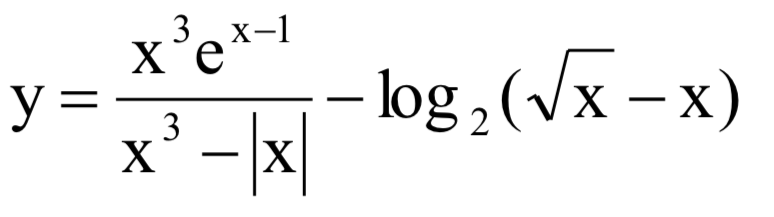
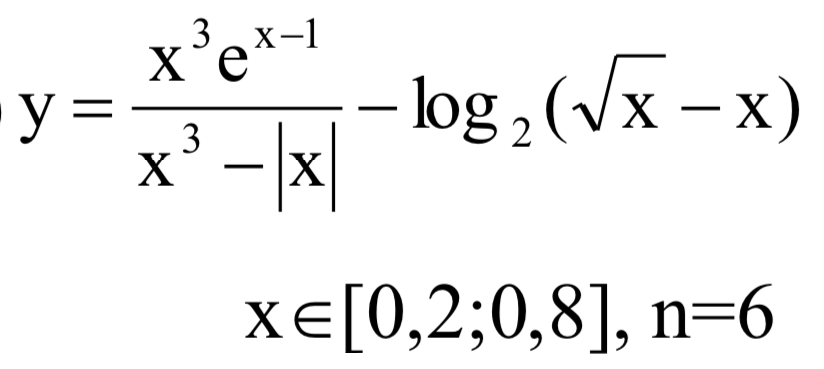
# Задание №1, 3

## Условия для оператора if:



## Условия для оператора for:



## Исходный код программы:

*"""  
Деменчук Г.М., вариант 6, стандартные задания  
"""***import** math  
  
**import** matplotlib.pyplot **as** plt  
  
  
**class** GraphClass():  
 *"""  
 Вариант задания для операторов цикла + график  
  
 Класс для вывода графика варианта задания   
 для операторов цикла  
 """* **def** \_\_init\_\_(self):  
 self.graph()  
  
 **def** graph(self):  
 args = ([], [])  
 x = 0.2  
 end\_cycle = 0.8  
  
 **while** x != end\_cycle:  
 obj = MathUpper(x)  
 args[0].append(x)  
 args[1].append(obj.result)  
 x = round(x + 0.1, 2)  
  
 ax = plt.figure().gca()  
 ax.plot(args[0], args[1], linewidth=2, marker=**"o"**)  
 plt.show()  
  
  
**class** MathUpper():  
 *"""  
 Вариант задания на условные операторы  
  
 Класс для работы с первым заданием  
 на условные операторы  
 """* **def** \_\_init\_\_(self, x):  
 self.x = x  
 self.getter()  
  
 **def** getter(self):  
  
 x = self.x  
 upper = x \*\* 3 \* math.e \*\* (x - 1)  
 lower = x \*\* 3 - math.fabs(x)  
 **if** lower == 0:  
 print(**"Знаменатель равен нулю, деление на 0!"**)  
 self.result = 0  
 **return** first = upper / lower  
  
 log\_sqrt = math.sqrt(x) - x  
  
 **if** log\_sqrt >= 0:  
 buf\_log = math.log(log\_sqrt, 2)  
 **else**:  
 print(**"Выражение в log[sqrt(x)-x,2] меньше 0!"**)  
 self.result = 0  
 **return** self.result = first - buf\_log  
  
  
**class** CycleClass():  
 *"""  
 Вариант задания для операторов цикла  
  
 Класс для вызова MathUpper в цикле  
 """* **def** \_\_init\_\_(self):  
 self.cycle()  
  
 **def** cycle(self):  
 x = 0.2  
 end\_cycle = 0.8  
  
 **while** x != end\_cycle:  
 obj = MathUpper(x)  
  
 print(**"x="**, x, **"result = "**, obj.result)  
 x = round(x + 0.1, 2)  
  
 **pass  
  
  
def** main():  
 **try**:  
 x = float(input(**"Введите x: "**))  
 **except**:  
 print(**"Проблема ввода данных!"**)  
 **return** obj = MathUpper(x)  
 print(**"\n\*Условные операторы\*"**)  
 print(**"Результат:"** + str(obj.result))  
  
 print(**"\n\*Операторы цикла\*"**)  
 CycleClass()  
  
 GraphClass()  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

## Скриншоты программы:

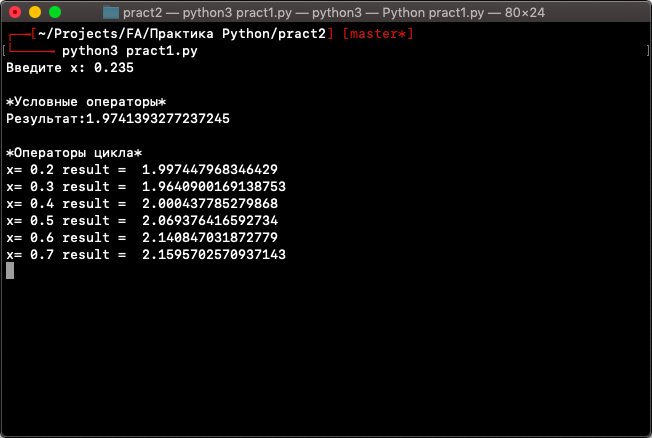


Рис 1 - Стандартный ввод данных

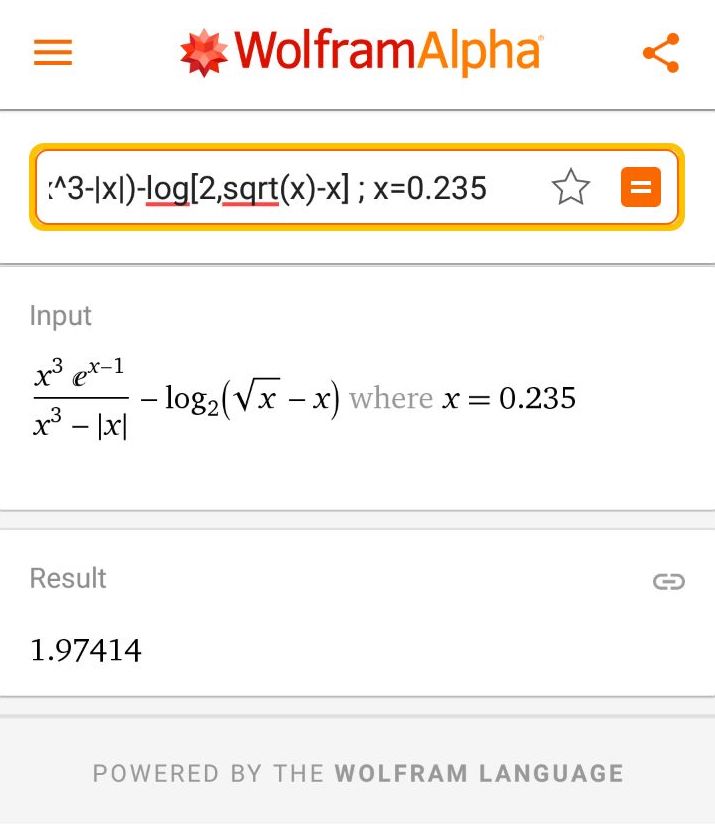


Рис 2 – Проверка результатов вычисления в WolframAlpha

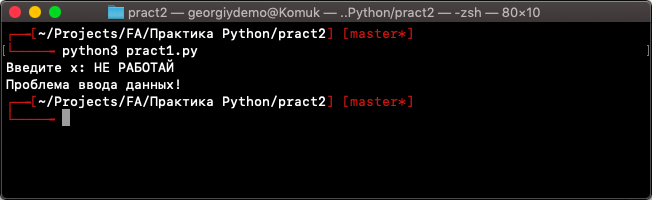


Рис 3 – Реализация фильтрации вводных данных

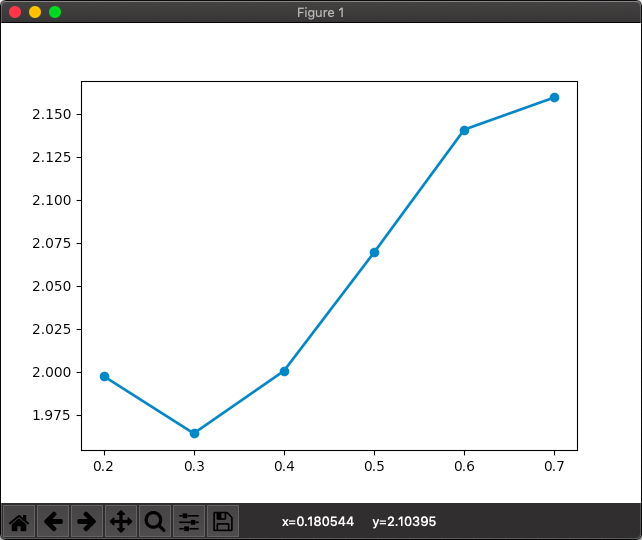


Рис 4 – График аргументов x,y в цикле for

# Задание №2

## Условие:

Вывести следующий график

## Исходный код программы:

*"""  
Деменчук Г.М., вариант 6, задания со звёздочкой \*  
"""***import** matplotlib.pyplot **as** plt  
  
  
**def** get\_scale(x, y):  
 *"""  
 Метод для возврата точки на основании c и d  
 Для динамического масштаба  
 """* x = abs(x)  
 y = abs(y)  
 **if** x > y:  
 **return** x  
 **return** y  
  
  
**class** LineGraphClass():  
 *"""  
 Функции y=y(x), заданные графически\*  
 """* **def** \_\_init\_\_(self, c, d):  
  
 self.c = c  
 self.d = d  
 self.graph()  
  
 **def** graph(self):  
 c = self.c  
 d = self.d  
  
 *# Необходимо для адекватного масштаба* p = get\_scale(c, d)  
  
 *# Координаты основной оси* m = ([-p \* 4, p \* 4], [0, 0])  
 *# Исходные точки* x = [-p \* 2, -c, 0]  
 y = [0, 0, d]  
 *# Аннотации к слевой линии* n1 = [**""**, **"-c"**, **"d"**]  
 *# Аннотации к правой линии* n2 = [**"-d"**, **"c"**, **""**]  
  
 plt.style.use(**'seaborn-dark'**)  
 ax = plt.figure().gca()  
  
 *# Ось координат X* ax.plot(m[0], m[1], linewidth=1, color=**"k"**)  
 *# Ось координат Y* ax.plot(m[1], m[0], linewidth=1, color=**"k"**)  
  
 *# Линия левая* ax.plot(x, y, linewidth=2, marker=**"o"**, color=**"b"**)  
 **for** i **in** range(len(x)):  
 ax.annotate(n1[i], (x[i], y[i] + 0.5))  
  
 *# Линия правая* x = [-e **for** e **in** reversed(x)]  
 y = [-e **for** e **in** reversed(y)]  
  
 ax.plot(x, y, linewidth=2, marker=**"o"**, color=**"b"**)  
 **for** i **in** range(len(x)):  
 ax.annotate(n2[i], (x[i], y[i] + 0.5))  
 plt.show()  
  
  
**def** main():  
 **try**:  
 d = float(input(**"Введите d -> "**))  
 c = float(input(**"Введите c -> "**))  
 **except**:  
 print(**"Проблема ввода данных!"**)  
 **return** LineGraphClass(c, d)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

## Скриншоты программы:

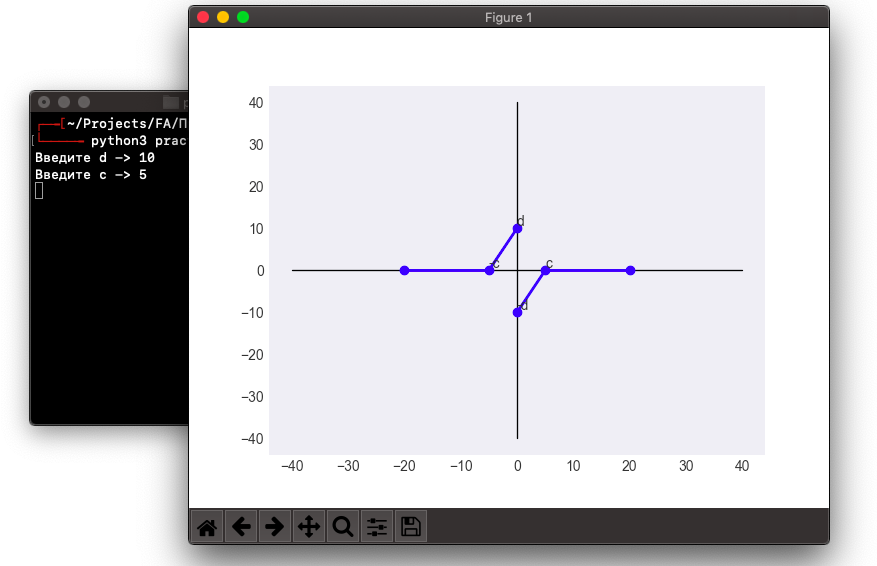
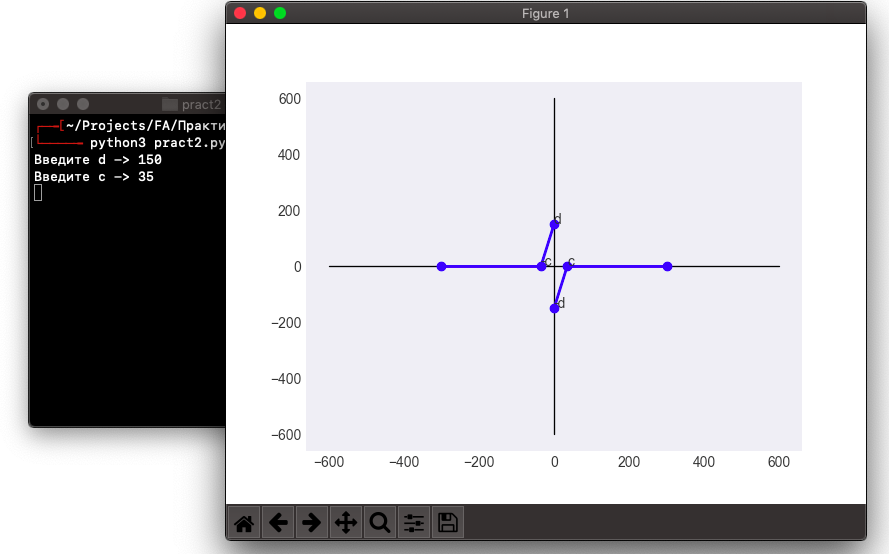


Рис 1 – Стандартный ввод данных

## 

Рис 2 – Реализация фильтрации входных данных