Build A Wheel: A Console Car Game using Python

PYTHON – PlaY wiTH it ON

Xiaoyong Wei (魏驍勇)

x1wei@polyu.edu.hk

Department of Computing
電子計算學系

Day 1: Pave The Road

Things you need to know

- >print("string content") is a function to print a string
- >* can be used to duplicate a string for a specified number of times
  - E.g., "ab"*3="ababab"
- >+ can be used to join two strings
  - E.g., "ab"+ "cd"="abcd"
- >[ ] denotes a list collection which can be used to save strings or integers. A list can be composed with the list comprehension based on a for-loop
  - E.g., [d for d in range(5)] generates [0,1,2,3,4]

Things you need to know

- chr() converts a ASCII code to the corresponding character, ord() returns the ASCII code of a character
  - E.g., chr('A') gives 65, ord(65) gives 'A'
- join() can be used to concatenate characters in a list into a string
  - """.join(['a','b','c']) gives "abc"

Try This

```
1 for i in range(10):
2     print("|*|"+" "*45+"|*|")
3     print("| |"+" "*45+"| |")
```

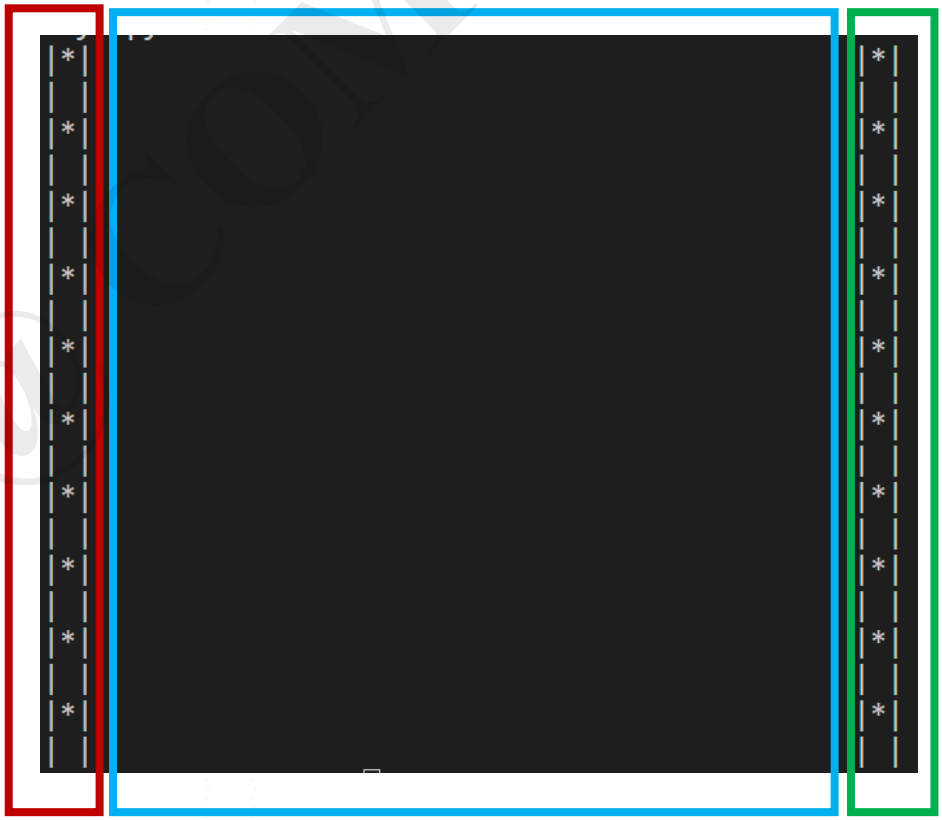
Try This

```
1 for i in range(10):
2     print("|*|" + " "*45 + "|*|")
3     print("| |" + " "*45 + "| |")
```

Left Curb

The Road

Right Curb



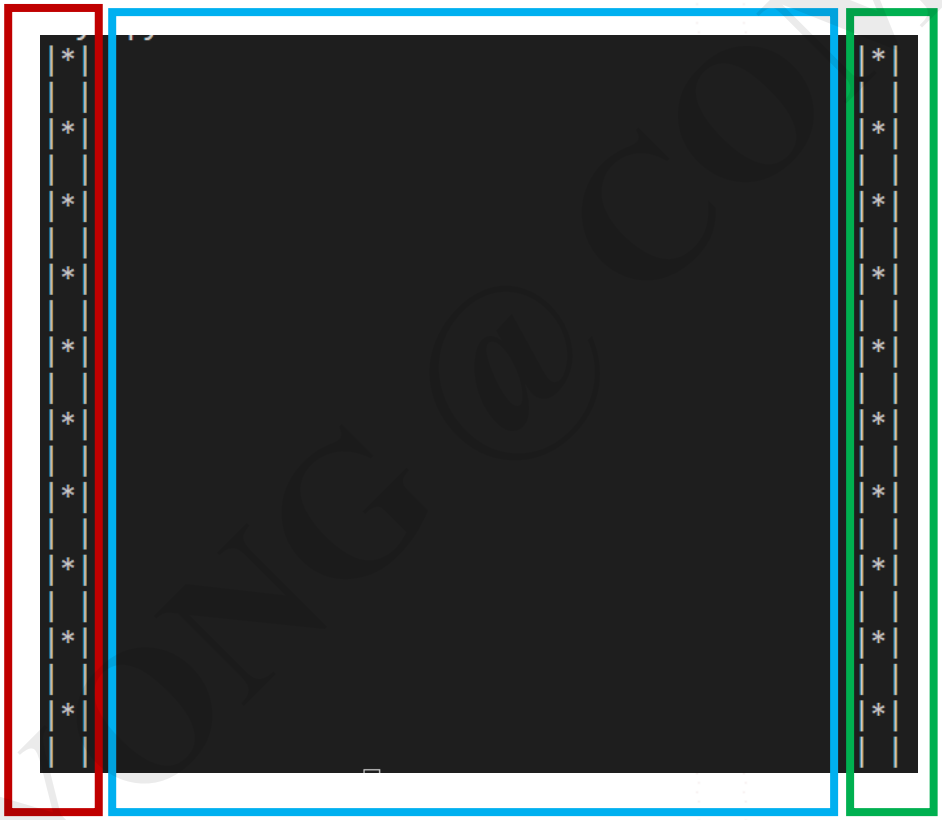
We just created a road with left and right curbs. Each curb takes 3-by-20 characters on screen while the road takes 45-by-20.

Looks good. But, we need to update the content in the scene frequently during the game. We need a better way to print.

A better way for the background

The scene is a 51-by-20 (character) array.

A better way for the curb



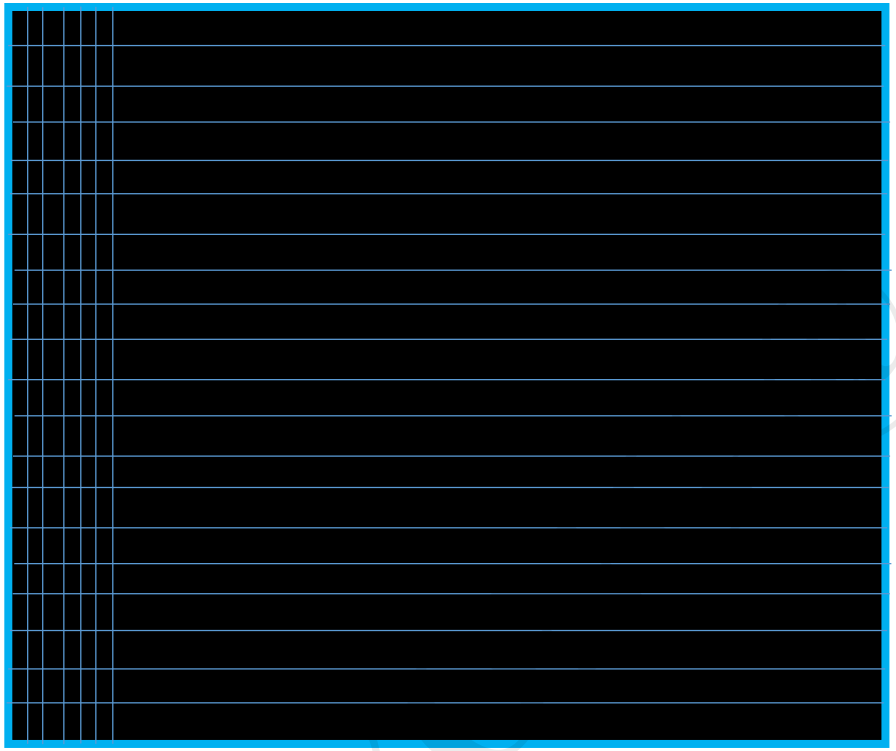
Each curb is a 3-by-20 (character) array with patterns.

Try This

```
# create a background of 51-by-20 characters
background=[[32 for i in range(51)] for j in range(20)]
# creat a curb of 3-by-20 characters
curb=[]
for i in range(10):
    curb.append([ord('|'),ord('*'),ord('|')])
    curb.append([ord('|'),ord(' '),ord('|')])
```

Note that we store the characters using ASCII codes because integers are easier to manipulate than strings.

A better way for the background

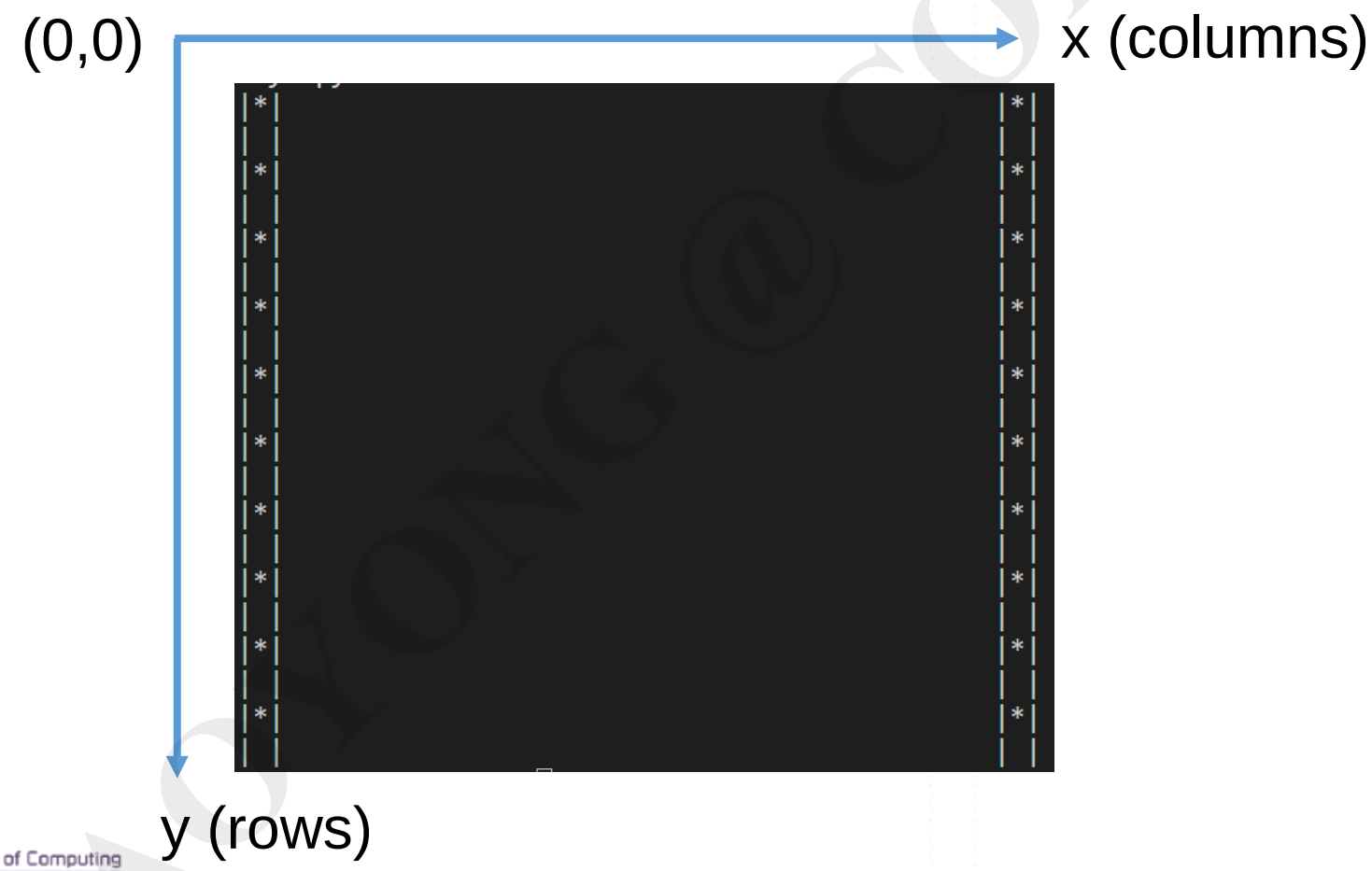


```
[
 [ord(‘ ’), ord(‘ ’), ord(‘ ’), ord(‘ ‘), …],
 [ord(‘ ’), ord(‘ ’), ord(‘ ’), ord(‘ ‘), …],
 [ord(‘ ’), ord(‘ ’), ord(‘ ’), ord(‘ ‘), …],
 [ord(‘ ’), ord(‘ ’), ord(‘ ’), ord(‘ ‘), …],
 [ord(‘ ’), ord(‘ ’), ord(‘ ’), ord(‘ ‘), …],
 [ord(‘ ’), ord(‘ ’), ord(‘ ’), ord(‘ ‘), …],
 [ord(‘ ’), ord(‘ ’), ord(‘ ’), ord(‘ ‘), …],
…
]
```

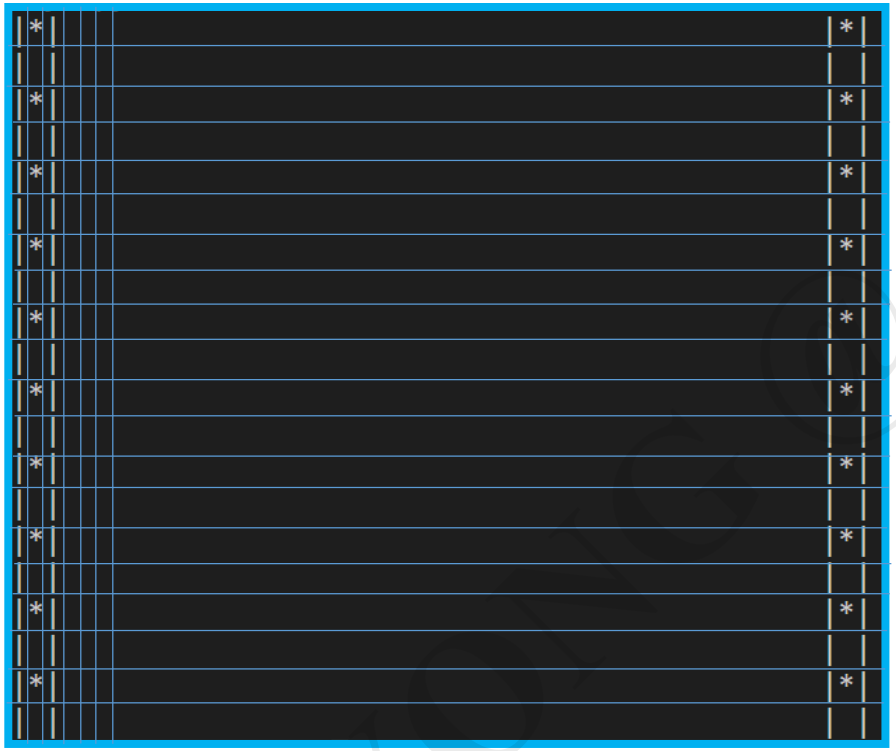
The scene is a 51-by-20 (character) array. Here we store it as an integer list.

How can we put curbs into the background?

The coordinate system



A better way for the background



```
[
 [ord('|'), ord('*'), ord('|'), ord(' '), …],
 [ord('|'), ord('*'), ord('|'), ord(' '), …],
 [ord('|'), ord('*'), ord('|'), ord(' '), …],
 [ord('|'), ord('*'), ord('|'), ord(' '), …],
 [ord('|'), ord('*'), ord('|'), ord(' '), …],
 [ord('|'), ord('*'), ord('|'), ord(' '), …],
 [ord('|'), ord('*'), ord('|'), ord(' '), …],
…
]
```

We put an object into the scene by updating the integers at a given position.

Put an object into the scene

```
6  # define a funtion to put an object into the background
7  # x, y are the corrdinates of the top-lef corner of the object
8  def put_object(background, x, y, object):
9      bg_width,bg_height=len(background[0]),len(background)
10     wid_obj,hei_obj=len(object[0]),len(object)
11     for i in range(hei_obj):
12         for j in range(wid_obj):
13             tag_x,tag_y=x+j,y+i
14             if object[i][j]==' ' or tag_x>=bg_width or tag_y>=bg_height: continue
15             background[tag_y][tag_x]=object[i][j]
16     return background
```

Put curbs into the scene

```
# put the left curb into the scene
cur_bg=put_object(background.copy(),0,0,curb)
# put the right curb into the scene
cur_bg=put_object(cur_bg,48,0,curb)
```

Let's display the scene

```
# print a scene
def print_scene(background):
    for row in background:
        row_str="".join([chr(d) for d in row])
        print(row_str)
```

We have to convert the list into strings before we an print it out.

Try This

```
# put the left curb into the scene
cur_bg=put_object(background.copy(),0,0,curb)
# put the right curb into the scene
cur_bg=put_object(cur_bg,48,0,curb)
# print the updated scene
print_scene(cur_bg)
```

Works like a charm!

Let's try put more stuff into the scene

Convert a string into a list of codes

```
startup="  _______    ______    ___       ___  ___\n"\
'  |    __ "\\  /    " \\  |"  |     |"  \\/"  |\n'\
'  (. |__) :)// ____  \\ ||  |     \\    \\  /\n'  \
'  |:  ____// /    ) :)|:  |      \\  \\/ \n' \
'  (|  /   (: (____/ //  \\  |__    /    / \n' \
' /|__/ \\   \\        /  ( \\_|:  \\  /    /  \n' \
'(_______)  \\"_____/    \\_______)|___/  \n' \
'                                  \n' \
'                                  \n' \
'  ______      _       ______     \n' \
' /"  _  "\\    /""\\     /"     \\   \n' \
'(: ( \\___)   /    \\    |:       |  \n' \
' \/ \\       /'' /\\  \\   |_____/   )  \n' \
' //  \\ _    //  __''  \\   //     /   \n' \
'(:   _) \\  /   /  \\  \\ |:  __   \\   \n' \
' \_______)(___/    \\___)|__|  \\___)  \n'\
'\n By Xiaoyong Wei @ COMP 1012\n'\
' SEP 24, 2022'
```

Convert a string into a list of codes

```
def encode2rect(pic):
    rows=pic.split('\n')
    rectcode_c=[]
    maxlen=0
    for row in rows:
        maxlen=max(maxlen,len(row))
    for row in rows:
        row_new=row+" "*(maxlen-len(row))
        rectcode_c.append(row_new)
    rectcode=[]
    for s in rectcode_c:
        rectcode.append([ord(c) for c in s])
    return rectcode

startcode=encode2rect(startup)
print(startcode)
```

Generate a logo as a sequence of ASCII codes

```
start_logo_code=[[32, 32, 32, 95, 95, 95, 95, 95, 95, 95, 32, 32, 32, 32, 95, 95, 95, 95,
32, 124, 32, 32, 32, 95, 95, 32, 34, 92, 32, 32, 47, 32, 32, 32, 32, 34, 32, 92, 32, 32, 1
95, 95, 95, 32, 32, 92, 32, 124, 124, 32, 32, 124, 32, 32, 32, 32, 32, 32, 92, 32, 32, 32,
32, 32, 32, 32, 92, 32, 32, 92, 47, 32, 32, 32], [32, 32, 40, 124, 32, 32, 47, 32, 32, 32,
32, 32, 32, 47, 32, 32, 32, 47, 32, 32, 32], [32, 47, 124, 95, 95, 47, 32, 92, 32, 32, 32,
32, 47, 32, 32, 32, 47, 32, 32, 32, 32], [40, 95, 95, 95, 95, 95, 95, 95, 41, 32, 32, 92,
58, 32, 32, 32, 32, 32, 32, 32, 32, 124, 32, 32, 32, 32, 32, 32, 32, 32], [32, 92, 47, 32,
95, 95, 95, 47, 32, 32, 32, 41, 32, 32, 32, 32, 32, 32, 32,
32, 32], [32, 47, 47, 32, 32, 92, 32, 95, 32, 32, 32, 32, 47, 47, 32, 32, 95, 95, 32, 32,
32, 47, 32, 32, 92, 32, 32, 92, 32, 124, 58, 32, 32, 95, 95, 32, 32, 32, 92, 32, 32, 32, 3
69, 80, 32, 50, 52, 44, 32, 50, 48, 50, 50, 32, 32, 32, 32,
32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,
32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32]]
```

Put the logo into the scene as a startup screen

```
# put logo into scene
scene_start=put_object(cur_bg.copy(),4,1,start_logo_code)
# display the updated scene
print_scene(scene_start)
```


By Xiaoyong Wei @ COMP 1012
SEP 24, 2022

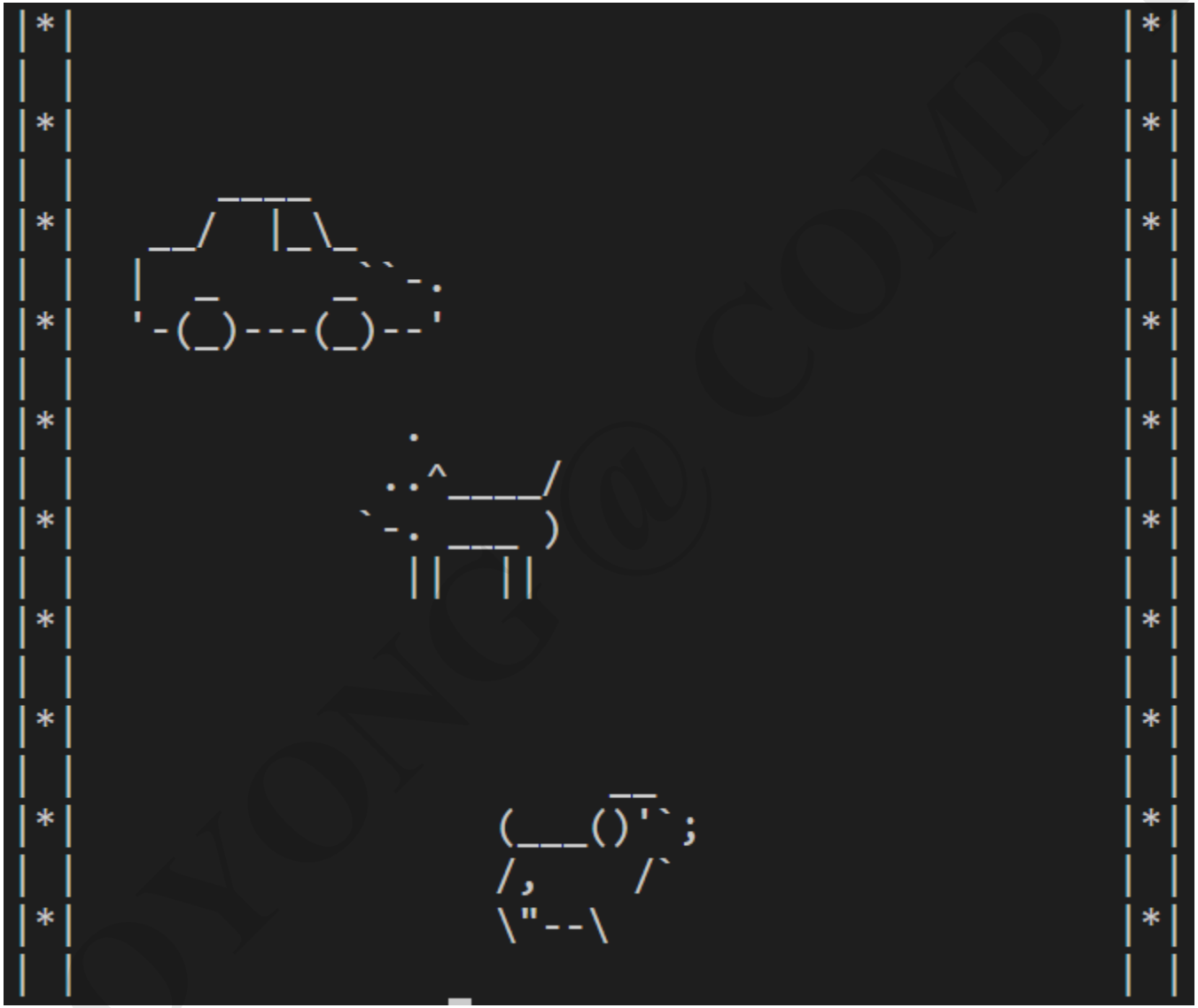
More objects

```
car_code=[[32, 32, 32, 32, 95, 95, 95, 95, 32, 32, 32, 32, 32,

dog1_code=[[32, 32, 46, 32, 32, 32, 32, 32, 32], [32, 46, 46, 94

dog2_code=[[32, 32, 32, 32, 32, 32, 32, 32, 95, 95, 32, 32], [32
```

```
# put the more object into a new scene
new_scene=put_object(copy.deepcopy(cur_bg),5,3,car_code)
new_scene=put_object(new_scene,15,8,dog1_code)
new_scene=put_object(new_scene,18,15,dog2_code)
# print the updated scene
print_scene(new_scene)
```



Day 2: Animation

We need to make the objects move in the scene

The idea is to update the positions of objects in the scene every tick.

Taking the car as an example, let's fix its horizontal position at the 15 row of the scene and let it move right by 1 character every tick.

Before that, we need a helper function which helps us to clean the old content in the screen. Otherwise, the new scene will be printed on the old ones. It's not an animation.

The Helper Function

```
LINE_UP = '\033[1A'
LINE_CLEAR = '\x1b[2K'
# a function to clean the printed content on screen
def clean_screen():
    for idx in range(20):
        print(LINE_UP, end=LINE_CLEAR)
```

Move the car

```
car_pos=[15,4] #set the initial car position at y=15, x=4
car_speed=[0,1]
while True:
    # clean the screen before printing new content
    clean_screen()
    car_pos=[car_pos[0]+car_speed[0],car_pos[1]+car_speed[1]]
    new_scene=put_object(copy.deepcopy(cur_bg),car_pos[1],car_pos[0],car_code)
    # print the updated scene
    print_scene(new_scene)
    time.sleep(0.1)
```

Now, let's control the car using the keyboard rather setting a fixed speed.

More specifically, we need the car to move left/right by 1 character when the left/right arrow key is pressed.

To this end, we need the help of a library called pynput which comes with a keyboard listener

```
car_pos=[15,4] #set the initial car position at y=15, x=4
car_speed=[0,1]

from pynput import keyboard
def on_press(key):
    global car_pos
    if key == keyboard.Key.esc:
        return False  # stop listener
    try:
        k = key.char  # single-char keys
    except:
        k = key.name  # other keys
    if k=='right':
        car_pos=[car_pos[0],car_pos[1]+1]
    elif k=='left':
        car_pos=[car_pos[0],car_pos[1]-1]


listener = keyboard.Listener(on_press=on_press)
listener.start()  # start to listen on a separate thread
```

```
listener = keyboard.Listener(on_press=on_press)
listener.start()  # start to listen on a separate thread

while True:
    # clean the screen before printing new content
    clean_screen()
    #car_pos=[car_pos[0]+car_speed[0],car_pos[1]+car_speed[1]]
    new_scene=put_object(copy.deepcopy(cur_bg),car_pos[1],car_pos[0],car_code)
    # print the updated scene
    print_scene(new_scene)
    time.sleep(0.1)
```

Day 3: Put dogs in

The game is about moving the car left/right to avoid hitting on the dogs or curbs.

Let's pull the dogs in the scene.

We can generate a dog on the top of the scene and let it move downwards with a speed.

Its initiate position is randomly determined at somewhere between the two curbs.

```
import random
# a list of dog genres
dog_genres=[dog1_code, dog2_code]
# a list to record the dogs in the scene
# each dog is record with a genre and a position
list_dogs=[]
# max #dog can be put into the scene
max_num_dogs=1
# speed of dogs at y and x directions
dog_speed=[1,0]
```

```python
def udpate_dogs(scene):
    global list_dogs, dog_speed
    dog2remove=[]
    for doginfo in list_dogs:
        # update position of each dog
        doginfo[1]=[doginfo[1][0]+dog_speed[0],doginfo[1][1]+dog_speed[1]]
        if doginfo[1][0]>scene_height:
            # the dog runs out from the scene, remove it
            dog2remove.append(doginfo)
    for doginfo in dog2remove:
        list_dogs.remove(doginfo)
    if len(list_dogs)<max_num_dogs:
        # generate a dog by chance (a probability of 30%)
        if random.randint(0,100)<30:
            dog_gen=dog_genres[random.randint(0,1)]
            hei_dog,wid_dog=len(dog_gen),len(dog_gen[0])
            pos_init=[0,random.randint(4,scene_width-3-wid_dog-1)]
            list_dogs.append([dog_gen,pos_init])
    for doginfo in list_dogs:
        scene=put_object(scene,doginfo[1][1],doginfo[1][0],doginfo[0])
    return scene
```

```
while True:
    # clean the screen before printing new content
    clean_screen()
    #car_pos=[car_pos[0]+car_speed[0],car_pos[1]+car_speed[1]]
    new_scene=put_object(copy.deepcopy(cur_bg),car_pos[1],car_pos[0],car_code)
    # udpate dogs
    new_scene=udpate_dogs(new_scene)
    # print the updated scene
    print_scene(new_scene)
    time.sleep(0.2)
```

Day 4: Collisions

Dogs now are moving in the scene and we can control the car as well. But it's a not game yet.

We need to need to know whether the car is hitting on a curb/dog.

This can be done easily by slightly modifying the put_object function.

```python
# define a funtion to put an object into the background
# x, y are the corrdinates of the top-lef corner of the object
def put_object(bg, x, y, object):
    bg_width,bg_height=len(bg[0]),len(bg)
    wid_obj,hei_obj=len(object[0]),len(object)
    crashed=False
    for r in range(hei_obj):
        for c in range(wid_obj):
            tag_x,tag_y=x+c,y+r
            if object[r][c]==32 or tag_x>=bg_width or tag_y>=bg_height: continue
            # in case the target scene position (ie., bg[tag_y][tag_x]) is
            # with a non-space chacter, the car is hitting something (curb/dog)
            if not bg[tag_y][tag_x]==32:
                crashed=True
            bg[tag_y][tag_x]=object[r][c]
    return crashed,bg
```

We have to modify at where the put_object is called.

At Scene Generation

```
# put the left curb into the scene
crashed,cur_bg=put_object(copy.deepcopy(background),0,0,curb)
# put the right curb into the scene
crashed,cur_bg=put_object(cur_bg,scene_width-3,0,curb)
# print the updated scene
#print_scene(cur_bg)

# put logo into scene
crashed,scene_start=put_object(copy.deepcopy(cur_bg),20,1,start_logo_code)
```

```python
def udpate_dogs(scene):
    global list_dogs, dog_speed
    dog2remove=[]
    for doginfo in list_dogs:
        # update position of each dog
        doginfo[1]=[doginfo[1][0]+dog_speed[0],doginfo[1][1]+dog_speed[1]]
        if doginfo[1][0]>scene_height:
            # the dog runs out from the scene, remove it
            dog2remove.append(doginfo)
    for doginfo in dog2remove:
        list_dogs.remove(doginfo)
    if len(list_dogs)<max_num_dogs:
        # generate a dog by chance (a probability of 30%)
        if random.randint(0,100)<30:
            dog_gen=dog_genres[random.randint(0,1)]
            hei_dog,wid_dog=len(dog_gen),len(dog_gen[0])
            pos_init=[0,random.randint(4,scene_width-3-wid_dog-1)]
            list_dogs.append([dog_gen,pos_init])
    for doginfo in list_dogs:
        crashed,scene=put_object(scene,doginfo[1][1],doginfo[1][0],doginfo[0])
    return scene
```

At Car Update

```
while True:
    # clean the screen before printing new content
    clean_screen()
    # udpate dogs
    new_scene=udpate_dogs(copy.deepcopy(cur_bg))
    # put car into the scene
    crashed,new_scene=put_object(new_scene,car_pos[1],car_pos[0],car_code)
    # check if crashed
    if crashed:
        break
    # print the updated scene
    print_scene(new_scene)
    time.sleep(0.2)

print("**************** Game Over! ****************")
```

But it would be nice to display a crash sign and reset the game.

```
def reset_game():
    global list_dogs,car_pos
    list_dogs=[]
    car_pos=[15,6]

while True:
    # clean the screen before printing new content
    clean_screen()
    # udpate dogs
    new_scene=udpate_dogs(copy.deepcopy(cur_bg))
    # put car into the scene
    crashed,new_scene=put_object(new_scene,car_pos[1],car_pos[0],car_code)
    # check if crashed
    if crashed:
        crashed,new_scene=put_object(new_scene,20,3,crashedcode)
        print_scene(new_scene)
        time.sleep(2)
        reset_game()
        continue
    # print the updated scene
    print_scene(new_scene)
    time.sleep(0.2)

print("**************** Game Over! ****************")
```


Thank you!

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學
Department of Computing
電子計算學系

Thank you!