Universidad de Guadalajara
Sistema de Educación Media Superior
Centro Universitario de Ciencias Exactas e Ingenierías

Estatus

Materia: Computación Tolerante a Fallas

D06 2023 B

Alumno: Esquivel Barbosa Diego Humberto

Código: 211401635

Carrera: INCO

Fecha: 18/09/2023

Introducción

La programación concurrente, que permite que una aplicación ejecute simultáneamente varias operaciones en el mismo espacio de proceso, es esencial en el desarrollo de aplicaciones modernas. Uno de los enfoques para lograr la concurrencia en Python es mediante el uso del módulo threading, que permite la creación y gestión de hilos o subprocesos. En este documento, presentaremos un simulador de carreras de caballos desarrollado en Python, que utiliza hilos para representar a cada caballo en la carrera. Además, el programa es capaz de revisar periódicamente el estado de la aplicación, brindando así una funcionalidad de monitoreo en tiempo real de la carrera.

Código

```
import threading
import random
import time

# Definición de la clase Caballo que hereda de threading.Thread
class Caballo(threading.Thread):
    def __init__(self, nombre, apuesta_numero, distancia_total):
        super().__init__()
        self.nombre = nombre
        self.distancia_recorrida = 0
        self.velocidad = random.randint(1, 5)
        self.apuesta_numero = apuesta_numero
        self.distancia_total = distancia_total

    def run(self):
        while self.distancia_recorrida < self.distancia_total:
            self.distancia_recorrida += self.velocidad
            print(f"{self.nombre} ha recorrido {self.distancia_recorrida} metros.")
            time.sleep(1)  # Simula el tiempo de carrera

# Función para revisar el estado de los caballos durante la carrera
def revisar_estado(caballos):
    while True:
        time.sleep(2)  # Revisa el estado cada 2 segundos
        for caballo in caballos:
            print(f"Estado de {caballo.nombre}: {caballo.distancia_recorrida} metros.")

# Función principal del simulador de carrera
def simulador_carrera(caballos, apuesta_numero):
    hilos = []
    for i, caballo in enumerate(caballos):
        hilo = Caballo(f"Caballo {i + 1}", apuesta_numero, 100)
        hilos.append(hilo)
        hilo.start()

    for hilo in hilos:
        hilo.join()
```

```
    ganador = max(caballos, key=lambda x: x.distancia_recorrida)

    if ganador.apuesta_numero == apuesta_numero:
        print(f"¡Has ganado! {ganador.nombre} ha ganado la carrera.")
    else:
        print(f"Lo siento, {ganador.nombre} ha ganado la carrera.")

if __name__ == "__main__":
    num_caballos = 4
    caballos = [Caballo(f"Caballo {i + 1}", i + 1, 100) for i in range(num_caballos)]

    print("Bienvenido al simulador de carreras de caballos.")
    for i, caballo in enumerate(caballos):
        print(f"{i + 1}. {caballo.nombre}")

    while True:
        try:
            apuesta_numero = int(input("Elige el número del caballo en el que deseas apostar (1-4): "))
            if 1 <= apuesta_numero <= num_caballos:
                break
            else:
                print("Número no válido. Debes elegir un número de caballo entre 1 y 4.")
        except ValueError:
            print("Por favor, ingresa un número válido.")

    # Creación y configuración del hilo revisor de estado
    revisor = threading.Thread(target=revisar_estado, args=(caballos,))
    revisor.daemon = True
    revisor.start()

    # Inicio del simulador de carrera
    simulador_carrera(caballos, apuesta_numero)
```

```
Bienvenido al simulador de carreras de caballos.
1. Caballo 1
2. Caballo 2
3. Caballo 3
4. Caballo 4
Elige el número del caballo en el que deseas apostar (1-4): 4
Caballo 1 ha recorrido 2 metros.
Caballo 2 ha recorrido 3 metros.
Caballo 3 ha recorrido 5 metros.
Caballo 4 ha recorrido 1 metros.
Caballo 1 ha recorrido 4 metros.
Caballo 2 ha recorrido 6 metros.
Caballo 3 ha recorrido 10 metros.
Caballo 4 ha recorrido 2 metros.
Estado de Caballo 1: 0 metros.
Estado de Caballo 2: 0 metros.
Estado de Caballo 3: 0 metros.
Estado de Caballo 4: 0 metros.
Caballo 1 ha recorrido 6 metros.
Caballo 2 ha recorrido 9 metros.
Caballo 3 ha recorrido 15 metros.
Caballo 4 ha recorrido 3 metros.
Caballo 1 ha recorrido 8 metros.
Caballo 2 ha recorrido 12 metros.
```

```
Caballo 4 ha recorrido 96 metros.
Caballo 4 ha recorrido 97 metros.
Estado de Caballo 1: 0 metros.
Estado de Caballo 2: 0 metros.
Estado de Caballo 3: 0 metros.
Estado de Caballo 4: 0 metros.
Caballo 4 ha recorrido 98 metros.
Caballo 4 ha recorrido 99 metros.
Estado de Caballo 1: 0 metros.
Estado de Caballo 2: 0 metros.
Estado de Caballo 3: 0 metros.
Estado de Caballo 4: 0 metros.
Caballo 4 ha recorrido 100 metros.
Lo siento, Caballo 1 ha ganado la carrera.
```

Explicación

Caballo: Se define una clase Caballo que hereda de threading.Thread. Esta clase representa a cada caballo en la carrera y tiene un método run que simula el avance del caballo en la carrera. Cada caballo tiene un nombre, una distancia recorrida, una velocidad aleatoria, un número de apuesta y una distancia total de la carrera.

revisar_estado: Esta función se ejecuta en un hilo separado y periódicamente muestra el estado de cada caballo, incluyendo la distancia recorrida.

simulador_carrera: Esta función principal crea instancias de la clase Caballo para cada caballo en la carrera, inicia los hilos correspondientes, y luego espera a que todos los hilos terminen. Finalmente, determina al ganador de la carrera basado en la distancia recorrida.

Conclusión

El uso de hilos en Python, a través del módulo threading, permite la programación concurrente y la ejecución simultánea de tareas. En este simulador de carreras de caballos, hemos visto cómo los hilos representan a los caballos y cómo se pueden monitorear continuamente utilizando hilos adicionales para revisar su estado. Esta aplicación es un ejemplo práctico de la concurrencia en Python y puede servir como punto de partida para proyectos más complejos.

Bibliografía

Python threading documentation: https://docs.python.org/3/library/threading.html

Real Python - Python Threading: An Introductory Tutorial: https://realpython.com/intro-to-python-threading/

Python Official Documentation (APA Style): Author(s). (Year). Title of the Document. Retrieved from URL