Full версия

import telebot

from telebot import types

import time

import threading

import random

import json

# Вставьте ваш токен бота здесь

BOT\_TOKEN = 'YOUR\_BOT\_TOKEN\_HERE' # Замените на ваш токен

bot = telebot.TeleBot(BOT\_TOKEN)

ADMIN\_TELEGRAM\_ID = 6400017164 # Замените на ваш Telegram ID

# Словарь для хранения данных пользователей

user\_data = {}

user\_data\_lock = threading.Lock() # Блокировка для синхронизации доступа к user\_data

# Список руд для каждой планеты с уникальными названиями и весами

ores = {

'Earth': [

{'name': 'Камень', 'weight': 50},

{'name': 'Уголь', 'weight': 40},

{'name': 'Железная руда', 'weight': 30},

{'name': 'Медная руда', 'weight': 25},

{'name': 'Золотая руда', 'weight': 20},

{'name': 'Серебряная руда', 'weight': 15},

{'name': 'Алмазная руда', 'weight': 10},

{'name': 'Платиновая руда', 'weight': 5},

{'name': 'Редкая руда', 'weight': 3},

{'name': 'Легендарная руда', 'weight': 1}

],

'Moon': [

{'name': 'Лунный гранит', 'weight': 45},

{'name': 'Селенит', 'weight': 35},

{'name': 'Криолит', 'weight': 25},

{'name': 'Старый Лунный металл', 'weight': 15},

{'name': 'Метеоритная руда', 'weight': 5}

],

'Mars': [

{'name': 'Марсианский базальт', 'weight': 40},

{'name': 'Красный гранит', 'weight': 30},

{'name': 'Фобийский металл', 'weight': 20},

{'name': 'Валькирия руда', 'weight': 10},

{'name': 'Космическая руда', 'weight': 5}

]

}

# Характеристики кирок (15 типов)

pickaxes = {

'Earth': [

{'name': 'Деревянная кирка', 'luck': 0},

{'name': 'Каменная кирка', 'luck': 5},

{'name': 'Железная кирка', 'luck': 10},

{'name': 'Медная кирка', 'luck': 15},

{'name': 'Золотая кирка', 'luck': 20},

{'name': 'Серебряная кирка', 'luck': 25},

{'name': 'Алмазная кирка', 'luck': 30},

{'name': 'Платиновая кирка', 'luck': 35},

{'name': 'Редкая кирка', 'luck': 40},

{'name': 'Легендарная кирка', 'luck': 50},

{'name': 'Эпическая кирка', 'luck': 60},

{'name': 'Мифическая кирка', 'luck': 70},

{'name': 'Божественная кирка', 'luck': 80},

{'name': 'Титановая кирка', 'luck': 90},

{'name': 'Ультра кирка', 'luck': 100}

],

'Moon': [

{'name': 'Лунная кирка Начинающего', 'luck': 10},

{'name': 'Лунная кирка Усовершенствованная', 'luck': 15},

{'name': 'Лунная кирка Эксперта', 'luck': 20},

{'name': 'Лунная кирка Мастера', 'luck': 25},

{'name': 'Лунная кирка Легенды', 'luck': 30},

{'name': 'Лунная кирка Императора', 'luck': 35},

{'name': 'Лунная кирка Совершенства', 'luck': 40},

{'name': 'Лунная кирка Великого', 'luck': 45},

{'name': 'Лунная кирка Небес', 'luck': 50},

{'name': 'Лунная кирка Божественная', 'luck': 55},

{'name': 'Лунная кирка Космическая', 'luck': 60},

{'name': 'Лунная кирка Эксельсиор', 'luck': 65},

{'name': 'Лунная кирка Галактус', 'luck': 70},

{'name': 'Лунная кирка Звёздная', 'luck': 75},

{'name': 'Лунная кирка Абсолютная', 'luck': 80}

],

'Mars': [

{'name': 'Марсианская кирка Начинающего', 'luck': 20},

{'name': 'Марсианская кирка Усовершенствованная', 'luck': 25},

{'name': 'Марсианская кирка Эксперта', 'luck': 30},

{'name': 'Марсианская кирка Мастера', 'luck': 35},

{'name': 'Марсианская кирка Легенды', 'luck': 40},

{'name': 'Марсианская кирка Императора', 'luck': 45},

{'name': 'Марсианская кирка Совершенства', 'luck': 50},

{'name': 'Марсианская кирка Великого', 'luck': 55},

{'name': 'Марсианская кирка Небес', 'luck': 60},

{'name': 'Марсианская кирка Божественная', 'luck': 65},

{'name': 'Марсианская кирка Космическая', 'luck': 70},

{'name': 'Марсианская кирка Эксельсиор', 'luck': 75},

{'name': 'Марсианская кирка Галактус', 'luck': 80},

{'name': 'Марсианская кирка Звёздная', 'luck': 85},

{'name': 'Марсианская кирка Абсолютная', 'luck': 90}

]

}

# Характеристики портфелей (15 типов)

backpacks = {

'Earth': [

{'name': 'Маленький портфель', 'capacity': 20},

{'name': 'Обычный портфель', 'capacity': 40},

{'name': 'Большой портфель', 'capacity': 60},

{'name': 'Походный портфель', 'capacity': 80},

{'name': 'Профессиональный портфель', 'capacity': 100},

{'name': 'Экспедиционный портфель', 'capacity': 120},

{'name': 'Горный портфель', 'capacity': 140},

{'name': 'Шахтерский портфель', 'capacity': 160},

{'name': 'Гигантский портфель', 'capacity': 180},

{'name': 'Бесконечный портфель', 'capacity': 200},

{'name': 'Портфель Хранителя', 'capacity': 220},

{'name': 'Портфель Магната', 'capacity': 240},

{'name': 'Портфель Колосса', 'capacity': 260},

{'name': 'Портфель Громовержца', 'capacity': 280},

{'name': 'Портфель Богов', 'capacity': 300}

],

'Moon': [

{'name': 'Лунный рюкзак Начинающего', 'capacity': 25},

{'name': 'Лунный рюкзак Усовершенствованный', 'capacity': 50},

{'name': 'Лунный рюкзак Эксперта', 'capacity': 75},

{'name': 'Лунный рюкзак Мастера', 'capacity': 100},

{'name': 'Лунный рюкзак Легенды', 'capacity': 125},

{'name': 'Лунный рюкзак Императора', 'capacity': 150},

{'name': 'Лунный рюкзак Совершенства', 'capacity': 175},

{'name': 'Лунный рюкзак Великого', 'capacity': 200},

{'name': 'Лунный рюкзак Небес', 'capacity': 225},

{'name': 'Лунный рюкзак Божественный', 'capacity': 250},

{'name': 'Лунный рюкзак Космический', 'capacity': 275},

{'name': 'Лунный рюкзак Эксельсиор', 'capacity': 300},

{'name': 'Лунный рюкзак Галактус', 'capacity': 325},

{'name': 'Лунный рюкзак Звёздный', 'capacity': 350},

{'name': 'Лунный рюкзак Абсолютный', 'capacity': 375}

],

'Mars': [

{'name': 'Марсианский рюкзак Начинающего', 'capacity': 30},

{'name': 'Марсианский рюкзак Усовершенствованный', 'capacity': 60},

{'name': 'Марсианский рюкзак Эксперта', 'capacity': 90},

{'name': 'Марсианский рюкзак Мастера', 'capacity': 120},

{'name': 'Марсианский рюкзак Легенды', 'capacity': 150},

{'name': 'Марсианский рюкзак Императора', 'capacity': 180},

{'name': 'Марсианский рюкзак Совершенства', 'capacity': 210},

{'name': 'Марсианский рюкзак Великого', 'capacity': 240},

{'name': 'Марсианский рюкзак Небес', 'capacity': 270},

{'name': 'Марсианский рюкзак Божественный', 'capacity': 300},

{'name': 'Марсианский рюкзак Космический', 'capacity': 330},

{'name': 'Марсианский рюкзак Эксельсиор', 'capacity': 360},

{'name': 'Марсианский рюкзак Галактус', 'capacity': 390},

{'name': 'Марсианский рюкзак Звёздный', 'capacity': 420},

{'name': 'Марсианский рюкзак Абсолютный', 'capacity': 450}

]

}

# Характеристики печей (10 типов)

furnaces = {

'Earth': [

{'name': 'Простая печь', 'smelt\_capacity': 5},

{'name': 'Улучшенная печь', 'smelt\_capacity': 10},

{'name': 'Высокотемпературная печь', 'smelt\_capacity': 20},

{'name': 'Мощная печь', 'smelt\_capacity': 40},

{'name': 'Индустриальная печь', 'smelt\_capacity': 60},

{'name': 'Термоядерная печь', 'smelt\_capacity': 80},

{'name': 'Печь Бога Огня', 'smelt\_capacity': 100},

{'name': 'Печь Ледяного Короля', 'smelt\_capacity': 120},

{'name': 'Печь Демона', 'smelt\_capacity': 140},

{'name': 'Печь Всесильного', 'smelt\_capacity': 160}

],

'Moon': [

{'name': 'Лунная печь Начинающего', 'smelt\_capacity': 10},

{'name': 'Лунная печь Усовершенствованная', 'smelt\_capacity': 20},

{'name': 'Лунная печь Эксперта', 'smelt\_capacity': 30},

{'name': 'Лунная печь Мастера', 'smelt\_capacity': 40},

{'name': 'Лунная печь Легенды', 'smelt\_capacity': 50},

{'name': 'Лунная печь Императора', 'smelt\_capacity': 60},

{'name': 'Лунная печь Совершенства', 'smelt\_capacity': 70},

{'name': 'Лунная печь Великого', 'smelt\_capacity': 80},

{'name': 'Лунная печь Небес', 'smelt\_capacity': 90},

{'name': 'Лунная печь Божественная', 'smelt\_capacity': 100}

],

'Mars': [

{'name': 'Марсианская печь Начинающего', 'smelt\_capacity': 15},

{'name': 'Марсианская печь Усовершенствованная', 'smelt\_capacity': 30},

{'name': 'Марсианская печь Эксперта', 'smelt\_capacity': 45},

{'name': 'Марсианская печь Мастера', 'smelt\_capacity': 60},

{'name': 'Марсианская печь Легенды', 'smelt\_capacity': 75},

{'name': 'Марсианская печь Императора', 'smelt\_capacity': 90},

{'name': 'Марсианская печь Совершенства', 'smelt\_capacity': 105},

{'name': 'Марсианская печь Великого', 'smelt\_capacity': 120},

{'name': 'Марсианская печь Небес', 'smelt\_capacity': 135},

{'name': 'Марсианская печь Божественная', 'smelt\_capacity': 150}

]

}

# Список шахт (30 типов)

mines = {

'Earth': [

'Заброшенная шахта',

'Небольшая шахта',

'Обычная шахта',

'Глубокая шахта',

'Богатая шахта',

'Подземный рудник',

'Кристальная шахта',

'Алмазная шахта',

'Золотой карьер',

'Скрытая шахта',

'Лунная шахта',

'Солнечная шахта',

'Песчаная шахта',

'Водная шахта',

'Ледяная шахта',

'Лавовая шахта',

'Эфирная шахта',

'Темная шахта',

'Светлая шахта',

'Магическая шахта',

'Техно-шахта',

'Кибер-шахта',

'Туманная шахта',

'Горная шахта',

'Вулканическая шахта',

'Радужная шахта',

'Призрачная шахта',

'Древняя шахта',

'Фантазийная шахта',

'Титаническая шахта',

'Эльфийская шахта'

],

'Moon': [

'Лунная шахта Начинающего',

'Лунная шахта Усовершенствованная',

'Лунная шахта Эксперта',

'Лунная шахта Мастера',

'Лунная шахта Легенды',

'Лунная шахта Императора',

'Лунная шахта Совершенства',

'Лунная шахта Великого',

'Лунная шахта Небес',

'Лунная шахта Божественная',

'Лунная шахта Космическая',

'Лунная шахта Эксельсиор',

'Лунная шахта Галактус',

'Лунная шахта Звёздная',

'Лунная шахта Абсолютная',

'Лунная шахта Омега',

'Лунная шахта Альфа',

'Лунная шахта Бета',

'Лунная шахта Гамма',

'Лунная шахта Дельта',

'Лунная шахта Эпсилон',

'Лунная шахта Дзета',

'Лунная шахта Йота',

'Лунная шахта Каппа',

'Лунная шахта Лямбда',

'Лунная шахта Мю',

'Лунная шахта Ню',

'Лунная шахта Кси',

'Лунная шахта Омикрон',

'Лунная шахта Пси',

'Лунная шахта Омега'

],

'Mars': [

'Марсианская шахта Начинающего',

'Марсианская шахта Усовершенствованная',

'Марсианская шахта Эксперта',

'Марсианская шахта Мастера',

'Марсианская шахта Легенды',

'Марсианская шахта Императора',

'Марсианская шахта Совершенства',

'Марсианская шахта Великого',

'Марсианская шахта Небес',

'Марсианская шахта Божественная',

'Марсианская шахта Космическая',

'Марсианская шахта Эксельсиор',

'Марсианская шахта Галактус',

'Марсианская шахта Звёздная',

'Марсианская шахта Абсолютная',

'Марсианская шахта Омега',

'Марсианская шахта Альфа',

'Марсианская шахта Бета',

'Марсианская шахта Гамма',

'Марсианская шахта Дельта',

'Марсианская шахта Эпсилон',

'Марсианская шахта Дзета',

'Марсианская шахта Йота',

'Марсианская шахта Каппа',

'Марсианская шахта Лямбда',

'Марсианская шахта Мю',

'Марсианская шахта Ню',

'Марсианская шахта Кси',

'Марсианская шахта Омикрон',

'Марсианская шахта Пси',

'Марсианская шахта Омега'

]

}

# 🧰 Кейсы: Добавляем список кейсов без смайликов с уникальными названиями для каждой планеты

cases = {

'Earth': [

{'name': 'Бронзовый кейс', 'price': 1500, 'min\_reward': 1000, 'max\_reward': 2500},

{'name': 'Серебряный кейс', 'price': 3000, 'min\_reward': 2500, 'max\_reward': 5000},

{'name': 'Золотой кейс', 'price': 6000, 'min\_reward': 5000, 'max\_reward': 10000},

{'name': 'Платиновый кейс', 'price': 12000, 'min\_reward': 10000, 'max\_reward': 20000},

{'name': 'Алмазный кейс', 'price': 24000, 'min\_reward': 20000, 'max\_reward': 40000},

{'name': 'Рубиновый кейс', 'price': 48000, 'min\_reward': 40000, 'max\_reward': 80000},

{'name': 'Эльфийский кейс', 'price': 96000, 'min\_reward': 80000, 'max\_reward': 160000},

{'name': 'Драконов кейс', 'price': 192000, 'min\_reward': 160000, 'max\_reward': 320000},

{'name': 'Фениксов кейс', 'price': 384000, 'min\_reward': 320000, 'max\_reward': 640000},

{'name': 'Божественный кейс', 'price': 768000, 'min\_reward': 640000, 'max\_reward': 1280000}

],

'Moon': [

{'name': 'Кейс 1', 'price': 5000, 'min\_reward': 3000, 'max\_reward': 7000},

{'name': 'Кейс 2', 'price': 10000, 'min\_reward': 6000, 'max\_reward': 14000},

{'name': 'Кейс 3', 'price': 20000, 'min\_reward': 12000, 'max\_reward': 28000},

{'name': 'Кейс 4', 'price': 40000, 'min\_reward': 24000, 'max\_reward': 56000},

{'name': 'Кейс 5', 'price': 80000, 'min\_reward': 48000, 'max\_reward': 112000},

{'name': 'Кейс 6', 'price': 160000, 'min\_reward': 96000, 'max\_reward': 224000},

{'name': 'Кейс 7', 'price': 320000, 'min\_reward': 192000, 'max\_reward': 448000},

{'name': 'Кейс 8', 'price': 640000, 'min\_reward': 384000, 'max\_reward': 896000},

{'name': 'Кейс 9', 'price': 1280000, 'min\_reward': 768000, 'max\_reward': 1792000},

{'name': 'Кейс 10', 'price': 2560000, 'min\_reward': 1536000, 'max\_reward': 3584000}

],

'Mars': [

{'name': 'Кейс 1', 'price': 10000, 'min\_reward': 7000, 'max\_reward': 13000},

{'name': 'Кейс 2', 'price': 20000, 'min\_reward': 14000, 'max\_reward': 26000},

{'name': 'Кейс 3', 'price': 40000, 'min\_reward': 28000, 'max\_reward': 52000},

{'name': 'Кейс 4', 'price': 80000, 'min\_reward': 56000, 'max\_reward': 104000},

{'name': 'Кейс 5', 'price': 160000, 'min\_reward': 112000, 'max\_reward': 208000},

{'name': 'Кейс 6', 'price': 320000, 'min\_reward': 224000, 'max\_reward': 416000},

{'name': 'Кейс 7', 'price': 640000, 'min\_reward': 448000, 'max\_reward': 832000},

{'name': 'Кейс 8', 'price': 1280000, 'min\_reward': 896000, 'max\_reward': 1664000},

{'name': 'Кейс 9', 'price': 2560000, 'min\_reward': 1792000, 'max\_reward': 3328000},

{'name': 'Кейс 10', 'price': 5120000, 'min\_reward': 3584000, 'max\_reward': 6656000}

]

}

# Список ребиртов (10 уровней)

rebirths = [

{'name': 'Новичёк', 'price': 100000},

{'name': 'Опытный', 'price': 200000},

{'name': 'Профессионал', 'price': 300000},

{'name': 'Мастер', 'price': 400000},

{'name': 'Гуру', 'price': 500000},

{'name': 'Эксперт', 'price': 600000},

{'name': 'Легенда', 'price': 700000},

{'name': 'Великий', 'price': 800000},

{'name': 'Император', 'price': 900000},

{'name': 'Бессмертный', 'price': 1000000}

]

# Геймпассы

gamepasses = {

'double\_luck': {

'name': 'Удвоенная удача',

'price': 500000, # Цена

'description': 'Двойная удача на добытые руды.',

'effect': 'double\_luck'

},

'ultra\_luck': {

'name': 'Ультра удача',

'price': 900000,

'description': 'Удача увеличена в 5 раз на руды.',

'effect': 'ultra\_luck'

},

'double\_ore': {

'name': 'Двойная руда',

'price': 700000,

'description': 'Двойная руда выпадает при добыче руды.',

'effect': 'double\_ore'

},

'instant\_smelting': {

'name': 'Моментальная переплавка',

'price': 500000,

'description': 'Моментальная переплавка без ожидания 3 секунд.',

'effect': 'instant\_smelting'

}

}

# Кликер

clicker = {

'cooldown': 2, # секунды

'reward': 1 # монеты за клик

}

# Майнеры с уникальными названиями

miners = {

'Mars': [

{'name': 'Марсианский Нож', 'price': 7500, 'reward': 300, 'interval': 36000}, # 10 часов

{'name': 'Красный Клинок', 'price': 14000, 'reward': 600, 'interval': 36000},

{'name': 'Фобийский Дробитель', 'price': 21000, 'reward': 900, 'interval': 36000},

{'name': 'Валькирия Боевая', 'price': 28000, 'reward': 1200, 'interval': 36000},

{'name': 'Космический Извилин', 'price': 35000, 'reward': 1500, 'interval': 36000}

]

}

# Команда /start

@bot.message\_handler(commands=['start'])

def start\_game(message):

user\_id = message.from\_user.id

with user\_data\_lock:

if user\_id not in user\_data:

user\_data[user\_id] = {

'current\_planet': 'Earth', # Начальная планета: Земля

'purchased\_planets': ['Earth'], # Игрок начинает только с Земли

'pickaxe\_level': {'Earth': 1},

'backpack\_level': {'Earth': 1},

'furnace\_level': {'Earth': 1},

'mine\_level': {'Earth': 1},

'balance': 0, # Начальный баланс

'ores': {'Earth': {}, 'Moon': {}, 'Mars': {}},

'smelting': {'Earth': {}, 'Moon': {}, 'Mars': {}},

'coal': {'Earth': 0, 'Moon': 0, 'Mars': 0},

'is\_mining': {'Earth': False, 'Moon': False, 'Mars': False},

'purchased\_pickaxes': {'Earth': [1], 'Moon': [], 'Mars': []},

'purchased\_backpacks': {'Earth': [1], 'Moon': [], 'Mars': []},

'purchased\_furnaces': {'Earth': [1], 'Moon': [], 'Mars': []},

'purchased\_mines': {'Earth': [1], 'Moon': [], 'Mars': []},

'ingots': {'Earth': {}, 'Moon': {}, 'Mars': {}},

'rebirth\_count': 0, # Количество ребиртов

'multiplier': 1, # Множитель монет

'gamepasses': {}, # Покупленные геймпассы

'clicker\_last\_click': 0, # Время последнего клика

'clicker\_balance': 0, # Баланс кликера

'miners': {} # Покупленные майнеры

}

welcome\_text = (

"Добро пожаловать в игру \*Шахтёр\*!\n\n"

"Вы начинаете своё путешествие на планете \*Земля\*.\n"

"Исследуйте рудные поля, улучшайте своё оборудование и открывайте новые планеты для ещё больших возможностей!"

)

bot.send\_message(message.chat.id, welcome\_text, parse\_mode='Markdown')

show\_main\_menu(message)

# Функция для отображения главного меню

def show\_main\_menu(message):

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

mine\_button = types.KeyboardButton('⛏ Добывать руду')

furnace\_button = types.KeyboardButton('🔥 Печь')

sell\_button = types.KeyboardButton('💰 Продать руду')

shop\_button = types.KeyboardButton('🛒 Магазин')

balance\_button = types.KeyboardButton('💎 Баланс')

inventory\_button = types.KeyboardButton('🎒 Мой портфель')

switch\_planet\_button = types.KeyboardButton('🔄 Переключиться на планету')

help\_button = types.KeyboardButton('❓ Помощь')

user\_id = message.from\_user.id

with user\_data\_lock:

if user\_id == ADMIN\_TELEGRAM\_ID:

admin\_button = types.KeyboardButton('🔴 Админ панель 🔴')

markup.add(admin\_button)

markup.add(mine\_button, furnace\_button)

markup.add(sell\_button, shop\_button)

markup.add(balance\_button, inventory\_button)

markup.add(switch\_planet\_button)

markup.add(help\_button)

bot.send\_message(message.chat.id, "Главное меню", reply\_markup=markup)

# Обработчик для раздела "Помощь"

@bot.message\_handler(func=lambda message: message.text == '❓ Помощь')

def show\_help(message):

faq\_text = """

\*Часто задаваемые вопросы (FAQ):\*

1️⃣ \*Как начать игру?\*

- Используйте команду /start, чтобы начать игру и инициализировать ваш профиль.

2️⃣ \*Как добывать руду?\*

- В главном меню нажмите на кнопку ⛏ Добывать руду. Каждая добыча занимает 3 секунды.

- Чем выше уровень вашей \*кирки\*, тем более ценные руды вы сможете добывать.

- Чем выше уровень вашей \*шахты\*, тем больше шанс найти редкие и ценные руды.

3️⃣ \*Как переплавлять руду?\*

- В главном меню выберите 🔥 Печь, затем добавьте уголь и переплавьте руду.

- Чем выше уровень вашей \*печи\*, тем больше руды можно переплавить за раз.

4️⃣ \*Как продавать руду?\*

- Нажмите на 💰 Продать руду в главном меню, чтобы продать все переплавленные руды.

5️⃣ \*Как улучшить оборудование?\*

- Воспользуйтесь 🛒 Магазином для покупки новых кирок, рюкзаков, печей и шахт.

- Вы можете выбрать из доступных вариантов, включая предыдущие уровни.

- При покупке оборудование помечается как \*(Куплено)\*, и вы можете бесплатно переключаться между купленными предметами.

- При покупке оборудования отображаются его характеристики.

6️⃣ \*Как узнать свой баланс и содержимое портфеля?\*

- Нажмите на 💎 Баланс или 🎒 Мой портфель в главном меню.

- Чем выше уровень вашего \*портфеля\*, тем больше руды вы можете носить с собой.

7️⃣ \*🔴 Админ панель 🔴:\*

- Доступна только администратору бота.

- Позволяет управлять пользователями: добавлять/убирать баланс, удалять пользователей и просматривать список активных игроков.

8️⃣ \*Что делать, если бот не реагирует на команды?\*

- Используйте кнопку ⏪ Меню, чтобы вернуться в главное меню или перезапустите бота.

9️⃣ \*Как получить больше угля?\*

- Уголь можно добывать вместе с рудой, при выборе для переплавки.

- Вы можете добавлять уголь в печь неограниченно, сколько захотите.

🔟 \*Есть ли ограничение на уровень оборудования?\*

- Да, каждый тип оборудования имеет максимальный уровень.

1️⃣1️⃣ \*Как открыть новые планеты?\*

- В магазине доступна вкладка "Купить планеты" для приобретения Луна и Марс за 150,000 и 350,000 монет соответственно.

- После покупки планеты вы сможете бесплатно переключаться между Луной и Марсом.

- Открыв планету, вы получите доступ к её уникальному магазину с более дорогими предметами и увеличенными наградами.

1️⃣2️⃣ \*Что такое Ребирты?\*

- Ребирты позволяют вам сбросить весь прогресс в игре (баланс, купленные предметы, планеты и т.д.) в обмен на увеличенный множитель монет.

- Каждый новый ребирт увеличивает ваш множитель монет на ×1.

- Например, после первого ребирта множитель будет ×2, после второго ×3 и т.д.

- Ребирты можно покупать только на планете Марс.

- Максимальное количество ребиртов: 10.

1️⃣3️⃣ \*Что такое Кейсы?\*

- Кейсы позволяют вам испытать свою удачу.

- При открытии кейса вы можете как выиграть монеты, так и проиграть часть или всю вложенную сумму.

- Удача зависит от выбранного кейса и множителя монет вашего аккаунта.

- Открывайте кейсы для увеличения вашего баланса и получения редких наград.

1️⃣4️⃣ \*Что такое Геймпассы?\*

- Геймпассы предоставляют специальные бонусы, такие как удвоенная удача, ультра удача, двойная руда и моментальная переплавка.

- Геймпассы можно приобрести в специальном разделе веб-приложения.

- Геймпассы не сбрасываются при ребиртах и сохраняются в вашем профиле.

1️⃣5️⃣ \*Что такое Кликер?\*

- Кликер позволяет вам получать монеты при нажатии на кнопку.

- Каждые 2 секунды вы можете получить 1 монету, кликая на кнопку.

1️⃣6️⃣ \*Что такое Майнеры?\*

- Майнеры автоматически добавляют монеты в ваш баланс через определённые интервалы времени.

- Каждый майнер имеет уникальное название и стоимость.

- Количество майнеров влияет на скорость накопления монет.

\*Для возврата в главное меню используйте кнопку ⏪ Меню.\*

\*Описание планет:\*

🌍 \*Земля:\*

- Начальная планета с базовыми предметами и рудой.

🌕 \*Луна:\*

- Дополнительная планета с уникальными предметами и рудой.

🔴 \*Марс:\*

- Стоимость: 350,000 монет.

- Уникальные предметы: Марсианские кирки, рюкзаки, печи и шахты.

- Руды и кейсы на Марсе стоят дороже и приносят больше монет, чем на Луне.

- Руды на Марсе имеют уникальные названия и продаются дороже, чем на Луне.

- Ребирты доступны только на Марсе.

\*Описание ребиртов:\*

🔹 \*Ребирты:\*

- Позволяют сбросить весь прогресс в игре.

- Каждый ребирт увеличивает ваш множитель монет на ×1.

- Ребирты продаются исключительно на планете Марс.

- Максимальное количество ребиртов: 10.

\*Описание геймпассов:\*

✨ \*Геймпассы:\*

- \*\*Удвоенная удача\*\*: Двойная удача на добытые руды.

- \*\*Ультра удача\*\*: Удача увеличена в 5 раз на руды.

- \*\*Двойная руда\*\*: Двойная руда выпадает при добыче руды.

- \*\*Моментальная переплавка\*\*: Моментальная переплавка без ожидания 3 секунд.

\*Для возврата в главное меню используйте кнопку ⏪ Меню.\*

"""

bot.send\_message(message.chat.id, faq\_text, parse\_mode='Markdown')

# Команда админ-панели

@bot.message\_handler(func=lambda message: message.text == '🔴 Админ панель 🔴')

def admin\_panel(message):

user\_id = message.from\_user.id

with user\_data\_lock:

if user\_id == ADMIN\_TELEGRAM\_ID:

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

add\_balance\_button = types.KeyboardButton('Добавить баланс пользователю')

remove\_balance\_button = types.KeyboardButton('Убрать баланс у пользователя')

delete\_user\_button = types.KeyboardButton('Удалить пользователя из бота')

view\_users\_button = types.KeyboardButton('Просмотреть пользователей')

back\_button = types.KeyboardButton('⏪ Меню')

markup.add(add\_balance\_button, remove\_balance\_button)

markup.add(delete\_user\_button, view\_users\_button)

markup.add(back\_button)

# Оформление заголовка админ-панели

admin\_header = "🔴 \*АДМИН ПАНЕЛЬ\* 🔴"

bot.send\_message(message.chat.id, admin\_header, reply\_markup=markup, parse\_mode='Markdown')

bot.register\_next\_step\_handler(message, admin\_commands)

else:

bot.send\_message(message.chat.id, "У вас нет доступа к этой команде.")

def admin\_commands(message):

if message.text == 'Добавить баланс пользователю':

bot.send\_message(message.chat.id, "Введите ID пользователя, которому вы хотите добавить баланс:")

bot.register\_next\_step\_handler(message, add\_balance\_to\_user)

elif message.text == 'Убрать баланс у пользователя':

bot.send\_message(message.chat.id, "Введите ID пользователя, у которого вы хотите убрать баланс:")

bot.register\_next\_step\_handler(message, remove\_balance\_from\_user)

elif message.text == 'Удалить пользователя из бота':

bot.send\_message(message.chat.id, "Введите ID пользователя, которого вы хотите удалить:")

bot.register\_next\_step\_handler(message, delete\_user)

elif message.text == 'Просмотреть пользователей':

with user\_data\_lock:

if not user\_data:

users\_list = "Нет активных пользователей."

else:

users\_list = "Список пользователей:\n"

for uid, data in user\_data.items():

users\_list += f"ID: {uid}, Планета: {data['current\_planet']}, Баланс: {data['balance']} монет, Ребирты: {data['rebirth\_count']}\n"

bot.send\_message(message.chat.id, users\_list)

admin\_panel(message)

elif message.text == '⏪ Меню':

show\_main\_menu(message)

else:

bot.send\_message(message.chat.id, "Неизвестная команда. Возвращаюсь в админ-панель.")

admin\_panel(message)

def add\_balance\_to\_user(message):

try:

target\_user\_id = int(message.text)

except ValueError:

bot.send\_message(message.chat.id, "Некорректный ID пользователя. Возвращаюсь в админ-панель.")

admin\_panel(message)

return

with user\_data\_lock:

if target\_user\_id in user\_data:

bot.send\_message(message.chat.id, f"Введите сумму, которую вы хотите добавить пользователю {target\_user\_id}:")

bot.register\_next\_step\_handler(message, process\_add\_balance, target\_user\_id)

else:

bot.send\_message(message.chat.id, "Пользователь не найден. Возвращаюсь в админ-панель.")

admin\_panel(message)

def process\_add\_balance(message, target\_user\_id):

try:

amount = int(message.text)

if amount <= 0:

bot.send\_message(message.chat.id, "Сумма должна быть положительной.")

admin\_panel(message)

return

except ValueError:

bot.send\_message(message.chat.id, "Некорректная сумма. Возвращаюсь в админ-панель.")

admin\_panel(message)

return

with user\_data\_lock:

user\_data[target\_user\_id]['balance'] += amount

bot.send\_message(message.chat.id, f"Пользователю {target\_user\_id} добавлено {amount} монет.")

try:

bot.send\_message(target\_user\_id, f"Администратор добавил вам {amount} монет.")

except Exception:

pass # Если пользователь не найден или не может получить сообщение

admin\_panel(message)

def remove\_balance\_from\_user(message):

try:

target\_user\_id = int(message.text)

except ValueError:

bot.send\_message(message.chat.id, "Некорректный ID пользователя. Возвращаюсь в админ-панель.")

admin\_panel(message)

return

with user\_data\_lock:

if target\_user\_id in user\_data:

bot.send\_message(message.chat.id, f"Введите сумму, которую вы хотите убрать у пользователя {target\_user\_id}:")

bot.register\_next\_step\_handler(message, process\_remove\_balance, target\_user\_id)

else:

bot.send\_message(message.chat.id, "Пользователь не найден. Возвращаюсь в админ-панель.")

admin\_panel(message)

def process\_remove\_balance(message, target\_user\_id):

try:

amount = int(message.text)

if amount <= 0:

bot.send\_message(message.chat.id, "Сумма должна быть положительной.")

admin\_panel(message)

return

except ValueError:

bot.send\_message(message.chat.id, "Некорректная сумма. Возвращаюсь в админ-панель.")

admin\_panel(message)

return

with user\_data\_lock:

if user\_data[target\_user\_id]['balance'] < amount:

bot.send\_message(message.chat.id, f"У пользователя {target\_user\_id} недостаточно средств. Текущий баланс: {user\_data[target\_user\_id]['balance']} монет.")

else:

user\_data[target\_user\_id]['balance'] -= amount

bot.send\_message(message.chat.id, f"У пользователя {target\_user\_id} убрано {amount} монет.")

try:

bot.send\_message(target\_user\_id, f"Администратор убрал у вас {amount} монет.")

except Exception:

pass # Если пользователь не найден или не может получить сообщение

admin\_panel(message)

def delete\_user(message):

try:

target\_user\_id = int(message.text)

except ValueError:

bot.send\_message(message.chat.id, "Некорректный ID пользователя. Возвращаюсь в админ-панель.")

admin\_panel(message)

return

with user\_data\_lock:

if target\_user\_id in user\_data:

del user\_data[target\_user\_id]

bot.send\_message(message.chat.id, f"Пользователь {target\_user\_id} удалён из бота.")

try:

bot.send\_message(target\_user\_id, "Ваш профиль был удалён администратором.")

except Exception:

pass # Если пользователь не найден или не может получить сообщение

else:

bot.send\_message(message.chat.id, "Пользователь не найден. Возвращаюсь в админ-панель.")

admin\_panel(message)

# Основной обработчик сообщений

@bot.message\_handler(func=lambda message: True)

def main\_handler(message):

user\_id = message.from\_user.id

with user\_data\_lock:

if user\_id not in user\_data:

# Инициализируем пользователя, если он не был инициализирован ранее

user\_data[user\_id] = {

'current\_planet': 'Earth',

'purchased\_planets': ['Earth'],

'pickaxe\_level': {'Earth': 1},

'backpack\_level': {'Earth': 1},

'furnace\_level': {'Earth': 1},

'mine\_level': {'Earth': 1},

'balance': 0,

'ores': {'Earth': {}, 'Moon': {}, 'Mars': {}},

'smelting': {'Earth': {}, 'Moon': {}, 'Mars': {}},

'coal': {'Earth': 0, 'Moon': 0, 'Mars': 0},

'is\_mining': {'Earth': False, 'Moon': False, 'Mars': False},

'purchased\_pickaxes': {'Earth': [1], 'Moon': [], 'Mars': []},

'purchased\_backpacks': {'Earth': [1], 'Moon': [], 'Mars': []},

'purchased\_furnaces': {'Earth': [1], 'Moon': [], 'Mars': []},

'purchased\_mines': {'Earth': [1], 'Moon': [], 'Mars': []},

'ingots': {'Earth': {}, 'Moon': {}, 'Mars': {}},

'rebirth\_count': 0,

'multiplier': 1,

'gamepasses': {},

'clicker\_last\_click': 0,

'clicker\_balance': 0,

'miners': {}

}

current\_planet = user\_data[user\_id]['current\_planet']

# Обработка различных команд и кнопок

if message.text == '⛏ Добывать руду':

start\_mining(message)

elif message.text == '🔥 Печь':

furnace\_menu(message)

elif message.text == '💰 Продать руду':

sell\_ores(message)

elif message.text == '🛒 Магазин':

shop\_menu(message)

elif message.text == '💎 Баланс':

show\_balance(message)

elif message.text == '🎒 Мой портфель':

show\_inventory(message)

elif message.text == 'Кейсы':

show\_cases\_menu(message)

elif message.text == '❓ Помощь':

show\_help(message)

elif message.text == '⏪ Меню':

show\_main\_menu(message)

elif message.text == '➕ Добавить уголь':

add\_coal(message)

elif message.text == '⚒ Переплавить руду':

smelt\_ore(message)

elif message.text in ['Купить Марс', 'Купить Луну']:

# Эти покупки обрабатываются через register\_next\_step\_handler в buy\_planets\_menu

# Ничего не делаем здесь, чтобы избежать попадания в общий обработчик

pass

elif message.text.startswith('Купить ') and not message.text.startswith('Купить ребирт') and message.text != 'Купить планеты' and message.text not in ['Купить Марс', 'Купить Луну']:

item\_type = message.text.split('Купить ')[1].lower()

if item\_type == 'кирку':

buy\_pickaxe(message)

elif item\_type == 'рюкзак':

buy\_backpack(message)

elif item\_type == 'печь':

buy\_furnace(message)

elif item\_type == 'шахту':

buy\_mine(message)

elif item\_type == 'кейсы':

show\_cases\_menu(message)

else:

bot.send\_message(message.chat.id, "Неизвестный тип товара. Используйте кнопки магазина.")

elif message.text.startswith('Купить ребирт'):

buy\_rebirth(message)

elif message.text == '🔴 Админ панель 🔴':

admin\_panel(message)

elif message.text == '🔄 Переключиться на планету':

switch\_planet\_menu(message)

elif message.text == 'Купить планеты':

buy\_planets\_menu(message)

elif message.web\_app\_data:

handle\_web\_app\_data(message)

else:

bot.send\_message(message.chat.id, "Неизвестная команда. Используйте кнопку ⏪ Меню для возврата в главное меню.")

# Обработка данных из Web App

def handle\_web\_app\_data(message):

user\_id = message.from\_user.id

data = message.web\_app\_data.data # JSON строка

try:

data\_json = json.loads(data)

action = data\_json.get('action')

if action == 'purchase\_gamepass':

gamepass\_key = data\_json.get('gamepass\_key')

purchase\_gamepass(user\_id, gamepass\_key, message)

elif action == 'purchase\_click':

purchase\_clicker(user\_id, message)

elif action == 'purchase\_miner':

miner\_key = data\_json.get('miner\_key')

purchase\_miner(user\_id, miner\_key, message)

elif action == 'clicker\_action':

clicker\_action(user\_id, message)

else:

bot.send\_message(message.chat.id, "Неизвестное действие из Web App.")

except Exception as e:

bot.send\_message(message.chat.id, f"Ошибка обработки данных из Web App: {e}")

# Функция покупки геймпасса

def purchase\_gamepass(user\_id, gamepass\_key, message):

with user\_data\_lock:

if gamepass\_key not in gamepasses:

bot.send\_message(message.chat.id, "Некорректный геймпасс.")

return

if gamepass\_key in user\_data[user\_id]['gamepasses']:

bot.send\_message(message.chat.id, "Вы уже приобрели этот геймпасс.")

return

gamepass = gamepasses[gamepass\_key]

price = gamepass['price']

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки {gamepass['name']}. Нужно {price} монет.")

return

# Покупка геймпасса

user\_data[user\_id]['balance'] -= price

user\_data[user\_id]['gamepasses'][gamepass\_key] = True

bot.send\_message(message.chat.id, f"Вы успешно приобрели геймпасс '{gamepass['name']}'.")

# Функция покупки кликера

def purchase\_clicker(user\_id, message):

with user\_data\_lock:

# Допустим, кликер активируется бесплатно или за определенную цену

# Здесь можно добавить цену, если требуется

if 'clicker\_active' not in user\_data[user\_id]:

user\_data[user\_id]['clicker\_active'] = False

if user\_data[user\_id]['clicker\_active']:

bot.send\_message(message.chat.id, "Ваш кликер уже активен.")

return

# Активируем кликер

user\_data[user\_id]['clicker\_active'] = True

user\_data[user\_id]['clicker\_balance'] = 0

bot.send\_message(message.chat.id, "Кликер активирован! Нажимайте на кнопку на сайте для получения монет.")

# Запуск потока для автоматического начисления монет каждые 2 секунды при кликах

threading.Thread(target=clicker\_task, args=(user\_id,)).start()

def clicker\_task(user\_id):

while True:

with user\_data\_lock:

if not user\_data[user\_id].get('clicker\_active', False):

break

last\_click = user\_data[user\_id].get('clicker\_last\_click', 0)

current\_time = time.time()

if current\_time - last\_click >= clicker['cooldown']:

# Проверяем, есть ли клики

if user\_data[user\_id]['clicker\_balance'] > 0:

user\_data[user\_id]['balance'] += clicker['reward'] \* user\_data[user\_id]['multiplier']

user\_data[user\_id]['clicker\_balance'] -= 1

try:

bot.send\_message(user\_id, f"Ваш кликер добавил {clicker['reward'] \* user\_data[user\_id]['multiplier']} монет на ваш баланс.")

except Exception:

pass

time.sleep(1)

# Функция покупки майнера

def purchase\_miner(user\_id, miner\_name, message):

with user\_data\_lock:

# Найти майнера по имени

miner = next((m for m in miners['Mars'] if m['name'] == miner\_name), None)

if not miner:

bot.send\_message(message.chat.id, "Некорректный майнер.")

return

price = miner['price']

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки майнера '{miner['name']}'. Нужно {price} монет.")

return

# Покупка майнера

user\_data[user\_id]['balance'] -= price

if miner['name'] not in user\_data[user\_id]['miners']:

user\_data[user\_id]['miners'][miner['name']] = {'count': 1, 'next\_reward\_time': time.time() + miner['interval']}

else:

user\_data[user\_id]['miners'][miner['name']]['count'] += 1

user\_data[user\_id]['miners'][miner['name']]['next\_reward\_time'] = time.time() + miner['interval']

bot.send\_message(message.chat.id, f"Вы успешно приобрели майнера '{miner['name']}'. Он будет добавлять {miner['reward']} монет каждые {miner['interval'] / 3600} часов.")

# Запуск майнера в отдельном потоке

threading.Thread(target=miner\_task, args=(user\_id, miner)).start()

# Задача майнера

def miner\_task(user\_id, miner):

while True:

with user\_data\_lock:

miner\_info = user\_data[user\_id]['miners'].get(miner['name'])

if not miner\_info:

break # Майнер был удалён

current\_time = time.time()

if current\_time >= miner\_info['next\_reward\_time']:

reward = miner['reward'] \* user\_data[user\_id]['multiplier']

user\_data[user\_id]['balance'] += reward

miner\_info['next\_reward\_time'] = current\_time + miner['interval']

try:

bot.send\_message(user\_id, f"Ваш майнер '{miner['name']}' добавил {reward} монет на ваш баланс.")

except Exception:

pass

time.sleep(60) # Проверять каждую минуту

# Функция начала добычи

def start\_mining(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

if user\_data[user\_id].get('is\_mining', {}).get(current\_planet, False):

bot.send\_message(message.chat.id, "Не флуди! Подождите, пока добыча руды завершится.")

return

user\_data[user\_id]['is\_mining'][current\_planet] = True # Устанавливаем флаг добычи

bot.send\_message(message.chat.id, f"Вы начали добычу руды на планете {current\_planet}...")

threading.Thread(target=mine\_ore, args=(message,)).start()

# Функция добычи руды с учетом геймпассов

def mine\_ore(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

time.sleep(3) # Добыча руды занимает 3 секунды

with user\_data\_lock:

pickaxe\_level = user\_data[user\_id]['pickaxe\_level'][current\_planet]

mine\_level = user\_data[user\_id]['mine\_level'][current\_planet]

backpack\_level = user\_data[user\_id]['backpack\_level'][current\_planet]

backpack\_capacity = backpacks[current\_planet][backpack\_level - 1]['capacity']

pickaxe\_luck = pickaxes[current\_planet][pickaxe\_level - 1]['luck']

# Проверка наличия геймпассов

double\_luck = user\_data[user\_id]['gamepasses'].get('double\_luck', False)

ultra\_luck = user\_data[user\_id]['gamepasses'].get('ultra\_luck', False)

double\_ore = user\_data[user\_id]['gamepasses'].get('double\_ore', False)

# Определение количества добытой руды

if pickaxe\_level <= 3:

amount = random.randint(1, 3)

elif 3 < pickaxe\_level <= 7:

amount = random.randint(1, 5)

else:

amount = random.randint(2, 7)

# Применение геймпассов

if double\_luck:

pickaxe\_luck \*= 2

if ultra\_luck:

pickaxe\_luck \*= 5

# Получение списка руд на текущей планете

ore\_list = ores[current\_planet]

base\_weights = [ore['weight'] for ore in ore\_list]

# Увеличение веса более ценных руд на уровне шахты

total\_ores = len(ore\_list)

modified\_weights = []

for idx, ore in enumerate(ore\_list):

# Определяем, насколько ценной является руда

# Чем выше индекс, тем ценнее руда

# Расчитываем долю ценности

value\_fraction = (idx + 1) / total\_ores # от 1/total до 1

# Определяем модификатор на основе уровня шахты

# Например, каждый уровень шахты добавляет 5% к весу пропорционально ценности руды

modifier = 1 + (mine\_level \* 0.05 \* value\_fraction)

modified\_weight = ore['weight'] \* modifier

modified\_weights.append(modified\_weight)

# Выбор руды на основе модифицированных весов

mined\_ore = random.choices(ore\_list, weights=modified\_weights, k=1)[0]['name']

# Применение геймпасса двойной руды

if double\_ore:

amount \*= 2

with user\_data\_lock:

current\_ores = sum(user\_data[user\_id]['ores'][current\_planet].values())

if current\_ores + amount > backpack\_capacity:

amount = backpack\_capacity - current\_ores

if amount <= 0:

bot.send\_message(message.chat.id, "Ваш портфель переполнен!")

user\_data[user\_id]['is\_mining'][current\_planet] = False

return

user\_data[user\_id]['ores'][current\_planet][mined\_ore] = user\_data[user\_id]['ores'][current\_planet].get(mined\_ore, 0) + amount

bot.send\_message(message.chat.id, f"Вы добыли {amount} x {mined\_ore} на планете {current\_planet}.")

with user\_data\_lock:

user\_data[user\_id]['is\_mining'][current\_planet] = False

# Меню печи

def furnace\_menu(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

add\_coal\_button = types.KeyboardButton('➕ Добавить уголь')

smelt\_ore\_button = types.KeyboardButton('⚒ Переплавить руду')

back\_button = types.KeyboardButton('⏪ Меню')

markup.add(add\_coal\_button, smelt\_ore\_button)

markup.add(back\_button)

with user\_data\_lock:

ores\_available = user\_data[user\_id]['ores'][current\_planet].copy()

if ores\_available:

ores\_list = "Доступные для переплавки руды:\n"

for ore, amount in ores\_available.items():

ores\_list += f"{ore}: {amount} шт.\n"

ores\_list += "\nВыберите действие:"

bot.send\_message(message.chat.id, ores\_list, reply\_markup=markup)

else:

bot.send\_message(message.chat.id, f"В вашем портфеле на планете {current\_planet} нет руды для переплавки.", reply\_markup=markup)

# Добавить уголь

def add\_coal(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

bot.send\_message(message.chat.id, f"Сколько угля вы хотите добавить в печь на планете {current\_planet}?")

bot.register\_next\_step\_handler(message, process\_add\_coal)

def process\_add\_coal(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

try:

amount = int(message.text)

if amount <= 0:

bot.send\_message(message.chat.id, "Количество должно быть положительным.")

return

except ValueError:

bot.send\_message(message.chat.id, "Пожалуйста, введите корректное число.")

return

with user\_data\_lock:

user\_data[user\_id]['coal'][current\_planet] += amount

bot.send\_message(message.chat.id, f"Вы добавили {amount} угля в печь на планете {current\_planet}.")

# Переплавить руду

def smelt\_ore(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

ores\_user = user\_data[user\_id]['ores'][current\_planet].copy()

if not ores\_user:

bot.send\_message(message.chat.id, "У вас нет руды для переплавки.")

return

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

for ore, amount in ores\_user.items():

markup.add(types.KeyboardButton(f"{ore} {amount} шт."))

back\_button = types.KeyboardButton('⏪ Меню')

markup.add(back\_button)

bot.send\_message(message.chat.id, f"Выберите руду для переплавки на планете {current\_planet}:", reply\_markup=markup)

bot.register\_next\_step\_handler(message, process\_smelt\_ore)

def process\_smelt\_ore(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

selected\_text = message.text

if selected\_text == '⏪ Меню':

show\_main\_menu(message)

return

# Извлекаем название руды из текста кнопки, например, "Железная руда 3 шт." -> "Железная руда"

if ' шт.' in selected\_text:

ore = selected\_text.rsplit(' ', 2)[0]

else:

ore = selected\_text # На всякий случай

with user\_data\_lock:

if ore not in user\_data[user\_id]['ores'][current\_planet]:

bot.send\_message(message.chat.id, "У вас нет такой руды.")

return

if user\_data[user\_id]['coal'][current\_planet] <= 0:

bot.send\_message(message.chat.id, "У вас нет угля для переплавки.")

return

bot.send\_message(message.chat.id, f"Сколько '{ore}' вы хотите переплавить на планете {current\_planet}?")

with user\_data\_lock:

user\_data[user\_id]['selected\_ore'] = ore

bot.register\_next\_step\_handler(message, process\_smelting\_amount)

def process\_smelting\_amount(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

try:

amount = int(message.text)

if amount <= 0:

bot.send\_message(message.chat.id, "Количество должно быть положительным.")

return

except ValueError:

bot.send\_message(message.chat.id, "Пожалуйста, введите корректное число.")

return

with user\_data\_lock:

ore = user\_data[user\_id].get('selected\_ore')

if not ore:

bot.send\_message(message.chat.id, "Произошла ошибка. Пожалуйста, попробуйте снова.")

return

available\_amount = user\_data[user\_id]['ores'][current\_planet].get(ore, 0)

if amount > available\_amount:

bot.send\_message(message.chat.id, f"У вас нет столько '{ore}'. Доступно для переплавки: {available\_amount}.")

return

if user\_data[user\_id]['coal'][current\_planet] <= 0:

bot.send\_message(message.chat.id, "У вас нет угля для переплавки.")

return

furnace\_level = user\_data[user\_id]['furnace\_level'][current\_planet]

smelting\_speed = furnaces[current\_planet][furnace\_level - 1]['smelt\_capacity']

if amount > smelting\_speed:

bot.send\_message(message.chat.id, f"Ваша печь может переплавить за раз не более {smelting\_speed} единиц руды.")

amount = smelting\_speed

# Проверка геймпасса моментальной переплавки

instant\_smelting = user\_data[user\_id]['gamepasses'].get('instant\_smelting', False)

smelt\_time = 0 if instant\_smelting else 3 # Если геймпасс активен, время переплавки 0

# Обновляем данные пользователя

user\_data[user\_id]['ores'][current\_planet][ore] -= amount

if user\_data[user\_id]['ores'][current\_planet][ore] == 0:

del user\_data[user\_id]['ores'][current\_planet][ore]

user\_data[user\_id]['smelting'][current\_planet][ore] = user\_data[user\_id]['smelting'][current\_planet].get(ore, 0) + amount

user\_data[user\_id]['coal'][current\_planet] -= 1

if smelt\_time > 0:

bot.send\_message(message.chat.id, f"Начата переплавка {amount} x {ore} на планете {current\_planet}. Это займет {smelt\_time} секунд.")

else:

bot.send\_message(message.chat.id, f"Переплавка {amount} x {ore} на планете {current\_planet} начата моментально.")

threading.Thread(target=finish\_smelting, args=(message, ore, amount, smelt\_time)).start()

def finish\_smelting(message, ore, amount, smelt\_time):

try:

if smelt\_time > 0:

time.sleep(smelt\_time)

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

user\_data[user\_id]['smelting'][current\_planet][ore] -= amount

if user\_data[user\_id]['smelting'][current\_planet][ore] == 0:

del user\_data[user\_id]['smelting'][current\_planet][ore]

user\_data[user\_id]['ingots'][current\_planet][ore] = user\_data[user\_id]['ingots'][current\_planet].get(ore, 0) + amount

except Exception as e:

bot.send\_message(message.chat.id, f"Ошибка при переплавке: {e}")

with user\_data\_lock:

user\_data[user\_id]['is\_mining'][current\_planet] = False

admin\_panel(message)

finally:

if smelt\_time > 0:

bot.send\_message(message.chat.id, f"Переплавка {amount} x {ore} на планете {current\_planet} завершена.")

# Автоматический возврат в меню печи после завершения переплавки

furnace\_menu(message)

# Продать руду

def sell\_ores(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

ingots = user\_data[user\_id]['ingots'][current\_planet].copy()

if not ingots:

bot.send\_message(message.chat.id, "У вас нет переплавленной руды для продажи.")

return

total\_earnings = 0

earnings\_details = ""

earth\_ores = [ore\_item['name'] for ore\_item in ores['Earth']]

with user\_data\_lock:

for ore, amount in ingots.items():

if ore in earth\_ores:

base\_price = 10 \* (earth\_ores.index(ore) + 1)

else:

base\_price = 10 # Можно настроить по желанию

if current\_planet == 'Earth':

multiplier = 1

elif current\_planet == 'Moon':

multiplier = 1.5

elif current\_planet == 'Mars':

multiplier = 2

earnings = int(amount \* base\_price \* multiplier \* user\_data[user\_id]['multiplier'])

total\_earnings += earnings

earnings\_details += f"{ore}: {amount} x {base\_price} x {multiplier} x {user\_data[user\_id]['multiplier']} = {earnings} монет\n"

user\_data[user\_id]['balance'] += total\_earnings

user\_data[user\_id]['ingots'][current\_planet] = {}

sell\_message = f"Вы продали все переплавленные руды на планете {current\_planet} и заработали {total\_earnings} монет.\n\nДетали:\n{earnings\_details}"

bot.send\_message(message.chat.id, sell\_message)

# Показать баланс

def show\_balance(message):

user\_id = message.from\_user.id

with user\_data\_lock:

balance = user\_data[user\_id]['balance']

multiplier = user\_data[user\_id]['multiplier']

bot.send\_message(message.chat.id, f"Ваш баланс: {balance} монет.\nМножитель монет: ×{multiplier}")

# Показать портфель

def show\_inventory(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

ores\_in\_inventory = user\_data[user\_id]['ores'][current\_planet].copy()

ingots = user\_data[user\_id]['ingots'][current\_planet].copy()

if not ores\_in\_inventory and not ingots:

bot.send\_message(message.chat.id, f"Ваш портфель на планете {current\_planet} пуст.")

else:

inventory = f"Портфель содержит:\n"

if ores\_in\_inventory:

inventory += "\n\*Не переплавленной руды:\*\n"

for ore, amount in ores\_in\_inventory.items():

inventory += f"{ore}: {amount} шт.\n"

else:

inventory += "\n\*Не переплавленной руды:\* Нет\n"

if ingots:

inventory += "\n\*Переплавленной руды:\*\n"

for ore, amount in ingots.items():

inventory += f"{ore}: {amount} шт.\n"

else:

inventory += "\n\*Переплавленной руды:\* Нет\n"

bot.send\_message(message.chat.id, inventory, parse\_mode='Markdown')

# Меню магазина с отдельной вкладкой для покупки планет и ребиртов

def shop\_menu(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

rebirth\_available = current\_planet == 'Mars' and user\_data[user\_id]['rebirth\_count'] < len(rebirths)

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

buy\_pickaxe\_button = types.KeyboardButton('Купить кирку')

buy\_backpack\_button = types.KeyboardButton('Купить рюкзак')

buy\_furnace\_button = types.KeyboardButton('Купить печь')

buy\_mine\_button = types.KeyboardButton('Купить шахту')

buy\_cases\_button = types.KeyboardButton('Кейсы')

buy\_rebirth\_button = types.KeyboardButton('Купить ребирт') if rebirth\_available else None

buy\_planets\_button = types.KeyboardButton('Купить планеты')

back\_button = types.KeyboardButton('⏪ Меню')

# Добавляем кнопки в магазин

buttons = [buy\_pickaxe\_button, buy\_backpack\_button]

buttons += [buy\_furnace\_button, buy\_mine\_button]

buttons += [buy\_cases\_button]

if buy\_rebirth\_button:

buttons.append(buy\_rebirth\_button)

buttons.append(buy\_planets\_button)

buttons.append(back\_button)

markup.add(\*buttons)

bot.send\_message(message.chat.id, f"Магазин на планете {current\_planet}", reply\_markup=markup)

# Функция покупки и переключения оборудования

def buy\_pickaxe(message):

buy\_item(message, 'pickaxe')

def buy\_backpack(message):

buy\_item(message, 'backpack')

def buy\_furnace(message):

buy\_item(message, 'furnace')

def buy\_mine(message):

buy\_item(message, 'mine')

def buy\_item(message, item\_type):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

if item\_type == 'pickaxe':

items = pickaxes[current\_planet]

user\_inventory = user\_data[user\_id]['purchased\_pickaxes'][current\_planet]

current\_level = user\_data[user\_id]['pickaxe\_level'][current\_planet]

price\_multiplier = 450 if current\_planet == 'Earth' else (675 if current\_planet == 'Moon' else 900) # Увеличены цены на 1.5x

elif item\_type == 'backpack':

items = backpacks[current\_planet]

user\_inventory = user\_data[user\_id]['purchased\_backpacks'][current\_planet]

current\_level = user\_data[user\_id]['backpack\_level'][current\_planet]

price\_multiplier = 450 if current\_planet == 'Earth' else (675 if current\_planet == 'Moon' else 900) # Увеличены цены на 1.5x

elif item\_type == 'furnace':

items = furnaces[current\_planet]

user\_inventory = user\_data[user\_id]['purchased\_furnaces'][current\_planet]

current\_level = user\_data[user\_id]['furnace\_level'][current\_planet]

price\_multiplier = 600 if current\_planet == 'Earth' else (900 if current\_planet == 'Moon' else 1200) # Увеличены цены на 1.5x

elif item\_type == 'mine':

items = mines[current\_planet]

user\_inventory = user\_data[user\_id]['purchased\_mines'][current\_planet]

current\_level = user\_data[user\_id]['mine\_level'][current\_planet]

price\_multiplier = 750 if current\_planet == 'Earth' else (1125 if current\_planet == 'Moon' else 1500) # Увеличены цены на 1.5x

else:

return

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

for idx, item in enumerate(items):

level = idx + 1

if item\_type == 'mine':

item\_name = item

price = level \* price\_multiplier

else:

item\_name = item['name']

price = level \* price\_multiplier

with user\_data\_lock:

if level in user\_inventory:

label = f"{item\_name} (Куплено)"

else:

if item\_type == 'pickaxe':

label = f"{item\_name} (+{item['luck']}% удачи) - {price} монет"

elif item\_type == 'backpack':

label = f"{item\_name} (вместимость: {item['capacity']}) - {price} монет"

elif item\_type == 'furnace':

label = f"{item\_name} (переплавка: {item['smelt\_capacity']} руд) - {price} монет"

elif item\_type == 'mine':

label = f"{item\_name} - {price} монет"

markup.add(types.KeyboardButton(label))

back\_button = types.KeyboardButton('🛒 Продолжить покупки')

exit\_button = types.KeyboardButton('⏪ Меню')

markup.add(back\_button, exit\_button)

bot.send\_message(message.chat.id, f"Выберите {item\_type} на планете {current\_planet}:", reply\_markup=markup)

bot.register\_next\_step\_handler(message, process\_buy\_item, item\_type, items, price\_multiplier)

def process\_buy\_item(message, item\_type, items, price\_multiplier):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

selected = message.text

if selected == '⏪ Меню':

show\_main\_menu(message)

return

elif selected == '🛒 Продолжить покупки':

shop\_menu(message)

return

with user\_data\_lock:

user\_inventory = user\_data[user\_id][f'purchased\_{item\_type}s'][current\_planet].copy()

for idx, item in enumerate(items):

level = idx + 1

if item\_type == 'mine':

item\_name = item

price = level \* price\_multiplier

if level in user\_inventory:

option = f"{item\_name} (Куплено)"

if selected == option:

user\_data[user\_id][f'{item\_type}\_level'][current\_planet] = level

bot.send\_message(message.chat.id, f"Вы переключились на {item\_name} на планете {current\_planet}.")

return

else:

option = f"{item\_name} - {price} монет"

if selected == option:

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки {item\_name}. Нужно {price} монет.")

return

user\_data[user\_id]['balance'] -= price

user\_data[user\_id][f'{item\_type}\_level'][current\_planet] = level

user\_data[user\_id][f'purchased\_{item\_type}s'][current\_planet].append(level)

bot.send\_message(message.chat.id, f"Вы купили {item\_name} за {price} монет на планете {current\_planet}.")

return

else:

item\_name = item['name']

price = level \* price\_multiplier

if level in user\_inventory:

option = f"{item\_name} (Куплено)"

if selected == option:

user\_data[user\_id][f'{item\_type}\_level'][current\_planet] = level

bot.send\_message(message.chat.id, f"Вы переключились на {item\_name} на планете {current\_planet}.")

return

else:

if item\_type == 'pickaxe':

option = f"{item\_name} (+{item['luck']}% удачи) - {price} монет"

elif item\_type == 'backpack':

option = f"{item\_name} (вместимость: {item['capacity']}) - {price} монет"

elif item\_type == 'furnace':

option = f"{item\_name} (переплавка: {item['smelt\_capacity']} руд) - {price} монет"

elif item\_type == 'mine':

option = f"{item\_name} - {price} монет"

if selected == option:

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки {item\_name}. Нужно {price} монет.")

return

user\_data[user\_id]['balance'] -= price

user\_data[user\_id][f'{item\_type}\_level'][current\_planet] = level

user\_data[user\_id][f'purchased\_{item\_type}s'][current\_planet].append(level)

bot.send\_message(message.chat.id, f"Вы купили {item\_name} за {price} монет на планете {current\_planet}.")

return

bot.send\_message(message.chat.id, "Некорректный выбор. Попробуйте снова.")

buy\_item(message, item\_type)

# Функция покупки планет

def buy\_planet(message, planet\_name, price):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

if planet\_name in user\_data[user\_id]['purchased\_planets']:

bot.send\_message(message.chat.id, f"Вы уже приобрели планету {planet\_name}.")

return

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки планеты {planet\_name}. Нужно {price} монет.")

return

user\_data[user\_id]['balance'] -= price

user\_data[user\_id]['purchased\_planets'].append(planet\_name)

# Инициализация оборудования на новой планете

user\_data[user\_id]['pickaxe\_level'][planet\_name] = 1

user\_data[user\_id]['backpack\_level'][planet\_name] = 1

user\_data[user\_id]['furnace\_level'][planet\_name] = 1

user\_data[user\_id]['mine\_level'][planet\_name] = 1

user\_data[user\_id]['purchased\_pickaxes'][planet\_name] = [1]

user\_data[user\_id]['purchased\_backpacks'][planet\_name] = [1]

user\_data[user\_id]['purchased\_furnaces'][planet\_name] = [1]

user\_data[user\_id]['purchased\_mines'][planet\_name] = [1]

user\_data[user\_id]['ingots'][planet\_name] = {}

user\_data[user\_id]['ores'][planet\_name] = {}

user\_data[user\_id]['smelting'][planet\_name] = {}

user\_data[user\_id]['coal'][planet\_name] = 0

user\_data[user\_id]['is\_mining'][planet\_name] = False

# Майнеры и кликер инициализируются при покупке на сайте

bot.send\_message(message.chat.id, f"Поздравляем! Вы приобрели планету {planet\_name} за {price} монет.")

bot.send\_message(message.chat.id, f"Теперь вы можете переключаться между планетами: {', '.join(user\_data[user\_id]['purchased\_planets'])}.")

# Меню для покупки планет

def buy\_planets\_menu(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

available\_planets = [planet for planet in ['Moon', 'Mars'] if planet not in user\_data[user\_id]['purchased\_planets']]

if not available\_planets:

bot.send\_message(message.chat.id, "У вас уже куплены все доступные планеты.", reply\_markup=types.ReplyKeyboardRemove())

shop\_menu(message)

return

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

for planet in available\_planets:

if planet == 'Moon':

label = 'Купить Луну'

else:

label = 'Купить Марс'

markup.add(types.KeyboardButton(label))

back\_button = types.KeyboardButton('🛒 Продолжить покупки')

exit\_button = types.KeyboardButton('⏪ Меню')

markup.add(back\_button, exit\_button)

bot.send\_message(message.chat.id, "Выберите планету для покупки:", reply\_markup=markup)

bot.register\_next\_step\_handler(message, process\_buy\_planet)

def process\_buy\_planet(message):

user\_id = message.from\_user.id

selected = message.text

if selected == '⏪ Меню':

show\_main\_menu(message)

return

elif selected == '🛒 Продолжить покупки':

shop\_menu(message)

return

elif selected == 'Купить Луну':

buy\_planet(message, 'Moon', 150000) # Цена Луна: 150,000 монет

elif selected == 'Купить Марс':

buy\_planet(message, 'Mars', 350000) # Цена Марс: 350,000 монет

else:

bot.send\_message(message.chat.id, "Некорректный выбор. Возвращаюсь в меню покупки планет.")

buy\_planets\_menu(message)

# Функция переключения планеты

def switch\_planet\_menu(message):

user\_id = message.from\_user.id

with user\_data\_lock:

purchased\_planets = user\_data[user\_id]['purchased\_planets'].copy()

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

for planet in purchased\_planets:

if planet == 'Earth':

label = 'Земля'

elif planet == 'Moon':

label = 'Луна'

elif planet == 'Mars':

label = 'Марс'

else:

label = planet

markup.add(types.KeyboardButton(label))

back\_button = types.KeyboardButton('⏪ Меню')

markup.add(back\_button)

bot.send\_message(message.chat.id, "Выберите планету для переключения:", reply\_markup=markup)

bot.register\_next\_step\_handler(message, process\_switch\_planet)

def process\_switch\_planet(message):

user\_id = message.from\_user.id

selected = message.text

if selected == '⏪ Меню':

show\_main\_menu(message)

return

planet\_mapping = {'Земля': 'Earth', 'Луна': 'Moon', 'Марс': 'Mars'}

if selected in planet\_mapping and planet\_mapping[selected] in user\_data[user\_id]['purchased\_planets']:

with user\_data\_lock:

user\_data[user\_id]['current\_planet'] = planet\_mapping[selected]

bot.send\_message(message.chat.id, f"Вы переключились на планету {selected}.")

else:

bot.send\_message(message.chat.id, "Некорректный выбор. Возвращаюсь в меню переключения планет.")

show\_main\_menu(message)

# 🧰 Кейсы: Функции для отображения и открытия кейсов без смайликов

def show\_cases\_menu(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

available\_cases = cases[current\_planet].copy()

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

for case in available\_cases:

label = f"{case['name']} - {case['price']} монет"

markup.add(types.KeyboardButton(label))

back\_button = types.KeyboardButton('⏪ Меню')

markup.add(back\_button)

bot.send\_message(message.chat.id, f"Выберите кейс для покупки и открытия на планете {current\_planet}:", reply\_markup=markup)

bot.register\_next\_step\_handler(message, process\_case\_selection)

def process\_case\_selection(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

selected = message.text

if selected == '⏪ Меню':

show\_main\_menu(message)

return

with user\_data\_lock:

selected\_case = next((case for case in cases[current\_planet] if f"{case['name']} - {case['price']} монет" == selected), None)

if not selected\_case:

bot.send\_message(message.chat.id, "Некорректный выбор. Возвращаюсь в меню кейсов.")

show\_cases\_menu(message)

return

with user\_data\_lock:

if user\_data[user\_id]['balance'] < selected\_case['price']:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки {selected\_case['name']}. Нужно {selected\_case['price']} монет.")

show\_cases\_menu(message)

return

user\_data[user\_id]['balance'] -= selected\_case['price']

bot.send\_message(message.chat.id, f"Вы приобрели {selected\_case['name']} за {selected\_case['price']} монет.")

bot.send\_message(message.chat.id, f"Открытие {selected\_case['name']} началось! Подождите 10 секунд...")

# Запуск процесса открытия кейса в отдельном потоке

threading.Thread(target=open\_case, args=(message, selected\_case)).start()

def open\_case(message, case):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

min\_reward = case['min\_reward']

max\_reward = case['max\_reward']

try:

# Имитация кручения чисел (10 секунд)

for i in range(1, 11):

with user\_data\_lock:

if current\_planet == 'Earth':

spinning\_number = random.randint(min\_reward, max\_reward)

elif current\_planet == 'Moon':

spinning\_number = random.randint(int(min\_reward \* 1.2), int(max\_reward \* 1.2)) # Увеличение на 20%

elif current\_planet == 'Mars':

spinning\_number = random.randint(int(min\_reward \* 1.4), int(max\_reward \* 1.4)) # Увеличение на 40%

else:

spinning\_number = random.randint(min\_reward, max\_reward) # По умолчанию

bot.send\_message(message.chat.id, f"🔄 Крутится... {spinning\_number} монет")

time.sleep(1)

# Генерация окончательной награды

with user\_data\_lock:

if current\_planet == 'Earth':

reward = random.randint(min\_reward, max\_reward)

elif current\_planet == 'Moon':

reward = random.randint(int(min\_reward \* 1.2), int(max\_reward \* 1.2))

elif current\_planet == 'Mars':

reward = random.randint(int(min\_reward \* 1.4), int(max\_reward \* 1.4))

else:

reward = random.randint(min\_reward, max\_reward) # По умолчанию

reward \*= user\_data[user\_id]['multiplier']

user\_data[user\_id]['balance'] += reward

bot.send\_message(message.chat.id, f"🎉 Поздравляем! Вы получили {reward} монет из {case['name']} на планете {current\_planet}.")

except Exception as e:

bot.send\_message(message.chat.id, f"Ошибка при открытии кейса: {e}")

finally:

# Автоматический возврат в меню кейсов после открытия

show\_cases\_menu(message)

# Функция покупки ребирта

def buy\_rebirth(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

rebirth\_available = current\_planet == 'Mars' and user\_data[user\_id]['rebirth\_count'] < len(rebirths)

if not rebirth\_available:

if current\_planet != 'Mars':

reason = "Ребирты можно покупать только на планете Марс."

else:

reason = "Вы достигли максимального количества ребиртов."

bot.send\_message(message.chat.id, reason)

return

rebirth = rebirths[user\_data[user\_id]['rebirth\_count']]

price = rebirth['price']

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки ребирта '{rebirth['name']}'. Нужно {price} монет.")

return

# Подтверждение покупки

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

yes\_button = types.KeyboardButton('Да')

no\_button = types.KeyboardButton('Нет')

markup.add(yes\_button, no\_button)

bot.send\_message(message.chat.id, f"Вы уверены, что хотите купить ребирт '{rebirth['name']}' за {price} монет?\nВсе ваши данные будут сброшены, и баланс станет 0, но ваш множитель монет увеличится на ×1.", reply\_markup=markup)

bot.register\_next\_step\_handler(message, confirm\_rebirth\_purchase, rebirth)

def confirm\_rebirth\_purchase(message, rebirth):

user\_id = message.from\_user.id

if message.text.lower() == 'да':

price = rebirth['price']

with user\_data\_lock:

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки ребирта '{rebirth['name']}'. Нужно {price} монет.")

return

user\_data[user\_id]['balance'] -= price

user\_data[user\_id]['rebirth\_count'] += 1

user\_data[user\_id]['multiplier'] += 1

# Сброс всех данных, кроме rebirth\_count, multiplier и gamepasses

user\_data[user\_id]['current\_planet'] = 'Earth' # После сброса игрок начинает с Земли

user\_data[user\_id]['purchased\_planets'] = ['Earth'] # Только Земля как купленная планета

user\_data[user\_id]['pickaxe\_level'] = {'Earth': 1}

user\_data[user\_id]['backpack\_level'] = {'Earth': 1}

user\_data[user\_id]['furnace\_level'] = {'Earth': 1}

user\_data[user\_id]['mine\_level'] = {'Earth': 1}

user\_data[user\_id]['purchased\_pickaxes'] = {'Earth': [1], 'Moon': [], 'Mars': []}

user\_data[user\_id]['purchased\_backpacks'] = {'Earth': [1], 'Moon': [], 'Mars': []}

user\_data[user\_id]['purchased\_furnaces'] = {'Earth': [1], 'Moon': [], 'Mars': []}

user\_data[user\_id]['purchased\_mines'] = {'Earth': [1], 'Moon': [], 'Mars': []}

user\_data[user\_id]['ingots'] = {'Earth': {}, 'Moon': {}, 'Mars': {}}

user\_data[user\_id]['ores'] = {'Earth': {}, 'Moon': {}, 'Mars': {}}

user\_data[user\_id]['smelting'] = {'Earth': {}, 'Moon': {}, 'Mars': {}}

user\_data[user\_id]['coal'] = {'Earth': 0, 'Moon': 0, 'Mars': 0}

user\_data[user\_id]['is\_mining'] = {'Earth': False, 'Moon': False, 'Mars': False}

user\_data[user\_id]['clicker\_last\_click'] = 0

user\_data[user\_id]['clicker\_balance'] = 0

user\_data[user\_id]['miners'] = {}

# Геймпассы сохраняются

bot.send\_message(message.chat.id, f"Поздравляем! Вы успешно совершили ребирт '{rebirth['name']}'. Ваш множитель монет увеличен на ×1.")

else:

bot.send\_message(message.chat.id, "Покупка ребирта отменена.")

show\_main\_menu(message)

# Функция покупки майнера

def purchase\_miner(user\_id, miner\_name, message):

with user\_data\_lock:

# Найти майнера по имени

miner = next((m for m in miners['Mars'] if m['name'] == miner\_name), None)

if not miner:

bot.send\_message(message.chat.id, "Некорректный майнер.")

return

price = miner['price']

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки майнера '{miner['name']}'. Нужно {price} монет.")

return

# Покупка майнера

user\_data[user\_id]['balance'] -= price

if miner['name'] not in user\_data[user\_id]['miners']:

user\_data[user\_id]['miners'][miner['name']] = {'count': 1, 'next\_reward\_time': time.time() + miner['interval']}

else:

user\_data[user\_id]['miners'][miner['name']]['count'] += 1

user\_data[user\_id]['miners'][miner['name']]['next\_reward\_time'] = time.time() + miner['interval']

bot.send\_message(message.chat.id, f"Вы успешно приобрели майнера '{miner['name']}'. Он будет добавлять {miner['reward']} монет каждые {miner['interval'] / 3600} часов.")

# Запуск майнера в отдельном потоке

threading.Thread(target=miner\_task, args=(user\_id, miner)).start()

# Задача майнера

def miner\_task(user\_id, miner):

while True:

with user\_data\_lock:

miner\_info = user\_data[user\_id]['miners'].get(miner['name'])

if not miner\_info:

break # Майнер был удалён

current\_time = time.time()

if current\_time >= miner\_info['next\_reward\_time']:

reward = miner['reward'] \* user\_data[user\_id]['multiplier']

user\_data[user\_id]['balance'] += reward

miner\_info['next\_reward\_time'] = current\_time + miner['interval']

try:

bot.send\_message(user\_id, f"Ваш майнер '{miner['name']}' добавил {reward} монет на ваш баланс.")

except Exception:

pass

time.sleep(60) # Проверять каждую минуту

# Функция начала добычи

def start\_mining(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

if user\_data[user\_id].get('is\_mining', {}).get(current\_planet, False):

bot.send\_message(message.chat.id, "Не флуди! Подождите, пока добыча руды завершится.")

return

user\_data[user\_id]['is\_mining'][current\_planet] = True # Устанавливаем флаг добычи

bot.send\_message(message.chat.id, f"Вы начали добычу руды на планете {current\_planet}...")

threading.Thread(target=mine\_ore, args=(message,)).start()

# Функция добычи руды с учетом геймпассов

def mine\_ore(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

time.sleep(3) # Добыча руды занимает 3 секунды

with user\_data\_lock:

pickaxe\_level = user\_data[user\_id]['pickaxe\_level'][current\_planet]

mine\_level = user\_data[user\_id]['mine\_level'][current\_planet]

backpack\_level = user\_data[user\_id]['backpack\_level'][current\_planet]

backpack\_capacity = backpacks[current\_planet][backpack\_level - 1]['capacity']

pickaxe\_luck = pickaxes[current\_planet][pickaxe\_level - 1]['luck']

# Проверка наличия геймпассов

double\_luck = user\_data[user\_id]['gamepasses'].get('double\_luck', False)

ultra\_luck = user\_data[user\_id]['gamepasses'].get('ultra\_luck', False)

double\_ore = user\_data[user\_id]['gamepasses'].get('double\_ore', False)

# Определение количества добытой руды

if pickaxe\_level <= 3:

amount = random.randint(1, 3)

elif 3 < pickaxe\_level <= 7:

amount = random.randint(1, 5)

else:

amount = random.randint(2, 7)

# Применение геймпассов

if double\_luck:

pickaxe\_luck \*= 2

if ultra\_luck:

pickaxe\_luck \*= 5

# Получение списка руд на текущей планете

ore\_list = ores[current\_planet]

base\_weights = [ore['weight'] for ore in ore\_list]

# Увеличение веса более ценных руд на уровне шахты

total\_ores = len(ore\_list)

modified\_weights = []

for idx, ore in enumerate(ore\_list):

# Определяем, насколько ценной является руда

# Чем выше индекс, тем ценнее руда

# Расчитываем долю ценности

value\_fraction = (idx + 1) / total\_ores # от 1/total до 1

# Определяем модификатор на основе уровня шахты

# Например, каждый уровень шахты добавляет 5% к весу пропорционально ценности руды

modifier = 1 + (mine\_level \* 0.05 \* value\_fraction)

modified\_weight = ore['weight'] \* modifier

modified\_weights.append(modified\_weight)

# Выбор руды на основе модифицированных весов

mined\_ore = random.choices(ore\_list, weights=modified\_weights, k=1)[0]['name']

# Применение геймпасса двойной руды

if double\_ore:

amount \*= 2

with user\_data\_lock:

current\_ores = sum(user\_data[user\_id]['ores'][current\_planet].values())

if current\_ores + amount > backpack\_capacity:

amount = backpack\_capacity - current\_ores

if amount <= 0:

bot.send\_message(message.chat.id, "Ваш портфель переполнен!")

user\_data[user\_id]['is\_mining'][current\_planet] = False

return

user\_data[user\_id]['ores'][current\_planet][mined\_ore] = user\_data[user\_id]['ores'][current\_planet].get(mined\_ore, 0) + amount

bot.send\_message(message.chat.id, f"Вы добыли {amount} x {mined\_ore} на планете {current\_planet}.")

with user\_data\_lock:

user\_data[user\_id]['is\_mining'][current\_planet] = False

# Меню печи

def furnace\_menu(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

add\_coal\_button = types.KeyboardButton('➕ Добавить уголь')

smelt\_ore\_button = types.KeyboardButton('⚒ Переплавить руду')

back\_button = types.KeyboardButton('⏪ Меню')

markup.add(add\_coal\_button, smelt\_ore\_button)

markup.add(back\_button)

with user\_data\_lock:

ores\_available = user\_data[user\_id]['ores'][current\_planet].copy()

if ores\_available:

ores\_list = "Доступные для переплавки руды:\n"

for ore, amount in ores\_available.items():

ores\_list += f"{ore}: {amount} шт.\n"

ores\_list += "\nВыберите действие:"

bot.send\_message(message.chat.id, ores\_list, reply\_markup=markup)

else:

bot.send\_message(message.chat.id, f"В вашем портфеле на планете {current\_planet} нет руды для переплавки.", reply\_markup=markup)

# Добавить уголь

def add\_coal(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

bot.send\_message(message.chat.id, f"Сколько угля вы хотите добавить в печь на планете {current\_planet}?")

bot.register\_next\_step\_handler(message, process\_add\_coal)

def process\_add\_coal(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

try:

amount = int(message.text)

if amount <= 0:

bot.send\_message(message.chat.id, "Количество должно быть положительным.")

return

except ValueError:

bot.send\_message(message.chat.id, "Пожалуйста, введите корректное число.")

return

with user\_data\_lock:

user\_data[user\_id]['coal'][current\_planet] += amount

bot.send\_message(message.chat.id, f"Вы добавили {amount} угля в печь на планете {current\_planet}.")

# Переплавить руду

def smelt\_ore(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

ores\_user = user\_data[user\_id]['ores'][current\_planet].copy()

if not ores\_user:

bot.send\_message(message.chat.id, "У вас нет руды для переплавки.")

return

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

for ore, amount in ores\_user.items():

markup.add(types.KeyboardButton(f"{ore} {amount} шт."))

back\_button = types.KeyboardButton('⏪ Меню')

markup.add(back\_button)

bot.send\_message(message.chat.id, f"Выберите руду для переплавки на планете {current\_planet}:", reply\_markup=markup)

bot.register\_next\_step\_handler(message, process\_smelt\_ore)

def process\_smelt\_ore(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

selected\_text = message.text

if selected\_text == '⏪ Меню':

show\_main\_menu(message)

return

# Извлекаем название руды из текста кнопки, например, "Железная руда 3 шт." -> "Железная руда"

if ' шт.' in selected\_text:

ore = selected\_text.rsplit(' ', 2)[0]

else:

ore = selected\_text # На всякий случай

with user\_data\_lock:

if ore not in user\_data[user\_id]['ores'][current\_planet]:

bot.send\_message(message.chat.id, "У вас нет такой руды.")

return

if user\_data[user\_id]['coal'][current\_planet] <= 0:

bot.send\_message(message.chat.id, "У вас нет угля для переплавки.")

return

bot.send\_message(message.chat.id, f"Сколько '{ore}' вы хотите переплавить на планете {current\_planet}?")

with user\_data\_lock:

user\_data[user\_id]['selected\_ore'] = ore

bot.register\_next\_step\_handler(message, process\_smelting\_amount)

def process\_smelting\_amount(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

try:

amount = int(message.text)

if amount <= 0:

bot.send\_message(message.chat.id, "Количество должно быть положительным.")

return

except ValueError:

bot.send\_message(message.chat.id, "Пожалуйста, введите корректное число.")

return

with user\_data\_lock:

ore = user\_data[user\_id].get('selected\_ore')

if not ore:

bot.send\_message(message.chat.id, "Произошла ошибка. Пожалуйста, попробуйте снова.")

return

available\_amount = user\_data[user\_id]['ores'][current\_planet].get(ore, 0)

if amount > available\_amount:

bot.send\_message(message.chat.id, f"У вас нет столько '{ore}'. Доступно для переплавки: {available\_amount}.")

return

if user\_data[user\_id]['coal'][current\_planet] <= 0:

bot.send\_message(message.chat.id, "У вас нет угля для переплавки.")

return

furnace\_level = user\_data[user\_id]['furnace\_level'][current\_planet]

smelting\_speed = furnaces[current\_planet][furnace\_level - 1]['smelt\_capacity']

if amount > smelting\_speed:

bot.send\_message(message.chat.id, f"Ваша печь может переплавить за раз не более {smelting\_speed} единиц руды.")

amount = smelting\_speed

# Проверка геймпасса моментальной переплавки

instant\_smelting = user\_data[user\_id]['gamepasses'].get('instant\_smelting', False)

smelt\_time = 0 if instant\_smelting else 3 # Если геймпасс активен, время переплавки 0

# Обновляем данные пользователя

user\_data[user\_id]['ores'][current\_planet][ore] -= amount

if user\_data[user\_id]['ores'][current\_planet][ore] == 0:

del user\_data[user\_id]['ores'][current\_planet][ore]

user\_data[user\_id]['smelting'][current\_planet][ore] = user\_data[user\_id]['smelting'][current\_planet].get(ore, 0) + amount

user\_data[user\_id]['coal'][current\_planet] -= 1

if smelt\_time > 0:

bot.send\_message(message.chat.id, f"Начата переплавка {amount} x {ore} на планете {current\_planet}. Это займет {smelt\_time} секунд.")

else:

bot.send\_message(message.chat.id, f"Переплавка {amount} x {ore} на планете {current\_planet} начата моментально.")

threading.Thread(target=finish\_smelting, args=(message, ore, amount, smelt\_time)).start()

def finish\_smelting(message, ore, amount, smelt\_time):

try:

if smelt\_time > 0:

time.sleep(smelt\_time)

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

user\_data[user\_id]['smelting'][current\_planet][ore] -= amount

if user\_data[user\_id]['smelting'][current\_planet][ore] == 0:

del user\_data[user\_id]['smelting'][current\_planet][ore]

user\_data[user\_id]['ingots'][current\_planet][ore] = user\_data[user\_id]['ingots'][current\_planet].get(ore, 0) + amount

except Exception as e:

bot.send\_message(message.chat.id, f"Ошибка при переплавке: {e}")

with user\_data\_lock:

user\_data[user\_id]['is\_mining'][current\_planet] = False

admin\_panel(message)

finally:

if smelt\_time > 0:

bot.send\_message(message.chat.id, f"Переплавка {amount} x {ore} на планете {current\_planet} завершена.")

# Автоматический возврат в меню печи после завершения переплавки

furnace\_menu(message)

# Продать руду

def sell\_ores(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

ingots = user\_data[user\_id]['ingots'][current\_planet].copy()

if not ingots:

bot.send\_message(message.chat.id, "У вас нет переплавленной руды для продажи.")

return

total\_earnings = 0

earnings\_details = ""

earth\_ores = [ore\_item['name'] for ore\_item in ores['Earth']]

with user\_data\_lock:

for ore, amount in ingots.items():

if ore in earth\_ores:

base\_price = 10 \* (earth\_ores.index(ore) + 1)

else:

base\_price = 10 # Можно настроить по желанию

if current\_planet == 'Earth':

multiplier = 1

elif current\_planet == 'Moon':

multiplier = 1.5

elif current\_planet == 'Mars':

multiplier = 2

earnings = int(amount \* base\_price \* multiplier \* user\_data[user\_id]['multiplier'])

total\_earnings += earnings

earnings\_details += f"{ore}: {amount} x {base\_price} x {multiplier} x {user\_data[user\_id]['multiplier']} = {earnings} монет\n"

user\_data[user\_id]['balance'] += total\_earnings

user\_data[user\_id]['ingots'][current\_planet] = {}

sell\_message = f"Вы продали все переплавленные руды на планете {current\_planet} и заработали {total\_earnings} монет.\n\nДетали:\n{earnings\_details}"

bot.send\_message(message.chat.id, sell\_message)

# Показать баланс

def show\_balance(message):

user\_id = message.from\_user.id

with user\_data\_lock:

balance = user\_data[user\_id]['balance']

multiplier = user\_data[user\_id]['multiplier']

bot.send\_message(message.chat.id, f"Ваш баланс: {balance} монет.\nМножитель монет: ×{multiplier}")

# Показать портфель

def show\_inventory(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

ores\_in\_inventory = user\_data[user\_id]['ores'][current\_planet].copy()

ingots = user\_data[user\_id]['ingots'][current\_planet].copy()

if not ores\_in\_inventory and not ingots:

bot.send\_message(message.chat.id, f"Ваш портфель на планете {current\_planet} пуст.")

else:

inventory = f"Портфель содержит:\n"

if ores\_in\_inventory:

inventory += "\n\*Не переплавленной руды:\*\n"

for ore, amount in ores\_in\_inventory.items():

inventory += f"{ore}: {amount} шт.\n"

else:

inventory += "\n\*Не переплавленной руды:\* Нет\n"

if ingots:

inventory += "\n\*Переплавленной руды:\*\n"

for ore, amount in ingots.items():

inventory += f"{ore}: {amount} шт.\n"

else:

inventory += "\n\*Переплавленной руды:\* Нет\n"

bot.send\_message(message.chat.id, inventory, parse\_mode='Markdown')

# Меню магазина с отдельной вкладкой для покупки планет и ребиртов

def shop\_menu(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

rebirth\_available = current\_planet == 'Mars' and user\_data[user\_id]['rebirth\_count'] < len(rebirths)

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

buy\_pickaxe\_button = types.KeyboardButton('Купить кирку')

buy\_backpack\_button = types.KeyboardButton('Купить рюкзак')

buy\_furnace\_button = types.KeyboardButton('Купить печь')

buy\_mine\_button = types.KeyboardButton('Купить шахту')

buy\_cases\_button = types.KeyboardButton('Кейсы')

buy\_rebirth\_button = types.KeyboardButton('Купить ребирт') if rebirth\_available else None

buy\_planets\_button = types.KeyboardButton('Купить планеты')

back\_button = types.KeyboardButton('⏪ Меню')

# Добавляем кнопки в магазин

buttons = [buy\_pickaxe\_button, buy\_backpack\_button]

buttons += [buy\_furnace\_button, buy\_mine\_button]

buttons += [buy\_cases\_button]

if buy\_rebirth\_button:

buttons.append(buy\_rebirth\_button)

buttons.append(buy\_planets\_button)

buttons.append(back\_button)

markup.add(\*buttons)

bot.send\_message(message.chat.id, f"Магазин на планете {current\_planet}", reply\_markup=markup)

# Функция покупки и переключения оборудования

def buy\_pickaxe(message):

buy\_item(message, 'pickaxe')

def buy\_backpack(message):

buy\_item(message, 'backpack')

def buy\_furnace(message):

buy\_item(message, 'furnace')

def buy\_mine(message):

buy\_item(message, 'mine')

def buy\_item(message, item\_type):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

if item\_type == 'pickaxe':

items = pickaxes[current\_planet]

user\_inventory = user\_data[user\_id]['purchased\_pickaxes'][current\_planet]

current\_level = user\_data[user\_id]['pickaxe\_level'][current\_planet]

price\_multiplier = 450 if current\_planet == 'Earth' else (675 if current\_planet == 'Moon' else 900) # Увеличены цены на 1.5x

elif item\_type == 'backpack':

items = backpacks[current\_planet]

user\_inventory = user\_data[user\_id]['purchased\_backpacks'][current\_planet]

current\_level = user\_data[user\_id]['backpack\_level'][current\_planet]

price\_multiplier = 450 if current\_planet == 'Earth' else (675 if current\_planet == 'Moon' else 900) # Увеличены цены на 1.5x

elif item\_type == 'furnace':

items = furnaces[current\_planet]

user\_inventory = user\_data[user\_id]['purchased\_furnaces'][current\_planet]

current\_level = user\_data[user\_id]['furnace\_level'][current\_planet]

price\_multiplier = 600 if current\_planet == 'Earth' else (900 if current\_planet == 'Moon' else 1200) # Увеличены цены на 1.5x

elif item\_type == 'mine':

items = mines[current\_planet]

user\_inventory = user\_data[user\_id]['purchased\_mines'][current\_planet]

current\_level = user\_data[user\_id]['mine\_level'][current\_planet]

price\_multiplier = 750 if current\_planet == 'Earth' else (1125 if current\_planet == 'Moon' else 1500) # Увеличены цены на 1.5x

else:

return

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

for idx, item in enumerate(items):

level = idx + 1

if item\_type == 'mine':

item\_name = item

price = level \* price\_multiplier

else:

item\_name = item['name']

price = level \* price\_multiplier

with user\_data\_lock:

if level in user\_inventory:

label = f"{item\_name} (Куплено)"

else:

if item\_type == 'pickaxe':

label = f"{item\_name} (+{item['luck']}% удачи) - {price} монет"

elif item\_type == 'backpack':

label = f"{item\_name} (вместимость: {item['capacity']}) - {price} монет"

elif item\_type == 'furnace':

label = f"{item\_name} (переплавка: {item['smelt\_capacity']} руд) - {price} монет"

elif item\_type == 'mine':

label = f"{item\_name} - {price} монет"

markup.add(types.KeyboardButton(label))

back\_button = types.KeyboardButton('🛒 Продолжить покупки')

exit\_button = types.KeyboardButton('⏪ Меню')

markup.add(back\_button, exit\_button)

bot.send\_message(message.chat.id, f"Выберите {item\_type} на планете {current\_planet}:", reply\_markup=markup)

bot.register\_next\_step\_handler(message, process\_buy\_item, item\_type, items, price\_multiplier)

def process\_buy\_item(message, item\_type, items, price\_multiplier):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

selected = message.text

if selected == '⏪ Меню':

show\_main\_menu(message)

return

elif selected == '🛒 Продолжить покупки':

shop\_menu(message)

return

with user\_data\_lock:

user\_inventory = user\_data[user\_id][f'purchased\_{item\_type}s'][current\_planet].copy()

for idx, item in enumerate(items):

level = idx + 1

if item\_type == 'mine':

item\_name = item

price = level \* price\_multiplier

if level in user\_inventory:

option = f"{item\_name} (Куплено)"

if selected == option:

user\_data[user\_id][f'{item\_type}\_level'][current\_planet] = level

bot.send\_message(message.chat.id, f"Вы переключились на {item\_name} на планете {current\_planet}.")

return

else:

option = f"{item\_name} - {price} монет"

if selected == option:

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки {item\_name}. Нужно {price} монет.")

return

user\_data[user\_id]['balance'] -= price

user\_data[user\_id][f'{item\_type}\_level'][current\_planet] = level

user\_data[user\_id][f'purchased\_{item\_type}s'][current\_planet].append(level)

bot.send\_message(message.chat.id, f"Вы купили {item\_name} за {price} монет на планете {current\_planet}.")

return

else:

item\_name = item['name']

price = level \* price\_multiplier

if level in user\_inventory:

option = f"{item\_name} (Куплено)"

if selected == option:

user\_data[user\_id][f'{item\_type}\_level'][current\_planet] = level

bot.send\_message(message.chat.id, f"Вы переключились на {item\_name} на планете {current\_planet}.")

return

else:

if item\_type == 'pickaxe':

option = f"{item\_name} (+{item['luck']}% удачи) - {price} монет"

elif item\_type == 'backpack':

option = f"{item\_name} (вместимость: {item['capacity']}) - {price} монет"

elif item\_type == 'furnace':

option = f"{item\_name} (переплавка: {item['smelt\_capacity']} руд) - {price} монет"

elif item\_type == 'mine':

option = f"{item\_name} - {price} монет"

if selected == option:

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки {item\_name}. Нужно {price} монет.")

return

user\_data[user\_id]['balance'] -= price

user\_data[user\_id][f'{item\_type}\_level'][current\_planet] = level

user\_data[user\_id][f'purchased\_{item\_type}s'][current\_planet].append(level)

bot.send\_message(message.chat.id, f"Вы купили {item\_name} за {price} монет на планете {current\_planet}.")

return

bot.send\_message(message.chat.id, "Некорректный выбор. Попробуйте снова.")

buy\_item(message, item\_type)

# Функция покупки планет

def buy\_planet(message, planet\_name, price):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

if planet\_name in user\_data[user\_id]['purchased\_planets']:

bot.send\_message(message.chat.id, f"Вы уже приобрели планету {planet\_name}.")

return

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки планеты {planet\_name}. Нужно {price} монет.")

return

user\_data[user\_id]['balance'] -= price

user\_data[user\_id]['purchased\_planets'].append(planet\_name)

# Инициализация оборудования на новой планете

user\_data[user\_id]['pickaxe\_level'][planet\_name] = 1

user\_data[user\_id]['backpack\_level'][planet\_name] = 1

user\_data[user\_id]['furnace\_level'][planet\_name] = 1

user\_data[user\_id]['mine\_level'][planet\_name] = 1

user\_data[user\_id]['purchased\_pickaxes'][planet\_name] = [1]

user\_data[user\_id]['purchased\_backpacks'][planet\_name] = [1]

user\_data[user\_id]['purchased\_furnaces'][planet\_name] = [1]

user\_data[user\_id]['purchased\_mines'][planet\_name] = [1]

user\_data[user\_id]['ingots'][planet\_name] = {}

user\_data[user\_id]['ores'][planet\_name] = {}

user\_data[user\_id]['smelting'][planet\_name] = {}

user\_data[user\_id]['coal'][planet\_name] = 0

user\_data[user\_id]['is\_mining'][planet\_name] = False

bot.send\_message(message.chat.id, f"Поздравляем! Вы приобрели планету {planet\_name} за {price} монет.")

bot.send\_message(message.chat.id, f"Теперь вы можете переключаться между планетами: {', '.join(user\_data[user\_id]['purchased\_planets'])}.")

# Меню для покупки планет

def buy\_planets\_menu(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

available\_planets = [planet for planet in ['Moon', 'Mars'] if planet not in user\_data[user\_id]['purchased\_planets']]

if not available\_planets:

bot.send\_message(message.chat.id, "У вас уже куплены все доступные планеты.", reply\_markup=types.ReplyKeyboardRemove())

shop\_menu(message)

return

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

for planet in available\_planets:

if planet == 'Moon':

label = 'Купить Луну'

else:

label = 'Купить Марс'

markup.add(types.KeyboardButton(label))

back\_button = types.KeyboardButton('🛒 Продолжить покупки')

exit\_button = types.KeyboardButton('⏪ Меню')

markup.add(back\_button, exit\_button)

bot.send\_message(message.chat.id, "Выберите планету для покупки:", reply\_markup=markup)

bot.register\_next\_step\_handler(message, process\_buy\_planet)

def process\_buy\_planet(message):

user\_id = message.from\_user.id

selected = message.text

if selected == '⏪ Меню':

show\_main\_menu(message)

return

elif selected == '🛒 Продолжить покупки':

shop\_menu(message)

return

elif selected == 'Купить Луну':

buy\_planet(message, 'Moon', 150000) # Цена Луна: 150,000 монет

elif selected == 'Купить Марс':

buy\_planet(message, 'Mars', 350000) # Цена Марс: 350,000 монет

else:

bot.send\_message(message.chat.id, "Некорректный выбор. Возвращаюсь в меню покупки планет.")

buy\_planets\_menu(message)

# Функция переключения планеты

def switch\_planet\_menu(message):

user\_id = message.from\_user.id

with user\_data\_lock:

purchased\_planets = user\_data[user\_id]['purchased\_planets'].copy()

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

for planet in purchased\_planets:

if planet == 'Earth':

label = 'Земля'

elif planet == 'Moon':

label = 'Луна'

elif planet == 'Mars':

label = 'Марс'

else:

label = planet

markup.add(types.KeyboardButton(label))

back\_button = types.KeyboardButton('⏪ Меню')

markup.add(back\_button)

bot.send\_message(message.chat.id, "Выберите планету для переключения:", reply\_markup=markup)

bot.register\_next\_step\_handler(message, process\_switch\_planet)

def process\_switch\_planet(message):

user\_id = message.from\_user.id

selected = message.text

if selected == '⏪ Меню':

show\_main\_menu(message)

return

planet\_mapping = {'Земля': 'Earth', 'Луна': 'Moon', 'Марс': 'Mars'}

if selected in planet\_mapping and planet\_mapping[selected] in user\_data[user\_id]['purchased\_planets']:

with user\_data\_lock:

user\_data[user\_id]['current\_planet'] = planet\_mapping[selected]

bot.send\_message(message.chat.id, f"Вы переключились на планету {selected}.")

else:

bot.send\_message(message.chat.id, "Некорректный выбор. Возвращаюсь в меню переключения планет.")

show\_main\_menu(message)

# 🧰 Кейсы: Функции для отображения и открытия кейсов без смайликов

def show\_cases\_menu(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

available\_cases = cases[current\_planet].copy()

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

for case in available\_cases:

label = f"{case['name']} - {case['price']} монет"

markup.add(types.KeyboardButton(label))

back\_button = types.KeyboardButton('⏪ Меню')

markup.add(back\_button)

bot.send\_message(message.chat.id, f"Выберите кейс для покупки и открытия на планете {current\_planet}:", reply\_markup=markup)

bot.register\_next\_step\_handler(message, process\_case\_selection)

def process\_case\_selection(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

selected = message.text

if selected == '⏪ Меню':

show\_main\_menu(message)

return

with user\_data\_lock:

selected\_case = next((case for case in cases[current\_planet] if f"{case['name']} - {case['price']} монет" == selected), None)

if not selected\_case:

bot.send\_message(message.chat.id, "Некорректный выбор. Возвращаюсь в меню кейсов.")

show\_cases\_menu(message)

return

with user\_data\_lock:

if user\_data[user\_id]['balance'] < selected\_case['price']:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки {selected\_case['name']}. Нужно {selected\_case['price']} монет.")

show\_cases\_menu(message)

return

user\_data[user\_id]['balance'] -= selected\_case['price']

bot.send\_message(message.chat.id, f"Вы приобрели {selected\_case['name']} за {selected\_case['price']} монет.")

bot.send\_message(message.chat.id, f"Открытие {selected\_case['name']} началось! Подождите 10 секунд...")

# Запуск процесса открытия кейса в отдельном потоке

threading.Thread(target=open\_case, args=(message, selected\_case)).start()

def open\_case(message, case):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

min\_reward = case['min\_reward']

max\_reward = case['max\_reward']

try:

# Имитация кручения чисел (10 секунд)

for i in range(1, 11):

with user\_data\_lock:

if current\_planet == 'Earth':

spinning\_number = random.randint(min\_reward, max\_reward)

elif current\_planet == 'Moon':

spinning\_number = random.randint(int(min\_reward \* 1.2), int(max\_reward \* 1.2)) # Увеличение на 20%

elif current\_planet == 'Mars':

spinning\_number = random.randint(int(min\_reward \* 1.4), int(max\_reward \* 1.4)) # Увеличение на 40%

else:

spinning\_number = random.randint(min\_reward, max\_reward) # По умолчанию

bot.send\_message(message.chat.id, f"🔄 Крутится... {spinning\_number} монет")

time.sleep(1)

# Генерация окончательной награды

with user\_data\_lock:

if current\_planet == 'Earth':

reward = random.randint(min\_reward, max\_reward)

elif current\_planet == 'Moon':

reward = random.randint(int(min\_reward \* 1.2), int(max\_reward \* 1.2))

elif current\_planet == 'Mars':

reward = random.randint(int(min\_reward \* 1.4), int(max\_reward \* 1.4))

else:

reward = random.randint(min\_reward, max\_reward) # По умолчанию

reward \*= user\_data[user\_id]['multiplier']

user\_data[user\_id]['balance'] += reward

except Exception as e:

bot.send\_message(message.chat.id, f"Ошибка при открытии кейса: {e}")

finally:

# Автоматический возврат в меню кейсов после открытия

bot.send\_message(message.chat.id, f"🎉 Поздравляем! Вы получили {reward} монет из {case['name']} на планете {current\_planet}.")

show\_cases\_menu(message)

# Функция покупки ребирта

def buy\_rebirth(message):

user\_id = message.from\_user.id

current\_planet = user\_data[user\_id]['current\_planet']

with user\_data\_lock:

rebirth\_available = current\_planet == 'Mars' and user\_data[user\_id]['rebirth\_count'] < len(rebirths)

if not rebirth\_available:

if current\_planet != 'Mars':

reason = "Ребирты можно покупать только на планете Марс."

else:

reason = "Вы достигли максимального количества ребиртов."

bot.send\_message(message.chat.id, reason)

return

rebirth = rebirths[user\_data[user\_id]['rebirth\_count']]

price = rebirth['price']

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки ребирта '{rebirth['name']}'. Нужно {price} монет.")

return

# Подтверждение покупки

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

yes\_button = types.KeyboardButton('Да')

no\_button = types.KeyboardButton('Нет')

markup.add(yes\_button, no\_button)

bot.send\_message(message.chat.id, f"Вы уверены, что хотите купить ребирт '{rebirth['name']}' за {price} монет?\nВсе ваши данные будут сброшены, и баланс станет 0, но ваш множитель монет увеличится на ×1.", reply\_markup=markup)

bot.register\_next\_step\_handler(message, confirm\_rebirth\_purchase, rebirth)

def confirm\_rebirth\_purchase(message, rebirth):

user\_id = message.from\_user.id

if message.text.lower() == 'да':

price = rebirth['price']

with user\_data\_lock:

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки ребирта '{rebirth['name']}'. Нужно {price} монет.")

return

user\_data[user\_id]['balance'] -= price

user\_data[user\_id]['rebirth\_count'] += 1

user\_data[user\_id]['multiplier'] += 1

# Сброс всех данных, кроме rebirth\_count, multiplier и gamepasses

user\_data[user\_id]['current\_planet'] = 'Earth' # После сброса игрок начинает с Земли

user\_data[user\_id]['purchased\_planets'] = ['Earth'] # Только Земля как купленная планета

user\_data[user\_id]['pickaxe\_level'] = {'Earth': 1}

user\_data[user\_id]['backpack\_level'] = {'Earth': 1}

user\_data[user\_id]['furnace\_level'] = {'Earth': 1}

user\_data[user\_id]['mine\_level'] = {'Earth': 1}

user\_data[user\_id]['purchased\_pickaxes'] = {'Earth': [1], 'Moon': [], 'Mars': []}

user\_data[user\_id]['purchased\_backpacks'] = {'Earth': [1], 'Moon': [], 'Mars': []}

user\_data[user\_id]['purchased\_furnaces'] = {'Earth': [1], 'Moon': [], 'Mars': []}

user\_data[user\_id]['purchased\_mines'] = {'Earth': [1], 'Moon': [], 'Mars': []}

user\_data[user\_id]['ingots'] = {'Earth': {}, 'Moon': {}, 'Mars': {}}

user\_data[user\_id]['ores'] = {'Earth': {}, 'Moon': {}, 'Mars': {}}

user\_data[user\_id]['smelting'] = {'Earth': {}, 'Moon': {}, 'Mars': {}}

user\_data[user\_id]['coal'] = {'Earth': 0, 'Moon': 0, 'Mars': 0}

user\_data[user\_id]['is\_mining'] = {'Earth': False, 'Moon': False, 'Mars': False}

user\_data[user\_id]['clicker\_last\_click'] = 0

user\_data[user\_id]['clicker\_balance'] = 0

user\_data[user\_id]['miners'] = {}

# Геймпассы сохраняются

else:

bot.send\_message(message.chat.id, "Покупка ребирта отменена.")

show\_main\_menu(message)

# Функция покупки геймпасса

def purchase\_gamepass(user\_id, gamepass\_key, message):

with user\_data\_lock:

if gamepass\_key not in gamepasses:

bot.send\_message(message.chat.id, "Некорректный геймпасс.")

return

if gamepass\_key in user\_data[user\_id]['gamepasses']:

bot.send\_message(message.chat.id, "Вы уже приобрели этот геймпасс.")

return

gamepass = gamepasses[gamepass\_key]

price = gamepass['price']

if user\_data[user\_id]['balance'] < price:

bot.send\_message(message.chat.id, f"У вас недостаточно монет для покупки {gamepass['name']}. Нужно {price} монет.")

return

# Покупка геймпасса

user\_data[user\_id]['balance'] -= price

user\_data[user\_id]['gamepasses'][gamepass\_key] = True

bot.send\_message(message.chat.id, f"Вы успешно приобрели геймпасс '{gamepass['name']}'.")

# Функция покупки кликера

def purchase\_clicker(user\_id, message):

with user\_data\_lock:

# Проверяем, активирован ли уже кликер

if user\_data[user\_id].get('clicker\_active', False):

bot.send\_message(message.chat.id, "Ваш кликер уже активен.")

return

# Активируем кликер

user\_data[user\_id]['clicker\_active'] = True

user\_data[user\_id]['clicker\_balance'] = 0

bot.send\_message(message.chat.id, "Кликер активирован! Нажимайте на кнопку на сайте для получения монет.")

# Запуск потока для автоматического начисления монет при кликах

threading.Thread(target=clicker\_task, args=(user\_id,)).start()

def clicker\_task(user\_id):

while True:

with user\_data\_lock:

if not user\_data[user\_id].get('clicker\_active', False):

break

last\_click = user\_data[user\_id].get('clicker\_last\_click', 0)

current\_time = time.time()

if current\_time - last\_click >= clicker['cooldown']:

# Проверяем, есть ли клики

if user\_data[user\_id]['clicker\_balance'] > 0:

user\_data[user\_id]['balance'] += clicker['reward'] \* user\_data[user\_id]['multiplier']

user\_data[user\_id]['clicker\_balance'] -= 1

try:

bot.send\_message(user\_id, f"Ваш кликер добавил {clicker['reward'] \* user\_data[user\_id]['multiplier']} монет на ваш баланс.")

except Exception:

pass

time.sleep(1)

# Запуск бота

def run\_bot():

try:

bot.polling(none\_stop=True)

except Exception as e:

print(f"Ошибка в работе бота: {e}")

time.sleep(15)

run\_bot()

if \_\_name\_\_ == '\_\_main\_\_':

run\_bot()