Университет ИТМО

Факультет ФПИ и КТ

### Отчёт

### по лабораторной работе 6

# «ЧИСЛЕННОЕ РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ»

Студент:

Гр.P32111

Преподаватель:  
Малышева Татьяна Алексеевна

**Цель работы**:

Найти приближение решения задачи Коши программой.

**Ключевые код программы:**

Исходные функции:

文本

描述已自动生成

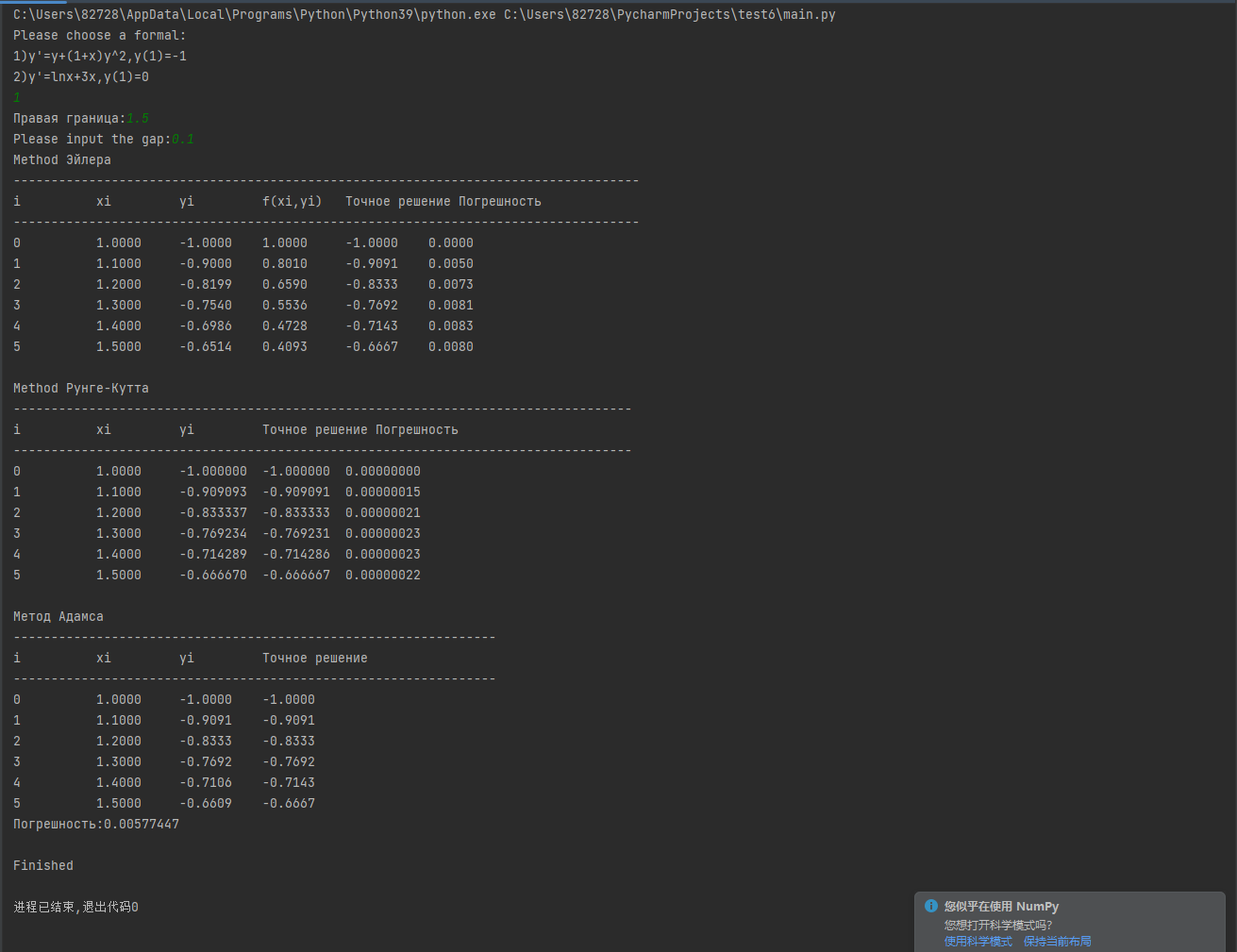
Реализация методов:

def Euler(choose: str, result: [], gap):  
 print("Method Эйлера")  
 deviation = []  
 if choose == "1":  
 y = -1  
 y1= -1  
 i = 0  
 left = 1  
 x0 = left  
 x1 = left  
 error = 0  
 t = gap / 2  
 print("-----------------------------------------------------------------------------------")  
 print("%-10s %-10s %-10s %-10s %-10s %-10s" % ("i", "xi", "yi", "f(xi,yi)", "Точное решение", "Погрешность"))  
 print("-----------------------------------------------------------------------------------")  
 while round(x0, 4) <= right:  
 result.append([x0, y])  
 deviation.append(abs(y - function1(x0)))  
 print("%-10.0f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f" % (i, x0, y, get\_f1(x0, y), function1(x0), error))  
 y = y + gap \* get\_f1(x0, y)  
 i = i + 1  
  
 y1 = y1 + t \* get\_f1(x1, y1)  
 x1 = x1 + t  
 y1 = y1 + t \* get\_f1(x1, y1)  
 x1 = x1 + t  
  
 x0 = x0 + gap  
 error = abs(y1 - y) / (2 \*\* 1 - 1)  
 return result  
 elif choose == "2":  
 y = 0  
 y1 = -1  
 i = 0  
 left = 1  
 x0 = left  
 x1 = left  
 error = 0  
 t = gap / 2  
 print("%-10s %-10s %-10s %-10s %-10s %-10s" % ("i", "xi", "yi", "f(xi,yi)", "Точное решение", "Погрешность"))  
 while round(x0, 4) <= right:  
 result.append([x0, y])  
 deviation.append(abs(y - function2(x0)))  
 print("%-10.0f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f" % (i, x0, y, get\_f2(x0, y), function2(x0), error))  
 y = y + gap \* get\_f2(x0, y)  
 i = i + 1  
  
 y1 = y1 + t \* get\_f2(x1, y1)  
 x1 = x1 + t  
 y1 = y1 + t \* get\_f2(x1, y1)  
 x1 = x1 + t  
  
 x0 = x0 + gap  
 error = abs(y1 - y) / (2 \*\* 1 - 1)  
 return result  
  
  
def Runge\_Kutta(choose: str, result: [], gap):  
 print("Method Рунге-Кутта")  
 if choose == "1":  
 y = -1  
 i = 0  
 left = 1  
 x0 = left  
 x1 = left  
 y1 = -1  
 t = gap / 2  
 error=0  
 print("----------------------------------------------------------------------------------")  
 print("%-10s %-10s %-10s %-10s %-10s" % ("i", "xi", "yi", "Точное решение", "Погрешность"))  
 print("----------------------------------------------------------------------------------")  
 result = []  
 deviation = []  
 while round(x0, 4) <= right:  
 deviation.append(abs(function1(x0) - y))  
 result.append([x0, y])  
 print("%-10.0f %-10.4f %-10.6f %-10.6f %-10.8f " % (i, x0, y, function1(x0),error))  
 k1 = get\_f1(x0, y) \* gap  
 k2 = get\_f1(x0 + gap / 2, y + k1 / 2) \* gap  
 k3 = get\_f1(x0 + gap / 2, y + k2 / 2) \* gap  
 k4 = get\_f1(x0 + gap, y + k3) \* gap  
 y = y + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6  
  
 k5 = get\_f1(x1, y1) \* t  
 k6 = get\_f1(x1 + t / 2, y1 + k5 / 2) \* t  
 k7 = get\_f1(x1 + t / 2, y1 + k6 / 2) \* t  
 k8 = get\_f1(x1 + t, y1 + k7) \* t  
 y1 = y1 + (k5 + 2 \* k6 + 2 \* k7 + k8) / 6  
  
 x1 = x1 + t  
  
 k5 = get\_f1(x1, y1) \* t  
 k6 = get\_f1(x1 + t / 2, y1 + k5 / 2) \* t  
 k7 = get\_f1(x1 + t / 2, y1 + k6 / 2) \* t  
 k8 = get\_f1(x1 + t, y1 + k7) \* t  
 y1 = y1 + (k5 + 2 \* k6 + 2 \* k7 + k8) / 6  
  
 x1 = x1 + t  
  
 i = i + 1  
 x0 = x0 + gap  
 error = abs(y1 - y) / (2 \*\* 4 - 1)  
 final = {'result': result, 'gap': gap}  
 return final  
 else:  
 y = 0  
 i = 0  
 left = 1  
 x0 = left  
 print("%-10s %-10s %-10s %-10s " % ("i", "xi", "yi", "Точное решение"))  
 result = []  
 deviation = []  
 while round(x0, 4) <= right:  
 deviation.append(abs(function2(x0) - y))  
 result.append([x0, y])  
 print("%-10.0f %-10.4f %-10.6f %-10.6f " % (i, x0, y, function1(x0)))  
 k1 = get\_f2(x0, y) \* gap  
 k2 = get\_f2(x0 + gap / 2, y + k1 / 2) \* gap  
 k3 = get\_f2(x0 + gap / 2, y + k2 / 2) \* gap  
 k4 = get\_f2(x0 + gap, y + k3) \* gap  
 y = y + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6  
 i = i + 1  
 x0 = x0 + gap  
 print("Погрешность:%.8f" % max(deviation))  
 final = {'result': result, 'gap': gap}  
 return final  
  
  
class Dot:  
 x: float  
 y: float  
  
 def \_\_init\_\_(self, x: float, y: float):  
 self.x = x  
 self.y = y  
  
 def to\_string(self):  
 print("(%.4f,%.4f)" % (self.x, self.y))  
  
 def get\_x(self):  
 return self.x  
  
 def get\_y(self):  
 return self.y  
  
  
def Andamc(choose, Ry, gap):  
 if choose == "1":  
 i = 0  
 y0\_3 = []  
 x0\_3 = [1, 1 + gap, 1 + 2 \* gap, 1 + 3 \* gap]  
 y0\_3.append(-1)  
 left = 1  
 result = []  
 times = (right - left) / gap  
 dievation = []  
 if times < 4:  
 print("Метод Адмаса не подходит")  
 return []  
 while i < 3:  
 y0\_3.append(Ry[i + 1][1])  
 i = i + 1  
 i = 0  
 print("Метод Адамса")  
 print("----------------------------------------------------------------")  
 print("%-10s %-10s %-10s %-10s" % ("i", "xi", "yi", "Точное решение"))  
 print("----------------------------------------------------------------")  
 while i < 4:  
 print("%-10.0f %-10.4f %-10.4f %-10.4f" % (i, x0\_3[i], y0\_3[i], function1(x0\_3[i])))  
 dievation.append(abs(function1(x0\_3[i]) - y0\_3[i]))  
 result.append([x0\_3[i], y0\_3[i]])  
 i = i + 1  
 ## i = 4  
 f0 = get\_f1(x0\_3[0], y0\_3[0])  
 f1 = get\_f1(x0\_3[1], y0\_3[1])  
 f2 = get\_f1(x0\_3[2], y0\_3[2])  
 f3 = get\_f1(x0\_3[3], y0\_3[3])  
 times = times - 3  
 y = y0\_3[3]  
 x = x0\_3[3]  
 while times > 0:  
 det\_1\_f = f3 - f2  
 det\_2\_f = f3 - 2 \* f2 + f1  
 det\_3\_f = f3 - 3 \* f2 + 3 \* f1 - f0  
 y = y + gap \* get\_f1(x,y) + gap \*\* 2 \* det\_1\_f / 2 + 5 \* gap \*\* 3 \* det\_2\_f / 12 + 3 \* gap \*\* 4 \* det\_3\_f / 8  
 x = x + gap  
 dievation.append(abs(function1(x) - y))  
 result.append([x, y])  
 print("%-10.0f %-10.4f %-10.4f %-10.4f" % (i, x, y, function1(x)))  
 f0 = f1  
 f1 = f2  
 f2 = f3  
 f3 = get\_f1(x, y)  
 times = times - 1  
 i = i + 1  
 print("Погрешность:%.8f" % max(dievation))  
 elif choose == "2":  
 i = 0  
 y0\_3 = []  
 x0\_3 = [1, 1 + gap, 1 + 2 \* gap, 1 + 3 \* gap]  
 y0\_3.append(0)  
 left = 1  
 result = []  
 times = (right - left) / gap  
 dievation = []  
 if times < 4:  
 print("Метод Адмаса не подходит")  
 return []  
 while i < 3:  
 y0\_3.append(Ry[i + 1][1])  
 i = i + 1  
 i = 0  
 print("Метод Адамса")  
 print("%-10s %-10s %-10s %-10s" % ("i", "xi", "yi", "Точное решение"))  
 while i < 4:  
 print("%-10.0f %-10.4f %-10.4f %-10.4f" % (i, x0\_3[i], y0\_3[i], function2(x0\_3[i])))  
 dievation.append(abs(function2(x0\_3[i]) - y0\_3[i]))  
 result.append([x0\_3[i], y0\_3[i]])  
 i = i + 1  
 ## i = 4  
 f0 = get\_f2(x0\_3[0], y0\_3[0])  
 f1 = get\_f2(x0\_3[1], y0\_3[1])  
 f2 = get\_f2(x0\_3[2], y0\_3[2])  
 f3 = get\_f2(x0\_3[3], y0\_3[3])  
 times = times - 3  
 y = y0\_3[3]  
 x = x0\_3[3]  
 while times > 0:  
 det\_1\_f = f3 - f2  
 det\_2\_f = f3 - 2 \* f2 + f1  
 det\_3\_f = f3 - 3 \* f2 + 3 \* f1 - f0  
 y = y + gap \* get\_f2(x,  
 y) + gap \*\* 2 \* det\_1\_f / 2 + 5 \* gap \*\* 3 \* det\_2\_f / 12 + 3 \* gap \*\* 4 \* det\_3\_f / 8  
 x = x + gap  
 dievation.append(abs(function2(x) - y))  
 result.append([x, y])  
 print("%-10.0f %-10.4f %-10.4f %-10.4f" % (i, x, y, function2(x)))  
 f0 = f1  
 f1 = f2  
 f2 = f3  
 f3 = get\_f2(x, y)  
 times = times - 1  
 i = i + 1  
 print("Погрешность:%.8f" % max(dievation))  
 return result

**Описание кода:**

Служат вводом выбор функции, интервал, правая граница. Каждая реализация метода возвращает list, в котором сохраняет результат точек. А метод Рунге ещё возвращает шаг для Адамса.

**Примеры:**



图表, 折线图

描述已自动生成

### Вывод:

Необходимо обратить внимание на разные способы расчета погрешностей в разных методах.

**Полный код:**

import math  
import numpy as np  
import matplotlib.pyplot as plt  
import math  
  
h = 0  
right = 0  
precision = 0.01  
  
  
## first choice  
  
  
def get\_f1(x, y):  
 return y + (1 + x) \* y \*\* 2  
  
  
## -1/x  
def function1(x):  
 return -1 / x  
  
  
def get\_f2(x, y):  
 return math.log(x) + 3 \* x  
  
  
##xlnx + 3/2x^2-x-1/2  
def function2(x):  
 a = np.log(x)  
 return x \* a + 3 \* x \*\* 2 / 2 - x - 0.5  
  
  
def Euler(choose: str, result: [], gap):  
 print("Method Эйлера")  
 deviation = []  
 if choose == "1":  
 y = -1  
 y1= -1  
 i = 0  
 left = 1  
 x0 = left  
 x1 = left  
 error = 0  
 t = gap / 2  
 print("-----------------------------------------------------------------------------------")  
 print("%-10s %-10s %-10s %-10s %-10s %-10s" % ("i", "xi", "yi", "f(xi,yi)", "Точное решение", "Погрешность"))  
 print("-----------------------------------------------------------------------------------")  
 while round(x0, 4) <= right:  
 result.append([x0, y])  
 deviation.append(abs(y - function1(x0)))  
 print("%-10.0f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f" % (i, x0, y, get\_f1(x0, y), function1(x0), error))  
 y = y + gap \* get\_f1(x0, y)  
 i = i + 1  
  
 y1 = y1 + t \* get\_f1(x1, y1)  
 x1 = x1 + t  
 y1 = y1 + t \* get\_f1(x1, y1)  
 x1 = x1 + t  
  
 x0 = x0 + gap  
 error = abs(y1 - y) / (2 \*\* 1 - 1)  
 return result  
 elif choose == "2":  
 y = 0  
 y1 = -1  
 i = 0  
 left = 1  
 x0 = left  
 x1 = left  
 error = 0  
 t = gap / 2  
 print("%-10s %-10s %-10s %-10s %-10s %-10s" % ("i", "xi", "yi", "f(xi,yi)", "Точное решение", "Погрешность"))  
 while round(x0, 4) <= right:  
 result.append([x0, y])  
 deviation.append(abs(y - function2(x0)))  
 print("%-10.0f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f" % (i, x0, y, get\_f2(x0, y), function2(x0), error))  
 y = y + gap \* get\_f2(x0, y)  
 i = i + 1  
  
 y1 = y1 + t \* get\_f2(x1, y1)  
 x1 = x1 + t  
 y1 = y1 + t \* get\_f2(x1, y1)  
 x1 = x1 + t  
  
 x0 = x0 + gap  
 error = abs(y1 - y) / (2 \*\* 1 - 1)  
 return result  
  
  
def Runge\_Kutta(choose: str, result: [], gap):  
 print("Method Рунге-Кутта")  
 if choose == "1":  
 y = -1  
 i = 0  
 left = 1  
 x0 = left  
 x1 = left  
 y1 = -1  
 t = gap / 2  
 error=0  
 print("----------------------------------------------------------------------------------")  
 print("%-10s %-10s %-10s %-10s %-10s" % ("i", "xi", "yi", "Точное решение", "Погрешность"))  
 print("----------------------------------------------------------------------------------")  
 result = []  
 deviation = []  
 while round(x0, 4) <= right:  
 deviation.append(abs(function1(x0) - y))  
 result.append([x0, y])  
 print("%-10.0f %-10.4f %-10.6f %-10.6f %-10.8f " % (i, x0, y, function1(x0),error))  
 k1 = get\_f1(x0, y) \* gap  
 k2 = get\_f1(x0 + gap / 2, y + k1 / 2) \* gap  
 k3 = get\_f1(x0 + gap / 2, y + k2 / 2) \* gap  
 k4 = get\_f1(x0 + gap, y + k3) \* gap  
 y = y + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6  
  
 k5 = get\_f1(x1, y1) \* t  
 k6 = get\_f1(x1 + t / 2, y1 + k5 / 2) \* t  
 k7 = get\_f1(x1 + t / 2, y1 + k6 / 2) \* t  
 k8 = get\_f1(x1 + t, y1 + k7) \* t  
 y1 = y1 + (k5 + 2 \* k6 + 2 \* k7 + k8) / 6  
  
 x1 = x1 + t  
  
 k5 = get\_f1(x1, y1) \* t  
 k6 = get\_f1(x1 + t / 2, y1 + k5 / 2) \* t  
 k7 = get\_f1(x1 + t / 2, y1 + k6 / 2) \* t  
 k8 = get\_f1(x1 + t, y1 + k7) \* t  
 y1 = y1 + (k5 + 2 \* k6 + 2 \* k7 + k8) / 6  
  
 x1 = x1 + t  
  
 i = i + 1  
 x0 = x0 + gap  
 error = abs(y1 - y) / (2 \*\* 4 - 1)  
 final = {'result': result, 'gap': gap}  
 return final  
 else:  
 y = 0  
 i = 0  
 left = 1  
 x0 = left  
 print("%-10s %-10s %-10s %-10s " % ("i", "xi", "yi", "Точное решение"))  
 result = []  
 deviation = []  
 while round(x0, 4) <= right:  
 deviation.append(abs(function2(x0) - y))  
 result.append([x0, y])  
 print("%-10.0f %-10.4f %-10.6f %-10.6f " % (i, x0, y, function1(x0)))  
 k1 = get\_f2(x0, y) \* gap  
 k2 = get\_f2(x0 + gap / 2, y + k1 / 2) \* gap  
 k3 = get\_f2(x0 + gap / 2, y + k2 / 2) \* gap  
 k4 = get\_f2(x0 + gap, y + k3) \* gap  
 y = y + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6  
 i = i + 1  
 x0 = x0 + gap  
 print("Погрешность:%.8f" % max(deviation))  
 final = {'result': result, 'gap': gap}  
 return final  
  
  
class Dot:  
 x: float  
 y: float  
  
 def \_\_init\_\_(self, x: float, y: float):  
 self.x = x  
 self.y = y  
  
 def to\_string(self):  
 print("(%.4f,%.4f)" % (self.x, self.y))  
  
 def get\_x(self):  
 return self.x  
  
 def get\_y(self):  
 return self.y  
  
  
def Andamc(choose, Ry, gap):  
 if choose == "1":  
 i = 0  
 y0\_3 = []  
 x0\_3 = [1, 1 + gap, 1 + 2 \* gap, 1 + 3 \* gap]  
 y0\_3.append(-1)  
 left = 1  
 result = []  
 times = (right - left) / gap  
 dievation = []  
 if times < 4:  
 print("Метод Адмаса не подходит")  
 return []  
 while i < 3:  
 y0\_3.append(Ry[i + 1][1])  
 i = i + 1  
 i = 0  
 print("Метод Адамса")  
 print("----------------------------------------------------------------")  
 print("%-10s %-10s %-10s %-10s" % ("i", "xi", "yi", "Точное решение"))  
 print("----------------------------------------------------------------")  
 while i < 4:  
 print("%-10.0f %-10.4f %-10.4f %-10.4f" % (i, x0\_3[i], y0\_3[i], function1(x0\_3[i])))  
 dievation.append(abs(function1(x0\_3[i]) - y0\_3[i]))  
 result.append([x0\_3[i], y0\_3[i]])  
 i = i + 1  
 ## i = 4  
 f0 = get\_f1(x0\_3[0], y0\_3[0])  
 f1 = get\_f1(x0\_3[1], y0\_3[1])  
 f2 = get\_f1(x0\_3[2], y0\_3[2])  
 f3 = get\_f1(x0\_3[3], y0\_3[3])  
 times = times - 3  
 y = y0\_3[3]  
 x = x0\_3[3]  
 while times > 0:  
 det\_1\_f = f3 - f2  
 det\_2\_f = f3 - 2 \* f2 + f1  
 det\_3\_f = f3 - 3 \* f2 + 3 \* f1 - f0  
 y = y + gap \* get\_f1(x,y) + gap \*\* 2 \* det\_1\_f / 2 + 5 \* gap \*\* 3 \* det\_2\_f / 12 + 3 \* gap \*\* 4 \* det\_3\_f / 8  
 x = x + gap  
 dievation.append(abs(function1(x) - y))  
 result.append([x, y])  
 print("%-10.0f %-10.4f %-10.4f %-10.4f" % (i, x, y, function1(x)))  
 f0 = f1  
 f1 = f2  
 f2 = f3  
 f3 = get\_f1(x, y)  
 times = times - 1  
 i = i + 1  
 print("Погрешность:%.8f" % max(dievation))  
 elif choose == "2":  
 i = 0  
 y0\_3 = []  
 x0\_3 = [1, 1 + gap, 1 + 2 \* gap, 1 + 3 \* gap]  
 y0\_3.append(0)  
 left = 1  
 result = []  
 times = (right - left) / gap  
 dievation = []  
 if times < 4:  
 print("Метод Адмаса не подходит")  
 return []  
 while i < 3:  
 y0\_3.append(Ry[i + 1][1])  
 i = i + 1  
 i = 0  
 print("Метод Адамса")  
 print("%-10s %-10s %-10s %-10s" % ("i", "xi", "yi", "Точное решение"))  
 while i < 4:  
 print("%-10.0f %-10.4f %-10.4f %-10.4f" % (i, x0\_3[i], y0\_3[i], function2(x0\_3[i])))  
 dievation.append(abs(function2(x0\_3[i]) - y0\_3[i]))  
 result.append([x0\_3[i], y0\_3[i]])  
 i = i + 1  
 ## i = 4  
 f0 = get\_f2(x0\_3[0], y0\_3[0])  
 f1 = get\_f2(x0\_3[1], y0\_3[1])  
 f2 = get\_f2(x0\_3[2], y0\_3[2])  
 f3 = get\_f2(x0\_3[3], y0\_3[3])  
 times = times - 3  
 y = y0\_3[3]  
 x = x0\_3[3]  
 while times > 0:  
 det\_1\_f = f3 - f2  
 det\_2\_f = f3 - 2 \* f2 + f1  
 det\_3\_f = f3 - 3 \* f2 + 3 \* f1 - f0  
 y = y + gap \* get\_f2(x,  
 y) + gap \*\* 2 \* det\_1\_f / 2 + 5 \* gap \*\* 3 \* det\_2\_f / 12 + 3 \* gap \*\* 4 \* det\_3\_f / 8  
 x = x + gap  
 dievation.append(abs(function2(x) - y))  
 result.append([x, y])  
 print("%-10.0f %-10.4f %-10.4f %-10.4f" % (i, x, y, function2(x)))  
 f0 = f1  
 f1 = f2  
 f2 = f3  
 f3 = get\_f2(x, y)  
 times = times - 1  
 i = i + 1  
 print("Погрешность:%.8f" % max(dievation))  
 return result  
  
  
print("Please choose a formal:")  
print("1)y'=y+(1+x)y^2,y(1)=-1")  
print("2)y'=lnx+3x,y(1)=0")  
func = input()  
inter = input("Правая граница:")  
right = float(inter)  
h\_string = input("Please input the gap:")  
h = float(h\_string)  
##Эйлера  
result\_O = []  
result\_O = Euler(func, result\_O, h)  
count = 0  
while count < len(result\_O) - 1:  
 if count == 0:  
 plt.plot([result\_O[count][0], result\_O[count + 1][0]], [result\_O[count][1], result\_O[count + 1][1]],  
 color="red",  
 label="Эйлера")  
 else:  
 plt.plot([result\_O[count][0], result\_O[count + 1][0]], [result\_O[count][1], result\_O[count + 1][1]],  
 color="red")  
 count = count + 1  
print("")  
##Рунге-Кутта  
result\_R = []  
result\_R\_and\_gap = Runge\_Kutta(func, result\_R, h)  
result\_R = result\_R\_and\_gap['result']  
Andamc\_gap = result\_R\_and\_gap['gap']  
count = 0  
while count < len(result\_R) - 1:  
 if count == 0:  
 plt.plot([result\_R[count][0], result\_R[count + 1][0]], [result\_R[count][1], result\_R[count + 1][1]],  
 color="blue",  
 label="Рунге-Кутта")  
 else:  
 plt.plot([result\_R[count][0], result\_R[count + 1][0]], [result\_R[count][1], result\_R[count + 1][1]],  
 color="blue")  
 count = count + 1  
plt.legend()  
print("")  
##Адамса  
result\_A = []  
result\_A = Andamc(func, result\_R, Andamc\_gap)  
if len(result\_A) != 0:  
 count = 0  
 while count < len(result\_A) - 1:  
 if count == 0:  
 plt.plot([result\_A[count][0], result\_A[count + 1][0]], [result\_A[count][1], result\_A[count + 1][1]],  
 color="orange",  
 label="Адамса")  
 else:  
 plt.plot([result\_A[count][0], result\_A[count + 1][0]], [result\_A[count][1], result\_A[count + 1][1]],  
 color="orange")  
 count = count + 1  
 plt.legend()  
print("")  
##Точное  
left = 1  
x\_range = np.arange(left, right, 0.01)  
if func == "1":  
 y\_range = function1(x\_range)  
 plt.plot(x\_range, y\_range, color="green", label="Точное решение")  
elif func == "2":  
 y\_range = function2(x\_range)  
 plt.plot(x\_range, y\_range, color="green", label="Точное решение")  
plt.legend()  
  
plt.show()  
  
print("Finished")