qLearn.ipynb:

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "import gym\n",
    "from random import randint\n",
    "import numpy as np"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "env = gym.make(\"MountainCar-v0\")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "def discretizar(valor):\n",
    "  aux = ((valor - env.observation_space.low) / (env.observation_space.high - env.observation_space.low))*20\n",
    "  return tuple(aux.astype(np.int32))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "q_table = np.random.uniform(low = -1, high = 1, size = [20,20,3])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "taza_aprendizaje = 0.1\n",
    "factor_descuento = 0.95\n",
    "episodios = 5000\n",
    "listado_recompensas = []\n",
    "\n",
    "for episodio in range(episodios):\n",
    "  estado = discretizar(env.reset())\n",
    "  final = False\n",
    "  recompensa_total = 0\n",
    "\n",
    "  while not final:\n",
    "\n",
    "    if randint(0,10) > 2:\n",
    "      accion = np.argmax(q_table[estado])\n",
    "    else:\n",
    "      accion = randint(0,2)\n",
    "\n",
    "    nuevo_estado, recompensa, final, info = env.step(accion)\n",
    "    q_table[estado][accion] = q_table[estado][accion] + taza_aprendizaje * (recompensa + factor_descuento * np.max(q_table[discretizar(nuevo_estado)]) - q_table[estado][accion])\n",
    "    estado = discretizar(nuevo_estado)\n",
    "    recompensa_total += recompensa\n",
    "\n",
    "    if (episodio + 1) % 500 == 0:\n",
    "      env.render()\n",
    "\n",
    "  listado_recompensas.append(recompensa_total)\n",
    "  if (episodio + 1) % 100 == 0:\n",
    "    print(f'Espisodio {episodio + 1} - Recompensa: {np.mean(listado_recompensas)}')\n",
    "\n",
    "  env.close()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.9.13"
  },
  "orig_nbformat": 4
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```