SNAKE GAME – USING RE-INFORCEMENT LEARNING

LIBRARIES: Pygame, Tensorflow.

PROGRAM CODE:

import pygame, random, sys

from pygame.locals import \*

def collide(x1, x2, y1, y2, w1, w2, h1, h2):

if x1+w1&gt;x2 and x1&lt;x2+w2 and y1+h1&gt;y2 and y1&lt;y2+h2:

return True

else:

return False

def die(screen, score):

f=pygame.font.SysFont(&#39;Arial&#39;, 0);t=f.render(&#39;Your score was: &#39;+str(score), True, (0, 0,

0));screen.blit(t, (10, 270));pygame.display.update();pygame.time.wait(2000);sys.exit(0)

xs = [290, 290, 290, 290, 290];ys = [290, 270, 250, 230, 210];dirs = 0;score = 0;

applepos = (random.randint(0, 590), random.randint(0, 590));

pygame.init();

s=pygame.display.set\_mode((600, 600));

pygame.display.set\_caption(&#39;Snake&#39;);

appleimage = pygame.Surface((10, 10));

appleimage.fill((0, 255, 0));

img = pygame.Surface((20, 20));

img.fill((255, 0, 0));

f = pygame.font.SysFont(&#39;Arial&#39;, 20);

clock = pygame.time.Clock()

while True:

clock.tick(20)

for e in pygame.event.get():

if e.type == QUIT:

sys.exit(0)

elif e.type == KEYDOWN:

if e.key == K\_UP and dirs != 0:dirs = 2

elif e.key == K\_DOWN and dirs != 2:dirs = 0

elif e.key == K\_LEFT and dirs != 1:dirs = 3

elif e.key == K\_RIGHT and dirs != 3:dirs = 1

i = len(xs)-1

while i &gt;= 2:

if collide(xs[0], xs[i], ys[0], ys[i], 20, 20, 20, 20):

die(s, score)

i-= 1

if collide(xs[0], applepos[0], ys[0], applepos[1], 20, 10, 20, 10):

score+=1;

xs.append(700);

ys.append(700);

applepos=(random.randint(0,590),random.randint(0,590))

print(xs,ys)

if xs[0] &lt; 0 or xs[0] &gt; 580 or ys[0] &lt; 0 or ys[0] &gt; 580:

die(s, score)

i = len(xs)-1

while i &gt;= 1:

xs[i] = xs[i-1];ys[i] = ys[i-1];i -= 1

if dirs==0:ys[0] += 20

elif dirs==1:xs[0] += 20

elif dirs==2:ys[0] -= 20

elif dirs==3:xs[0] -= 20

s.fill((255, 255, 255))

for i in range(0, len(xs)):

s.blit(img, (xs[i], ys[i]))

s.blit(appleimage, applepos);t=f.render(str(score), True, (0, 0, 0));s.blit(t, (10,

10));pygame.display.update()

Output:

[290, 290, 290, 290, 290] [290, 270, 250, 230, 210]

[290, 290, 290, 290, 290] [310, 290, 270, 250, 230]

[290, 290, 290, 290, 290] [330, 310, 290, 270, 250]

[290, 290, 290, 290, 290] [350, 330, 310, 290, 270]

[290, 290, 290, 290, 290] [370, 350, 330, 310, 290]

[290, 290, 290, 290, 290] [390, 370, 350, 330, 310]

[290, 290, 290, 290, 290] [410, 390, 370, 350, 330]

[290, 290, 290, 290, 290] [430, 410, 390………….

…………………290, 290] [470, 450, 430, 410, 390]

[290, 290, 290, 290, 290] [490, 470, 450, 430, 410]

[290, 290, 290, 290, 290] [510, 490, 470, 450, 430]

[290, 290, 290, 290, 290] [530, 510, 490, 470, 450]

[290, 290, 290, 290, 290] [550, 530, 510, 490, 470]

[290, 290, 290, 290, 290, 700] [570, 550, 530, 510, 490, 700]

[290, 290, 290, 290, 290, 290] [590, 570, 550, 530, 510, 490]

­ROGRAM CODE:lowRNING