## **Принцип реализации системы**

Мы решили воспользоваться прямым выводом. Для этого достаточно было воспользоваться CLIPS с помощью библиотеки на python Pyknow. Prolog мы помнили хуже, чем python, поэтому выбор лег именно на этот язык программирования. В начале система задает пользователю вопросы, а затем обрабатывает ответы и выдает результат.

## **Механизм вывода**

Мы узнаем интересы нашего пользователя и заносим их в список фактов о нем. В Pyknow заданы правила с помощью которых факты из списка проверяются и доказываются, а впоследствии добавляются в память. Это повторяется до тех пор пока мы не найдем интересующие нас утверждения.

Правила в системе Pyknow

class **Film**(**KnowledgeEngine**):

result=[]

**@Rule**(OR(Fact(age='подросток(13-18)'),

Fact('необычные способности')))

def **unreal\_heroes**(self):

self.declare(Fact('нравится смотреть на вымышленных\несуществующих героев'))

**@Rule**(Fact(age='подросток(13-18)'))

def **unreal\_history**(self):

self.declare(Fact('нравятся вымышленные истории'))

**@Rule**(Fact(age='старше 40'))

def **world\_news**(self):

self.declare(Fact('информация про мировые события'))

**@Rule**(OR(Fact(grad='высшее образование'),Fact(grad='неоконченное высшее образование'),Fact(age='подросток(13-18)'),Fact(age='половозрелый(19-40)')))

def **found\_of\_history**(self):

self.declare(Fact('увлекаемся историей'))

**@Rule**(OR(Fact('нравится сакура'),Fact('нравятся самураи'),Fact('нравится кимоно')))

def **japan\_cul**(self):

self.declare(Fact('нравится японская культурая'))

**@Rule**(AND(Fact('нравится смотреть на вымышленных\несуществующих героев'),Fact('нравятся вымышленные истории')))

def **unreal\_screen**(self):

self.declare(Fact('несуществующий сюжет'))

**@Rule**(OR(AND(Fact(sex='женщина'),Fact(age='половозрелый(19-40)')),Fact(temp='флегматик'),Fact(temp='сагвинник'),Fact(temp='меланхолик')))

def **emotional**(self):

self.declare(Fact('хочется эмоционального'))

**@Rule**(OR(AND(Fact(sex='мужчина'),Fact(age='половозрелый(19-40)')),Fact(temp='холерик'),Fact('драки'),Fact('убийства')))

def **cruel**(self):

self.declare(Fact('нравится жестокость'))

**@Rule**(OR(Fact(age='половозрелый(19-40)'),Fact('нравится борьба двух сторон')))

def **tension**(self):

self.declare(Fact('нравится напряжение'))

**@Rule**(OR(Fact('нравятся самураи'),Fact('нравится оружие'),AND(Fact(sex='мужчина'),Fact(age='половозрелый(19-40)'),

Fact('нравится военная техника'),Fact('нравится военная культура'))))

def **military**(self):

self.declare(Fact('нравится военная тема'))

**@Rule**(AND(Fact('нравится напряжение'),Fact('стилистические фильмы')))

def **dark\_cadr**(self):

self.declare(Fact('нравятся "тёмные" кадры'))

**@Rule**(OR(Fact('мрачная атмосфера'),Fact('малое количество диалогов')))

def **style\_film**(self):

self.declare(Fact('стилистические фильмы'))

**@Rule**(OR(Fact('нравятся животные'),Fact('увлекаемся историей'),Fact('информация про мировые события')))

def **education**(self):

self.declare(Fact(film='документальные'))

**@Rule**(AND(Fact('несуществующий сюжет'),Fact('нравится японская культурая')))

def **anime**(self):

self.declare(Fact(film='аниме'))

**@Rule**(AND(Fact('несуществующий сюжет'),Fact('увлекаемся историей')))

def **fantastic**(self):

self.declare(Fact(film='фантастика'))

**@Rule**(AND(Fact('ребёнок (<12)'),Fact('элемент сказок'),Fact('нравятся абсурдные ситуации'),Fact('семья')))

def **cartoon**(self):

self.declare(Fact(film='мультфильмы'))

**@Rule**(AND(Fact('грустный фильм'),Fact('хочется эмоционального')))

def **drama**(self):

self.declare(Fact(film='драмма'))

**@Rule**(AND(Fact('семья'),Fact('хочется эмоционального'),OR(Fact('хочется посмеяться'),Fact('нравятся абсурдные ситуации'))))

def **comedy**(self):

self.declare(Fact(film='комедия'))

**@Rule**(AND(Fact('нравится напряжение'),Fact('хочется эмоционального'),Fact('нравится жестокость'),Fact('нравится военная тема')))

def **boevik**(self):

self.declare(Fact(film='военные\боевики'))

**@Rule**(AND(Fact('нравятся страшные истории'),Fact('хочется эмоционального')))

def **screamer**(self):

self.declare(Fact(film='ужасы'))

**@Rule**(AND(Fact('Пораскинуть мозгами'),Fact('нравятся "тёмные" кадры')))

def **nuar**(self):

self.declare(Fact(film='нуар'))

**@Rule**(AND(Fact('нравится напряжение'),Fact('стилистические фильмы')))

def **vestern**(self):

self.declare(Fact(film='вестерн'))

**@Rule**(AND(Fact('нравится заниматься спортом'),Fact('информация про мировые события')))

def **sport**(self):

self.declare(Fact(film='спорт'))

**@Rule**(Fact(film=MATCH.a))

def **print\_result**(self,a):

self.result.append(a)

**print**(f'Жанр- {a}')

def **factz**(self,l):

for x in l:

self.declare(x)

Механизм ввода:

for key in my\_questions.keys():

**print**(key)

print\_list(my\_questions[key])

user\_answer=**input**()

user\_answer\_list=[]

for l in my\_questions[key]:

if l.find(user\_answer+'.')>-1:

user\_answer\_list=l.split('.')

if user\_answer\_list[1]=='да':

user\_ans.append(key[:-1])

elif user\_answer\_list[1]!='нет' :

user\_ans.append(user\_answer\_list[1])

ex1 = Film()

ex1.reset()

facts\_my=[Fact(age=user\_ans[0]),

Fact(sex=user\_ans[1]),

Fact(grad = user\_ans[2]),

Fact(temp = user\_ans[3])]

for i in **range**(4,**len**(user\_ans)):

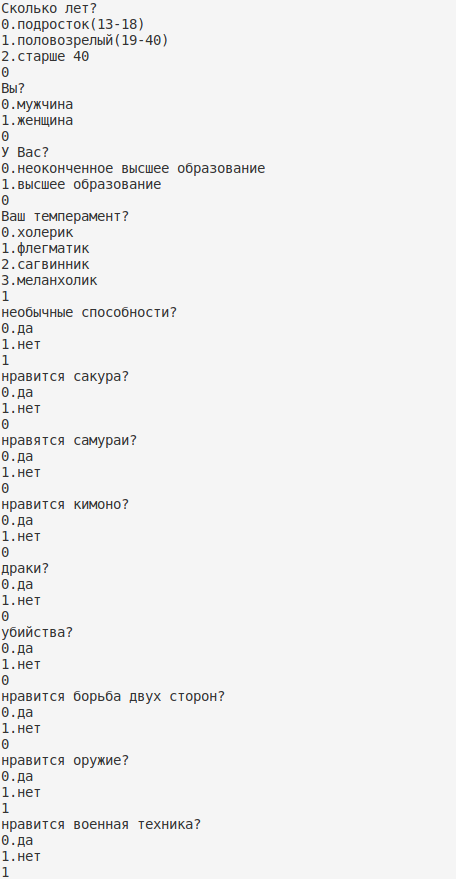
facts\_my.append(Fact(user\_ans[i]))

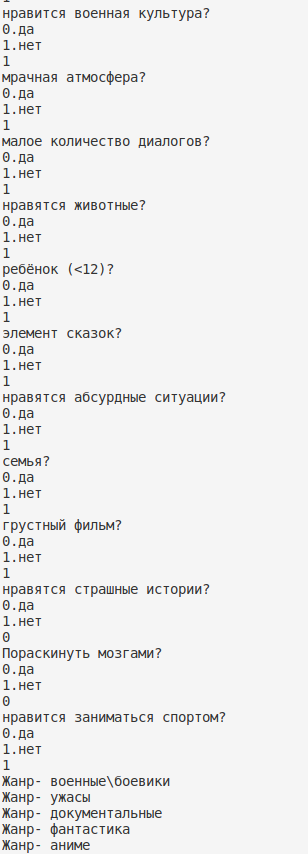
## **Извлечение знаний и база знаний**

Базу знаний жанров фильмов мы составили так: каждый из нас взял те жанры, которые ему наиболее интересны и представил каким людям они нравятся, по этому принципу составил правила.

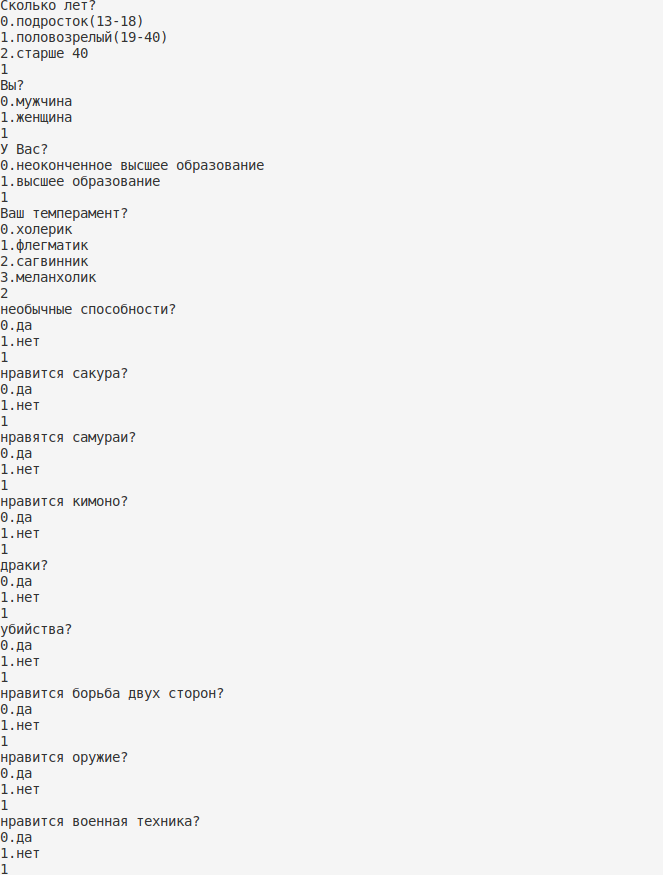
## **Протокол работы системы**

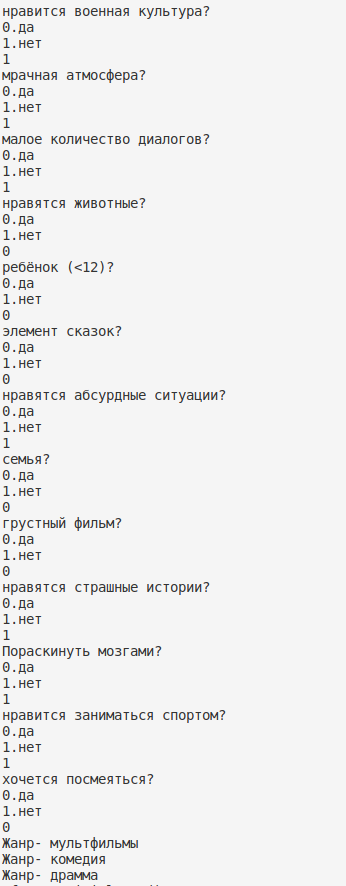
Пример 1





Пример 2





## **Выводы**

Данная работа научила нас работать в команде. Было немного трудно, так как команды составлялись рандомно, а не по взаимной симпатии. Основное общение у нас было в discord. Нам очень понравилась наша концепция экспертной системы , нам кажется она могла быть полезной для людей, которым трудно сделать выбор, а также компаниям людей во время совместного выбора фильма. Мы думаем в будущем нам понадобятся навыки создания экспертных систем, ведь не всегда нужно заниматься сложным машинным обучение, когда можно воспользоваться достаточно несложной логикой экспертных систем.