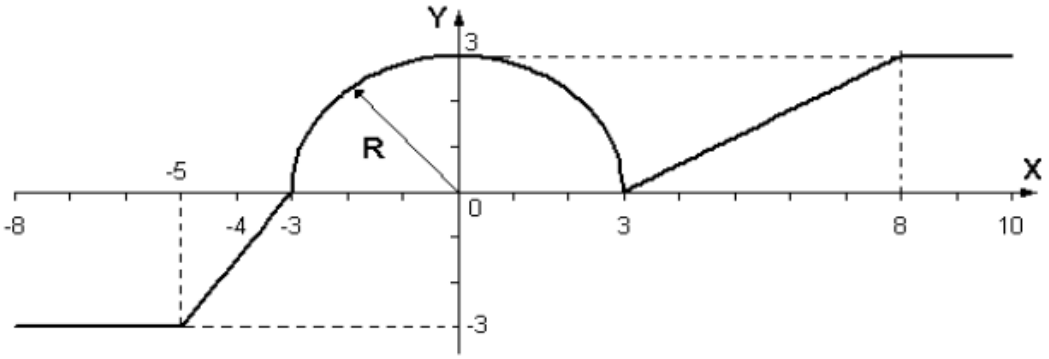
4)



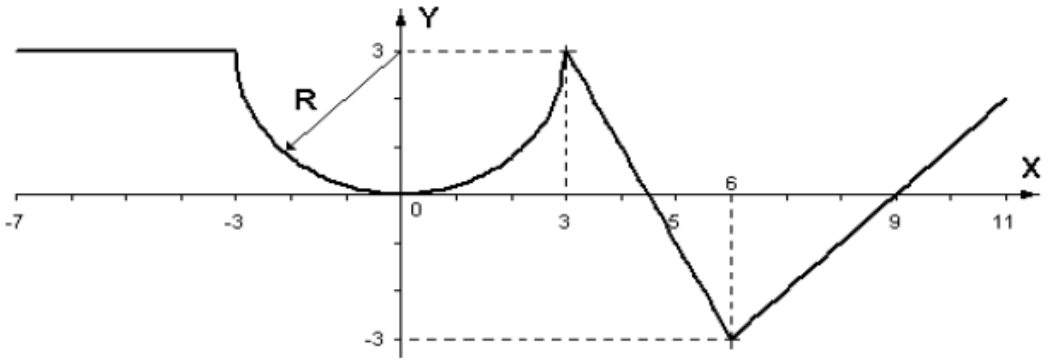
5)


Y
X
1
-1
-2
-4
-2
0
4
6
10
R

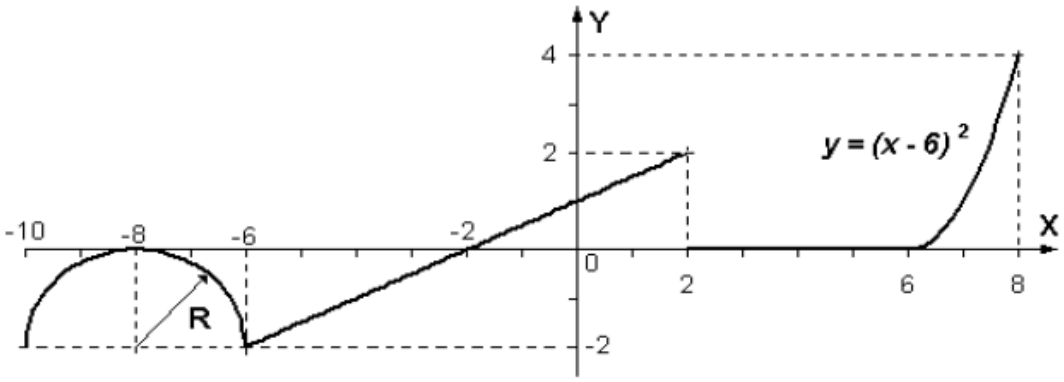
6)


Y
X
3
-3
R
-5
-8
-4
-3
0
3
8
10

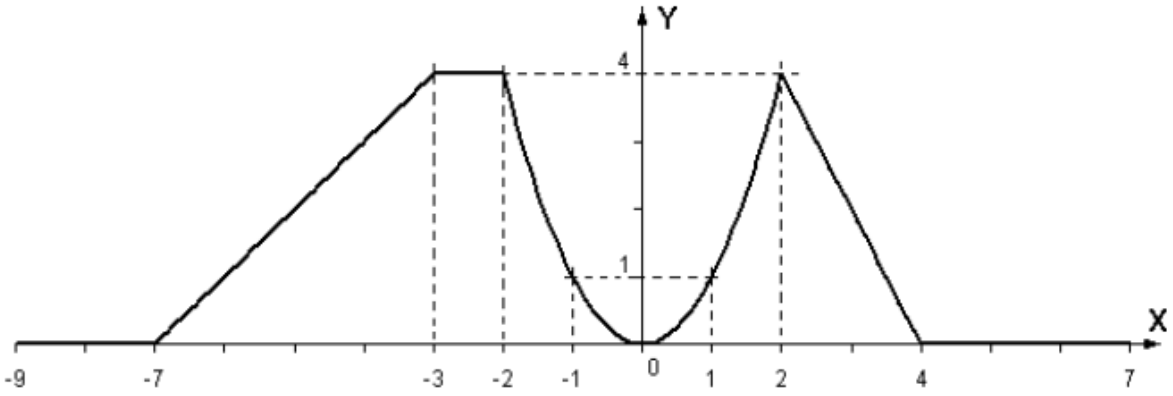
7)


Y
X
3
-3
R
-7
-3
0
3
5
6
9
11

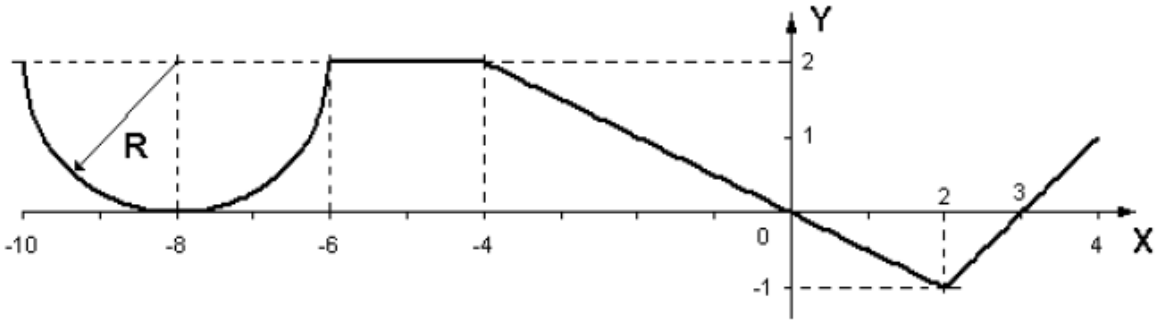
8)


Y
X
4
2
-2
y = (x - 6) ²
-10
-8
-6
-2
0
2
6
8
R

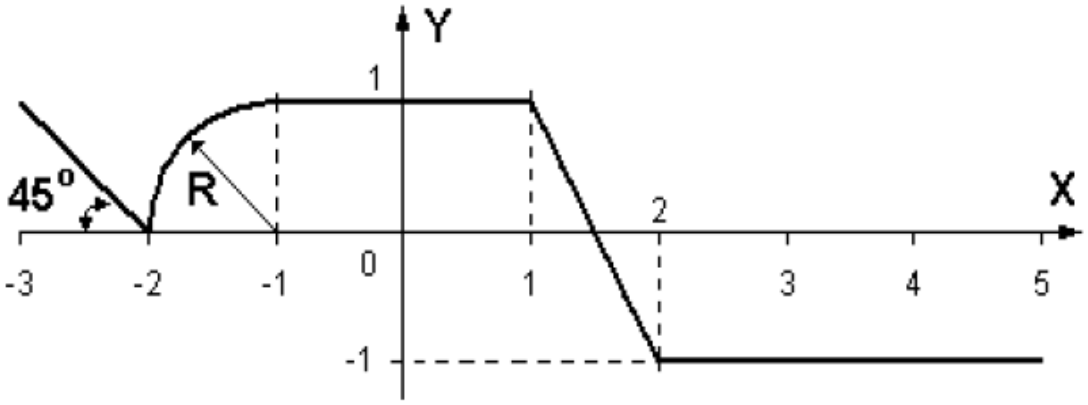
9)


Y
X
4
1
-9
-7
-3
-2
-1
0
1
2
4
7

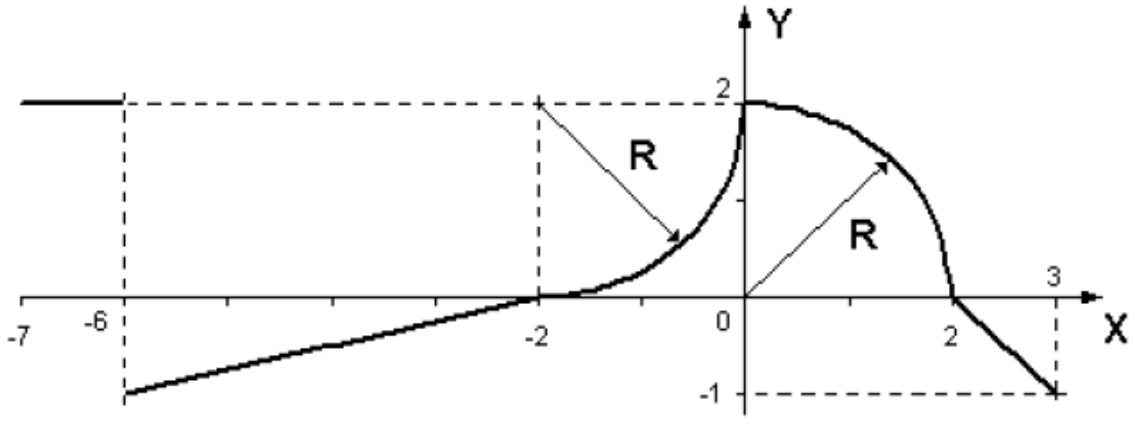
10)


Y
X
R
2
1
-1
-10
-8
-6
-4
0
2
3
4

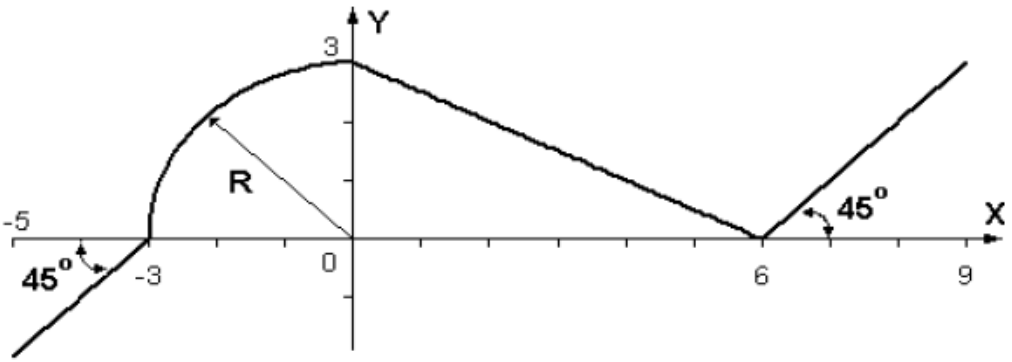
11)


Y
X
45°
R
1
-1
-3
-2
-1
0
1
2
3
4
5

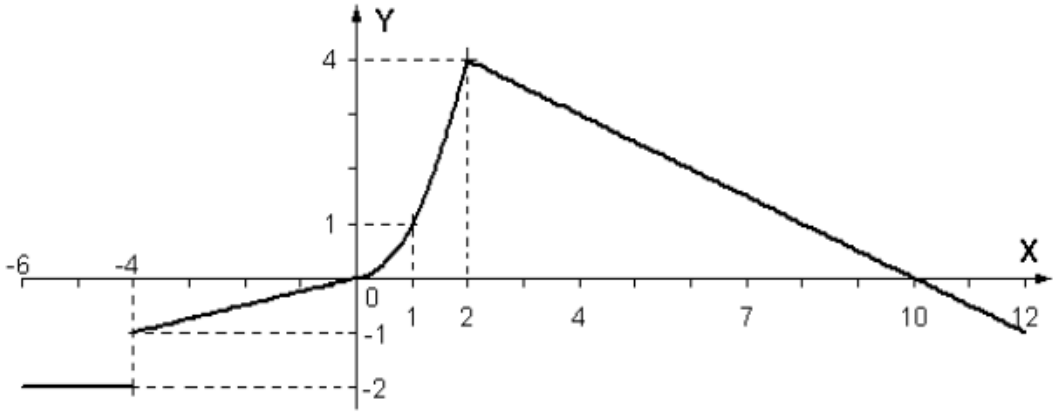
12)


Y
X
R
R
2
-1
-7
-6
-2
0
2
3

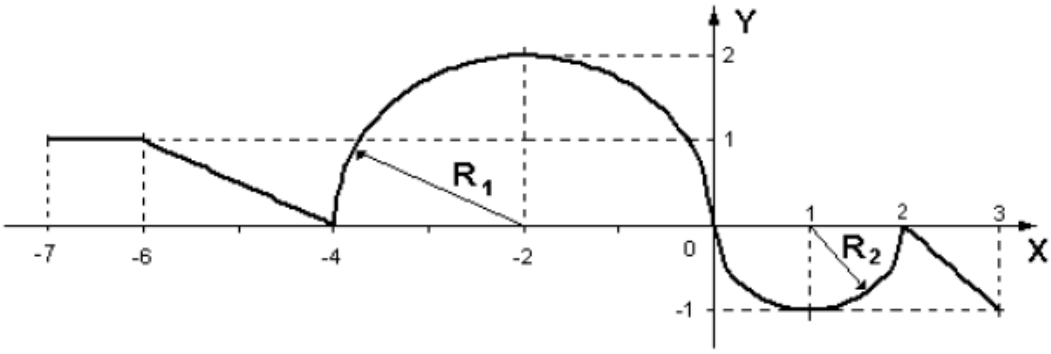
13)


Y
X
3
R
-5
45°
-3
0
6
45°
9

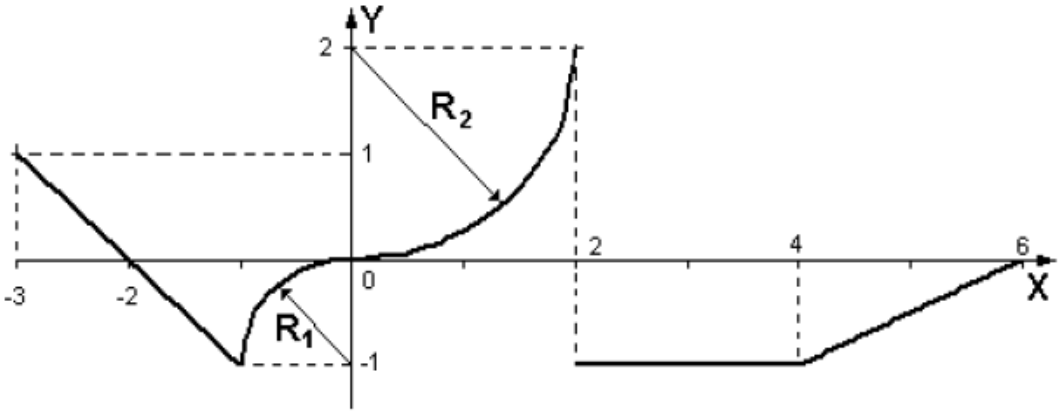
14)


Y
X
4
1
-6
-4
0
1
2
4
7
10
12
-1
-2

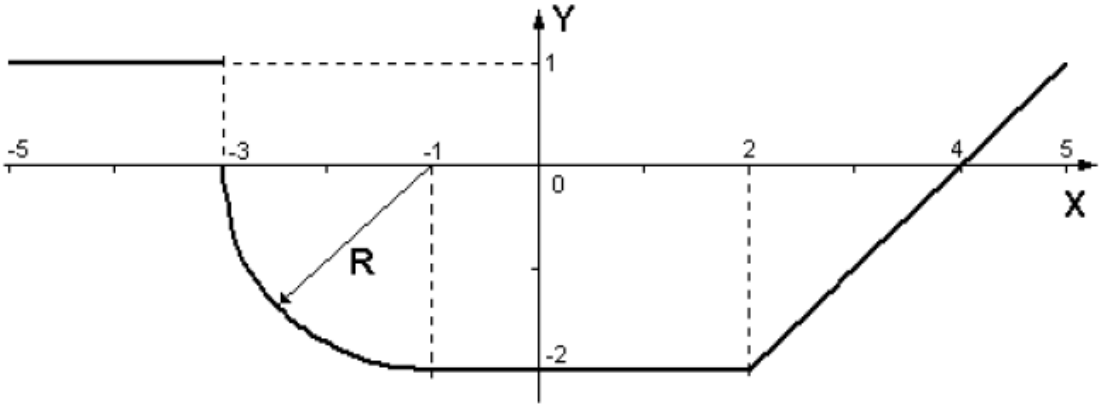
15)


Y
X
2
1
R1
-7
-6
-4
-2
0
1
2
3
R2
-1

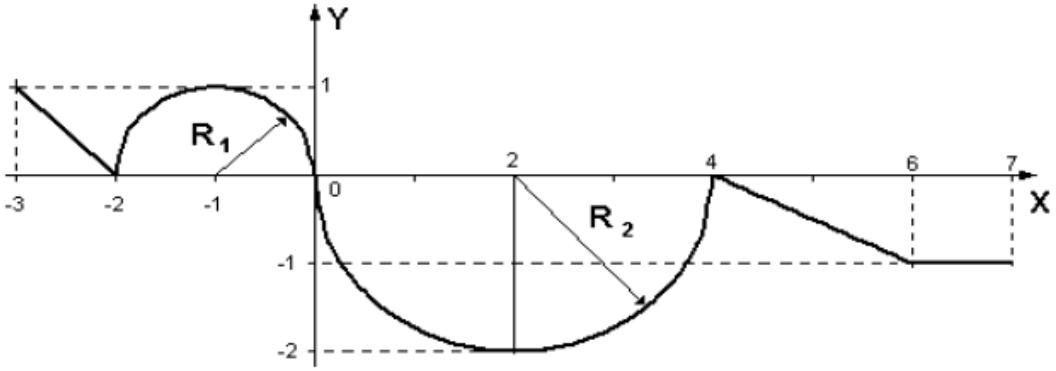
16)


Y
X
2
R2
1
2
4
6
-3
-2
0
R1
-1

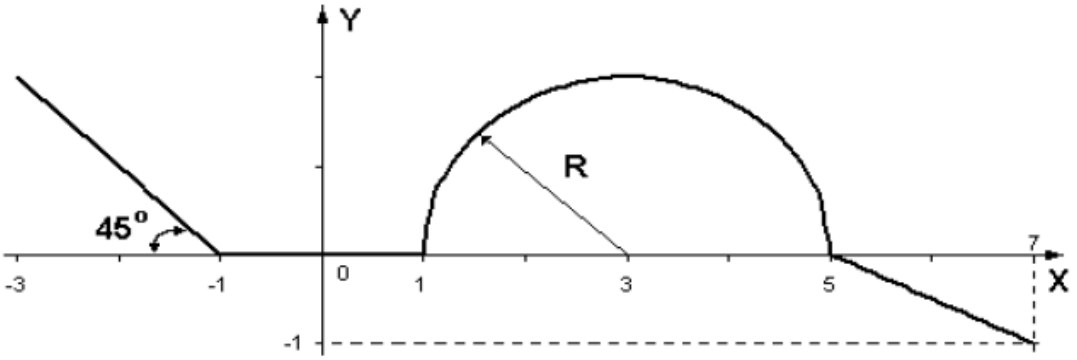
17)


Y
X
1
0
-2
-5
-3
-1
2
4
5
R

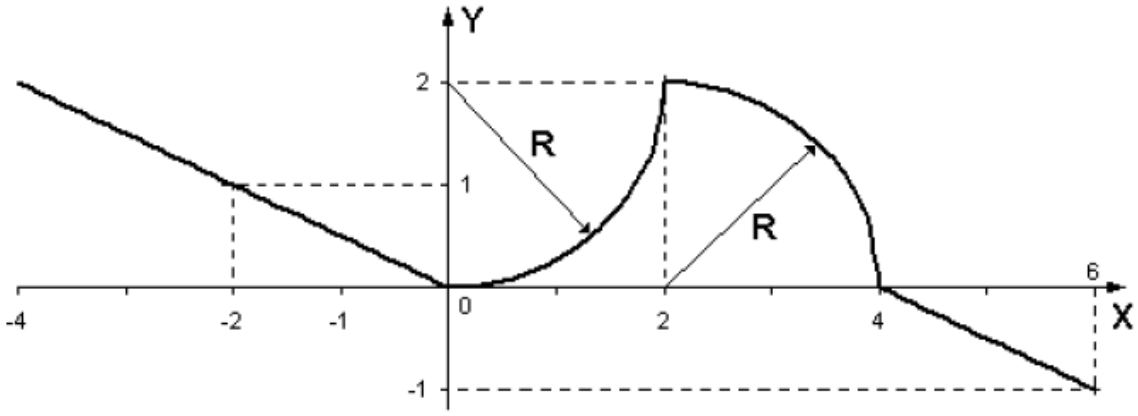
18)


Y
X
1
0
-1
-2
-3
-2
-1
2
4
6
7
R1
R2

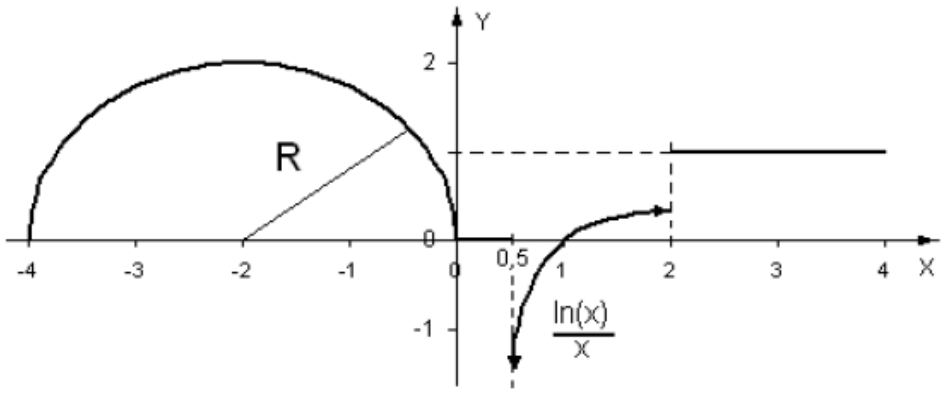
19)


Y
X
0
1
-1
-3
-1
1
3
5
7
45°
R

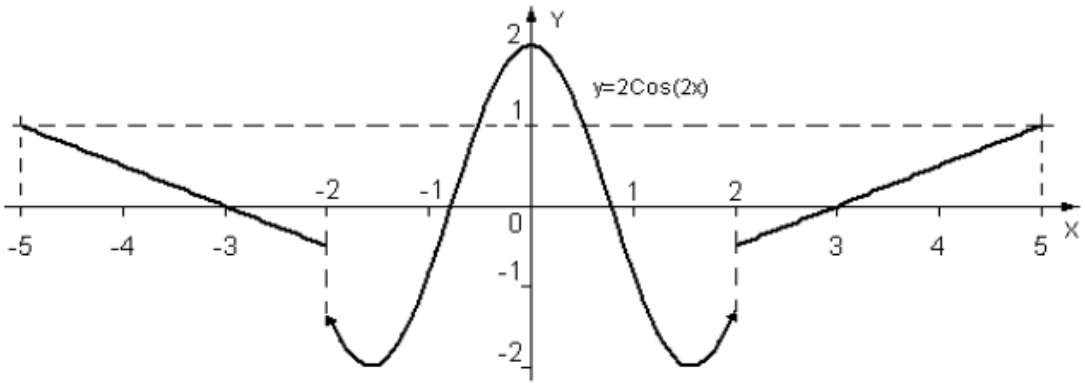
20)


Y
X
2
1
0
-1
-4
-2
-1
2
4
6
R
R

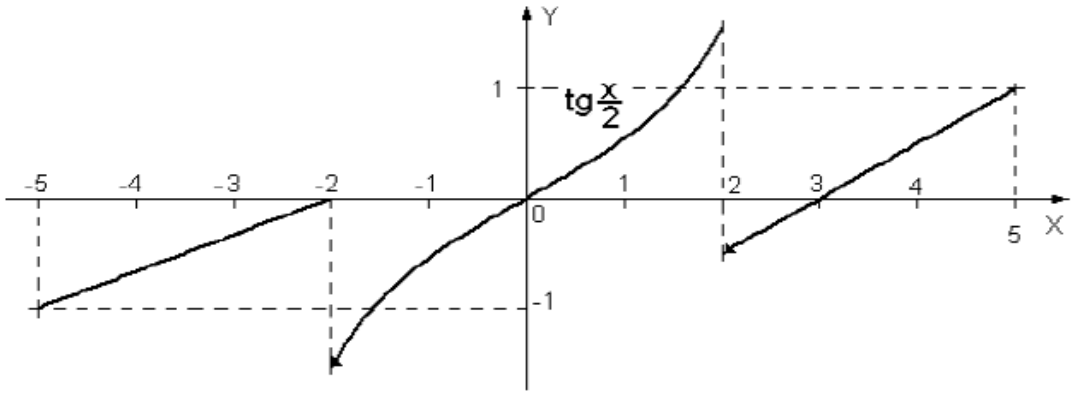
21)


Y
2
R
0
-4
-3
-2
-1
0
0,5
1
2
3
4
X
-1
ln(x)
x

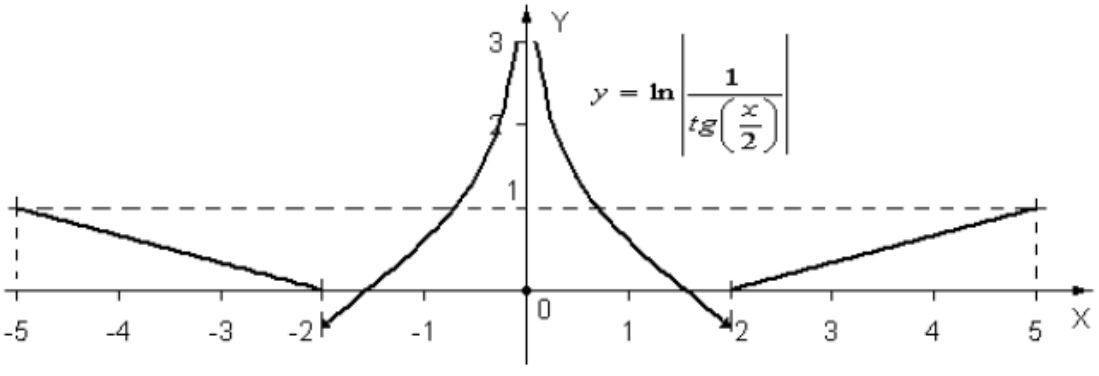
22)


Y
2
y=2Cos(2x)
1
-2
-1
1
2
-5
-4
-3
0
-1
-2
3
4
5
X

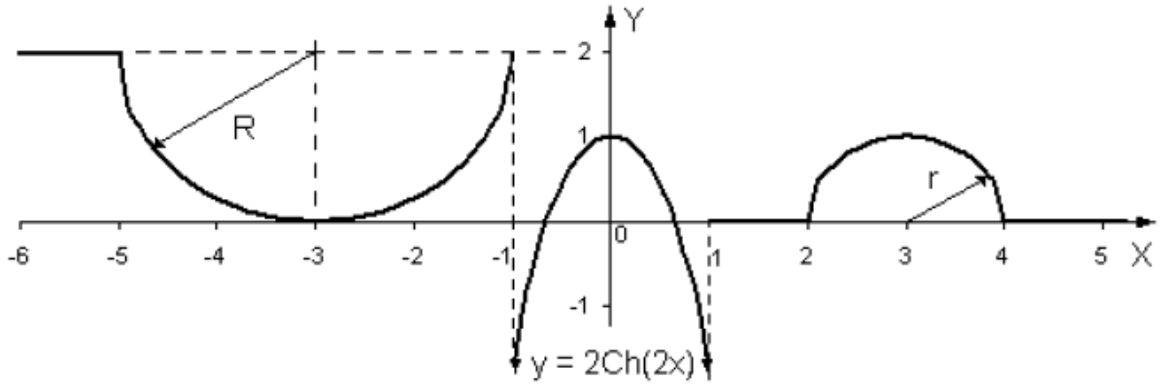
23)


Y
1
tg x/2
-5
-4
-3
-2
-1
1
2
3
4
0
5
X
-1

24)

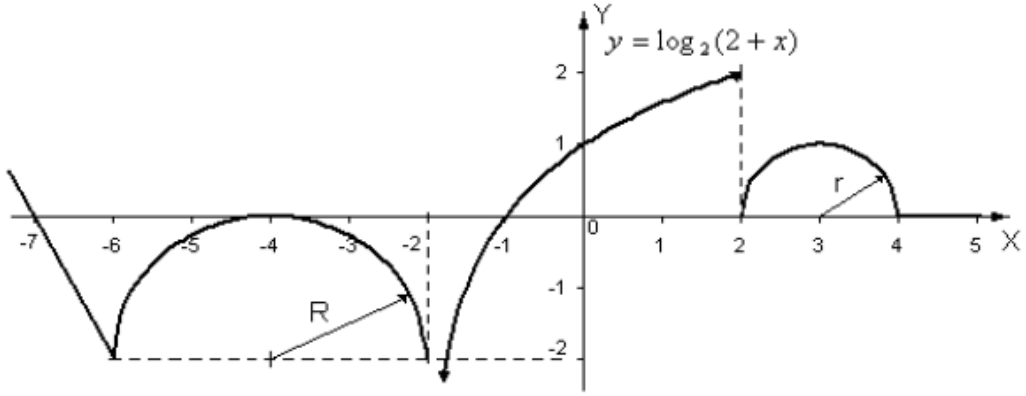

Y
3
y = ln |1 / tg(x/2)|
1
0
-5
-4
-3
-2
-1
1
2
3
4
5
X

25)

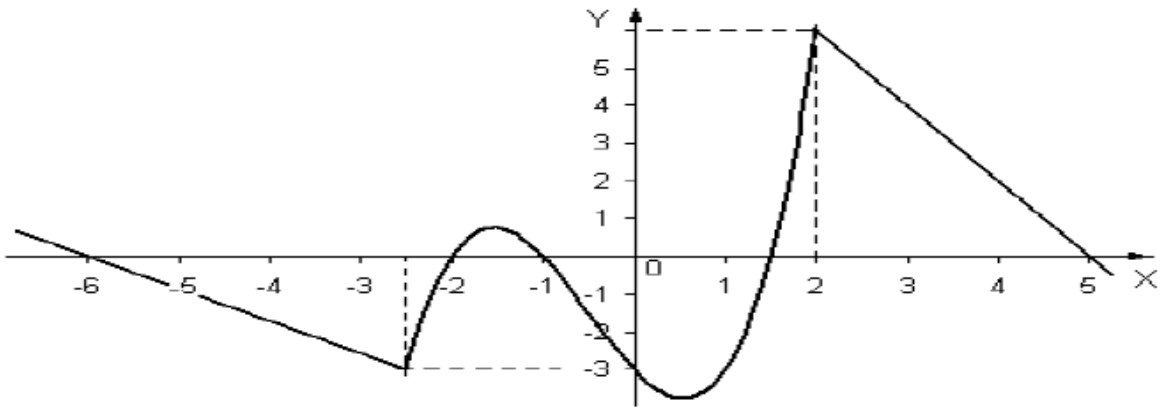


Гиперболический косинус может быть описан формулой: $Ch(x) = \frac{1}{2}\left(e^{x} + e^{-x}\right)$.

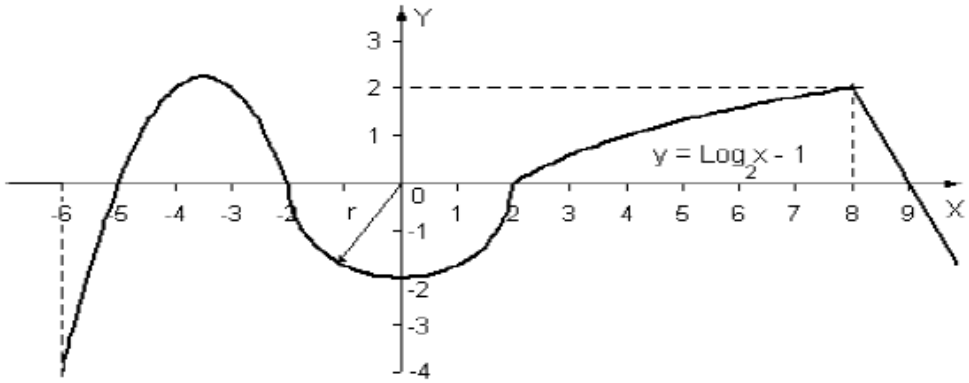
26)



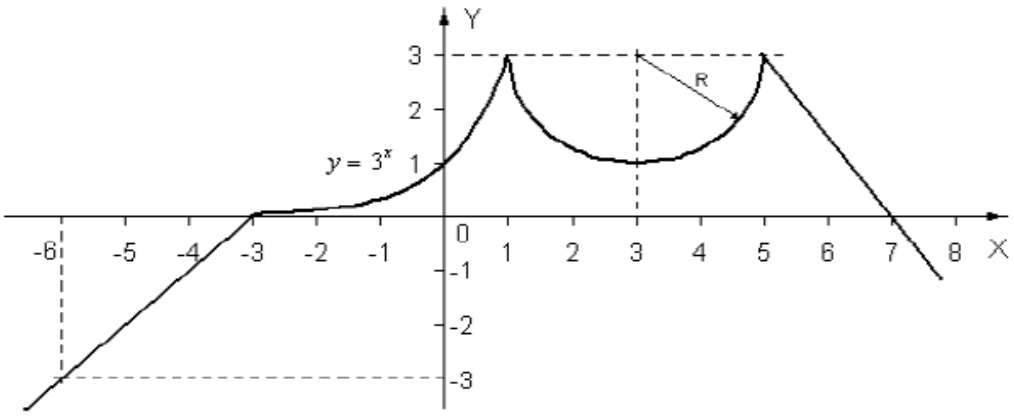
27)



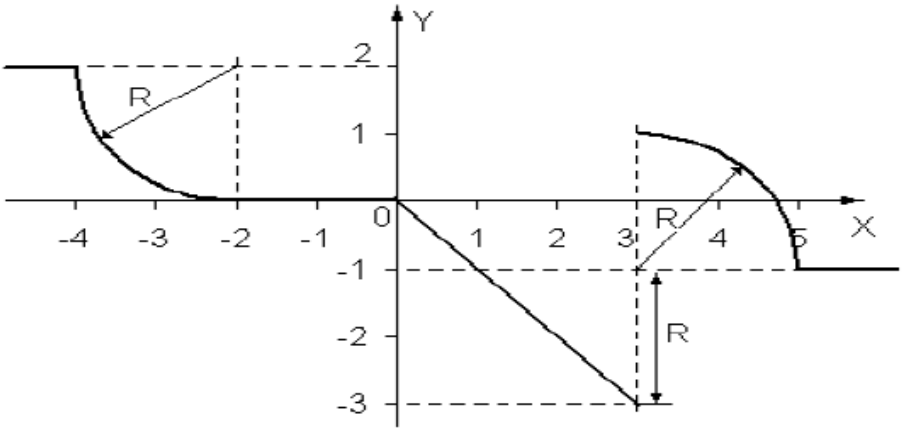
28)



29)



30)

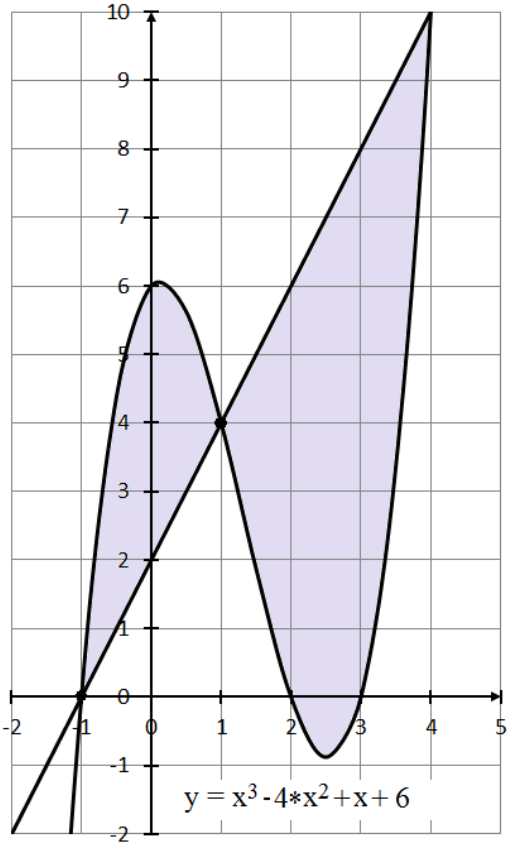


Дополнительное требование к заданию №30: Программа должна быть написана так, что бы решения получались при различных значениях R, вводимых с клавиатуры. Центр положения левой четверти окружности изменяется в соответствии с введённым радиусом, а правой – остаётся постоянным при `X = 3, Y = -1`.

Задание 2

Постановка задачи

Написать программу, которая определяет, попадает ли точка с заданными координатами в заштрихованную область. Точки на границе принадлежат области. Необходимые параметры получить из рисунка. Результат работы программы вывести в виде текстового сообщения: Попадает, Не попадает.



Теоретическое введение

Для решения задачи воспользуемся условным оператором:

```
if <Логическое выражение>:
    <Блок – выполняется, если условие истинно>
[elif <Логическое выражение>:
    <Блок – выполняется, если условие истинно>
]
[else:
    <Блок – выполняется, если все условия ложны>
]
```

<Блок> – это набор инструкций, которые выделяются одинаковым количеством пробелов (обычно четырьмя).

Необходимо правильно составить логическое выражение, параметрами которого будут значение координат точки (x,y) и уравнения линий.

Обмен с консолью выполняется стандартными функциями ввода/вывода: `input()` и `print()`.

Для решения задачи требуется знать уравнение прямой (уравнение кривой приведено на рисунке). Из рисунка получаем координаты двух точек, через которые проходит прямая линия (-1, 0) и (1,4). Подставив значения выбранных точек в уравнение общего вида `y = kx + b`, получим систему уравнений. Решив эту систему, найдём значения для параметров k и b. В итоге уравнение прямой примет вид: `y = 2x + 2`.

Точка с координатами (`x, y`), введённая пользователем, будет попадать в заштрихованную область на интервале [-1, 1] в том случае, если `x` будет не меньше -1 и не больше 1. Кроме этого, `y` должен быть не меньше значения полученного для прямой в точке `x` и не больше значения, вычисленного для кубической параболы в этой точке.

Описание алгоритма

1. Ввести координаты точки (x, y) и привести значения к типу `float`.
2. Выполнить проверку на попадание точки в заданную область.
3. Вывести результат в виде: "Точка `x, y` попадает в область." и "Точка `x, y` не попадает в область."

Описание входных и выходных данных

Входные данные - координаты точки, введённые пользователем. Тип данных и точность представления в задаче не заданы. Установим вещественный тип (`float`).

Выходные данные - сообщения, в текстовом виде, о попадании или непопадании точки в заданную область.

Тестовые примеры

X	Y	Результат
-1	0	Попадает
-0.5	-1	Не попадает
0	3	Попадает
1	4	Попадает
1	5	Не попадает
1.5	1	Не попадает
2	3	Попадает
2.5	-1	Не попадает
2.5	-0.3	Попадает
3.5	10	Не попадает

Листинг программы

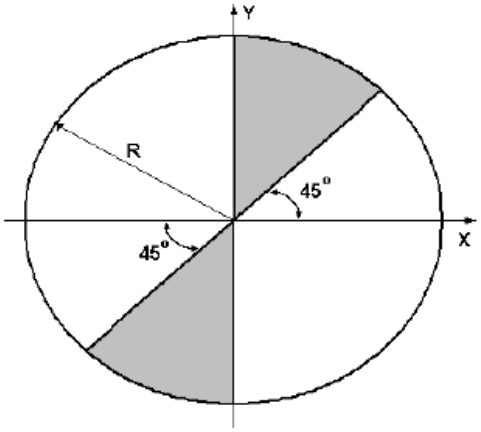
```
from math import *
flag = 0
print('Введите координаты X и Y для точки:')
x = float(input('X='))
y = float(input('Y='))
if (x < -1) or (x > 4):
    flag = 0     #False
```

```
if ((x>=-1) and (x<1) and (y>=2*x+2)
    and (y<=x**3-4*x**2+x+6) or (x>=1)
    and (x<=4) and (y>=x**3-4*x**2+x+6)and (y<=2*x+2)):
    flag = 1
else:
    flag = 0
print("Точка X={0: 6.2f} Y={1: 6.2f}"
        .format(x, y), end=" ")
if flag:
    print("попадает", end=" ")
else:
    print("не попадает", end=" ")
print("в область.")
```

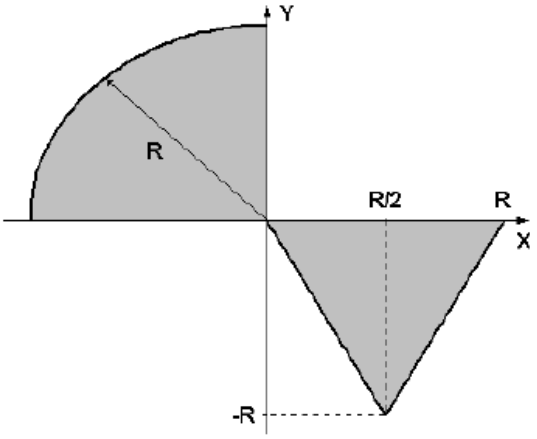
Задание к лабораторной работе №2
«Разветвляющиеся вычислительные процессы».
Задание 2

Написать программу, которая определяет, попадает ли точка с заданными координатами *X, Y* в область, закрашенную на рисунке серым цветом. Результат работы программы вывести в виде текстового сообщения. Параметр *R* вводится с клавиатуры.

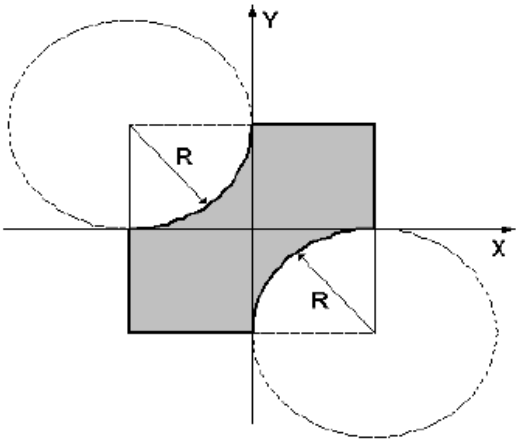
1)



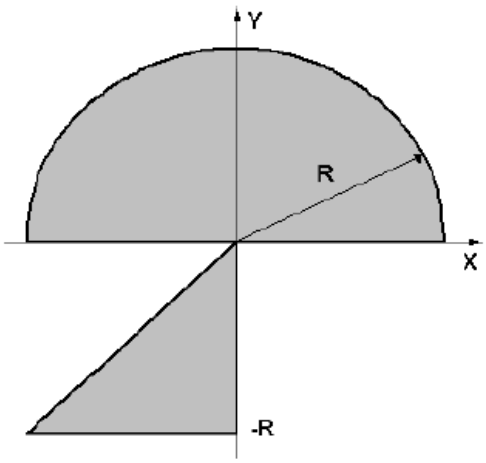
2)



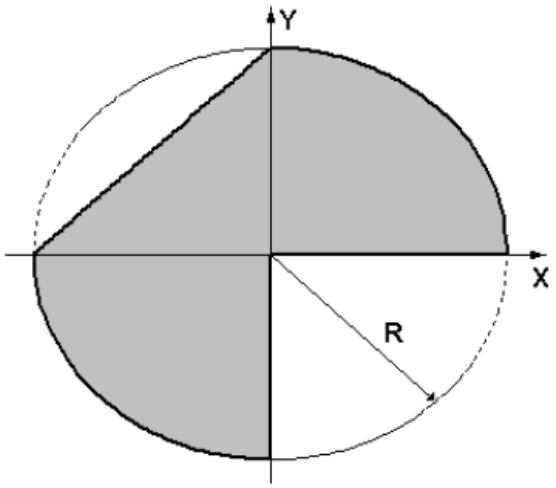
3)



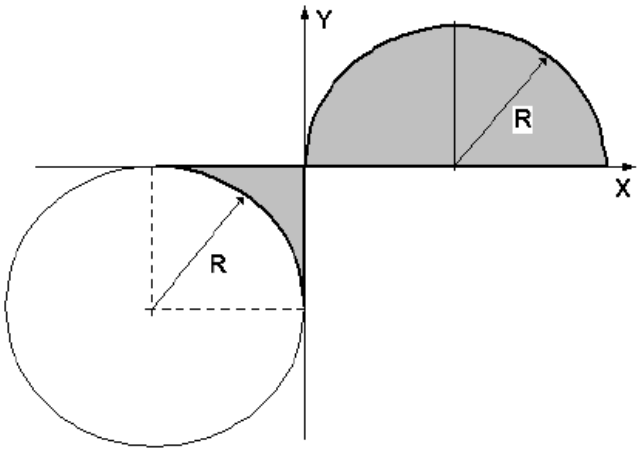
4)



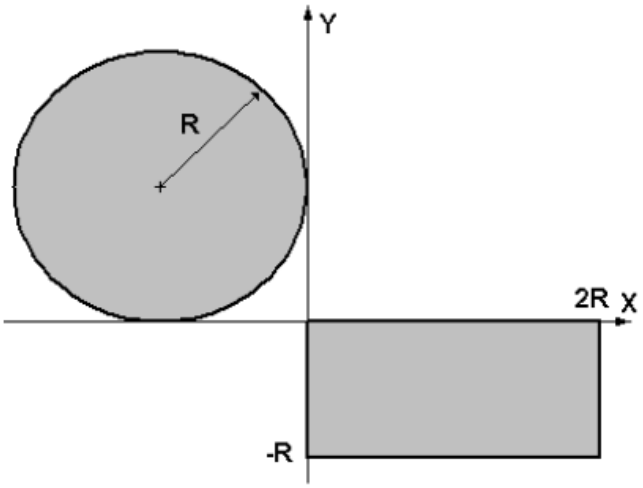
5)



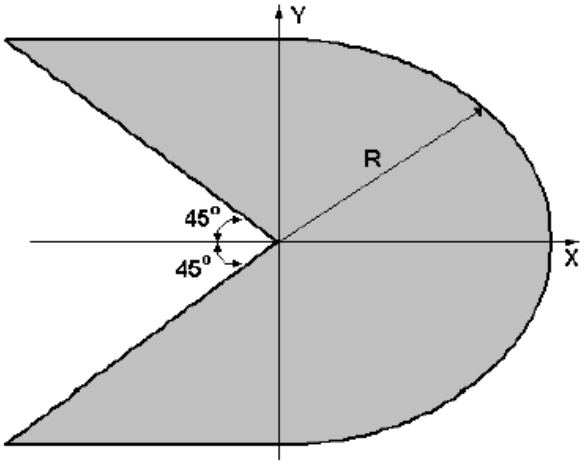
6)



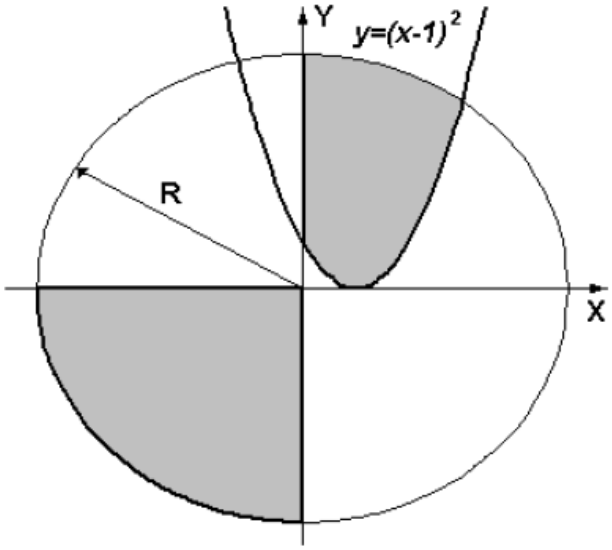
7)



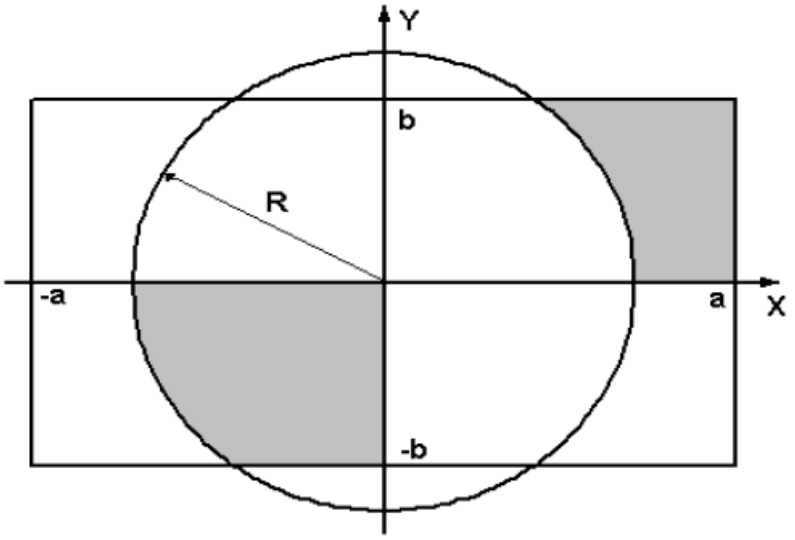
8)



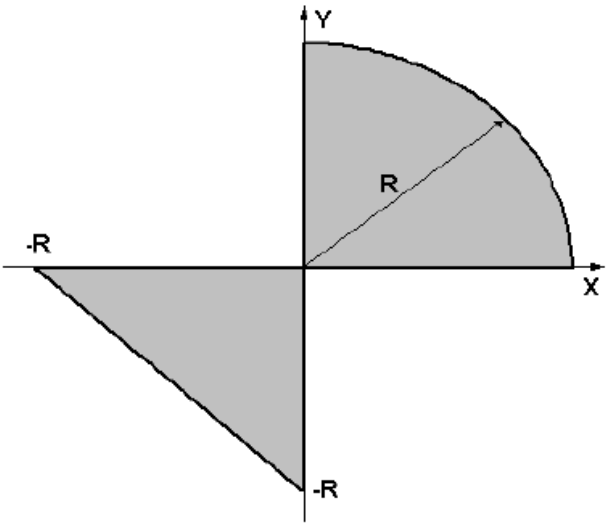
9)



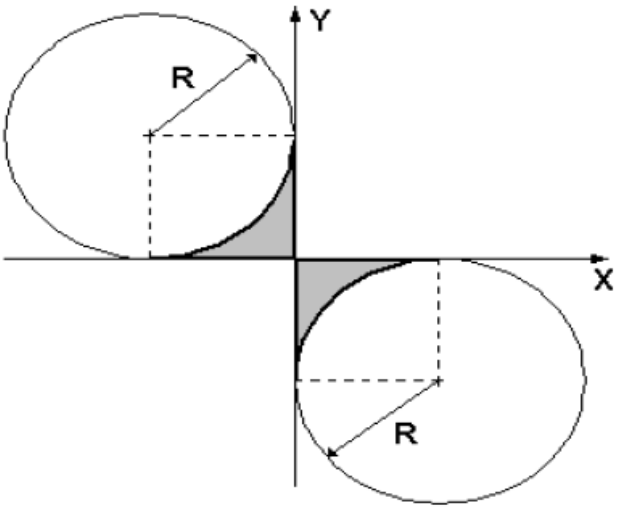
10)



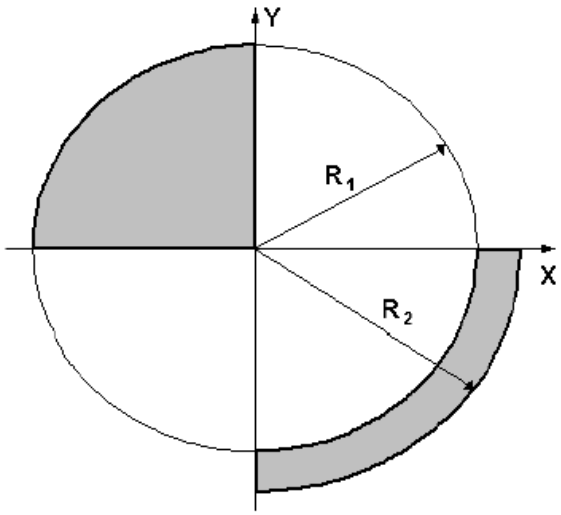
11)



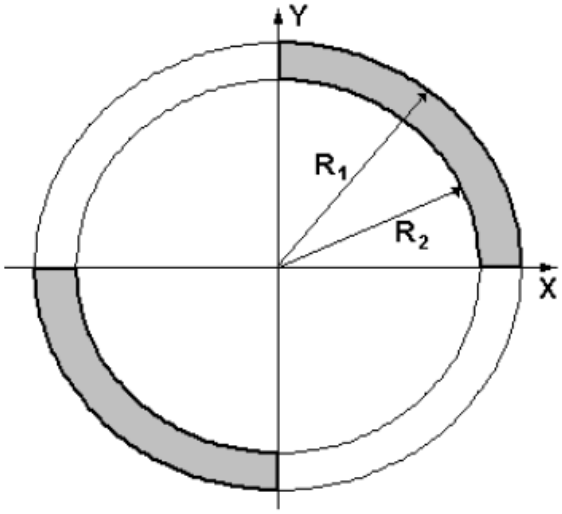
12)



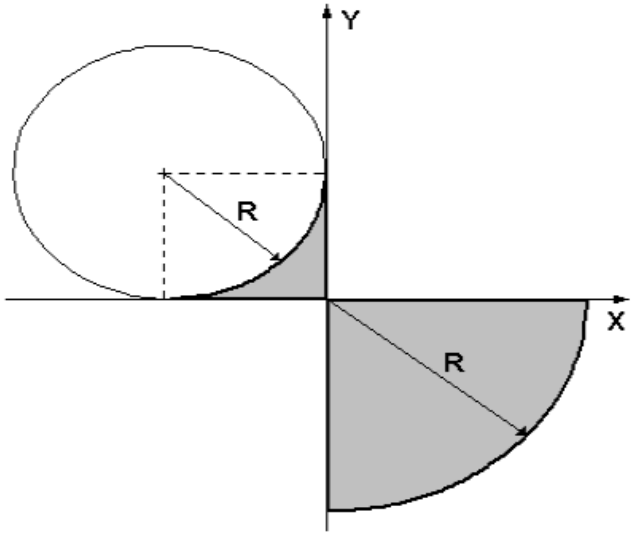
13)



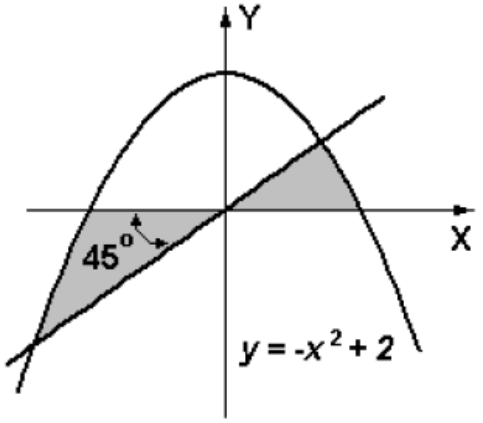
14)



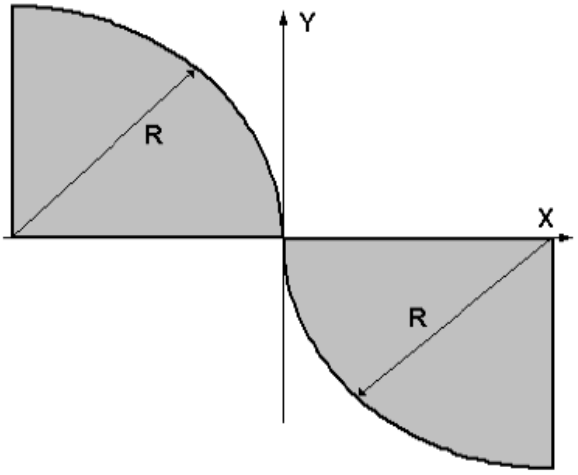
15)



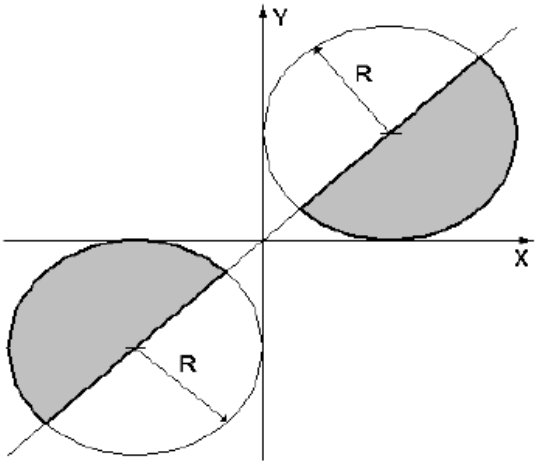
16)



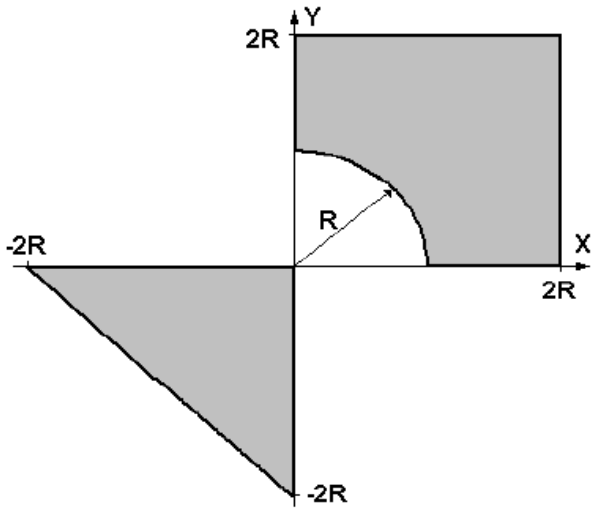
17)



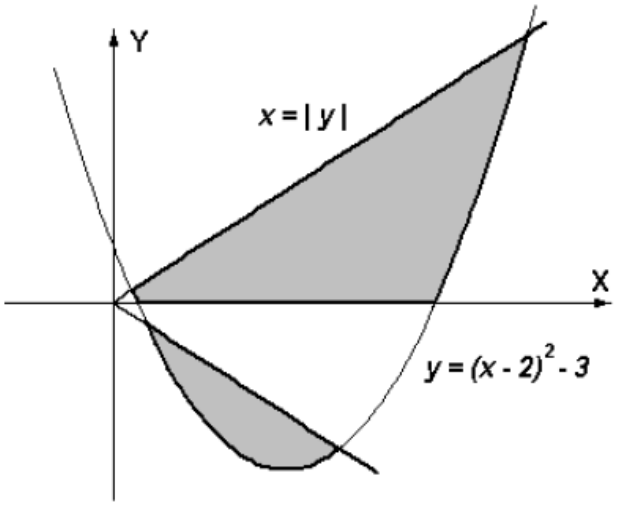
18)



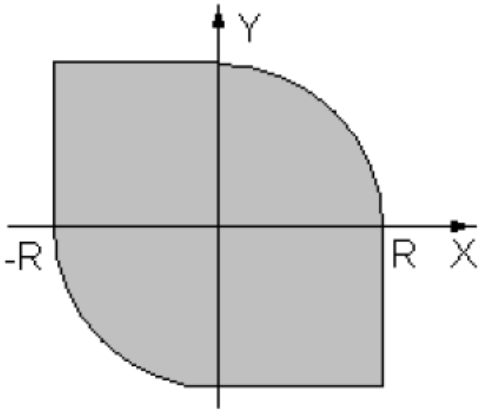
19)



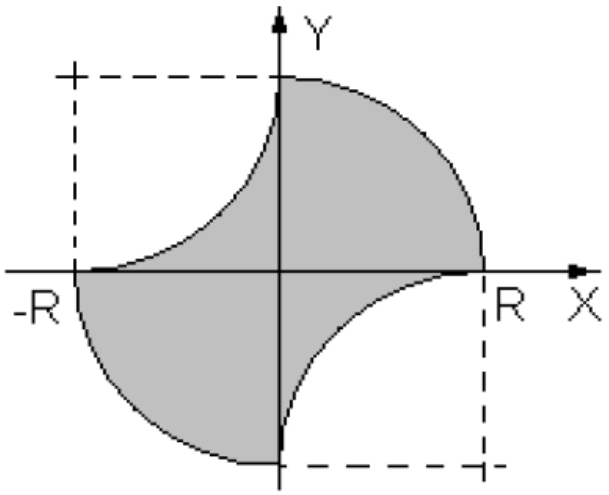
20)



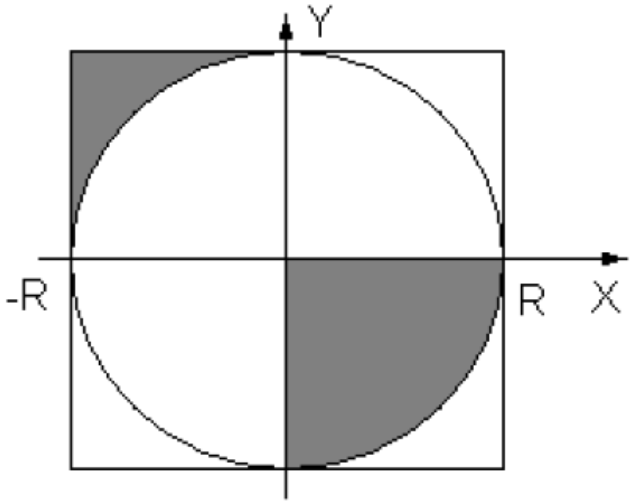
21)



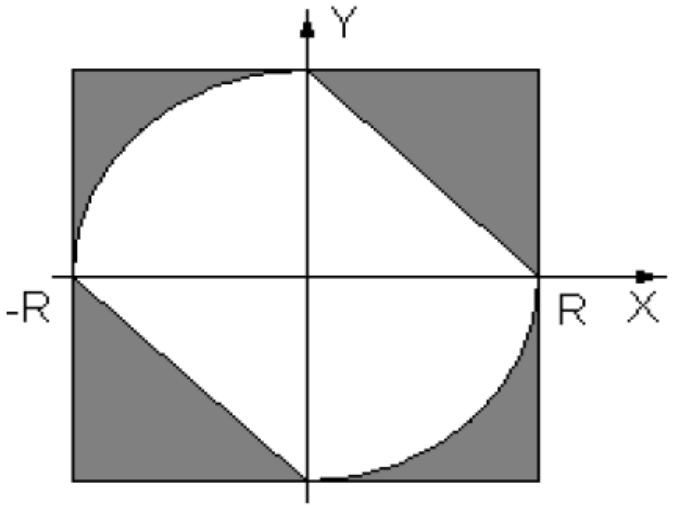
22)



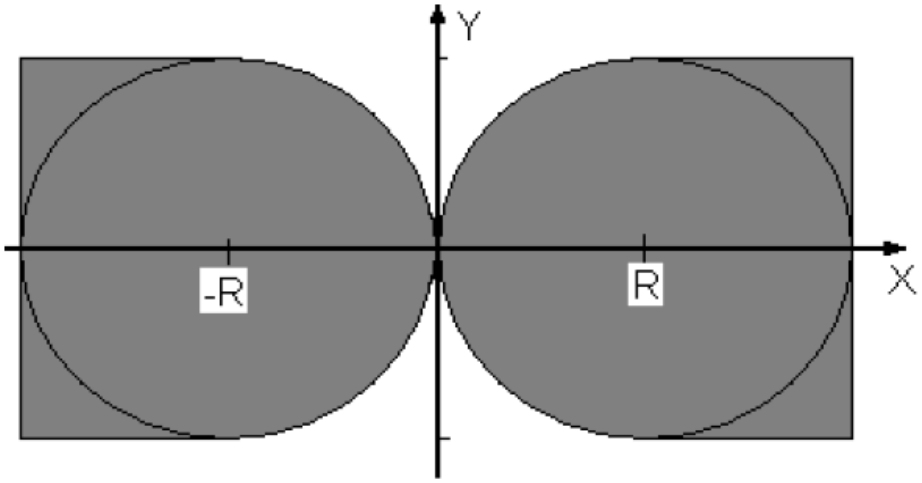
23)



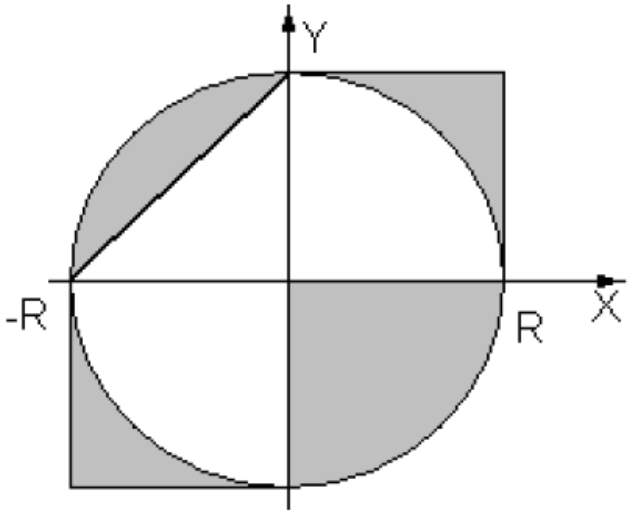
24)



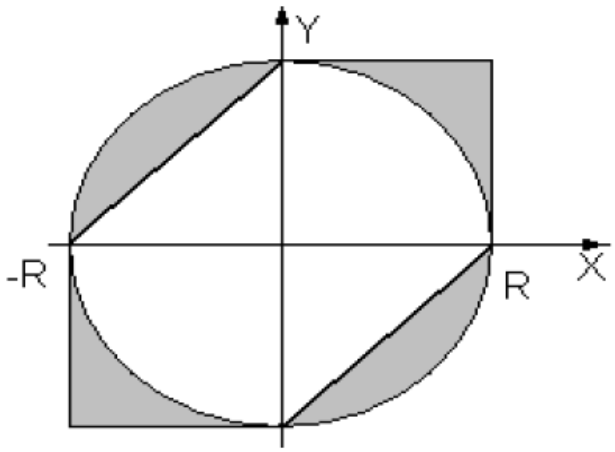
25)



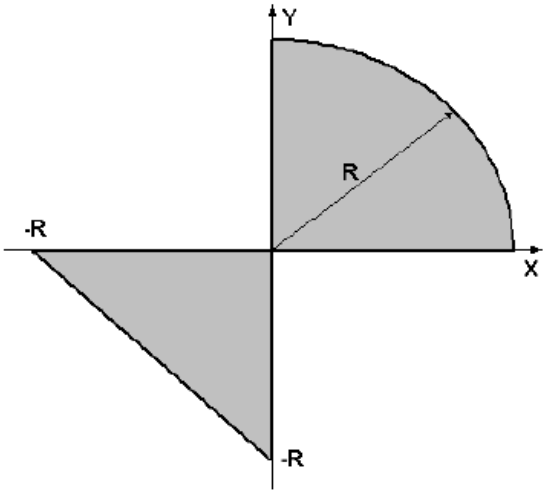
26)



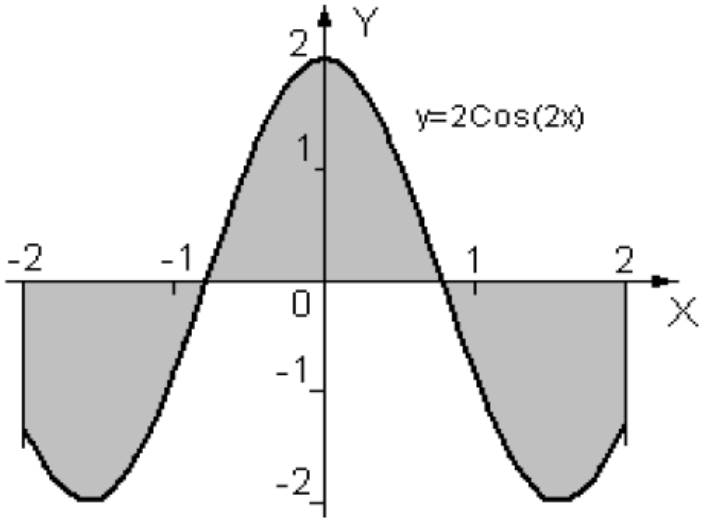
27)



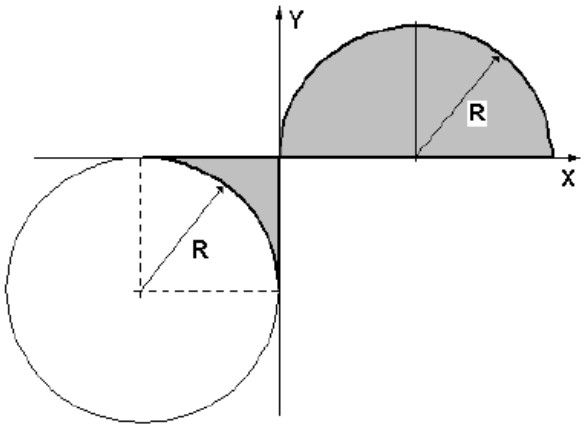
28)



29)



30)



2.3. Лабораторная работа №3: «Организация циклов»

Цель работы: научиться использовать операторы циклов и итерируемые объекты при программировании различных практических задач.

Задание 1

Постановка задачи

Вычислить и вывести на экран в виде таблицы значения функции, заданной графически (см. лабораторная работа № 2, задание 1), на интервале от Хнач до Хкон с шагом dx. Интервал и шаг задать таким образом, чтобы проверить все ветви программы. Таблицу снабдить заголовком и шапкой.

Теоретическое введение

Для решения задачи использована программа, подготовленная в лабораторной работе №2, задание 1 и оператор цикла с предусловием:

```
<Начальное значение>
while <Условие>:
    <Инструкции>
    <Приращение>
[else:
    <Блок, выполняемый, если не использовался break>
]
```

Для обмена с консолью (вывод сообщений и ввод начальных данных) использованы стандартные процедуры print() и input(). Результаты работы программы записываются в текстовый файл.

Описание алгоритма

1. Ввести значения переменных Xbeg, Xend, dx.
2. Присвоить текущему значению Xt начальное значение: Xt = Хнач.
3. Вычислить значение функции и вывести в виде строки таблицы.
4. Вычислить новое значение аргумента Xt = Xt + Dx.
5. Если значение аргумента меньше Xend, то перейти к пункту 3.
6. Завершить рисование таблицы и работу программы.

Описание входных и выходных данных

В предшествующей работе был принят вещественный тип данных (real). В этой работе тип данных сохранён. Для упрощения последующего контроля работы программы в выходной текстовый файл записываются и начальные данные.

Листинг программы

```
from math import *
print('Введите Xbeg, Xend и Dx')
xb = float(input('Xbeg='))
```

```
xe = float(input('Xend='))
dx = float(input('Dx='))
print("Xbeg={0: 7.2f} Xend={1: 7.2f}".format(xb, xe))
print(" Dx={0: 7.2f}".format(dx))
xt = xb
print("+--------+--------+")
print("I X I Y I")
print("+--------+--------+")
while xt <= xe:
    if xt < -5:
        y = 1
    elif xt >=-5 and xt<0:
        y = -(3/5)*xt-2
    elif xt >= 0 and xt<2:
        y = -sqrt(4-xt**2)
    elif xt >= 2 and xt<4:
        y = xt-2
    elif xt >= 4 and xt<8:
        y = 2+sqrt(4-(xt-6)**2)
    else: y = 2
    print("I{0: 7.2f} I{1: 7.2f} I".format(xt, y))
    xt += dx
print("+--------+--------+")
```

Результат работы программы

```
Xbeg= -10.00 Xend= 10.00
Dx= 1.00
+--------+--------+
I X I Y I
+--------+--------+
I -10.00 I 1.00 I
I -9.00 I 1.00 I
I -8.00 I 1.00 I
I -7.00 I 1.00 I
I -6.00 I 1.00 I
I -5.00 I 1.00 I
I -4.00 I 0.40 I
I -3.00 I -0.20 I
I -2.00 I -0.80 I
I -1.00 I -1.40 I
I 0.00 I -2.00 I
I 1.00 I -1.73 I
I 2.00 I 0.00 I
I 3.00 I 1.00 I
I 4.00 I 2.00 I
```

```
I 5.00 I 3.73 I
I 6.00 I 4.00 I
I 7.00 I 3.73 I
I 8.00 I 2.00 I
I 9.00 I 2.00 I
I 10.00 I 2.00 I
+--------+--------+
```

Задания к лабораторной работе №3 «Организация циклов».

Задание 1

Вычислить и вывести на экран в виде таблицы значения функции, заданной графически (см. задание 1 лабораторной работы № 2), на интервале от Хнач до Хкон с шагом dx. Интервал и шаг задать таким образом, чтобы проверить все ветви программы. Таблицу снабдить заголовком и шапкой.

Задание 2

Постановка задачи

Для десяти выстрелов, координаты которых задаются генератором случайных чисел, вывести текстовые сообщения о попадании в мишень (см. лабораторная работа № 2, задание 2).

Теоретическое введение

Для решения задачи использована программа, подготовленная в лабораторной работе №2, задание 2 и оператор цикла с параметром:

for <Текущий элемент> **in** <Последовательность>**:**

<Инструкции внутри цикла>

[**else:**

<Этот код выполняется, если в теле цикла>

<не использовался *break*>]

Вывод сообщения выполняется стандартной функцией print().

Для формирования координат точки используется модуль генератора случайных чисел, который подключается инструкцией:

```
import random
```

или

```
from random import *
```

Для формирования случайного вещественного числа воспользуемся функцией

```
uniform(<Начало>,<Конец>)
```

Эта функция генерирует псевдослучайное число в диапазоне от <Начало> до <Конец>.

При этом правая граница не входит в интервал генерируемых значений (интервал открыт справа). В нашей задаче значения X формируются в диапазоне (-1, 4), а для Y – (-1, 10).

Описание алгоритма

1. Вывести "шапку".
2. В цикле от 1 до 10.

3. Сформировать координаты точки X, Y.
4. Определить попадание точки в заданную область. Если есть попадание, то переменная flag получает значение 1, а иначе – 0.
5. Вывести координаты точки и маркер оставить на строке сообщения.
6. Вывести результат Yes или No в соответствии со значением переменной flag.
7. Изменить параметр цикла и проверить условие завершения.
8. Если условие False, то перейти к п. 3.
9. Завершить работу программы.

Описание входных и выходных данных

Типы переменных, использованные в предыдущей работе, не изменялись. Для организации цикла введена новая переменная целого типа (int).

Листинг программы

```
from math import *
from random import *
flag = 0
print(" X Y Res")
print("--------------------")
for n in range(10):
        x = uniform(-1, 4)
        y = uniform(-1, 10)
    if (x < -1) or (x > 4):
        flag = 0 #False
    if (((x>=-1) and (x< 1) and (y>=2*x+2)
        and (y<= x**3-4*x**2+x+6))
        or
        ((x>=1)and(x<=4)and(y>=x**3-4*x**2+x+6)
        and (y<= 2*x+2))):
        flag = 1
    else:
        flag = 0
    print("{0: 7.2f} {1: 7.2f}".format(x, y), end=" ")
    if flag:
        print("Yes")
    else:
        print("No")
```

Результат тестирования программы

```
X      Y    Res
-------------------
-0.68 7.15 No
```

```
3.53  3.44 No
-0.05 4.02 Yes
-0.24 0.01 No
2.48  5.58 Yes
0.41  4.77 Yes
0.09  0.89 No
0.68  0.98 No
-0.03 5.46 Yes
-0.73 0.73 Yes
```

2.4. Лабораторная работа №4: «Работа со строками»

Цель работы: научиться работать со строками и срезами на языке Python.

Теоретическое введение

Символьная строка – это последовательность символов, расположенных в памяти рядом (в соседних ячейках). Для работы с символами во многих языках программирования есть переменные специального типа: символы, символьные массивы и символьные строки (которые, в отличие от массивов, рассматриваются как цельные объекты). Основной тип данных для работы с символьными величинами в языке Python – это символьные строки (тип string). Для того, чтобы записать в строку значение, используют оператор присваивания

```
s='Вася пошел гулять'
```

Строка заключается в кавычки или в одиночные апострофы. Если строка ограничена кавычками, внутри неё могут быть апострофы, и наоборот. Для ввода строки с клавиатуры применяют функцию input ():

```
s = input( "Введите имя: " )
```

Для того, чтобы выделить отдельный символ строки, к нему нужно обращаться так же, как к элементам массива (списка): в квадратных скобках записывают номер символа. Например, так можно вывести на экран символ строки s с индексом 5 (длина строки в этом случае должна быть не менее 6 символов): `print ( s[5] ).`

Отрицательный индекс говорит о том, что отсчёт ведётся с конца строки. Например, s[-1] означает то же самое, что s[len(s)-1], то есть последний символ строки. В отличие от большинства современных языков программирования, в Python нельзя изменить символьную строку, поскольку

строка – это неизменяемый объект. Поэтому оператор присваивания s[5] = "a" не сработает – будет выдано сообщение об ошибке.

Для того, чтобы выделить часть строки (подстроку) применяется операция получения среза (англ. slicing), например s[3:8] означает символы строки s с 3-го по 7-й (то есть до 8-го, не включая его).

Следующий фрагмент копирует в строку s1 символы строки s с 3-го по 7- й (всего 5 символов):

```
s = "0123456789"
s1 = s[3:8]
```

В строку s1 будет записано значение "34567".

Для удаления части строки нужно составить новую строку, объединив части исходной строки до и после удаляемого участка:

```
s = "0123456789"
s1 = s[:3] + s[9:]
```

Срез s[:3] означает «от начала строки до символа s[3], не включая его», а запись s[9:] – «все символы, начиная с s[9] до конца строки». Таким образом, в переменной s1 остаётся значение "0129".

С помощью срезов можно вставить новый фрагмент внутрь строки:

```
s = "0123456789"
s1 = s[:3] + "ABC" + s[3:]
```

Переменная s получит значение "012ABC3456789".

Если заданы отрицательные индексы, к ним добавляется длина строки N. Таким образом, индекс «–1» означает то же самое, что N-1. При выполнении команд

```
s = "0123456789"
s1 = s[:-1] # "012345678"
s2 = s[-6:-2] # "4567"
```

мы получим s1 = "012345678" (строка s кроме последнего символа) и s2 = "4567". Срезы позволяют выполнить реверс строки (развернуть её наоборот):

```
s1 = s[::-1]
```

Так как начальный и конечный индексы элементов строки не указаны, задействована вся строка. Число «–1» означает шаг изменения индекса и говорит о том, что все символы перебираются в обратном порядке.

Пример обработки строк

Вводится строка, содержащая имя, отчество и фамилию человека, например:

Василий Алибабаевич Хрюндиков.

Каждые два слова разделены одним пробелом, в начале строки пробелов нет. В результате обработки должна получиться новая строка, содержащая фамилию и инициалы:

Хрюндиков В.А.

Описание алгоритма

1. Ввести строку s
2. Найти в строке s первый пробел
3. Имя = всё, что слева от первого пробела
4. Удалить из строки s имя с пробелом
5. Найти в строке s первый пробел
6. Отчество = всё, что слева от первого пробела
7. Удалить из строки s отчество с пробелом # осталась фамилия
8. s = s + " " + имя[0] + "." + отчество[0] + "."

Листинг программы

Вариант 1

```
print ( "Введите имя, отчество и фамилию:" )
s = input()
n = s.find ( " " )
name = s[:n] # вырезать имя
s = s[n+1:]
n = s.find ( " " )
name2 = s[:n] # вырезать отчество
s = s[n+1:] # осталась фамилия
s = s + " " + name[0] + "." + name2[0] + "."
print ( s )
```

Вариант2

```
print ( "Введите имя, отчество и фамилию:" )
s = input()
fio = s.split()
s = fio[2] + " " + fio[0][0] + "." + fio[1][0] + "."
print ( s )
```

Задания к лабораторной работе №4 «Работа со строками».

Выделить в строке-предложении *s* все слова, разделенные символами-разделителями «_.,;:\n\t!?». Обработать выделенные слова в соответствии с вариантом задания.

Регулярное слово – слово, состоящее только из больших латинских букв.

Палиндром – это слово, которое одинаково читается слева направо и справа налево.

1. Подсчитать количество слов, начинающихся на большую букву и содержащих хотя бы один арифметический знак. Напечатать все слова, содержащие две рядом стоящие одинаковые буквы.

2. Подсчитать количество регулярных слов. Напечатать в перевернутом виде все слова, которые содержат два экземпляра заданного символа.
3. Напечатать все слова, начинающиеся с большой буквы. Напечатать самое длинное регулярное слово, которое состоит из одинаковых символов.
4. Напечатать слово, содержащее наибольшее количество цифр. Напечатать количество слов, содержащих хотя бы два арифметических знака.
5. Напечатать все регулярные слова. Напечатать в перевернутом виде самое длинное слово, состоящее только из цифр и латинских букв.
6. Найти количество слов, содержащих более одной цифры, и, исключив все арифметические знаки из этих слов, напечатать их. Напечатать в порядке возрастания все числа, встретившиеся в словах.
7. Определить количество слов, которые содержат заданное подслово и хотя бы одну цифру, и напечатать их. Напечатать в порядке убывания все числа, встретившиеся в словах.
8. Определить количество слов, содержащих и буквы, и цифры, и арифметические знаки. Напечатать их. Напечатать все симметричные слова, содержащие наибольшее количество цифр.
9. Подсчитать количество регулярных слов, содержащих хотя бы две одинаковые буквы. Напечатать все слова, имеющие одну цифру, удалив из таких слов все арифметические знаки.
10. Найти самое длинное регулярное слово и удалить из него все гласные буквы. Найти все слова, в которых имеются либо только цифры, либо только латинские буквы.
11. Подсчитать количество слов, состоящих из одинаковых букв или одинаковых цифр. Напечатать в перевернутом виде слова, имеющие хотя бы один арифметический знак.
12. Напечатать все слова, которые начинаются с большой буквы и заканчиваются заданным двухбуквенным подсловом. Определить количество слов, содержащих согласные латинские буквы, и напечатать порядковые номера этих слов.
13. Напечатать все слова, имеющие в своем составе согласные латинские буквы. Определить количество слов, которые не имеют в своем составе ни одной цифры, и напечатать эти слова.
14. Напечатать все симметричные слова, предварительно удалив из них цифры. Напечатать все слова, состоящие только из согласных латинских букв.

15. Найти все слова, содержащие числа в диапазоне от 10 до 100, и подсчитать их сумму. Напечатать слова, не имеющие цифр, предварительно удалив арифметические знаки.
16. Подсчитать количество слов, начинающихся с большой буквы и оканчивающихся цифрой. Напечатать слова, содержащие заданное подслово и хотя бы один арифметический знак.
17. Подсчитать количество слов, содержащих хотя бы одну согласную латинскую букву и хотя бы одну цифру. Напечатать все слова, состоящие только из четных цифр, и подсчитать сумму этих цифр.
18. Напечатать все слова, которые содержат хотя бы один арифметический знак и заканчиваются на цифру. Определить количество слов, содержащих все маленькие латинские гласные буквы.
19. Найти количество симметричных регулярных слов и напечатать их. Напечатать в перевернутом виде все слова, содержащие согласные латинские буквы.
20. Найти и напечатать все слова, содержащие наибольшее количество букв, если только буквы расположены в алфавитном порядке. Подсчитать количество симметричных слов, имеющих более двух арифметических знаков.
21. Подсчитать количество слов, начинающихся на большую букву и содержащих хотя бы один арифметический знак. Напечатать все слова, содержащие две рядом стоящие одинаковые буквы.
22. Подсчитать количество регулярных слов. Напечатать в перевернутом виде все слова, которые содержат два экземпляра заданного символа.
23. Напечатать все слова, начинающиеся с большой буквы. Напечатать самое длинное регулярное слово, которое состоит из одинаковых символов.
24. Напечатать слово, содержащее наибольшее количество цифр. Напечатать количество слов, содержащих хотя бы два арифметических знака.
25. Напечатать все регулярные слова. Напечатать в перевернутом виде самое длинное слово, состоящее только из цифр и латинских букв.
26. Найти количество слов, содержащих более одной цифры, и, исключив все арифметические знаки из этих слов, напечатать их. Напечатать в порядке возрастания все числа, встретившиеся в словах.
27. Определить количество слов, которые содержат заданное подслово и хотя бы одну цифру, и напечатать их. Напечатать в порядке убывания все числа, встретившиеся в словах.

28. Определить количество слов, содержащих и буквы, и цифры, и арифметические знаки. Напечатать их. Напечатать все симметричные слова, содержащие наибольшее количество цифр.
29. Подсчитать количество регулярных слов, содержащих хотя бы две одинаковые буквы. Напечатать все слова, имеющие одну цифру, удалив из таких слов все арифметические знаки.
30. Найти самое длинное регулярное слово и удалить из него все гласные буквы. Найти все слова, в которых имеются либо только цифры, либо только латинские буквы.

ПРИЛОЖЕНИЕ. Функции и методы для работы со строками

Встроенные функции	**Описание и пример использования**
chr(<код_символа>)	возвращает символ по коду символа
ord(<символ>)	возвращает код символа `>>ord( '3' ) # вернет 51`
len(<s>)	возвращает длину строки s `>>len('Python') # Вернет 6`
str(<obj>)	создает строку из объекта obj
Методы класса str	
dir(str)	Вывод всех методов
str.count(<sub>)	возвращает сколько раз строка sub встречается в строке
str.endswith(<sub >)	Возвращает True если строка кончается на sub, иначе возвращает False.
str.find(<sub> [,<start> [,<end>]])	возвращает первую слева позицию, на которой находится строка sub, если не находит, то возвращает -1, поиск начинается с позиции start и до поз. end, если это указано, иначе поиск начинается с начала.
str.rfind(<sub> [,<start> [,<end>]])	возвращает первую справа позицию, на которой находится строка sub, если не находит, то возвращает -1, поиск начинается с позиции start и до позиции end, если это указано, иначе поиск начинается с начала
str.index(<sub> [,<start> [,<end>]])	работает так же как find, но если подстрока не найдена, вызывает исключение и останавливает программу
str.join(<iter>)	объединяет word с iter `>>> '*'.join( word ) #Здесь основная строка '*' 'P*y*t*h*o*n'`
str.startswith(<sub>)	возвращает True, если строка начинается с sub и False, если нет
str.upper()	создает строку из исходной в верхнем регистре `>>> word.upper() 'PYTHON'`
str.lower()	создает строку из исходной в нижнем регистре `>>> word.lower() 'python'`
str.swapcase()	создает строку, где верхний регистр символов заменен на нижний и наоборот.

	`>>> word.swapcase()` `'pYTHON'`
str.strip()	создает строку без пробелов слева и справа.
str.lstrip()	создает строку без пробелов слева.
str.rstrip()	создает строку без пробелов справа.

str.partition(<sep>)	создает три строки, первая часть строки до разделителя sep, потом строка разделитель sep и третья часть после разделителя.
	`>>> s1,s2,s3 = word.partition('t')`
	`>>> s1,s2,s3`
	`('Py', 't', 'hon')`

str.rpatition(<sep>)	делает тоже самое, но разделитель ищется справа.
str.split([<sep>])	возвращает несколько строк, разделенных sep. Если sep не указан, то строки разделяются по пробелам, а пустые строки (строки не содержащие символы или содержащие только пробелы) не возвращаются.
str.rsplit([<sep>])	делает тоже самое, но обрабатывает строку справа налево.

str.replace(<old> ,<new> [,<count>])	- заменяет все old в строке word на new и создает новую строку. Если указан параметр count, то замена производится count раз. count должно содержать число.
	`>>> word.replace( 'P','J')`
	`'Jython'`
	`>>> word #Обратите внимание, что оригинальная строка не изменилась!!!`
	`'Python'`

2.5. Лабораторная работа №5: «Одномерные массивы»

Цель работы: научиться работать с однородными массивами и списками на языке Python.

Замечание: Для организации массивов в Python могут использоваться разные варианты, например, списки или решения, предлагаемые в модулях `array`, `numpy`, В этих решениях могут быть готовые методы и функции, упрощающие программирование, например поиск максимума, минимума или сортировка. В этом руководстве предлагается использовать алгоритмы поиска решений без использования готовых методов и функций.

Постановка задачи

Сформировать одномерный список, состоящий из N вещественных чисел, полученных генератором случайных чисел. Количество элементов списка (N) запрашивается у пользователя, но не превышает 30. Диапазон значений элементов от -5.0 до 5.0. Вычислить:

1. Первый и второй максимальные по модулю элементы списка.
2. Сумму элементов, модуль которых меньше единицы.
3. Все элементы, модуль которых превышает `Amax` обнулить.
4. Отсортировать список, сохраняя порядок ненулевых элементов. Равные нулю элементы разместить в конце списка.

Теоретическое введение

Для работы с одномерными массивами и матрицами (многомерными массивами) в Python имеются специализированные модули и библиотеки (`array`, `numpy`, …).

Массив – это конечная именованная последовательность однотипных величин.

Для организации массива в Python можно использовать такие структуры данных, как списки, кортежи, множества и диапазоны, которые представляют нумерованные наборы объектов. Каждый элемент набора содержит ссылку на объект, который, в свою очередь, может представлять объект произвольного типа данных и иметь неограниченную степень вложенности.

В решении этого задания для хранения однотипных данных (массивов данных) предлагается использовать структуру данных, которая называется **список**.

Список представляет собой последовательность элементов, пронумерованных от **0 (нуля)**. Элементы списка могут иметь различные типы. Список можно задать перечислением элементов списка в квадратных скобках, например, список можно задать так:

```
Preshe = [1, 3, 5, 7, 11, 13]
PColor = ['Red', 'Orange', 'Yellow', 'Green']
```
или так:
```
Trest = [1, 3, 'Old', 7, 'Or', 13]
```
В списке `Preshe` — 6 элементов, а именно:
`Preshe[0] == 1`, `Preshe[1] == 3`, `Preshe[2] == 5`,
`Preshe[3] == 7`, `Preshe[4] == 11`, `Preshe[5] == 13`.

Список `PColor` состоит из 4-х элементов, каждый из которых является строкой, а вот список `Trest` состоит из шести элементов, часть которых – число, а часть – строка.

При обращении к элементам списка можно использовать как положительные индексы, так и отрицательные. Если индекс положительный, то счет ведется от нуля до максимального элемента, слева направо. Если индекс отрицательный, то счет ведется справа налево:
`Preshe[-1] == 13, Preshe[-6] == 1`.

Количество элементов в списке (длину списка), можно получить при помощи функции `len`, например, `len(Preshe) == 6`.

Существует несколько способов работы со списком. Разработчики предлагают различные варианты от специальных модулей до библиотек. Стандартные решения языка для нашего примера достаточны.

Рассмотрим один из способов создания списка и его наполнения:

- Запросить размер списка у пользователя. Пусть этот размер не должен быть меньше 5 элементов и не превышать 30;
- Создать пустой список (не содержащий элементов, длины 0);
- Наполним список случайными числами, применяя метод `append`.

```
n = int(input("Элементов в списке(N<=30) N: "))
if n > 30: n = 30
elif n < 5: n = 5
mas = [] # Создаем пустой список
for i in range(n): # Инициализация
mas.append(uniform(-5, 5))
```

Если использовать, например, модуль `array`, то изменения в программе будут минимальными:
```
# создание массива нулевой длины
# с элементами вещественного типа
mas = array('f')
for i in range(n): # Инициализация
mas.append(uniform(-5, 5))
```

В решении задачи используется цикл `for` с генерацией последовательности целых чисел, используемых для доступа к элементам массива:
for <Текущий элемент> **in** <Последовательность>:

<Инструкции внутри цикла>

Для обмена с консолью (ввод/вывод) использованы стандартные функции `input()` и `print()`.

После формирования списка, в следующем цикле подсчитывается сумма элементов, модуль которых не превышает единицу. В этом же цикле при обнаружении в массиве элемента, значение которого превышает пороговое значение, выполняется обнуление элемента.

Поиск первого и второго максимальных элементов построен по принципу однопроходного алгоритма:

- Модуль элемента `A[i]` сравнить со значением в `Max1`;
- Если `abs(A[i]) > Max1`, то `Max2 = Max1` и `Max1 = abs(A[i])`;
- Иначе модуль элемента `A[i]` сравнить с `Max2`. Если `abs(A[i]) > Max2`, то `Max2 = abs(A[i])`;

Для сжатия массива по заданному принципу (нулевые элементы размещаются в конце массива при сохранении порядка ненулевых элементов), используется следующий алгоритм:

- Массив просматривается с использованием индексируемой переменной `i`, дополнительный индекс `j` указывает на первый (нулевой) элемент массива;
- При обнаружении не нулевого элемента этот элемент копируется в элемент с индексом `j` и индекс `j` инкрементируется, указывая на следующую освободившуюся позицию;
- После просмотра массива все элементы, начиная с элемента с индексом `j` обнуляются.

Описание алгоритма

1. Запросить количество элементов N и пороговое значение `Amax`.
2. Инициировать массив случайными данными и вывести начальное состояние.
3. В цикле от `0` до `N-1`. Найти сумму элементов, модуль которых меньше `1`, и обнулить элементы, значение которых превысило установленный порог `Amax`.
4. Инициировать `Max1` и `Max2` модулем значения нулевого элемента массива.
5. В цикле от `1` до `N-1`. Если модуль элемента массива больше `Max1`, то `Max1` сохранить в `Max2`, а модуль элемента массива в `Max1`. Иначе, если модуль элемента массива больше `Max2`, то модуль элемента массива сохранить в `Max2`.
6. Инициировать переменную `j`.

7. В цикле от 0 до N-1. Если значение элемента больше нуля, то копировать его в элемент с индексом j. Увеличить j на 1.
8. В цикле от j до N-1. Все элементы приравнять нулю.
9. Вывести полученный массив и значения Max1, Max2 и суммы.

Описание входных и выходных данных

Поскольку тип элементов массива задан как вещественный, то тип переменных, используемых в подсчётах, также определим как вещественный (float).

Листинг программы

```
from math import *
from random import *
n = int(input("Элементов в массиве(N<=30) N: "))
if n > 30: n = 30
elif n < 5: n = 5
amax = float(input("Пороговое значение A: "))
# Генерация массива и вывод
print("Начальное состояние")
mas = []
for i in range(n):
    mas.append(uniform(-5, 5))
    print("{0: 7.3f}".format(mas[i]), end=" ")
print()
# Нахождение суммы
# Обнуление элементов превысивших порог
asum = 0.0
for i in range(n): # от 0 до n-1
    if abs(mas[i]) < 1.0:
        asum = asum + mas[i]
    if abs(mas[i]) > amax:
        mas[i] = 0.0
# Поиск максимального элемента
max1 = abs(mas[0])
max2 = abs(mas[0])
for i in range(1,n):
    if max1 < abs(mas[i]):
        max2 = max1
        max1 = abs(mas[i])
    else:
        if max2 < abs(mas[i]):
            max2 = abs(mas[i])
j = 0
for i in range(n): # Сортировка массива
```

```
    if abs(mas[i]) > 0.00001:
        mas[j] = mas[i]
        j = j + 1
for i in range(j,n): # от j до n-1
    mas[i] = 0.0
print("Конечное состояние")
for i in range(n): # Массив после сортировки
    print("{0: 7.3f}".format(mas[i]), end=" ")
print()
print("max1={0:7.3f} max2={1:7.3f} sum={2:7.3f}"
    .format(max1, max2, asum))
```

Результат работы программы

```
Элементов в массиве(N<=30) N: 6
Пороговое значение A: 2.8
Начальное состояние
    -2.655 2.194 1.034 2.804 -1.490 -0.474
Конечное состояние
    -2.655 2.194 1.034 -1.490 -0.474 0.000
max1= 2.655 max2= 2.655 sum= -0.474
```

Задание к лабораторной работе №5 «Одномерные массивы»

Вариант 1

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Сумму отрицательных элементов.
2. Произведение элементов, расположенных между максимальным и минимальным элементами. Упорядочить элементы массива по возрастанию.

Вариант 2

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Сумму положительных элементов.
2. Произведение элементов, расположенных между максимальным по модулю и минимальным по модулю элементами. Упорядочить элементы массива по убыванию.

Вариант 3

В одномерном массиве, состоящем из *n* целочисленных элементов, вычислить:

1. Произведение элементов с четными номерами.
2. Сумму элементов, расположенных между первым и последним нулевыми элементами. Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом - все отрицательные (элементы, равные нулю, считать положительными).

Вариант 4

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Сумму элементов с нечетными номерами.
2. Сумму элементов, расположенных между первым и последним отрицательными элементами. Сжать массив, удалив из него все элементы, модуль которых не превышает единицу. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 5

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Максимальный элемент массива.
2. Сумму элементов, расположенных до последнего положительного элемента. Сжать массив, удалив из него все элементы, модуль которых находится в интервале [*a, b*]. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 6

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Минимальный элемент массива.
2. Сумму элементов, расположенных между первым и последним положительными элементами. Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом - все остальные.

Вариант 7

В одномерном массиве, состоящем из п целочисленных элементов, вычислить:

1. Номер максимального элемента массива.
2. Произведение элементов массива, расположенных между первым и вторым нулевыми элементами. Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине - элементы, стоявшие в четных позициях.

Вариант 8

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Номер минимального элемента.
2. Сумму элементов, расположенных между первым и вторым отрицательными элементами. Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает единицу, а потом - все остальные.

Вариант 9

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Максимальный по модулю элемент.
2. Сумму элементов, расположенных между первым и вторым положительными элементами. Преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

Вариант 10

В одномерном массиве, состоящем из *n* целочисленных элементов,

вычислить:
1. Минимальный по модулю элемент.
2. Сумму модулей элементов, расположенных после первого элемента, равного нулю.
Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине - элементы, стоявшие в нечетных позициях.

Вариант 11

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:
1. Номер минимального по модулю элемента.
2. Сумму модулей элементов, расположенных после первого отрицательного элемента.
Сжать массив, удалив из него все элементы, величина которых находится в интервале [*a, b*]. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 12

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:
1. Номер максимального по модулю элемента.
2. Сумму элементов, расположенных после первого положительного элемента.
Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале [*a, b*], а потом — все остальные.

Вариант 13

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:
1. Количество элементов массива, лежащих в диапазоне от *A* до *B*.
2. Сумму элементов, расположенных после максимального элемента.
Упорядочить элементы массива по убыванию модулей.

Вариант 14

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:
1. Количество элементов массива, равных нулю.
2. Сумму элементов, расположенных после минимального элемента.
Упорядочить элементы массива по возрастанию модулей.

Вариант 15

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:
1. Количество элементов массива, больших *C*.
2. Произведение элементов, расположенных после максимального по модулю элемента.
Преобразовать массив таким образом, чтобы сначала располагались все отрицательные элементы, а потом - все положительные (элементы, равные нулю, считать положительными).

Вариант 16

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Количество отрицательных элементов.
2. Сумму модулей элементов, расположенных после минимального по модулю элемента.
Заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию.

Вариант 17
В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:
1. Количество положительных элементов.
2. Сумму элементов, расположенных после последнего элемента, равного нулю.
Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает единицу, а потом - все остальные.

Вариант 18
В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:
1. Количество элементов массива, меньших С.
2. Сумму целых частей элементов массива, расположенных после последнего отрицательного элемента.
Преобразовать массив таким образом, чтобы сначала располагались все элементы, отличающиеся от максимального не более чем на 20 %, а потом - все остальные:

Вариант 19
В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:
1. Произведение отрицательных элементов.
2. Сумму положительных элементов, расположенных до максимального элемента.
Изменить порядок следования элементов в массиве на обратный.

Вариант 20
В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:
1. Произведение положительных элементов.
2. Сумму элементов, расположенных до минимального элемента.
Упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Вариант 21
В одномерном массиве, состоящем из *n* целых элементов, вычислить:
1. Максимальный элемент массива.
2. Сумму остатков получаемых от деления элементов массива на максимальный элемент.
Упорядочить элементы массива по возрастанию остатков, полученных от их деления на максимальный элемент.

Вариант 22
В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Среднее значение, как отношение суммы всех элементов на их количество
2. Сумму квадратов разностей между элементами массива и полученным средним значением.
Упорядочить элементы массива по убыванию их разности со средним значением: вначале расположить элементы с положительной максимальной разностью, а затем остальные по убыванию разности.

Вариант 23
В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:
1. Второй по величине элемент.
2. Сумму элементов, расположенных между максимальным и вторым по величине элементами.
Упорядочить элементы массива по возрастанию остатков, полученных от их деления на второй по величине элемент.

Вариант 24
В одномерном массиве, состоящем из *n* вещественных элементов:
1. Вычислить максимальный и минимальный элементы (*Max* и *Min*).
2. Выполнить вставку нового значения в элемент, который расположен в середине между максимальным и минимальным элементами. Вставку выполнить следующим образом: запросить новое значение, которое должно быть в диапазоне ($Min \le a \le Max$); переместить вправо на одну позицию элементы массива от точки вставки (последний элемент теряется); освободившемуся элементу присвоить новое значение.
Упорядочить по возрастанию модулей разности элементов массива и нового значения: $|a_i - a|$

Вариант 25
С использованием модуля *Random* сформировать одномерный массив, состоящий из *n* вещественных элементов в котором элементы принимают случайное значение, и выполняется условие: $a_1 < a_2 < a_3 < \ldots < a_n$.
1. Для заданного числа *y*, такого, что $a_1 < y < a_n$, определить такой *k*, чтобы $a_k < y < a_{k+1}$.
2. Вычислить сумму элементов массива, расположенных до элемента a_k.
Упорядочить по убыванию элементы, стоящие после элемента a_k.

Вариант 26
С использованием модуля *Random* сформировать одномерный массив, состоящий из *n* вещественных элементов в котором элементы принимают случайное значение, и выполняется условие: $a_1 > a_2 > a_3 > \ldots > a_n$.
1. Для заданного числа *y*, такого, что $a_1 < y < a_n$, определить такой *k*, чтобы $a_k < y < a_{k+1}$.
2. Вычислить сумму элементов массива, расположенных после элемента a_k.
Упорядочить по убыванию элементы, стоящие до элемента a_k.

Вариант 27
С использованием модуля *Random* сформировать одномерный массив,

состоящий из *n* вещественных элементов в котором элементы случайным образом принимают положительный или отрицательный знак и значение от -5 до 5. Для заданного числа *y*, такого, что *amin* < *y* < *amax*, вычислить:
1. Сумму элементов массива, значения модуля которых меньше *y*.
2. Произведение остальных элементов.

Вариант 28

С использованием модуля *Random* сформировать одномерный массив, состоящий из *n* вещественных элементов в котором элементы случайным образом принимают положительный или отрицательный знак и значение от -10 до 10. Для заданного числа *y*, такого, что *amin* < *y* < *amax*, вычислить:
1. Произведение элементов массива, значения модуля которых больше *y*.
2. Сумму модулей остальных элементов.

Вариант 29

Координаты *n* точек заданы как элементы одномерного массива. Нечётные элементы - значения ординат, а чётные - абсцисс. Вычислить максимальное расстояние между ближайшими точками.
Вывести координаты всех точек, которые попадают в круг с координатами центра *X0, Y0* и радиусом *R*. Упорядочить положение точек по возрастанию их ординат.

Вариант 30

Координаты *n* точек заданы как элементы одномерного массива. Нечётные элементы - значения ординат, а чётные - абсцисс. Вычислить минимальное расстояние между ближайшими точками.
Вывести координаты всех точек, которые попадают в кольцо с координатами центра *X0, Y0*, внутренним радиусом *R1* и внешним радиусом *R2*.
Упорядочить положение точек по возрастанию их абсцисс.

2.6. Лабораторная работа №6: «Файлы»

Цель работы: научиться работать с файлами на языке Python.

Постановка задачи

В предыдущих заданиях необходимые для программы данные, вводились с клавиатуры, а результат выводился на экран монитора. Очевидно, что и при отладке программ, и при вводе большого объема данных в программу такой подход требует значительных трудозатрат и потому непригоден. Наиболее подходящее решение – это ввод данных из файла и вывод результатов работы в файл. При этом входные данные подготавливаются однократно и с должным многообразием, а результаты работы можно анализировать многократно.

Напишем программу, которая будет считывать входные данные из заранее подготовленного текстового файла, и выводить результат в текстовый файл. Для этой цели воспользуемся программами, написанными в предыдущих заданиях.

Теоретическое введение

Файловый тип данных введён в языках программирования для работы с внешними устройствами – файлами на диске, портами ввода/вывода, принтерами и т.д. Файловый тип подразделяют на текстовый и бинарный (двоичный).

Текстовый файл содержит данные типа строка (str), а бинарный – типа bytes.

Доступ к файлам может быть последовательным или прямым. При последовательном доступе каждый следующий элемент может быть прочитан только после выполнения аналогичной операции с предыдущим элементом. При прямом доступе операция чтения (записи) может быть выполнена для произвольного элемента с заданным адресом.

Текстовые файлы относятся к файлам с последовательным доступом. Они предназначены для хранения информации строкового типа. При этом ввод и вывод информации сопровождается преобразованием типов данных. При выводе в текстовый файл данные преобразуются из внутреннего представления в символы, а при вводе выполняется обратное преобразование.

Бинарные файлы относятся к файлам с прямым доступом. В них хранится информация в двоичном виде. При записи или чтении бинарного файла информация не подвергается дополнительному преобразованию.

Для организации работы с файлами, при программировании на языке высокого уровня, выполняются, как правило, четыре шага:

- создается объект файла. Для этого используются подпрограммы, которые связывают имя файла, задаваемое пользователем, с переменной, которая хранит ссылку на специально создаваемую

операционной системой структуру. Эта структура содержит информацию о файле, о буфере данных, через который будет проходить обмен между программой и файлом и о текущем состоянии процесса обмена данными;

- задается режим обмена, в котором будет происходить работа с файлом: будет ли это режим чтения, записи, добавления или какой-либо совмещенный режим, например, чтение и запись. Этот шаг реализуется либо после создания объекта файла, либо в процессе выполнения первого шага;
- производится запись или чтение данных. Процесс обмена данными между программой и файлом состоит в обмене данными между программой и буфером данных под управлением файловой подсистемы. При записи данных в буфер, файловая подсистема контролирует процесс записи и при заполнении буфера до некоторого уровня выполняет запись данных в файл, а буфер очищается, разрешая программе продолжать запись. При чтении данных из буфера файловая подсистема контролирует объем данных в буфере и, при необходимости, выполняет подкачку свежих данных из файла;
- выполняется операция закрытия файла. При этом остаток данных, находящийся в буфере записывается в файл и файл закрывается.

Операция закрытия файла обязательно должна выполняться, если файл был открыт на запись. Если файл не закрыть, то при завершении программы, ресурсы, выделенные операционной системой, будут закрыты. В этом случае может возникнуть состояние, когда файл окажется пуст (чаще всего) либо будет содержать не всю информацию, которую в него пытались записать. Это зависит от размера буфера, который в современных ОС может быть достаточно большим.

Файлы, открытые на чтение, так же необходимо закрывать. Для этого есть две причины:

- открытый на чтение файл блокируется и другие приложения не получают к нему доступа;
- и в программе, и в операционной системе есть ограничение на число открытых файлов.

В объектно-ориентированном языке программирования, как это, например, реализовано в Python, часть описанных шагов может быть реализована в скрытой форме.

Например, при создании файлового объекта первый и второй шаги выполняются в одной инструкции:

```
fh = open(<Имя_файла> [, mode = <mod>])
```

где `fh` – переменная, хранящая ссылку на файловый объект, `<Имя_файла>` – абсолютный или относительный путь и имя файла, `mode=<mod>` – режим в котором открывается файл: запись, чтение, добавление, …

Язык Python поддерживает протокол менеджеров контекста. Этот протокол гарантирует правильное закрытие файла в независимости от того, произошло исключение (ошибка) внутри блока кода или нет. Например, следующий код открывает файл на запись, записывает в файл строки, закрывает файл, а затем вновь открывает его, выводит текст на экран и закрывает файл. Обратите внимание на то, что операция закрытия файла в явном виде в коде отсутствует:

```
with open(r"lab6.txt","w") as fh:
fh.write("Меркурий\n") # Запись строк в файл
fh.write("Венера\n")
fh.write("Земля\n")
# В этом месте файл fh закрыт автоматически
with open(r"lab.6.txt","r") as fh:
print(fh.read())
# В этом месте файл fh так же закрыт
```

При создании объекта файла, необходимо указывать путь и имя существующего, либо будущего файла. Путь к файлу можно задавать как относительно текущей рабочей папки, так и абсолютно. При этом под относительным путем понимается путь относительно текущей рабочей папки, а под текущей рабочей папкой понимается папка, в которой находится пользователь в момент запуска файла (запускаемый файл может находиться в другой папке).

Далее рассмотрена модификация программы, написанной к лабораторной работе №1. Приведены способы работы с файлами.

В этой работе мы учились записывать выражения на языке Python. Внесем следующее изменение в нашу программу:

- подготовим текстовый файл с исходными данными;
- используя инструкции для работы с текстовым файлом, прочитаем записанные строки и выполним вычисления;
- результат вычисления запишем в текстовый файл в виде таблицы.

Вычисляемые выражения оформим в виде функций:

```
def f1(a,x):
    y = (tan(x**2/2-1)**2+(2*cos(x-pi/6))
        /(1/2+sin(a)**2))
    return y
def f2(x):
    y = pow(2, log(3-cos(pi/4+2*x),3+sin(x))
        /(1+tan(2*x/pi)**2))
    return y
```

Текстовый файл

Установим следующий формат текстового файла:

- две строки – это шапка, в которой указано назначение столбцов. Эти строки снабдим символом комментария, который используется в Python: '#';
- два столбца – это данные, для которых будут проводиться вычисления.

Пример:

```
# a x
#-----------
-2 -2
 0 -2
...
```

Используем упрощенную схему работы с текстовым файлом.

Создать файловый объект:

`fh = open(<Имя_файла> [, mode = <mod>])`,

где `fh` – переменная, хранящая ссылку на создаваемый файловый объект, `<Имя_файла>` – абсолютный или относительный путь и имя файла, `mode=<mod>` – режим в котором открывается файл. Вместо `<mod>` подставляется один из символов, см. таблицу.

mod	Режим	Примечание
'r'	чтение файла (read)	файл должен существовать, если файл не существует, то возбуждается исключение: FileNotFoundError
'w'	запись в файл (write)	если файла несуществует, то он создается
'a'	добавление в файл (append)	если файла несуществует, то он создается, запись выполняется в конец файла
'r+'	чтение и запись	файл должен существовать, если файл не существует, то возбуждается исключение: FileNotFoundError
'w+'	чтение и запись	если файла несуществует, то он создается, существующий файл перезаписывается
'a+'	чтение и запись	если файла несуществует, то он создается, запись выполняется в конец файла
'x'	создать файл для записи	если файл существует, то возбуждается исключение: FileExistsError
'x+'	создать файл для чтения и записи	если файл существует, то возбуждается исключение: FileExistsError

Вместе с указанием режима может следовать модификатор, определяющий режим открытия файла: `t` – текстовый или `b` – бинарный.

Для решения нашей задачи нам необходимо открыть два файла. Один файл будет содержать информацию для расчета выражений и будет открыт на чтение, а второй – для вывода результатов расчета – будет открыт для записи:

```
fi = open("lab1_pb_in.txt", mode = "rt")
```

```
fo = open("lab1_pb_ou.txt", mode = "wt")
```

Читать из файла:

Чтение файла можно организовать по-разному, например, считывать по строкам (метод `readline()`) или считать весь файл в буфер (метод `readlines()`) и затем обрабатывать строки. Обратим внимание на то, что строки заканчиваются символом конца строки (`'\n'`). Метод `readline()` читает строку, включая и символ конца строки.

При чтении по строкам итерацию (чтение следующей строки) можно выполнять через цикл `for`. Рассмотрим два примера:

```
1) while True:
    line = fi.readline() # чтение строки
    if not line: # строка пустая
        break # конец файла и обработки
    elif line=="\n": # конец строки
        continue # продолжим чтение строк
    (b, c) = line.split() # разделить строку
2) for line in fi: # для всех строк файла
    if line=="\n": # конец строки
        continue # продолжим чтение строк
    (b, c) = line.split("\t") # разделить строку
```

В первом примере итерации выполняются в цикле `while` при выполнении инструкции чтения строки. Считанное значение сохраняется в переменной `line`, и проверяется на то, что получено не пустое значение. Если инструкция `fi.readline()` вернет пустую строку, то это значит, что прочитан конец файла (`EOF`) и дальнейшую обработку можно прекратить (`break`). Кроме этого проверяется, что строка содержит информацию. Если строка не содержит информации, то в ней будет только символ конца строки. В этом случае обработку следует продолжить с чтения следующей строки (`continue`).

Замечание: Если строка не содержит информацию (в строке нет символов, которые можно было бы визуализировать), то в ней есть символ конца строки. Если строка пустая, то в ней НЕТ НИКАКИХ символов.

Во втором примере итерации (чтение строк) выполняются циклом `for`. Строки по очереди считываются в переменную `line`. В этом случае нет необходимости контролировать конец файла (`EOF`) .

Дальнейшая обработка считанной информации выполняется в соответствии с форматом записи данных.

В нашем примере мы поместили в начале файла две строки, описывающие данные, которые следуют за ними, а сами данные разместили по строкам в форме двух столбцов. Разделителем между столбцами может выступать знак табуляции или несколько пробелов.

При чтении такого файла поступим следующим образом:

- прочитаем две строки без обработки (пропустим эти строки). Эти строки – памятка, которой можно воспользоваться при подготовке файла в текстовом редакторе;
- в цикле читаем строку, и расщепляем ее для получения данных.

Формат записи метода расщепления (разделения) следующий:
`str.split(sep=None, maxsplit=-1),`
где `str` – строка символов, `sep` – разделитель, а `maxsplit` – количество груп, на которые делится строка.

Если указан разделитель `sep` (sep – от separate – разнимать) и количество групп `maxsplit`, то строка будет разделена на `maxsplit + 1` части. Если `maxsplit` не указан или равен -1, то число частей, на которые будет поделена строка неограниченно. Пример разделения строки:

```
'1,2,3'.split(',',maxsplit=1) # ['1', '2,3']
```

Мы можем не указывать параметры в методе `split()`, поскольку при расщеплении будет формироваться список из двух значений (в строке два столбца), а разделителем мы выбрали пробелы или знак табуляции.

В левой части инструкции мы укажем две переменные, которые примут значения, полученные при расщеплении строки.

```
b, c = line.split()
```

Полученные при расщеплении значения будут строкового типа и для дальнейшего их использования необходимо выполнить преобразование к вещественному типу (`float`).

Записывать в файл:

Для записи в файл будем использовать метод `write(<Данные>)`. Под данными тут выступает строка, в том числе и форматная строка, содержащая знаки форматирования. При записи строки в файл метод `write()` не добавляет символа конца строки. Для того, что бы следующие данные записывались с новой строки, необходимо, что бы текущая строка завершалась символом конца строки (`'\n'`).

Листинг программы

```
from math import *
def f1(a,x):
    y = (tan(x**2/2-1)**2+(2*cos(x-pi/6))
        /(1/2+sin(a)**2))
    return y
def f2(x):
    y = pow(2, log(3-cos(pi/4+2*x),3+sin(x))
        /(1+tan(2*x/pi)**2))
    return y
fi = open("lab1_pb_in.txt", "rt") #читать файл
fo = open("lab1_pb_ou.txt", "wt") #писать в файл
```

```
line = fi.readline() # Пропустить строки
line = fi.readline() # заголовка в файле
# Вывести шапку таблицы в файл
fo.write("+======+======+========+========+\n")
fo.write("I A I X I F1 I F2 I\n")
    fo.write("+======+======+========+========+\n")
for line in fi: # для всех строк файла
    if line=="\n":
        continue
    (b, c) = line.split() # расщепить
    a = float(b) # привести к вещест. типу
    x = float(c)
# Вывод в файл
    fo.write("I {0: .1f} I {1: .1f} I {2: 5.4f} I"
        .format(a, x, f1(a, x)))
    fo.write("{0: 6.4f} I\n".format(f2(x)))
    fo.write("+------+------+--------+" "--------+\n")
fi.close() # закроем файлы
fo.close()
```

Результат работы программы (текстовый файл lab1_pb_ou.txt)

```
+=======+=======+========+========+
I   A   I   X   I    F1   I  F2    I
+=======+=======+========+========+
I -2.00 I -2.00 I 1.1970  I 1.1184 I
+-------+-------+--------+--------+
I 0.00  I -2.00 I -0.8347 I 1.1184 I
+-------+-------+--------+--------+
I 0.00  I 0.00  I 5.8896  I 1.6880 I
+-------+-------+--------+--------+
I 2.00  I 0.00  I 3.7309  I 1.6880 I
+-------+-------+--------+--------+
I 1.50  I 0.50  I 2.7712  I 1.7955 I
+-------+-------+--------+--------+
I 4.00  I 3.00  I -1.3266 I 1.0517 I
+-------+-------+--------+--------+
```

Задание к лабораторной работе №6 «Файлы»

Выполнить корректировку программы, написанной для лабораторной работы №1, чтобы ввод данных и вывод результатов работы осуществлялся с использованием файлов.

3. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите наиболее популярные области применения Python.
2. Назовите основные алгоритмические конструкции языка Python.
3. Назовите основные встроенные структуры данных в языке Python. Для чего они используются? Есть ли альтернатива для них?
4. Каким образом можно получить автономный исполняемый файл из Python программы?
5. Можно ли использовать функциональное программирование в Python-программе?
Если можно, то какими средствами языка это поддерживается?
6. Какие основные средства существуют в Python для установки дополнительных библиотек?
7. Каким образом можно сделать общедоступный собственный Python-модуль?
8. Что такое лямбда-функция?
9. В чем заключаются особенности ООП в Python? Можно ли писать Python программы, не используя пользовательские классы?
10. Перечислите несколько модулей стандартной библиотеки языка Python.
11. Каким образом можно интегрировать Python с другими ЯП?
12. Каким образом можно использовать регулярные выражения вместе с Python?
13. Перечислите наиболее слабые стороны Python и области, где его применение нежелательно.
14. Какие реализации Python вы знаете, на каких платформах они доступны?

4. ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

Содержание отчета

1. Титульный лист с названием лабораторной работы, номером своего варианта, фамилией студента и группы.
2. Тексты программ.
3. Результаты работы программ.

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Ульяновский государственный университет»
Факультет математики, информационных и авиационных технологий

Лабораторная работа №

По курсу «Программирование на языке Python»

«Название работы»

Выполнил студент группы _____ _____
название группы ФИО

Проверил _____ _____
должность ФИ

Ульяновск

Вариант №.

Задание для выполнения

(Формулировка задания.)

Код программы:

(Вставить код написанной программы)

Вид командного окна с результатом выполнения программы:

(Вставить скрин с результатом работы)