**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

# Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №6

# «Основные конструкции языка Python»

Вариант 9

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-35Б |  |  |
| Крыжный. К.С. |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г

import tracemalloc  
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup  
from telegram.ext import Application, CommandHandler, CallbackQueryHandler, ContextTypes  
import random  
  
# Включаем tracemalloc  
tracemalloc.start()  
  
# Состояние игроков  
players = {}  
  
# Генерация кнопок меню  
def get\_main\_menu():  
 keyboard = [  
 [InlineKeyboardButton("Учиться", callback\_data="study"),  
 InlineKeyboardButton("Работать", callback\_data="work")],  
 [InlineKeyboardButton("Отдыхать", callback\_data="rest"),  
 InlineKeyboardButton("Купить еду", callback\_data="buy\_food")],  
 [InlineKeyboardButton("Проверить состояние", callback\_data="status")],  
 [InlineKeyboardButton("Начать карьеру", callback\_data="start\_career")]  
 ]  
 return InlineKeyboardMarkup(keyboard)  
  
# Начало игры  
async def start(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:  
 chat\_id = update.effective\_chat.id  
 players[chat\_id] = {  
 "health": 100,  
 "energy": 100,  
 "knowledge": 0,  
 "money": 50,  
 "luck": random.uniform(0.5, 1.5),  
 "exam\_counter": 5,  
 "turn": 0,  
 "career": None  
 }  
 await update.message.reply\_text(  
 "Добро пожаловать в игру 'Жизнь студента'! Используй кнопки ниже для взаимодействия.",  
 reply\_markup=get\_main\_menu()  
 )  
  
# Проверка состояния  
async def status(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:  
 query = update.callback\_query  
 await query.answer()  
 chat\_id = query.message.chat\_id  
  
 if chat\_id not in players:  
 await query.message.reply\_text("Ты ещё не начал игру! Используй команду /start.")  
 return  
  
 player = players[chat\_id]  
 await query.message.reply\_text(  
 f"Текущее состояние:\n"  
 f"Здоровье: {player['health']}\n"  
 f"Энергия: {player['energy']}\n"  
 f"Знания: {player['knowledge']}\n"  
 f"Деньги: {player['money']}\n"  
 f"Сессия через: {player['exam\_counter']} ходов\n"  
 f"Карьера: {player['career'] if player['career'] else 'Не выбрана'}",  
 reply\_markup=get\_main\_menu()  
 )  
  
# Учёба  
async def study(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:  
 query = update.callback\_query  
 await query.answer()  
 chat\_id = query.message.chat\_id  
  
 player = players[chat\_id]  
 if player["energy"] >= 20:  
 player["energy"] -= 20  
 knowledge\_gain = int(15 \* player["luck"])  
 player["knowledge"] += knowledge\_gain  
 player["exam\_counter"] -= 1  
 player["turn"] += 1  
 await query.message.reply\_text(  
 f"Ты учился и получил {knowledge\_gain} знаний. Энергия уменьшилась.",  
 reply\_markup=get\_main\_menu()  
 )  
 else:  
 await query.message.reply\_text(  
 "Недостаточно энергии для учёбы. Отдохни!",  
 reply\_markup=get\_main\_menu()  
 )  
 await check\_events(update, context)  
 await check\_exam(update, context)  
  
# Работа  
async def work(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:  
 query = update.callback\_query  
 await query.answer()  
 chat\_id = query.message.chat\_id  
  
 player = players[chat\_id]  
 if player["energy"] >= 30:  
 player["energy"] -= 30  
 money\_gain = int(20 \* player["luck"])  
 player["money"] += money\_gain  
 player["exam\_counter"] -= 1  
 player["turn"] += 1  
 await query.message.reply\_text(  
 f"Ты работал и заработал {money\_gain} денег. Энергия уменьшилась.",  
 reply\_markup=get\_main\_menu()  
 )  
 else:  
 await query.message.reply\_text(  
 "Недостаточно энергии для работы. Отдохни!",  
 reply\_markup=get\_main\_menu()  
 )  
 await check\_events(update, context)  
 await check\_exam(update, context)  
  
# Отдых  
async def rest(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:  
 query = update.callback\_query  
 await query.answer()  
 chat\_id = query.message.chat\_id  
  
 player = players[chat\_id]  
 player["health"] += 10  
 player["energy"] += 30  
 player["turn"] += 1  
 await query.message.reply\_text(  
 "Ты отдохнул и восстановил силы. Здоровье и энергия увеличились.",  
 reply\_markup=get\_main\_menu()  
 )  
 await check\_events(update, context)  
  
# Покупка еды  
async def buy\_food(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:  
 query = update.callback\_query  
 await query.answer()  
 chat\_id = query.message.chat\_id  
  
 player = players[chat\_id]  
 if player["money"] >= 20:  
 player["money"] -= 20  
 player["health"] += 30  
 await query.message.reply\_text(  
 "Ты купил еду и восстановил здоровье. Деньги уменьшились.",  
 reply\_markup=get\_main\_menu()  
 )  
 else:  
 await query.message.reply\_text(  
 "Недостаточно денег для покупки еды!",  
 reply\_markup=get\_main\_menu()  
 )  
  
# Начало карьеры  
async def start\_career(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:  
 query = update.callback\_query  
 await query.answer()  
 chat\_id = query.message.chat\_id  
  
 player = players[chat\_id]  
 if player["knowledge"] >= 50:  
 career\_options = ["Программист", "Инженер", "Менеджер"]  
 player["career"] = random.choice(career\_options)  
 player["money"] += 100  
 await query.message.reply\_text(  
 f"Ты начал карьеру: {player['career']}. Бонус: 100 денег.",  
 reply\_markup=get\_main\_menu()  
 )  
 else:  
 await query.message.reply\_text(  
 "Недостаточно знаний для начала карьеры. Учись больше!",  
 reply\_markup=get\_main\_menu()  
 )  
  
# Проверка событий  
async def check\_events(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:  
 chat\_id = update.callback\_query.message.chat\_id  
 player = players[chat\_id]  
  
 events = {  
 5: "Ты нашёл кошелёк на улице. Оставить себе?",  
 11: "Друг предлагает поучаствовать в стартапе. Присоединиться?",  
 20: "Тебе предложили стажировку. Пройти её?",  
 27: "Профессор дал возможность выполнить бонусный проект. Взяться?",  
 35: "Ты нашёл полезный ресурс для учёбы. Использовать?",  
 42: "Кто-то предложил курс по саморазвитию. Записаться?",  
 50: "Тебя пригласили на оплачиваемый проект. Принять?"  
 }  
  
 if player["turn"] in events:  
 event\_text = events[player["turn"]]  
 keyboard = [  
 [InlineKeyboardButton("Да", callback\_data=f"event\_yes\_{player['turn']}"),  
 InlineKeyboardButton("Нет", callback\_data=f"event\_no\_{player['turn']}")]  
 ]  
 await update.callback\_query.message.reply\_text(  
 event\_text,  
 reply\_markup=InlineKeyboardMarkup(keyboard)  
 )  
  
# Обработка событий  
async def handle\_event(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:  
 query = update.callback\_query  
 await query.answer()  
 chat\_id = query.message.chat\_id  
 player = players[chat\_id]  
  
 action = query.data.split('\_')[1]  
 event\_turn = int(query.data.split('\_')[-1])  
  
 if action == "yes":  
 if event\_turn == 5:  
 player["money"] += 50  
 await query.message.reply\_text("Ты оставил кошелёк себе и получил 50 денег.")  
 elif event\_turn == 11:  
 player["luck"] += 0.2  
 await query.message.reply\_text("Ты присоединился к стартапу и повысил удачу!")  
 elif event\_turn == 20:  
 player["knowledge"] += 30  
 await query.message.reply\_text("Ты прошёл стажировку и увеличил знания.")  
 elif event\_turn == 27:  
 player["knowledge"] += 20  
 await query.message.reply\_text("Ты выполнил проект и улучшил знания.")  
 elif event\_turn == 35:  
 player["knowledge"] += 15  
 await query.message.reply\_text("Ты воспользовался ресурсом и получил знания.")  
 elif event\_turn == 42:  
 player["knowledge"] += 10  
 await query.message.reply\_text("Ты записался на курс и стал умнее.")  
 elif event\_turn == 50:  
 player["money"] += 70  
 await query.message.reply\_text("Ты принял проект и получил деньги.")  
 else:  
 await query.message.reply\_text("Ты решил ничего не делать.")  
  
# Проверка экзамена  
async def check\_exam(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:  
 chat\_id = update.callback\_query.message.chat\_id  
 player = players[chat\_id]  
  
 if player['exam\_counter'] <= 0:  
 if player['knowledge'] >= 50:  
 player['knowledge'] = 0  
 player['exam\_counter'] = 5  
 await update.callback\_query.message.reply\_text(  
 "Ты успешно сдал сессию!"  
 )  
 else:  
 player['health'] -= 30  
 player['exam\_counter'] = 5  
 await update.callback\_query.message.reply\_text(  
 "Ты провалил сессию и потерял здоровье!"  
 )  
  
# Основной запуск  
def main() -> None:  
 application = Application.builder().token("7631428660:AAHfZiybtXP26bWvQgSKtk3nQRwsx6fePpQ").build()  
  
 application.add\_handler(CommandHandler("start", start))  
 application.add\_handler(CallbackQueryHandler(status, pattern="status"))  
 application.add\_handler(CallbackQueryHandler(study, pattern="study"))  
 application.add\_handler(CallbackQueryHandler(work, pattern="work"))  
 application.add\_handler(CallbackQueryHandler(rest, pattern="rest"))  
 application.add\_handler(CallbackQueryHandler(buy\_food, pattern="buy\_food"))  
 application.add\_handler(CallbackQueryHandler(start\_career, pattern="start\_career"))  
 application.add\_handler(CallbackQueryHandler(handle\_event, pattern="event\_.\*"))  
  
 application.run\_polling()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()