ИУ5-32Б,

Колесников Никита

**Отчет по лабораторным работам 1-6**

**Цель:** Создание системы оценки рисков и расчета надежности для IT-проектов

Разобьем задачу на подзадачи удовлетворяющие условия для каждой из 6 лабораторных работ.

1 Лаба

Реализация выбора значимых критериев для многокритериального принятия решения (выбора критичных компонентов ИТ-проекта).

*По заданному списку критериев собрать с пользователя:*

1. *текущая оценка значимости критерия по 5-бальной шкале*
2. *желаемая оценка значимости критерия по 5-бальной шкале*
3. *вес критерия (сумма всех весов = 100%)*
4. *уровень отсечения (принцип Парето, инженерный принцип)*

*Провести расчет взвешенной оценки, ранжирование критериев и отсечение значимых по заданному уровню. Изучить проект на наличие ранее определенных значимых критериев*

2 Лаба

Реализация группового многокритериального выбора критичных компонентов ИТ-проекта.

*По заданному списку компонентов провести оценку по значимых критериям из Лабы 1 по 5-бальной шкале со стороны заказчика и со стороны исполнителя.Составить бинарную матрицу и распределить компоненты на группы по их важности.*

*3 Лаба*

*Провести проверку согласованности решения по принципу Парето (построить матрицы попарного сравнения и определить коэф. согласованности.)*

*Определить группы критичных компонентов у заказчика и исполнителя.*

*Определить обобщенный согласованный список критичных компонентов.*

*Реализовать определение и сохранение критичных компонентов в зависимости от стадии IT-продукта*

4 Лаба

*Реализация решения системы линейных алгебраических уравнений, описывающих функционирование критичных компонентов ИТ-проекта. Метод динамики средних.*

*По заданным графам состояний критичных компонентов полученных в Лабе 2 -3 собрать с пользователя:*

1. *Кол-во объектов в компоненте (если выбрано множество)*
2. *Интенсивности перехода между состояниями*
3. *Ущерб, приносящий одним объектом, находящимся в неработоспособном состоянии (по каждому не работоспособному состоянию)*

*Рассчитать по заданным системам уравнений:*

1. *Кол-во объектов, находящихся в каждом состоянии (если выбрано множество)*
2. *Вероятность нахождения объекта в каждом состоянии (если выбран один объект)*
3. *Общая надежность компонента*
4. *Совокупный ущерб компонента*

5 Лаба

Расчет рисков и надежности компонентов не подчиняющихся системе линейных алгебраических уравнений

6 Лаба

*Реализация решения системы линейных алгебраических уравнений, описывающих взаимодействие критичных компонентов ИТ-проекта.*

*По заданным графам состояний критичных компонентов полученных в Лабе 2 собрать с пользователя:*

1. *Выбор методики расчета (множество объектов в компоненте или один)*
2. *Кол-во объектов в компоненте (если выбрано множество)*
3. *Интенсивности перехода между состояниями*
4. *Уравнения управляющих воздействий (равенства между интенсивностями переходов)*
5. *Ущерб, приносящий одним объектом, находящимся в неработоспособном состоянии (по каждому не работоспособному состоянию)*

*Рассчитать по заданным системам уравнений:*

1. *Параметры системы*
2. *Общая надежность проекта*
3. *Совокупный ущерб проекта*

**Текст программы**

import sys  
#Проверк наличия файла  
import os.path  
#Решение системы уравнений  
from sympy import symbols, Eq, nsolve  
#Вывод изображения  
from matplotlib import pyplot as plt  
from matplotlib import image as mpimg  
#Математический корень  
from math import sqrt  
#Математический факториал  
from math import factorial  
#Математический експонент  
from math import exp  
#Класс компонент  
class component:  
 importance=0  
 group\_executor=0  
 group\_customer=0  
 group\_difference=0  
 group=0  
 def \_\_init\_\_(self,name, group):  
 self.name = name  
 self.group=group  
 def \_\_str\_\_(self):  
 return f"{self.name}, {self.importance}, {self.group\_executor}, {self.group\_customer},{self.group\_difference} {self.group}"  
#Класс критерий  
class criteria:  
 Y=0  
 K=0  
 def \_\_init\_\_(self, name,Y,K):  
 self.name = name  
 self.Y = Y  
 self.K = K  
 def \_\_str\_\_(self):  
 return f"{self.name} {self.Y} {self.K}"  
#Нахождение, создание или изменения файла для критичных критериев, а так же запись в массив крит критериев  
def file\_criteria():  
 if os.path.exists("crit\_criteria")==1:  
 change=str(input("Файл с критическими критериями уже создан, желаете его поменять?"))  
 if change=="1" or change=="Да" or change=="да" or change=="Yes" or change=="yes":  
 list\_criteria=create\_list\_criteria()  
 list\_crit\_criteria=set\_crit\_criteria(list\_criteria)  
 else:  
 print("Работаем с иходными данными")  
 f\_criteria = open('crit\_criteria', 'r',encoding='utf-8')  
 list\_crit\_criteria=[]  
 for crit in f\_criteria:  
 X=criteria(crit,0,0)  
 list\_crit\_criteria.append(X)  
 f\_criteria.close()  
 if len(list\_crit\_criteria)==0:  
 print("Файл с иходными данными пуст, его нужно заполнить")  
 list\_criteria=create\_list\_criteria()  
 list\_crit\_criteria=set\_crit\_criteria(list\_criteria)  
 else:  
 print("Создаем файл с критическимим критериями")  
 list\_criteria=create\_list\_criteria()  
 list\_crit\_criteria=set\_crit\_criteria(list\_criteria)  
 return list\_crit\_criteria  
#Нахождение или создание файла для критичных компонентов, а так же запись в массив крит компонентов  
def file\_components(name, list\_crit\_criteria):  
 if os.path.exists(name)==1:  
 change=str(input("Файл с критическими компонентами уже создан, желаете его поменять? "))  
 if change=="1" or change=="Да" or change=="да" or change=="Yes" or change=="yes":  
 list\_components=create\_list\_components()  
 list\_crit\_components=crit\_comp(list\_components, list\_crit\_criteria,name)  
 write\_crit\_comp(list\_crit\_components, name)  
 else:  
 print("Работаем с иходными данными\n")  
 f\_components = open(name, 'r',encoding='utf-8')  
 list\_crit\_components=[]  
 for comp in f\_components:  
 X=component(comp,1)  
 list\_crit\_components.append(X)  
 f\_components.close()  
 if len(list\_crit\_components)==0:  
 print("Файл с иходными данными пуст, его нужно заполнить\n")  
 list\_components=create\_list\_components()  
 list\_crit\_components=crit\_comp(list\_components, list\_crit\_criteria,name)  
 write\_crit\_comp(list\_crit\_components, name)  
 else:  
 print("Создаем файл с критическимим компонентами")  
 list\_components=create\_list\_components()  
 list\_crit\_components=crit\_comp(list\_components, list\_crit\_criteria,name)  
 write\_crit\_comp(list\_crit\_components, name)  
 return list\_crit\_components  
#Считывание из файла компонентов  
def create\_list\_components():  
 f\_component = open('components', 'r',encoding='utf-8')  
 list\_components=[]  
 for comp in f\_component:  
 X=component(comp,0)  
 list\_components.append(X)  
 f\_component.close()  
 return list\_components  
#Считывание из файла критериев  
def create\_list\_criteria():  
 sum\_Y=0  
 count\_criteria=0  
 list\_criteria=[]  
 f\_criteria = open('criteria', 'r',encoding='utf-8')  
 for crit in f\_criteria:  
 print("Рассмотрим критерий: ",crit)  
 count\_criteria+=1  
 K\_D=int(input("Задайте текущую оценку отказа критерия при отсутствии мероприятий по управлению рисками(от 0-5, где 0 - значительный ущерб; 5 - незначительный ущерб или ущерб отсутствует): "))  
 while K\_D>5 or K\_D<0:  
 K\_D=int(input("Значение K(D) задан неверно! Измените параметр. Он может принимать значение только 0-5. Попробуйте еще раз. "))  
 K\_S=int(input("Задайте желаемое значение оценки отказа критерия с учетом возможным мероприятий по управлению рисками(от 0-5, где 0 - значительный ущерб; 5 - незначительный ущерб или ущерб отсутствует): "))  
 while K\_D>5 or K\_D<0:  
 K\_S=int(input("Значение K(S) задан неверно! Измените параметр. Он может принимать значение только 0-5. Попробуйте еще раз. "))  
 Y=int(input("Задайте относительный уровень важности критерия(сумма по всем критериям должна равняться 100%): "))  
 sum\_Y+=Y  
 while sum\_Y>100:  
 sum\_Y-=Y  
 Y=int(input("Сумма значений Y > 100%! Измените параметр Y "))  
 sum\_Y+=Y  
 K=Y\*(K\_S-K\_D)  
 while K<0:  
 print("Значения K(D) и K(S) заданы неверно! Измените эти параметры")  
 K\_D=int(input())  
 K\_S=int(input())  
 K=Y\*(K\_S-K\_D)  
 X=criteria(crit,Y,K)  
 list\_criteria.append(X)  
 f\_criteria.close()  
 return list\_criteria  
#Отсечение неважных критериев  
def cut\_off(list, perc):  
 sum\_Y=0  
 numb=0  
 for i in range(len(list)):  
 if (sum\_Y+list[i].Y) <= perc:  
 sum\_Y+=list[i].Y  
 else:  
 numb=i  
 break  
 return list[:i]  
#Выбор процента отсечения и сортировка  
def set\_crit\_criteria(list\_criteria):  
 persent=int(input("Задайте процент отсечения значимых критериев: "))  
 while persent>100 or persent<0:  
 persent=int(input("Выбран некорректный процент! Он должен принимать значение 1-100. Попробуйте еще раз "))  
 list\_criteria.sort(key=lambda x: x.K, reverse=True)  
 list\_crit\_criteria=cut\_off(list\_criteria,persent)  
 write\_crit\_criteria(list\_crit\_criteria)  
 return list\_crit\_criteria  
#Запись крит. критериев в файл  
def write\_crit\_criteria(list\_crit\_criteria):  
 f\_crit\_criteria = open('crit\_criteria', 'w',encoding='utf-8')  
 for i in range(len(list\_crit\_criteria)):  
 f\_crit\_criteria.write(list\_crit\_criteria[i].name)  
 f\_crit\_criteria.close()  
#Оценка значимости компонентов  
def evo\_comp(list\_com, list\_cri):  
 evaluation\_components=[]  
 for i in range(len(list\_com)):  
 evaluation\_criteria=[list\_com[i].name[:-2]]  
 print("Оценка важности компонента: ",list\_com[i].name)  
 for j in range(len(list\_cri)):  
 print("Оцените значимость компонета по критерию(от 0-5, 0 - незначительный ущерб; 5 - максимальный ущерб): ",list\_cri[j].name)  
 x=int(input())  
 while x<0 or x>5:  
 x=int(input("Некоректный ввод! Попробуйте снова "))  
 evaluation\_criteria.append(x)  
 evaluation\_components.append(evaluation\_criteria)  
 return evaluation\_components  
#Создание бинарной таблицы оценки занчимости компонентов  
def bin\_evaluation\_component(list\_com, list\_cri):  
 list\_evaluation\_components=evo\_comp(list\_com,list\_cri)  
 bin\_evaluation\_component=[]  
 for i in range(len(list\_evaluation\_components)):  
 bin\_evaluation\_component1=[]  
 bin\_evaluation\_component1.append(list\_com[i].name[:-2])  
 max\_evaluation\_component1=max(list\_evaluation\_components[i][1:])  
 for j in range(len(list\_evaluation\_components)):  
 max\_evaluation\_component2=max(list\_evaluation\_components[j][1:])  
 if max\_evaluation\_component1>max\_evaluation\_component2 and all(list\_evaluation\_components[i][k]>=list\_evaluation\_components[j][k] for k in range(1,len(list\_evaluation\_components[i]))):  
 bin\_evaluation\_component1.append(1)  
 else:  
 bin\_evaluation\_component1.append(0)  
 bin\_evaluation\_component.append(bin\_evaluation\_component1)  
 #print(\*bin\_evaluation\_component)  
 return bin\_evaluation\_component  
#Определение суммы 0 для каждого компонента  
def sum\_bin\_eva(bin\_evaluation\_component):  
 list\_sum\_bin\_eva=[0]\*(len(bin\_evaluation\_component))  
 list\_sum\_bin\_eva.insert(0,"Сумма")  
 for i in range(len(bin\_evaluation\_component)):  
 for j in range(1,len(bin\_evaluation\_component)+1):  
 if bin\_evaluation\_component[i][j]==0:  
 list\_sum\_bin\_eva[j]+=1  
 #print(list\_sum\_bin\_eva)  
 return list\_sum\_bin\_eva  
#Выставление важности для каждого сомпанента  
def set\_importance(list\_components,list\_sum\_bin\_eva):  
 for i in range(len(list\_components)):  
 list\_components[i].importance=list\_sum\_bin\_eva[i]  
#Выставление № группы для исполнителя  
def set\_group\_executor(list\_components,set\_sum\_bin\_eva):  
 group=1  
 for i in range(len(set\_sum\_bin\_eva)):  
 for j in range(len(list\_components)):  
 if list\_components[j].importance==set\_sum\_bin\_eva[i]:  
 list\_components[j].group\_executor=group  
 group+=1  
#Выставление № группы для заказчика  
def set\_group\_customer(list\_components,set\_sum\_bin\_eva):  
 group=1  
 for i in range(len(set\_sum\_bin\_eva)):  
 for j in range(len(list\_components)):  
 if list\_components[j].importance==set\_sum\_bin\_eva[i]:  
 list\_components[j].group\_customer=group  
 group+=1  
#Изменение номеров групп в множестве  
def change\_set\_group(list\_components,set\_sum\_bin\_eva):  
 group=1  
 for i in range(len(set\_sum\_bin\_eva)):  
 if set\_sum\_bin\_eva[i]!=group:  
 k=set\_sum\_bin\_eva[i]-group  
 set\_sum\_bin\_eva[i]=group  
 group+=1  
#Вывод критичных компонентов для исполнителя  
def print\_crit\_comp\_executor(list\_components,set\_sum\_bin\_eva):  
 for i in range(len(set\_sum\_bin\_eva)):  
 print("\nГруппа №",set\_sum\_bin\_eva[i], ": ")  
 for j in range(len(list\_components)):  
 if (set\_sum\_bin\_eva[i])==(list\_components[j].group\_executor):  
 print(list\_components[j].name)  
#Вывод критичных компонентов для заказчика  
def print\_crit\_comp\_customer(list\_components,set\_sum\_bin\_eva):  
 for i in range(len(set\_sum\_bin\_eva)):  
 print("\nГруппа №",set\_sum\_bin\_eva[i], ": ")  
 for j in range(len(list\_components)):  
 if (set\_sum\_bin\_eva[i])==(list\_components[j].group\_customer):  
 print(list\_components[j].name)  
#Основная функция определения критических компонентов для исполнителя  
def crit\_comp\_executor(list\_components,list\_crit\_criteria):  
 list\_sum\_bin\_eva=sum\_bin\_eva(bin\_evaluation\_component(list\_components,list\_crit\_criteria))  
 list\_set\_sum\_bin\_eva=sorted(set(list\_sum\_bin\_eva[1:]))  
 set\_importance(list\_components,list\_sum\_bin\_eva[1:])  
 set\_group\_executor(list\_components,list\_set\_sum\_bin\_eva[::-1])  
 change\_set\_group(list\_components,list\_set\_sum\_bin\_eva)  
 print("Распределение компонентов по группам для исполнителя")  
 print\_crit\_comp\_executor(list\_components,list\_set\_sum\_bin\_eva)  
#Основная функция определения критических компонентов для заказчика  
def crit\_comp\_customer(list\_components,list\_crit\_criteria):  
 list\_sum\_bin\_eva=sum\_bin\_eva(bin\_evaluation\_component(list\_components,list\_crit\_criteria))  
 list\_set\_sum\_bin\_eva=sorted(set(list\_sum\_bin\_eva[1:]))  
 set\_importance(list\_components,list\_sum\_bin\_eva[1:])  
 set\_group\_customer(list\_components,list\_set\_sum\_bin\_eva[::-1])  
 change\_set\_group(list\_components,list\_set\_sum\_bin\_eva)  
 print("Распределение компонентов по группам для заказчика")  
 print\_crit\_comp\_customer(list\_components,list\_set\_sum\_bin\_eva)  
#Оценка согласованности выбора критичный комапнентов  
def agreement\_crit\_comp(list\_components,list\_crit\_criteria,name):  
 persent\_agree=80  
 max\_group=0  
 for i in range(len(list\_components)):  
 max\_group\_comp=max(list\_components[i].group\_executor,list\_components[i].group\_customer)  
 list\_components[i].group\_difference=max\_group\_comp-min(list\_components[i].group\_executor,list\_components[i].group\_customer)  
 max\_group=max(max\_group,max\_group\_comp)  
 group\_agree=(max\_group\*(100-persent\_agree))/100  
 f=0  
 for i in range(len(list\_components)):  
 if (list\_components[i].group\_executor==1 or list\_components[i].group\_customer==1) and list\_components[i].group\_difference>group\_agree :  
 print(list\_components[i].name)  
 f=1  
 elif list\_components[i].group\_executor==1 or list\_components[i].group\_customer==1:  
 list\_components[i].group=1  
 if f==1:  
 #print("Ошибка согласования для данных компонентов, советуем более детально обсудить данную тему между заказчиком и исполнителем!")  
 #crit\_comp(list\_components,list\_crit\_criteria,name)  
 sys.exit("Ошибка согласования для данных компонентов, советуем более детально обсудить данную тему между заказчиком и исполнителем! Дальнейшая работа невозможна!")  
#Согласование по Поретто  
# def agreement\_crit\_comp(list1, list2):  
# list\_agree=[]  
# for i in range(len(list1)):  
# list\_agree\_comp=[]  
# list\_agree\_comp.append(list1[i].name)  
# for j in range(1,len(list[i])):  
# if list1[i][j]==list2[i][j]:  
# list\_agree\_comp.append(1)  
# else:  
# list\_agree\_comp.append(0)  
# list\_agree.append(list\_agree\_comp)  
# return list\_agree  
  
# def check\_agree(list\_agree):  
# persent\_agree=80  
# cnt\_agree=(len(list\_agree)\*persent\_agree)/100  
# f=0  
# for i in range(len(list\_agree)):  
# sum=0  
# for j in range(1,len(list\_agree[i])):  
# sum+=list\_agree[i][j]  
# if sum<cnt\_agree:  
# print(list\_agree[i][j-1])  
# f=1  
# if f==1:  
# print("Ошибка согласования для данных компонентов, советуем более детально обсудить данную тему между заказчиком и исполнителем!")  
# return f  
  
#Запись критичных критериев в файл  
def write\_crit\_comp(list\_components, name):  
 list\_crit\_components=[]  
 f\_crit\_comp=open(name,"w", encoding='utf-8')  
 for comp in list\_components:  
 if comp.group==1:  
 f\_crit\_comp.write(comp.name)  
 list\_crit\_components.append(comp)  
 f\_crit\_comp.close()  
 return list\_crit\_components  
#Основная функция определения критичных компонентов  
def crit\_comp(list\_components,list\_crit\_criteria,name):  
 post=int(input("Вы являетесь исполнителем(1) или заказчиком(2) "))  
 while post!=1 and post!=2:  
 post=int(input("Введено неправильное значение, попробуйте еще раз "))  
 if post==1:  
 crit\_comp\_executor(list\_components,list\_crit\_criteria)  
 change=str(input("Спасибо за вашу оценку! Теперь нам нужно узнать мнение заказчика.\nГотовы начать? "))  
 if change=="1" or change=="Да" or change=="да" or change=="Yes" or change=="yes":  
 crit\_comp\_customer(list\_components,list\_crit\_criteria)  
 else:  
 sys.exit("Для оценки соглассованности необходимо мнение заказчика")  
 elif post==2:  
 crit\_comp\_customer(list\_components,list\_crit\_criteria)  
 change=str(input("Спасибо за вашу оценку! Теперь нам нужно узнать мнение исполнителя.\nГотовы начать? "))  
 if change=="1" or change=="Да" or change=="да" or change=="Yes" or change=="yes":  
 crit\_comp\_executor(list\_components,list\_crit\_criteria)  
 else:  
 sys.exit("Для оценки соглассованности необходимо мнение заказчика")  
 agreement\_crit\_comp(list\_components,list\_crit\_criteria,name)  
 list\_crit\_components=write\_crit\_comp(list\_components, name)  
 return list\_crit\_components  
#Вывод критичных компонентов  
def print\_crit\_components(list\_crit\_companies):  
 for i in range(len(list\_crit\_companies)):  
 print(list\_crit\_companies[i].name)  
#Вывод критичных критериев  
def print\_crit\_criteria(list\_crit\_criteria):  
 for i in range(len(list\_crit\_criteria)):  
 print(list\_crit\_criteria[i].name)  
#Вывод изображения  
def print\_image(name):  
 name\_model="Модель компонента: "+name  
 plt.title(name\_model)  
 name\_image="graphs/"+name+".png"  
 image = mpimg.imread(name\_image)  
 plt.axis('off')  
 plt.imshow(image)  
 plt.show()  
#Вывод информации о модели  
def print\_ifo\_model\_and\_cnt(name):  
 file\_name="probability/"+name+".txt"  
 f=open(file\_name,'r',encoding='utf-8')  
 cnt=0  
 for i in f:  
 print(i)  
 cnt+=1  
 f.close()  
 return cnt  
#Расчет вероятности безотказной работы  
def P(numerical\_solution,N,method):  
 if method==1:  
 sum\_m=0  
 for i in range(len(numerical\_solution)):  
 sum\_m+=numerical\_solution[i]  
 P=numerical\_solution[0]/sum\_m  
 elif method==2:  
 P=numerical\_solution[0]  
 return P  
#Расчет совокупного риска компонента  
def R(list\_C, numerical\_solution, N,method):  
 R=0  
 if method==1:  
 for i in range(1,len(list\_C)):  
 R+=list\_C[i]\*sqrt(numerical\_solution[i]\*(1-(numerical\_solution[i]/N)))  
 elif method==2:  
 for i in range(1,len(list\_C)):  
 R+=list\_C[i]\*numerical\_solution[i]  
 return R  
#Введение C для компонента  
def c\_conpanent(x):  
 c\_conpanent=[]  
 for i in range(x):  
 print("Введите выгоду, которую привносит единица работоспособного объекта(C0) или потери, которые привносит объект при нахождении в неработоспособном состоянии(C1,C2,...). Введите значение С", i)  
 c\_conpanent.append(float(input()))  
 return c\_conpanent  
#Введение l для компонента бюджетного обеспечения (budget support).  
def l\_component\_BS():  
 list\_l=[]  
 list\_l0=[0,0]  
 list\_l1=[0,0]  
 list\_l0[1]=float(input("Введите интенсивность перехода системы из 0-го состояния в 1-ое. "))  
 list\_l1[0]=float(input("Введите интенсивность перехода системы из 1-го состояния в 0-ое. "))  
 list\_l.append(list\_l0)  
 list\_l.append(list\_l1)  
 return list\_l  
#Введение l для компонента контрактная среда (contract environment).  
def l\_component\_CE():  
 list\_l=[]  
 list\_l0=[0,0,0]  
 list\_l1=[0,0,0]  
 list\_l2=[0,0,0]  
 list\_l0[1]=float(input("Введите интенсивность перехода системы из 0-го состояния в 1-ое. "))  
 list\_l0[2]=float(input("Введите интенсивность перехода системы из 0-го состояния в 2-ое. "))  
 list\_l1[0]=float(input("Введите интенсивность перехода системы из 1-го состояния в 0-ое. "))  
 list\_l1[2]=float(input("Введите интенсивность перехода системы из 1-го состояния в 2-ое. "))  
 list\_l2[1]=float(input("Введите интенсивность перехода системы из 2-го состояния в 1-ое. "))  
 list\_l.append(list\_l0)  
 list\_l.append(list\_l1)  
 list\_l.append(list\_l2)  
 return list\_l  
#Введение l для компонента факторы среды предприятия (environmental factors of the enterprise).  
def l\_component\_EFE():  
 list\_l=[]  
 list\_l0=[0,0,0]  
 list\_l1=[0,0,0]  
 list\_l2=[0,0,0]  
 list\_l0[1]=float(input("Введите интенсивность перехода системы из 0-го состояния в 1-ое. "))  
 list\_l0[2]=float(input("Введите интенсивность перехода системы из 0-го состояния в 2-ое. "))  
 list\_l1[2]=float(input("Введите интенсивность перехода системы из 1-го состояния в 2-ое. "))  
 list\_l2[0]=float(input("Введите интенсивность перехода системы из 2-го состояния в 0-ое." ))  
 list\_l.append(list\_l0)  
 list\_l.append(list\_l1)  
 list\_l.append(list\_l2)  
 return list\_l  
#Введение l для компонента содержание проекта (project content).  
def l\_component\_PC():  
 list\_l=[]  
 list\_l0=[0,0,0,0]  
 list\_l1=[0,0,0,0]  
 list\_l2=[0,0,0,0]  
 list\_l3=[0,0,0,0]  
 list\_l0[1]=float(input("Введите интенсивность перехода системы из 0-го состояния в 1-ое. "))  
 list\_l1[2]=float(input("Введите интенсивность перехода системы из 1-го состояния в 2-ое. "))  
 list\_l1[3]=float(input("Введите интенсивность перехода системы из 1-го состояния в 3-е. "))  
 list\_l2[0]=float(input("Введите интенсивность перехода системы из 2-го состояния в 0-ое. "))  
 list\_l3[0]=float(input("Введите интенсивность перехода системы из 3-го состояния в 0-ое. "))  
 list\_l.append(list\_l0)  
 list\_l.append(list\_l1)  
 list\_l.append(list\_l2)  
 list\_l.append(list\_l3)  
 return list\_l  
#Введение l для компонента человеко-ресурсное обеспечение (human resource support).  
def l\_component\_HRS():  
 list\_l=[]  
 list\_l0=[0,0,0,0,0]  
 list\_l1=[0,0,0,0,0]  
 list\_l2=[0,0,0,0,0]  
 list\_l3=[0,0,0,0,0]  
 list\_l4=[0,0,0,0,0]  
 list\_l0[1]=float(input("Введите интенсивность перехода системы из 0-го состояния в 1-ое. "))  
 list\_l0[2]=float(input("Введите интенсивность перехода системы из 0-го состояния в 2-ое. "))  
 list\_l0[3]=float(input("Введите интенсивность перехода системы из 0-го состояния в 3-ое. "))  
 list\_l0[4]=float(input("Введите интенсивность перехода системы из 0-го состояния в 4-ое. "))  
 list\_l1[0]=float(input("Введите интенсивность перехода системы из 1-го состояния в 0-ое. "))  
 list\_l1[3]=float(input("Введите интенсивность перехода системы из 1-го состояния в 3-е. "))  
 list\_l1[4]=float(input("Введите интенсивность перехода системы из 1-го состояния в 4-ое. "))  
 list\_l2[0]=float(input("Введите интенсивность перехода системы из 2-го состояния в 0-ое. "))  
 list\_l3[0]=float(input("Введите интенсивность перехода системы из 3-го состояния в 0-ое. "))  
 list\_l3[1]=float(input("Введите интенсивность перехода системы из 3-го состояния в 1-ое. "))  
 list\_l3[4]=float(input("Введите интенсивность перехода системы из 3-го состояния в 4-ое. "))  
 list\_l4[0]=float(input("Введите интенсивность перехода системы из 4-го состояния в 0-ое. "))  
 list\_l.append(list\_l0)  
 list\_l.append(list\_l1)  
 list\_l.append(list\_l2)  
 list\_l.append(list\_l3)  
 list\_l.append(list\_l4)  
 return list\_l  
#Введение l для компонента аппаратно-технический комплекс HTC(Hardware and technical complex)..  
def l\_component\_HTC():  
 list\_l=[]  
 list\_l0=[0,0,0,0,0]  
 list\_l1=[0,0,0,0,0]  
 list\_l2=[0,0,0,0,0]  
 list\_l3=[0,0,0,0,0]  
 list\_l4=[0,0,0,0,0]  
 list\_l0[1]=float(input("Введите интенсивность перехода системы из 0-го состояния в 1-ое. "))  
 list\_l0[2]=float(input("Введите интенсивность перехода системы из 0-го состояния в 2-ое. "))  
 list\_l1[0]=float(input("Введите интенсивность перехода системы из 1-го состояния в 0-ое. "))  
 list\_l1[2]=float(input("Введите интенсивность перехода системы из 1-го состояния в 2-ое. "))  
 list\_l2[3]=float(input("Введите интенсивность перехода системы из 2-го состояния в 3-е. "))  
 list\_l2[4]=float(input("Введите интенсивность перехода системы из 2-го состояния в 4-ое. "))  
 list\_l3[1]=float(input("Введите интенсивность перехода системы из 3-го состояния в 1-ое. "))  
 list\_l3[4]=float(input("Введите интенсивность перехода системы из 3-го состояния в 4-ое. "))  
 list\_l4[0]=float(input("Введите интенсивность перехода системы из 4-го состояния в 0-ое. "))  
 list\_l.append(list\_l0)  
 list\_l.append(list\_l1)  
 list\_l.append(list\_l2)  
 list\_l.append(list\_l3)  
 list\_l.append(list\_l4)  
 return list\_l  
#В зависимости от метода мы считаем либо x=m, либо x=p. Но система не менят вид, а меняеся только обозначение переменных  
  
#Решение системы уравнений для компонента бюджетного обеспечения (budget support).  
def solution\_component\_BS(N,list\_l,list\_c):  
 x0, x1= symbols('x0 x1')  
 equations = [  
 Eq(-list\_l[0][1]\*x0+list\_l[1][0]\*x1, 0),  
 Eq(-list\_l[1][0]\*x1+list\_l[0][1]\*x0, 0),  
 Eq(x0+x1, N)  
 ]  
 initial\_guess = [list\_c[0], list\_c[0]]  
 numerical\_solution = nsolve(equations, (x0, x1), initial\_guess)  
 return numerical\_solution  
#Решение системы уравнений для компонента контрактная среда (contract environment).  
def solution\_component\_CE(N,list\_l,list\_c):  
 x0, x1, x2= symbols('x0 x1 x2')  
 equations = [  
 Eq(-(list\_l[0][1]+list\_l[0][2])\*x0+list\_l[1][0]\*x1, 0),  
 Eq(-(list\_l[1][0]+list\_l[1][2])\*x1+list\_l[0][1]\*x0+list\_l[2][1]\*x2, 0),  
 Eq(-list\_l[2][1]\*x2+list\_l[1][2]\*x1+list\_l[0][2]\*x0, 0),  
 Eq(x0+x1+x2, N)  
 ]  
 initial\_guess = [list\_c[0], list\_c[0],list\_c[0]]  
 numerical\_solution = nsolve(equations, (x0, x1, x2), initial\_guess)  
 return numerical\_solution  
#Решение системы уравнений для компонента факторы среды предприятия (environmental factors of the enterprise).  
def solution\_component\_EFE(N,list\_l,list\_c):  
 x0, x1, x2= symbols('x0 x1 x2')  
 equations = [  
 Eq(-(list\_l[0][1]+list\_l[0][2])\*x0+list\_l[2][0]\*x2, 0),  
 Eq(-list\_l[1][2]\*x1+list\_l[0][1]\*x0, 0),  
 Eq(-list\_l[2][0]\*x2+list\_l[0][2]\*x0+list\_l[1][2]\*x1, 0),  
 Eq(x0+x1+x2, N)  
 ]  
 initial\_guess = [list\_c[0], list\_c[0],list\_c[0]]  
 numerical\_solution = nsolve(equations, (x0, x1, x2), initial\_guess)  
 return numerical\_solution  
#Решение системы уравнений для компонента содержание проекта (project content).  
def solution\_component\_PC(N,list\_l,list\_c):  
 x0, x1, x2, x3= symbols('x0 x1 x2 x3')  
 equations = [  
 Eq(-list\_l[0][1]\*x0+list\_l[2][0]\*x2+list\_l[3][0]\*x3, 0),  
 Eq(-(list\_l[1][2]+list\_l[1][3])\*x1+list\_l[0][1]\*x0, 0),  
 Eq(-list\_l[2][0]\*x2+list\_l[1][2]\*x1, 0),  
 Eq(-list\_l[3][0]\*x3+list\_l[1][3]\*x1, 0),  
 Eq(x0+x1+x2+x3, N)  
 ]  
 initial\_guess = [list\_c[0], list\_c[0], list\_c[0], list\_c[0]]  
 numerical\_solution = nsolve(equations, (x0, x1, x2, x3), initial\_guess)  
 return numerical\_solution  
#Решение системы уравнений для компонента аппаратно-технический комплекс HTC(Hardware and technical complex).  
def solution\_component\_HTC(N,list\_l,list\_c):  
 x0, x1, x2, x3, x4 = symbols('x0 x1 x2 x3 x4')  
 equations = [  
 Eq(-x0\*(list\_l[0][1]+list\_l[0][2])+x1\*list\_l[1][0]+x4\*list\_l[4][0], 0),  
 Eq(-x1\*(list\_l[1][0]+list\_l[1][2])+x0\*list\_l[0][1]+x3\*list\_l[3][1], 0),  
 Eq(-x2\*(list\_l[2][3]+list\_l[2][4])+x0\*list\_l[0][2]+x1\*list\_l[1][2], 0),  
 Eq(-x3\*(list\_l[3][1]+list\_l[3][4])+x2\*list\_l[2][3], 0),  
 Eq(-x4\*list\_l[4][0]+x2\*list\_l[2][4]+x3\*list\_l[3][4], 0),  
 Eq(x0+x1+x2+x3+x4, N)  
 ]  
 initial\_guess = [list\_c[0], list\_c[0], list\_c[0], list\_c[0], list\_c[0]]  
 numerical\_solution = nsolve(equations, (x0, x1, x2, x3, x4), initial\_guess)  
 return numerical\_solution  
#Решение системы уравнений для компонента человеко-ресурсное обеспечение (human resource support).  
def solution\_component\_HRS(N,list\_l,list\_c,):  
 x0, x1, x2, x3, x4 = symbols('x0 x1 x2 x3 x4')  
 equations = [  
 Eq(-(list\_l[0][1]+list\_l[0][2]+list\_l[0][3]+list\_l[0][4])\*x0+x1\*list\_l[1][0]+x2\*list\_l[2][0]+x3\*list\_l[3][0]+x4\*list\_l[4][0], 0),  
 Eq(-(list\_l[1][0]+list\_l[1][3]+list\_l[1][4])\*x1+x0\*list\_l[0][1]+x3\*list\_l[3][1], 0),  
 Eq(-x2\*list\_l[2][0]+x0\*list\_l[0][2], 0),  
 Eq(-(list\_l[3][0]+list\_l[3][1]+list\_l[3][4])\*x3+x0\*list\_l[0][3]+x1\*list\_l[1][3], 0),  
 Eq(-x4\*list\_l[4][0]+x0\*list\_l[0][4]+x1\*list\_l[1][4]+x3\*list\_l[3][4], 0),  
 Eq(x0+x1+x2+x3+x4, N)  
 ]  
 initial\_guess = [list\_c[0], list\_c[0], list\_c[0], list\_c[0], list\_c[0]]  
 numerical\_solution = nsolve(equations, (x0, x1, x2, x3, x4), initial\_guess)  
 return numerical\_solution  
#Проверка системы  
def check\_solution(method,numerical\_solution):  
 if method==1:  
 for i in range(len(numerical\_solution)):  
 if not((numerical\_solution[i]//1)==int(numerical\_solution[i]) and numerical\_solution[i]%1==0):  
 print("Система имеет дробные или отрицательные результаты. Измените параметры l")  
 return 1  
 else:  
 numerical\_solution[i]=int(numerical\_solution[i])  
 elif method==2:  
 for i in range(len(numerical\_solution)):  
 if numerical\_solution[i]<0:  
 print("Система имеет отрицательные результаты. Измените параметры l")  
 return 1  
 else:  
 numerical\_solution[i]=float(numerical\_solution[i])  
 return 0  
#Полная реализация определения P и R для компонента аппаратно-технический комплекс HTC(Hardware and technical complex).  
def component\_HTC(N,list\_c, method):  
 list\_l=l\_component\_HTC()  
 numerical\_solution=solution\_component\_HTC(N,list\_l,list\_c)  
 f=check\_solution(method,numerical\_solution)  
 if f==1:  
 component\_HTC(N,list\_c,method)  
 elif f==0:  
 print("P=",P(numerical\_solution,N,method)\*100,'%')  
 print("R=",R(list\_c,numerical\_solution,N,method))  
#Полная реализация определения P и R для компонента бюджетного обеспечения (budget support).  
def component\_BS(N,list\_c, method):  
 list\_l=l\_component\_BS()  
 numerical\_solution=solution\_component\_BS(N,list\_l,list\_c)  
 f=check\_solution(method,numerical\_solution)  
 if f==1:  
 component\_BS(N,list\_c,method)  
 elif f==0:  
 print("P=",P(numerical\_solution,N,method)\*100,'%')  
 print("R=",R(list\_c,numerical\_solution,N,method))  
#Полная реализация определения P и R для компонента контрактная среда (contract environment).  
def component\_CE(N,list\_c, method):  
 list\_l=l\_component\_CE()  
 numerical\_solution=solution\_component\_CE(N,list\_l,list\_c)  
 f=check\_solution(method,numerical\_solution)  
 if f==1:  
 component\_CE(N,list\_c,method)  
 elif f==0:  
 print("P=",P(numerical\_solution,N,method)\*100,'%')  
 print("R=",R(list\_c,numerical\_solution,N,method))  
#Полная реализация определения P и R для компонента факторы среды предприятия (environmental factors of the enterprise).  
def component\_EFE(N,list\_c, method):  
 list\_l=l\_component\_EFE()  
 numerical\_solution=solution\_component\_EFE(N,list\_l,list\_c)  
 f=check\_solution(method,numerical\_solution)  
 if f==1:  
 component\_EFE(N,list\_c,method)  
 elif f==0:  
 print("P=",P(numerical\_solution,N,method)\*100,'%')  
 print("R=",R(list\_c,numerical\_solution,N,method))  
#Полная реализация определения P и R компонента человеко-ресурсное обеспечение (human resource support).  
def component\_HRS(N,list\_c, method):  
 list\_l=l\_component\_HRS()  
 numerical\_solution=solution\_component\_HRS(N,list\_l,list\_c)  
 f=check\_solution(method,numerical\_solution)  
 if f==1:  
 component\_HRS(N,list\_c,method)  
 elif f==0:  
 print("P=",P(numerical\_solution,N,method)\*100,'%')  
 print("R=",R(list\_c,numerical\_solution,N,method))  
#Реализация определения P и R компонентов ПО  
def component\_PO():  
 l=float(input("Введите l "))  
 M=float(input("Введите M "))  
 m=int(input("Введите m "))  
 n=int(input("Введите n "))  
 C1=float(input("Введите C1 "))  
 C2=float(input("Введите C2 "))  
 p=l/M  
 p0=P0(m,n,p)  
 print("P=",p0)  
 ls=Ls(m,n,p)  
 R=ls\*C1+Lq(m,ls)\*C2  
 print("R=",R)  
def P0(m,n,p):  
 x1=((factorial(m))/(factorial(m-n)))\*(p\*\*n)  
 x2=0  
 for k in range(0,n+1):  
 x2+=((factorial(m))/(factorial(m-k)))\*(p\*\*k)  
 P0=x1/x2  
 # x1=0  
 # for k in range(0,n+1):  
 # x1+=((n\*p\*(1/n))\*\*k)/factorial(k)  
 # P0=1/x1  
 return P0  
def Ls(m,n,p):  
 x=0  
 for k in range(0,n):  
 x+=((factorial(m)/factorial(m-k-1))\*(p\*\*k))  
 Ls=m-(1/G(m,n,p))\*x  
 return Ls  
def G(m,n,p):  
 G=0  
 for k in range(0,n-1):  
 G+=((factorial(m)/factorial(m-k))\*(p\*\*k))  
 return G  
def Lq(m,Ls):  
 Lq=m-Ls  
 return Lq  
#Определение вызываемой функции по названию компонента  
def component\_name(comp,N,list\_c,method):  
 match comp:  
 case "Методическая и правовая часть проекта":  
 component\_EFE(N,list\_c,method)  
 case "Бюджетное обеспечение проекта":  
 component\_BS(N,list\_c,method)  
 case "Активы процессов организации проекта":  
 component\_EFE(N,list\_c,method)  
 case "Факторы среды предприятия":  
 component\_EFE(N,list\_c,method)  
 case "Контрактная среда проекта":  
 component\_CE(N,list\_c,method)  
 case "Содержание проекта":  
 component\_PC(N,list\_c,method)  
 case "Человеко-ресурсное обеспечение":  
 component\_HRS(N,list\_c,method)  
 case "Организационное обеспечение АС":  
 component\_EFE(N,list\_c,method)  
 case "Техническое обеспечение АС":  
 component\_HTC(N,list\_c,method)  
#Расчет надежности и риска  
def assessment(list\_crit\_components):  
 for i in range(len(list\_crit\_components)):  
 N = 0  
 comp = list\_crit\_components[i].name[:-1]  
 print\_image(comp)  
 cnt = print\_ifo\_model\_and\_cnt(comp)  
 if comp == "Специализированное программное обеспечение (СПО)" or comp=="Общее программное обеспечение (ОПО)" or comp=="Программное обеспечение для разработки и тестирования":  
 component\_PO()  
 else:  
 if comp == "Бюджетное обеспечение проекта":  
 N = 1  
 else:  
 print("Выберите метод рассчета: 1-метод динамики средних, 2-классический метод")  
 while True:  
 method = int(input())  
 if method == 1:  
 N = int(input("Введите N (Нормируещее условие) "))  
 break  
 elif method == 2:  
 N = 1  
 break  
 else:  
 print("Неверный ввод. Попробейте снова")  
 list\_c = c\_conpanent(cnt)  
 component\_name(comp, N, list\_c, method)  
#Основная функция  
def main():  
 #component\_PO()  
 list\_crit\_criteria = file\_criteria()  
 print("Желаете посмотреть критические критерии?")  
 change=str(input())  
 if change=="1" or change=="Да" or change=="да" or change=="Yes" or change=="yes":  
 print\_crit\_criteria(list\_crit\_criteria)  
  
 print("На какой фазе проекта вы хотите определить критичные компоненты(назовите номер)?")  
 phase = int(input())  
 name = "crit\_comp\_phase№" + str(phase)  
  
 list\_crit\_components = file\_components(name, list\_crit\_criteria)  
 print("Желаете посмотреть критичные компоненты?")  
 change=str(input())  
 if change=="1" or change=="Да" or change=="да" or change=="Yes" or change=="yes":  
 print\_crit\_components(list\_crit\_components)  
  
 print("Желаете рассчитать риск и надежность критичных компонентов?")  
 change = str(input())  
 if change == "1" or change == "Да" or change == "да" or change == "Yes" or change == "yes":  
 assessment(list\_crit\_components)  
 print("Всего хорошего ;)")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Анализ результатов

   