Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Кафедра вычислительной математики и механики

**Лабораторная работа № 2**

**по дисциплине: «Интеллектуальные ИСИТ»**

Выполнил

студент группы ИСТ-19-2б

Репин М.Е.

Проверил

ассистент кафедры ВММБ

Нетбай Г.В.

Пермь, 2022

# Продукционные системы

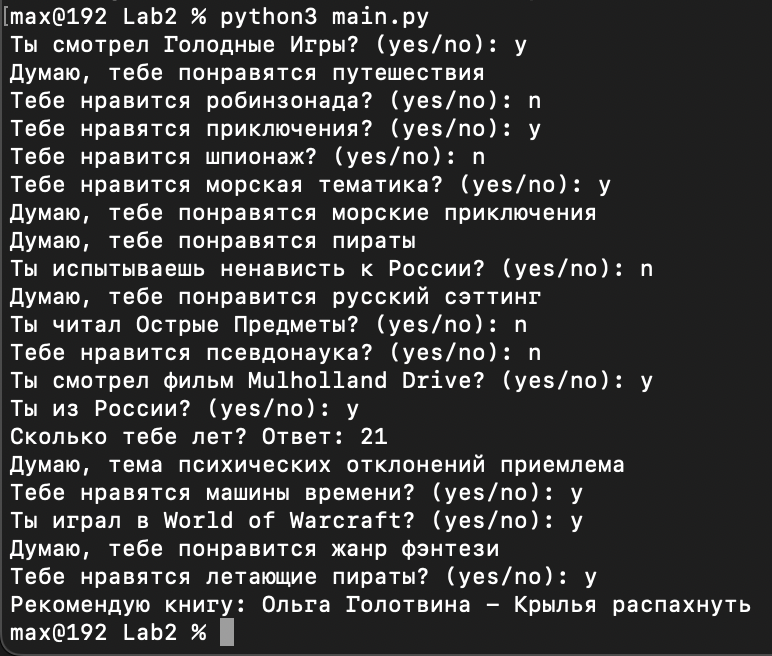
Продукционная система – система принятия решения, основанная на следовании правилам (продукциям). Система моделирует человеческое мышление.

Правила, из которых состоит система, имеют название, условие (левую часть), и действие/следствие (правую часть).

# Задача

Задача второй лабораторной заключается в написании программы, выполняющей роль продукционной системы. Был выбран язык Python. Программа должна выполнять действия в соответствии с правилами БЗ, сохранёнными в отдельный файл. Для хранения БЗ был выбран формат json.

# Пример запуска программы main.py



# Листинг кода main.py

from code import interact

import json

from random import choice

import pathlib

class rule:

def \_\_init\_\_(self, id:str, conditions: list, actions: list) -> None:

self.id = id

self.conditions = conditions

self.actions = actions

self.activated = False

if any([action["func"]=="yes\_or\_no" or action["func"]=="ask\_int" for action in self.actions]):

self.requires\_input = True

else:

self.requires\_input = False

class machine:

def \_\_init\_\_(self, jsonfile: str) -> None:

self.raw\_data = {}

self.rules = {}

self.answered = {}

with open(jsonfile, "r", encoding='utf-8') as read\_rules:

self.raw\_data = json.load(read\_rules)

for id, info in self.raw\_data.items():

self.rules[id] = rule(id, info["conditions"], info["actions"])

for r in self.rules.values():

for condition in r.conditions:

if isinstance(condition, list):

self.answered[condition[0]] = None

def perform\_actions(self, rule\_id: str) -> None:

for action in self.rules[rule\_id].actions:

if action["func"] == "ask\_int":

self.answered[action["arg\_1"]] = self.ask\_int(action["arg\_2"])

elif action["func"] == "yes\_or\_no":

self.answered[action["arg\_1"]] = self.yes\_or\_no(action["arg\_2"])

elif action["func"] == "assert":

self.answered[action["arg\_1"]] = action["arg\_2"]

elif action["func"] == "print":

self.print(action["arg\_1"])

@staticmethod

def ask\_int(question: str) -> int:

while True:

answer = input(question)

try:

answer = int(answer)

return answer

except:

continue

@staticmethod

def yes\_or\_no(question: str) -> str:

while True:

answer = input(question)

if answer.lower() in ["yes", "y"]:

return "yes"

elif answer.lower() in ["no", "n"]:

return "no"

@staticmethod

def print(text: str) -> None:

print(text)

def interprete(self, rule\_id: str) -> tuple:

text = ""

for item in self.rules[rule\_id].conditions:

temp = ""

if isinstance(item, list):

temp += f"self.answered[\"{item[0]}\"]"

if item[1] == "eq":

temp += " == "

elif item[1] == ">":

temp += " > "

elif item[1] == "<":

temp += " < "

if item[2] == "null":

temp += "None"

elif isinstance(item[2], int):

temp += f"{item[2]}"

else:

temp += f"\"{item[2]}\""

elif isinstance(item, str):

temp = f" {item} "

text += temp

try:

ready = eval(text)

except:

ready = False

return text, ready

def run(self) -> None:

while True:

# asking for input

options = []

for rule in self.rules.values():

cond, ready = self.interprete(rule.id)

if ready == True and rule.activated == False:

options += [rule.id]

try:

chosen\_rule = choice(options)

self.perform\_actions(chosen\_rule)

self.rules[chosen\_rule].activated = True

except:

break

self.implications()

def implications(self):

# calculating implications

while True:

done = True

for rule in self.rules.values():

cond, ready = self.interprete(rule.id)

if ready == True and rule.activated == False and rule.requires\_input == False:

self.perform\_actions(rule.id)

rule.activated = True

done = False

break

if done == True:

break

def main():

DEBUG = False

jsonfile = pathlib.Path(\_\_file\_\_).parent.resolve()

jsonfile = str(jsonfile) + "/rules.json"

m = machine(jsonfile)

if DEBUG == False:

m.run()

elif DEBUG == True:

m.perform\_actions("ask-if-read-sharp-objects")

m.perform\_actions("ask-for-age")

m.implications()

if \_\_name\_\_ == "\_\_main\_\_":

main()