**WCST平台代码说明文档**

修订记录

| 日期 | 修订版本 | 修改章节 | 修改描述 | 作者 |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1. 原理介绍 4](#_Toc512525127)

[2. 总流程图 4](#_Toc512525128)

[3. 分模块介绍 5](#_Toc512525129)

[3.1 逻辑模块1: 5](#_Toc512525130)

[3.1.1 代码 5](#_Toc512525131)

[3.1.2 流程图 6](#_Toc512525132)

[3.1.3 详细说明 6](#_Toc512525133)

[3.2 接口模块1: 6](#_Toc512525134)

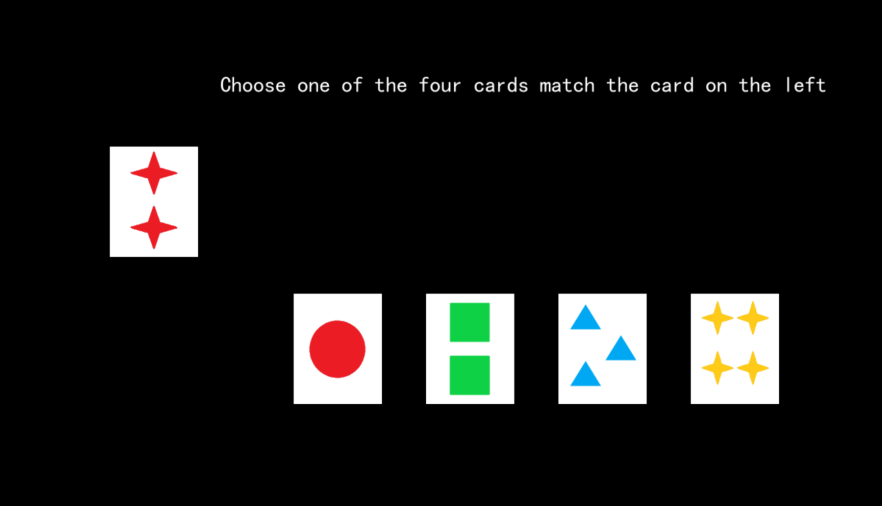
[3.2.1 代码 6](#_Toc512525135)

[3.2.2 功能描述 7](#_Toc512525136)

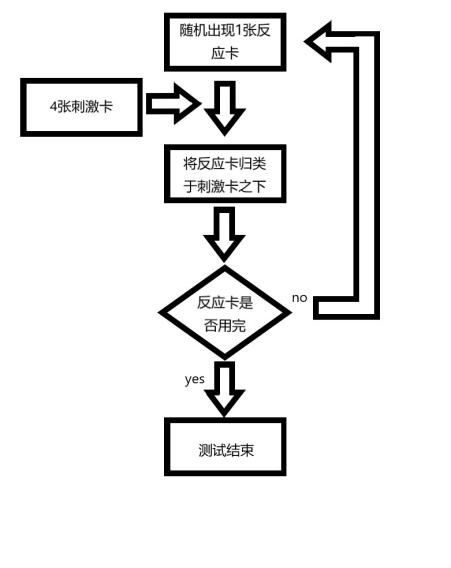
[3.2.3 接口描述 7](#_Toc512525137)

# 原理介绍

威斯康辛卡片分类测试（Wisconsin Card Sorting Task)，简称WCST，是一种单向神经心理测定，用来检测正常人的抽象思维能力。WCST共有4张刺激卡片和若干张反应卡片，每张卡片上会画有1-4个4种不同颜色的4种图形。4张刺激卡片的个数，颜色和图形都不相同。测试者通过随机出现的反应卡片，掌握分类规则，将反应卡片归类于4张刺激卡片之一。分类规则会随着试验次数或正确次数而改变。



# 总流程图



具体步骤如下：

1、4张刺激卡置于前方

2、被试将反应卡依次放在刺激卡下方

3、主试掌握分类原则（颜色/图形/数量）并告诉被试“对”或者“错误 ”

4、每个原则10张正确卡片或10张卡片

5、用完所有反映卡结束

# 分模块介绍

## 逻辑模块1:

### 代码

|  |
| --- |
| 核心代码  class Card(object): //定义卡片的类  class game(object): //整个游戏的类  ...  def run(self): //运行游戏的函数  ...  while self.Trial <60: //游戏主循环  ...  for event in pygame.event.get(): //获取游戏事件  if event.type == pygame.QUIT:  pygame.quit()  sys.exit()  if event.type == pygame.MOUSEBUTTONDOWN: //点击鼠标左键时判断鼠标坐标是否在牌上  pressed\_array = pygame.mouse.get\_pressed()  if pressed\_array[0]:  if self.mouse\_x<=520 and self.mouse\_x>=400 and self.mouse\_y<=550 and self.mouse\_y >=400:  self.flag = 1  self.n+=-1  if self.rule == 'num':  if self.Stim.num == self.Card1.num:  self.Correct += 1  self.Trial += 1  else:  self.Wrong += 1  self.Trial += 1  ...  self.Stim=Card(Num[random.randint(0,3)],Color[random.randint(0,3)],Shape[random.randint(0,3)]) //重置反应卡片  ...  elif event.type == pygame.MOUSEMOTION: //光标移动时记录鼠标坐标  pos = pygame.mouse.get\_pos()  self.mouse\_x = pos[0]  self.mouse\_y = pos[1]  pygame.display.flip()  ... |

### 详细说明

模块需求：

import pygame

import sys

import random

如何安装：

pip install pygame

pip install numpy

通过定义game类并执行run()来开始游戏。初始化窗口显示4张刺激卡片和1张反应卡片，当测试者点击鼠标左键并且鼠标坐标在四张刺激卡片之一上时，视为一次试验，根据当前的规则来判断并反馈给测试者正确或错误并随机更换下一张反应卡片，并且根据上次的规则判断这是否是持续性错误，记录数据。当测试结束时，窗口显示测试结果，包含正确数，错误数，持续性错误数。

## 接口模块1:

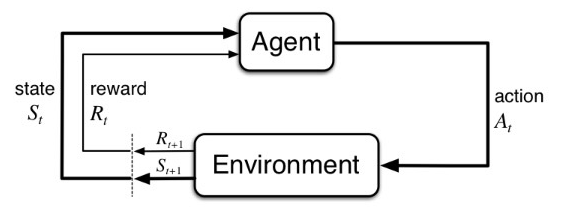
### 代码

|  |
| --- |
| 核心代码  def render(self)://重绘环境的一帧  def reset(self)://重置环境  def step(self,action)://推进一个时间步长，执行一次试验  ...  if action.index(max(action)) == 0:  self.Trial += 1  self.n -= 1  if self.rule == 'num':  reward = 1  self.Correct += 1  else:  reward = 0  self.Wrong += 1  if self.per\_flag:  if self.last\_rule == 'num':  self.per\_error += 1  ...  self.Stim=Card(Num[random.randint(0,3)],Color[random.randint(0,3)],Shape[random.randint(0,3)]) //重置反应卡片  return self.Stim.state, reward, self.Trial>60, [self.Correct,self.Wrong,self.per\_error] |

### 功能描述

reset(self):重置环境的状态，返回观察。step(self,action):推进一个时间步长，返回observation，reward，done，info。Step会执行一次试验，输入为三维向量的action，每一维代表一个特征，（1，0，0），（0，1，0），（0，0，1）分别代表三个策略，输入action并根据action和当前的分类规则判断是否正确，输出当前卡牌的特征，奖励，是否结束，当前的测试数据。render(self):重绘环境的一帧。

### 接口描述



Observation(object):返回一个特定环境的对象，描述对环境的观察。

Reward(float)：返回之前动作收获的总的奖励值。

Done(boolean)：返回是否应该重新设置（reset）环境。

Info(dict)：用于调试的诊断信息。

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | In/Out/parameter | Size/precision | Description/Value |
| action | in | 3\*1 | 输入动作 |
| Reward | Out | Int | 输出奖励 |
| self.Stim.state | Out | 3\*1 | 输出observation |
| [self.Correct,self.Wrong,self.per\_error] | Out | 3\*1 | 输出information |
| self.Trial>60 | Out | Boolean | 输出done |