**一、游戏规则说明**

**（一）游戏目标**

玩家的目标是通过在正规市场和黑市进行商品交易，积累尽可能多的货币，同时避免因亏损过多或在黑市被逮捕而导致游戏结束。

**（二）游戏流程**

1.游戏开始：

玩家初始拥有1000单位货币。

游戏界面分为两个市场：正规市场和黑市。

玩家需要决定在每个市场的供应数量和价格。

2.供应操作：

（1）正规市场：

玩家输入供应数量和价格。

系统根据输入的数量和价格计算成本（成本 = 供应数量 × 价格）。

如果玩家的货币不足以支付成本，系统会提示“货币不足”。

如果供应数量或价格小于等于0，系统会提示“供应数量和价格必须大于0”。

成功供应后，玩家的货币减少，相应市场的库存增加。

（2）黑市：

操作与正规市场相同，但存在被逮捕的风险。

**（三）交易操作**

玩家点击“进行交易”按钮后，系统模拟两个市场的交易。

正规市场：

系统随机生成需求量（0到玩家在该市场的库存之间）。

玩家的收入为需求量 × 价格。

玩家的货币增加，库存减少。

黑市：

操作与正规市场相同，但存在被逮捕的风险。被逮捕的概率为20%。

如果玩家被逮捕，游戏结束。

如果玩家的货币亏损超过500单位，游戏也会结束。

**（四）游戏结束**

玩家被逮捕。

玩家的货币亏损超过500单位。

**二、相关经济学知识**

**（一）供求关系**

需求：需求量通常与价格成反比关系。价格越高，需求量越低；价格越低，需求量越高。

供应：供应量通常与价格成正比关系。价格越高，供应量越高；价格越低，供应量越低。

市场均衡：当供应量等于需求量时，市场达到均衡。

**（二）价格弹性**

需求的价格弹性：衡量需求量对价格变化的敏感程度。如果需求对价格变化非常敏感，需求的价格弹性就高；反之则低。

供应的价格弹性：衡量供应量对价格变化的敏感程度。如果供应对价格变化非常敏感，供应的价格弹性就高；反之则低。

**（三）风险与收益**

风险：黑市交易存在被逮捕的风险，这种风险是随机的，概率为20%。

收益：黑市的初始价格高于正规市场，但风险也更高。

**（四）成本与收益分析**

成本：供应商品需要支付的成本（供应数量 × 价格）。

收益：交易后获得的收入（需求量 × 价格）。

净收益：收益减去成本。

**三、致胜技巧**

**（一）合理分配资源**

初始阶段：建议主要在正规市场操作，因为黑市存在被逮捕的风险。可以先积累一定的货币，降低风险。

中期阶段：当货币积累到一定程度后，可以适当增加黑市的供应量，但要控制风险。可以通过少量多次的方式试探黑市的需求和风险。

后期阶段：如果货币充足且风险可控，可以加大黑市的供应量，但要确保不会因一次被逮捕而全盘皆输。

**（二）灵活调整价格**

正规市场：可以根据库存和需求情况灵活调整价格。如果库存较多，可以适当降低价格以增加需求量；如果库存较少，可以适当提高价格以增加收益。

黑市：由于黑市的需求和风险都较高，价格可以适当高于正规市场，但也要根据库存和需求情况进行调整。

**（三）控制风险**

避免过度冒险：不要将所有货币都投入到黑市，要保留一部分在正规市场，以降低风险。

分散风险：可以通过在两个市场都进行操作，分散风险。例如，可以在正规市场供应一部分商品，同时在黑市供应一部分商品。

**（四）关注市场动态**

需求变化：每次交易后，系统会显示需求量和剩余库存。玩家可以根据这些信息调整下一次的供应策略。

货币管理：始终保持足够的货币以应对突发情况，避免因货币不足而无法进行供应。

**（五）利用概率优势**

黑市概率：虽然黑市存在被逮捕的风险，但概率只有20%。如果连续几次没有被逮捕，可以适当增加黑市的供应量以获取更高的收益，但要时刻注意风险。

**四、代码解构**

**（一）导入模块**

```python

import tkinter as tk

from tkinter import messagebox

import random

```

`tkinter`：用于创建图形用户界面（GUI）。

`messagebox`：用于显示消息框，如错误提示或游戏结束提示。

`random`：用于生成随机数，模拟市场需求和被捕概率。

**（二）类定义：`MarketGame`**

这是游戏的核心类，包含了游戏的所有逻辑和界面。

初始化方法：`\_\_init\_\_`

```python

def \_\_init\_\_(self, root):

接收一个`root`参数，即Tkinter的主窗口对象。

设置游戏的基本参数：

初始货币：`self.base\_currency = 1000`

常规市场和黑市的初始库存：`self.normal\_market\_stock = 0` 和 `self.black\_market\_stock = 0`

常规市场和黑市的初始价格：`self.normal\_market\_price = 10` 和 `self.black\_market\_price = 15`

损失限制：`self.loss\_limit = 500`

黑市被捕概率：`self.arrest\_probability = 0.2`

调用`create\_intro\_screen`方法创建游戏的介绍界面。

创建介绍界面：`create\_intro\_screen`

```python

def create\_intro\_screen(self):

创建一个介绍界面，包含游戏规则和开始按钮。

使用`tk.Frame`和`tk.Label`等组件显示文本信息。

使用`tk.Button`创建“开始游戏”按钮，点击后调用`start\_game`方法。

开始游戏：`start\_game`

```python

def start\_game(self):

```

清除介绍界面的所有组件。

调用`create\_game\_screen`方法创建游戏主界面。

创建游戏主界面：`create\_game\_screen`

```python

def create\_game\_screen(self):

```

清除介绍界面的所有组件。

创建游戏主界面，包含：

货币显示：`self.currency\_label`

常规市场和黑市的库存、价格和供应量输入框。

供应按钮，分别调用`supply\_normal\_market`和`supply\_black\_market`方法。

交易按钮，调用`trade`方法。

市场状态显示：`self.market\_status\_label`

常规市场供应：`supply\_normal\_market`

```python

def supply\_normal\_market(self):

```

获取用户输入的供应量和价格。

检查输入是否合法（大于0）。

检查是否有足够的货币进行供应。

更新货币和库存，并调用`update\_labels`方法更新界面。

黑市供应：`supply\_black\_market`

```python

def supply\_black\_market(self):

```

逻辑与`supply\_normal\_market`类似，但针对黑市。

交易：`trade`

```python

def trade(self):

```

模拟常规市场和黑市的交易：

随机生成需求量。

计算收入。

更新库存和货币。

检查是否在黑市被捕或损失是否超过限制，如果是则游戏结束。

调用`update\_labels`和`update\_market\_status`方法更新界面。

更新标签：`update\_labels`

```python

def update\_labels(self):

```

更新货币和库存的显示标签。

更新市场状态：`update\_market\_status`

```python

def update\_market\_status(self):

```

更新市场状态显示区域，显示常规市场和黑市的需求、剩余库存和收入。

**（三）主程序入口**

```python

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

root = tk.Tk()

game = MarketGame(root)

root.mainloop()

```

创建Tkinter主窗口。

实例化`MarketGame`类。

启动Tkinter的主循环，显示游戏界面并处理用户交互。

**（四）总结**

代码结构清晰，分为以下几个部分：

1.导入模块：提供必要的功能支持。

2.游戏类`MarketGame`：包含游戏的所有逻辑和界面。

初始化方法：设置游戏参数和创建介绍界面。

介绍界面：显示游戏规则和开始按钮。

游戏主界面：包含货币显示、常规市场和黑市的操作区域、交易按钮和市场状态显示。

供应和交易逻辑：处理用户操作，模拟市场需求和游戏结果。

更新界面：动态更新货币、库存和市场状态。

3.主程序入口：启动游戏。

这种结构使得代码易于维护和扩展，同时通过图形界面提供了良好的用户体验。