GAME #3

“RED ALERT”

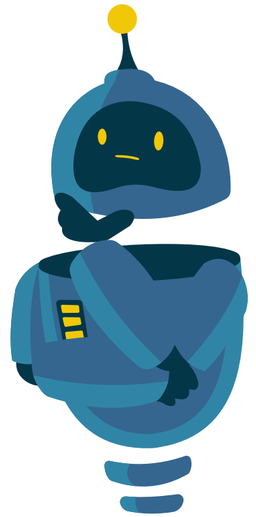
VORTEX ACADEMY
THE ART OF THINKING

VORTEX ACADEMY
THE ART OF THINKING

WELCOME BACK!

CAN YOU GUESS WHAT WE'LL CREATE TODAY?

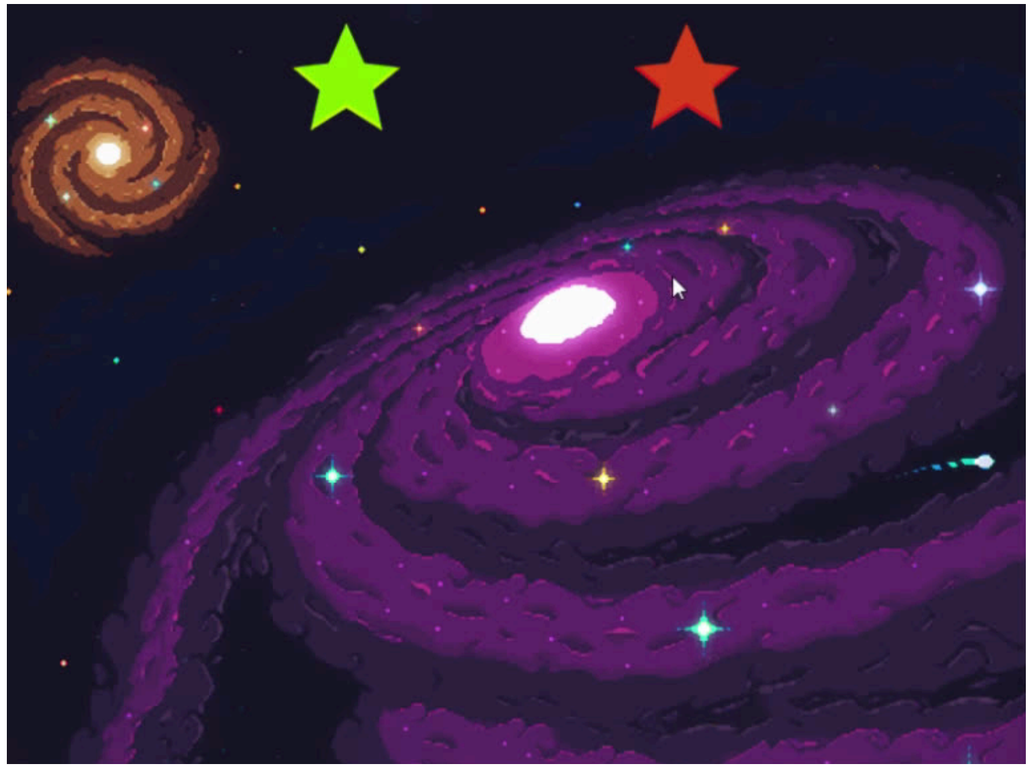
VORTEX ACADEMY
THE ART OF THINKING

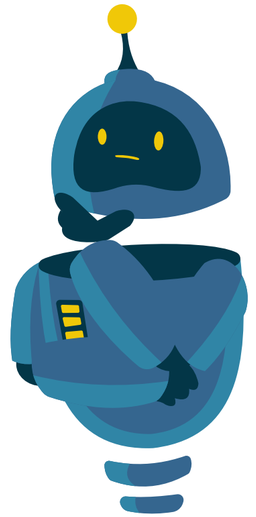


HOW DO WE MAKE RED ALERT?

VORTEX ACADEMY
THE ART OF THINKING

Using what you've learned from the previous sessions, how do you think we could build RED ALERT?





HOW DO WE MAKE RED ALERT?

VORTEX ACADEMY
THE ART OF THINKING

TOPICS

VORTEX ACADEMY
THE ART OF THINKING

01

More Random Options & Animate( )

- More options to vary the choices in our game.
- A new way to animate our actors.

02

Game Logic & More Text Options

- How do we make our game work?
- How to change our text's size, font and style?

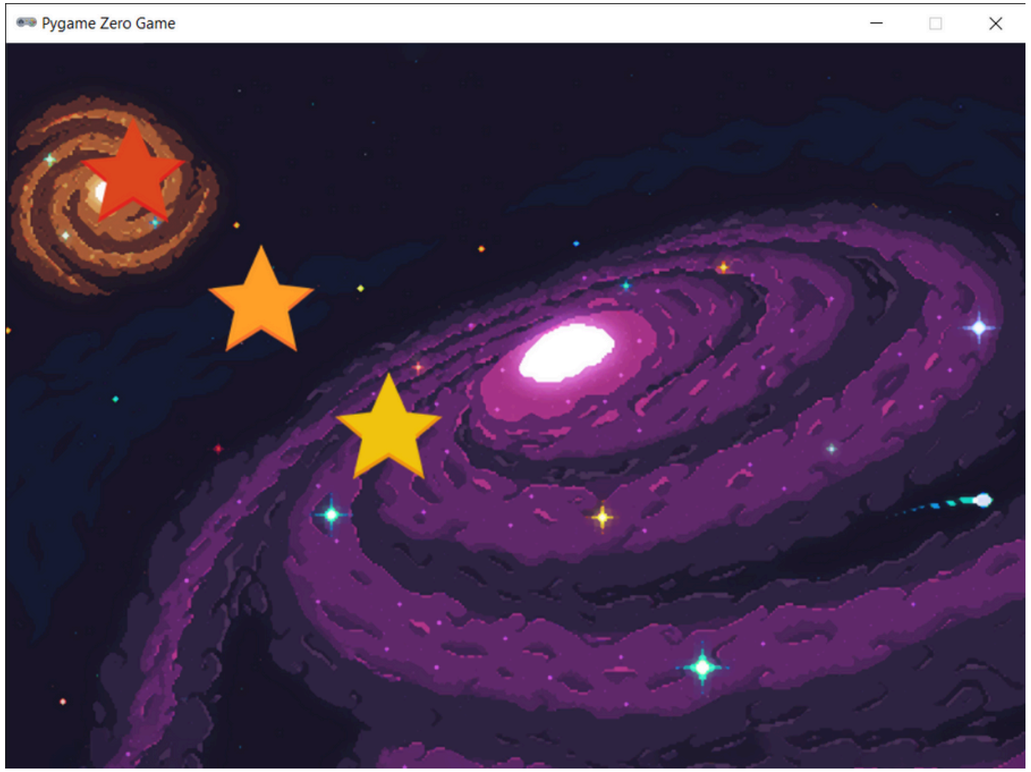
03

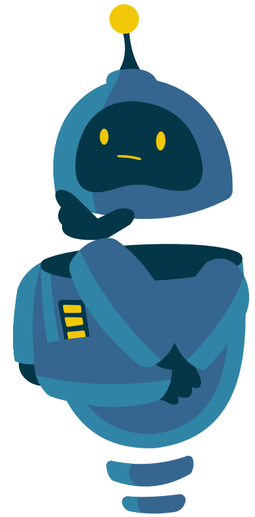
Music & Audio & RED ALERT!

- How do we add music and sound effects to our game?
- Building RED ALERT!

LET'S START

Use what you've learned to make the background space and make a list of stars then add three stars to it colored yellow, orange and red and place them anywhere on the screen.
You can look at your previous code and copy from it.





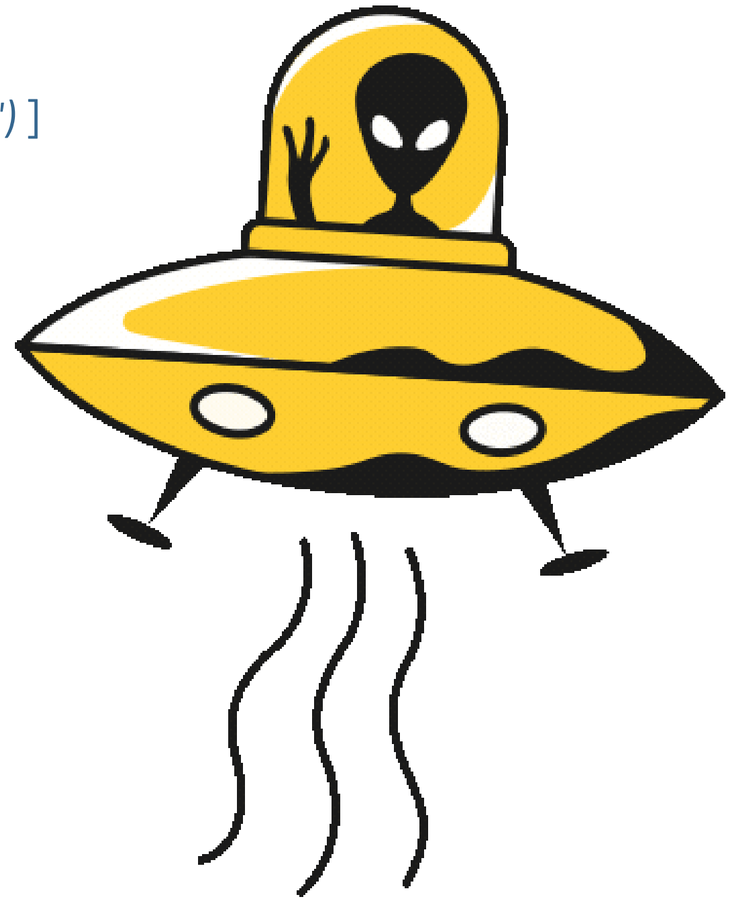
VORTEX ACADEMY
THE ART OF THINKING

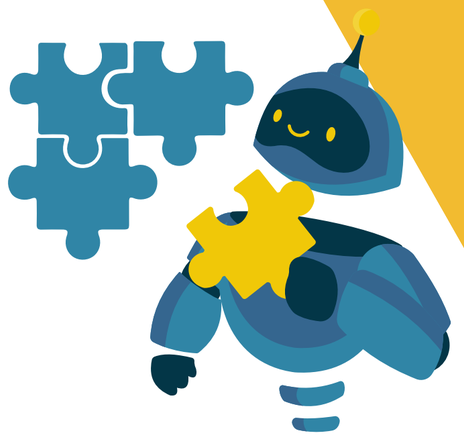
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#1

Try the following code

```
import pgzrun

WIDTH = 800
HEIGHT = 600

stars = [ Actor("red"), Actor("orange"), Actor("yellow") ]
stars[0].pos = 100, 100
stars[1].pos = 200, 200
stars[2].pos = 300, 300

def draw():
    screen.clear()
    screen.blit("space", (0, 0))
    for star in stars:
        star.draw()

def update():
    pass

pgzrun.go()
```

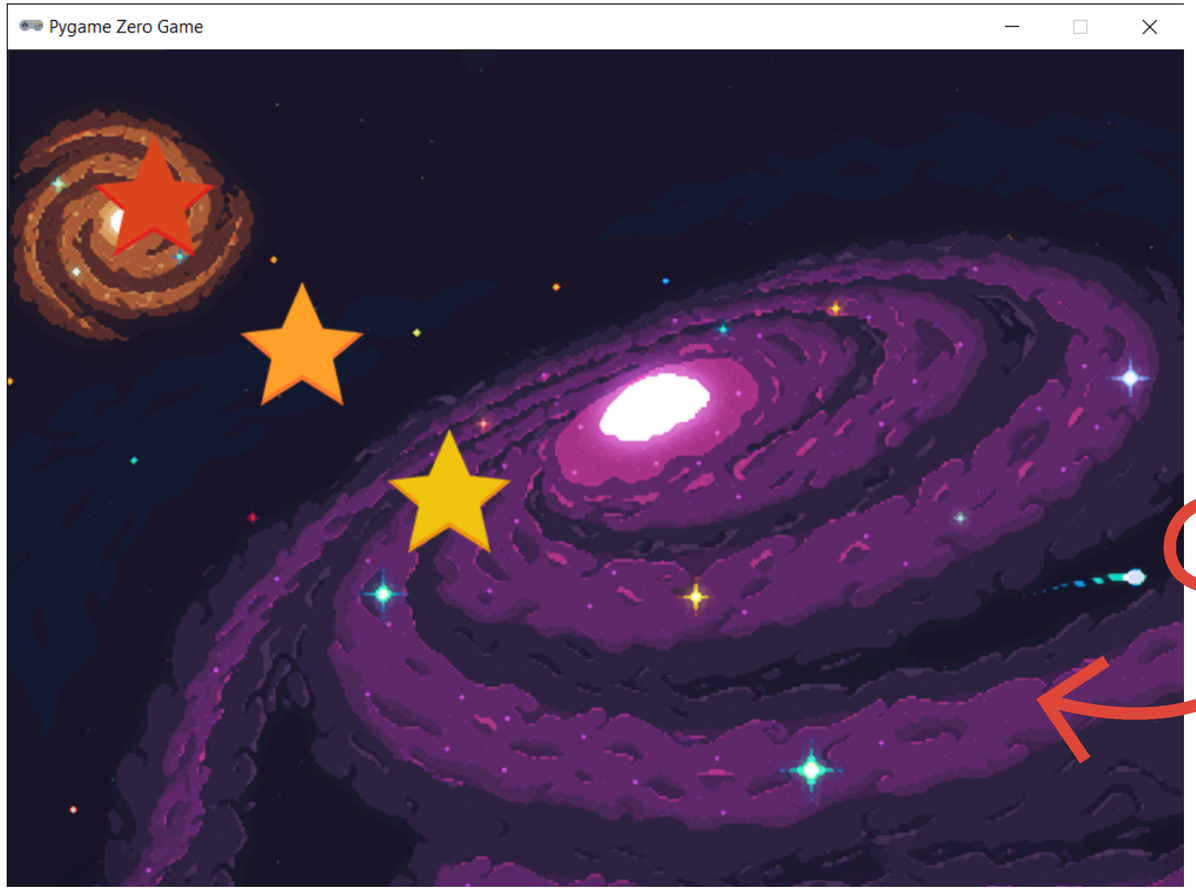




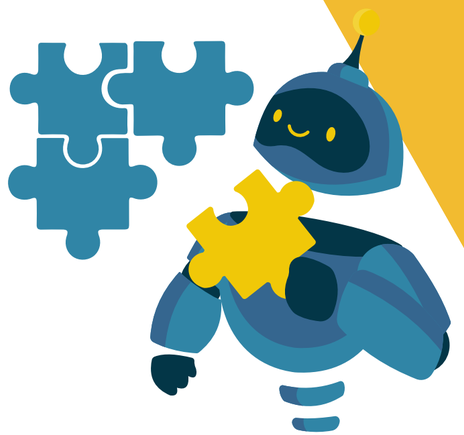
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#1

OUTPUT:



Now we have setup the base for our game, but how do we add and correctly place the rest of the stars?

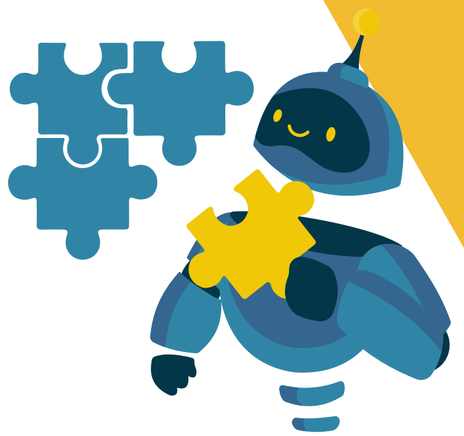


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#2.1

Try in another script the following code

```
import random

colors = ["red", "blue", "green", "yellow", "purple", "orange", "white", "black"]

for _ in range(0, 10):
    print(random.choice(colors))
```



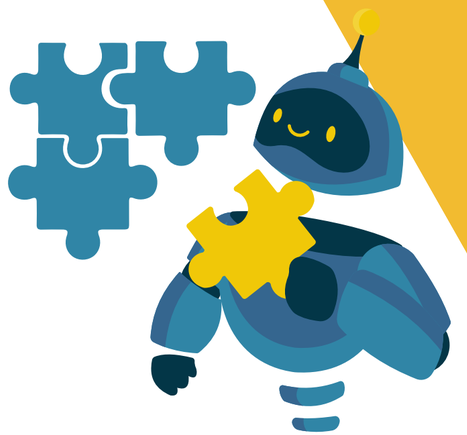
ACTIVITY#2.1

VORTEX ACADEMY
THE ART OF THINKING

OUTPUT:

black
blue
blue
purple
green
green
red
red
black
yellow

random.**choice()** chose a random element from our list to print every time.

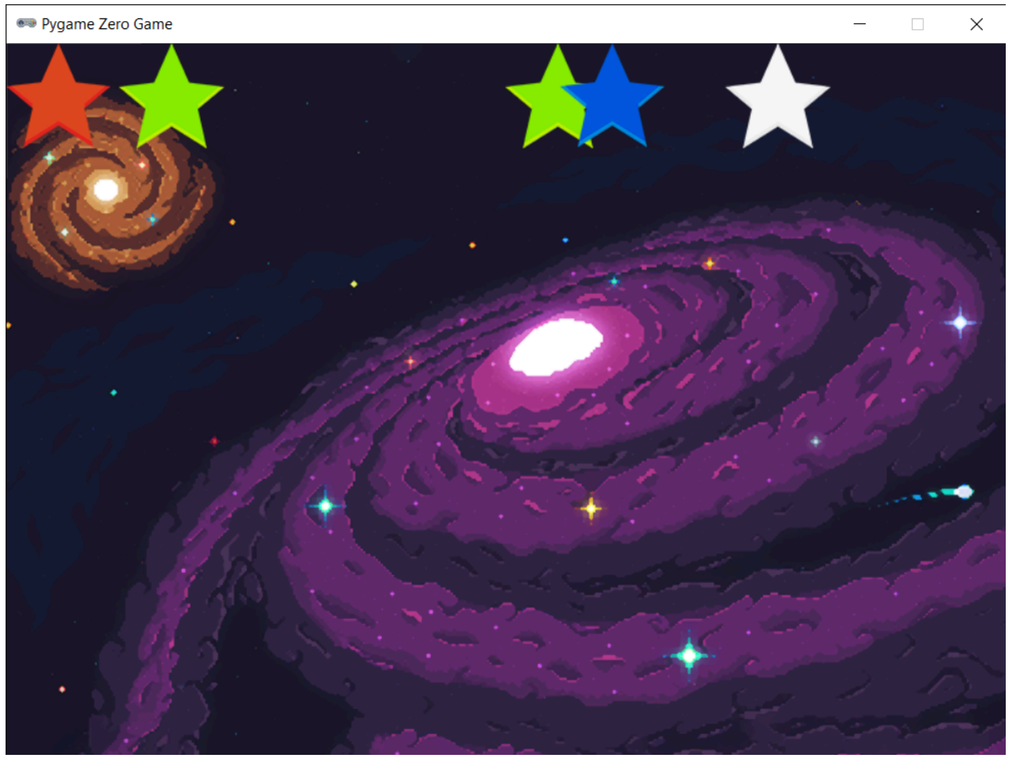


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#2.2

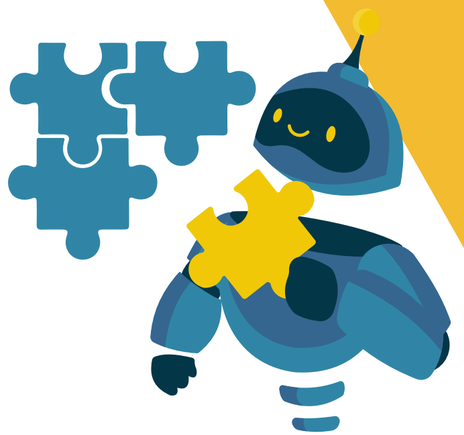
To Continue Our Game:

How can we use random.choice() to add randomly colored stars to our game screen?



Create a function **createStars**() that adds a *red star* to our stars list then fills it with *randomly colored* stars equal to our **current level-1** from the colors: *"yellow"*, *"blue"*, *"green"*, *"purple"*, *"orange"*, *"pink"*, *"white" and "black"* using the images you have, give them a *random x position* then draw them to the screen.

#Hint: add the colors to a list and use random.**choice**() to pick from them

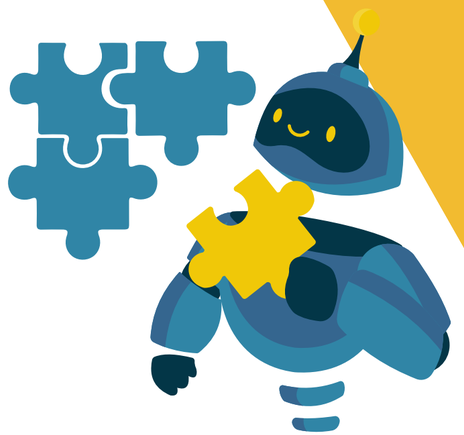


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#2.2

Modify Your Previous Code & Try:

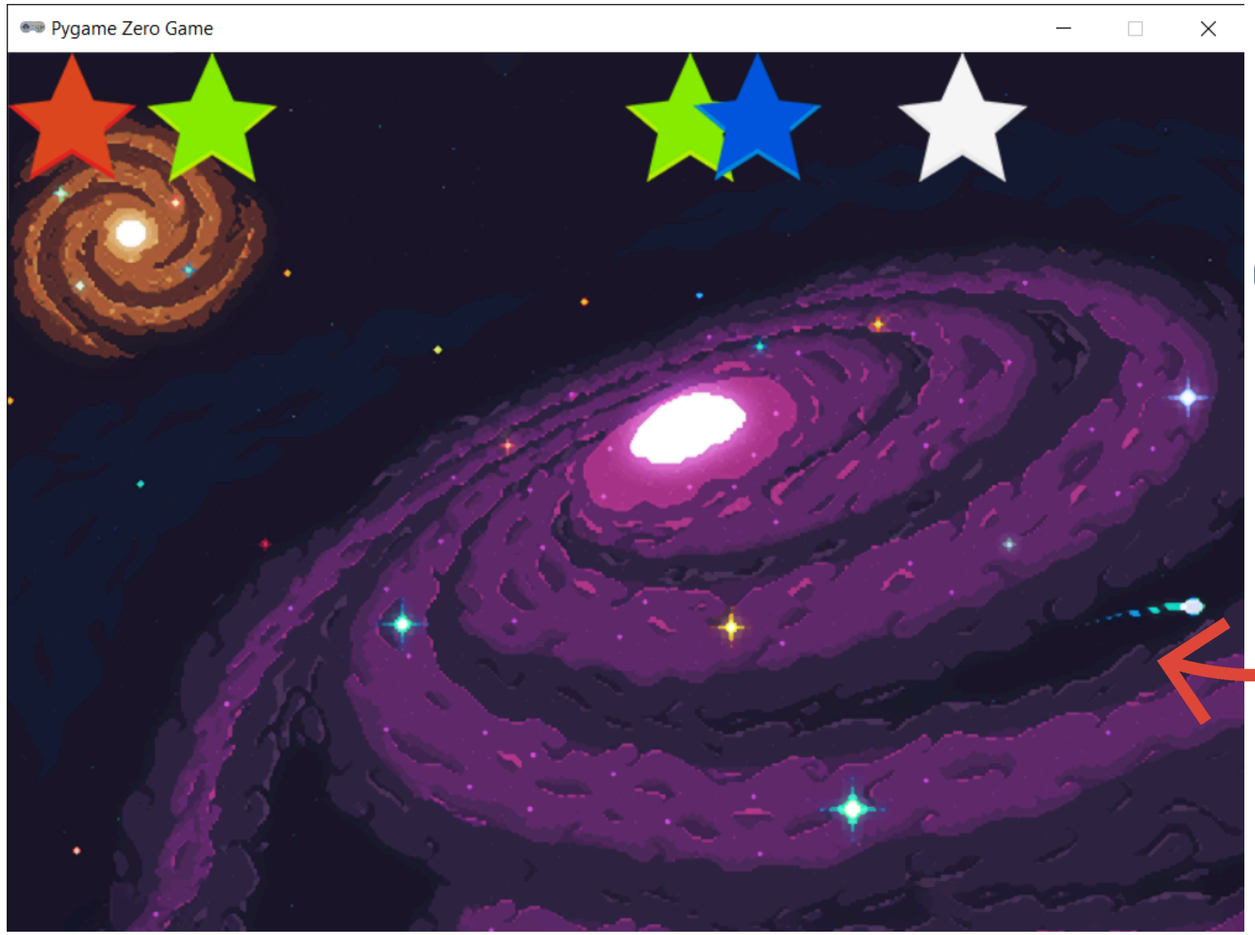
```
import random
# Add this to your game variables
currentLevel = 5 #Any number just to test
stars = [ ]
colors = [ "blue", "green", "yellow", "purple", "orange", "white", "black"]
# Add this function.
def createStars( ):
  global stars, colors, currentLevel

  stars.append(Actor("red"))
  for _ in range(currentLevel-1):
    star = Actor(random.choice(colors))
    star.x = random.randint(50, WIDTH - 50)
    stars.append(star)
# Modify this function.
def update( ):
  if len(stars) == 0:
    createStars( )
```



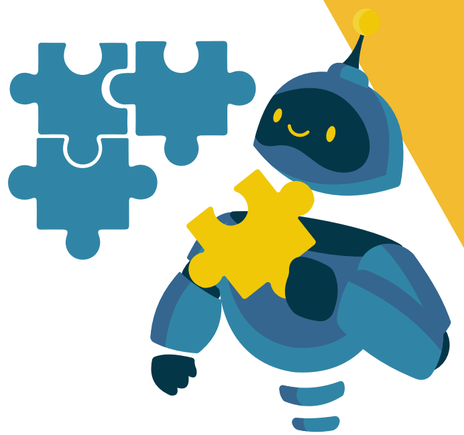
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#2.2

OUTPUT:



Now we have a number of stars equal to our current level

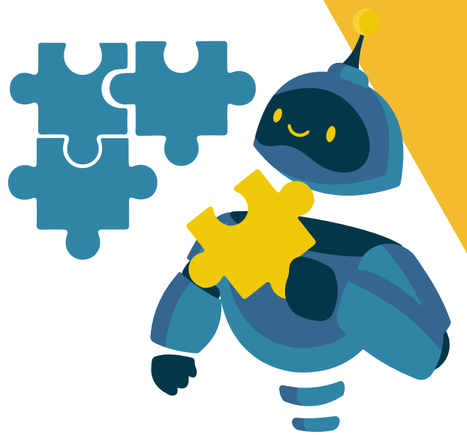


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#3.1

Try in another script the following code

```
import random

numbers = ["one", "two", "three", "four", "five", "six", "seven", "eight", "nine", "ten"]

random.shuffle(numbers)

for _ in numbers:
    print(_)
```



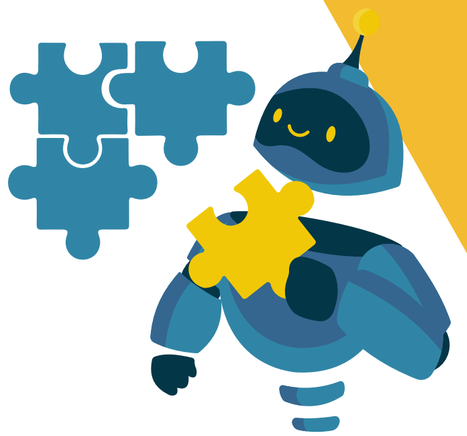
ACTIVITY#3.1

VORTEX ACADEMY
THE ART OF THINKING

OUTPUT:

four
five
six
two
eight
ten
seven
three
one
nine

random.**shuffle()**
shuffled our
numbers randomly.

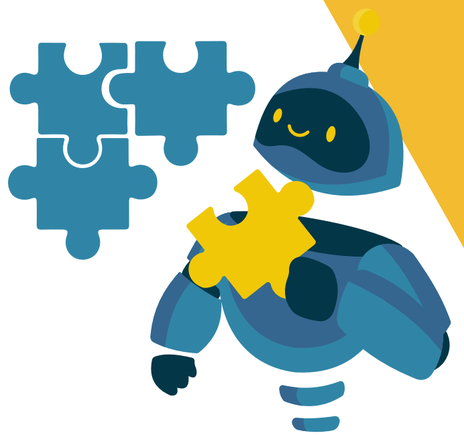


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#3.2

Try in another script the following code

```
# You can use the code from the previous script and modify the lines in yellow
import random

colors = ["yellow", "blue", "green", "purple", "orange", "pink", "white","black"]

random.shuffle(colors)

for _ in colors:
    print(_)
```



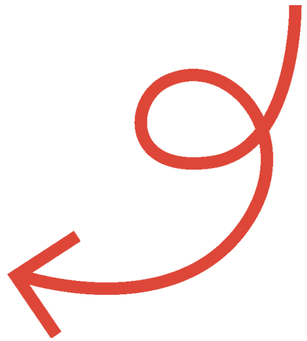
VORTEX ACADEMY
THE ART OF THINKING

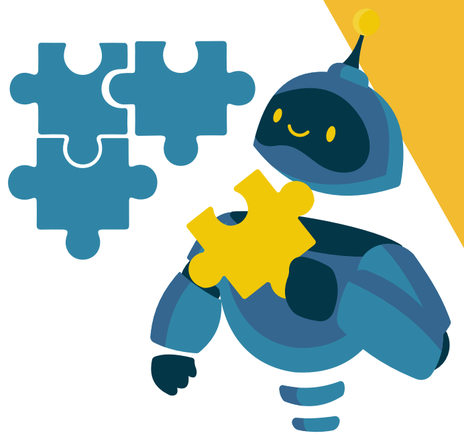
ACTIVITY#3.2

OUTPUT:

black
purple
orange
pink
yellow
white
blue
green

random.**shuffle()**
shuffled our colors randomly.





random.**shuffle()**

VORTEX ACADEMY
THE ART OF THINKING

When we want to shuffle our list randomly we use random.**shuffle**(LIST_NAME).

In the previous example we used random.**shuffle**() to shuffle our list of numbers and our list of colors.

Syntax:

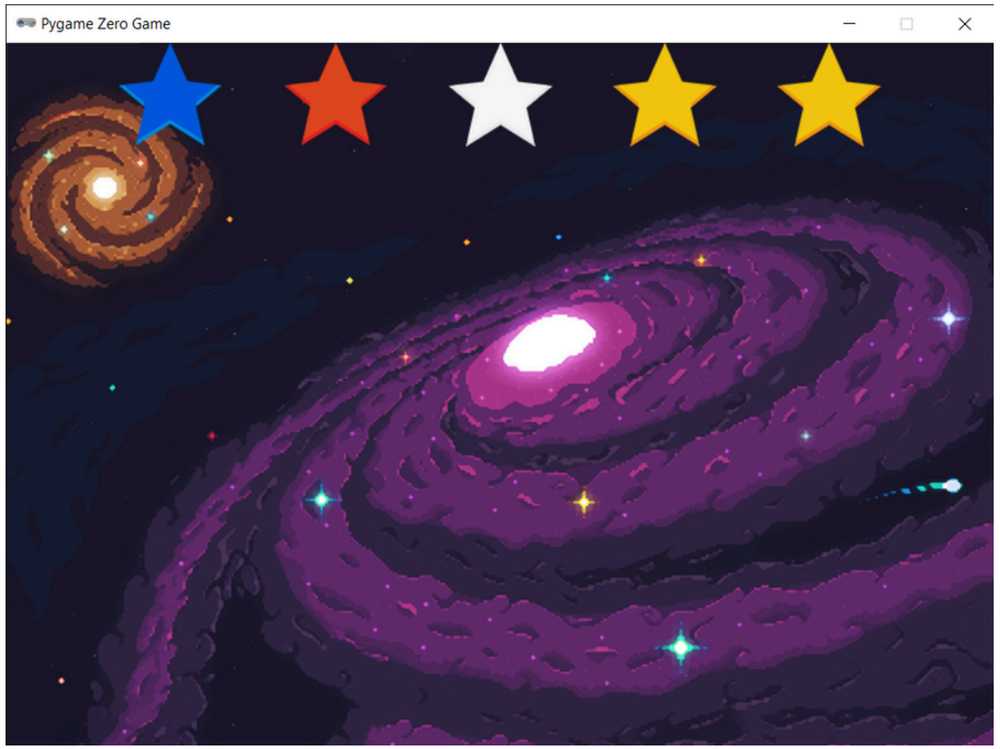
list = [*elements we want to suffle*]
random.**shuffle**(*list*)

VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#3.3

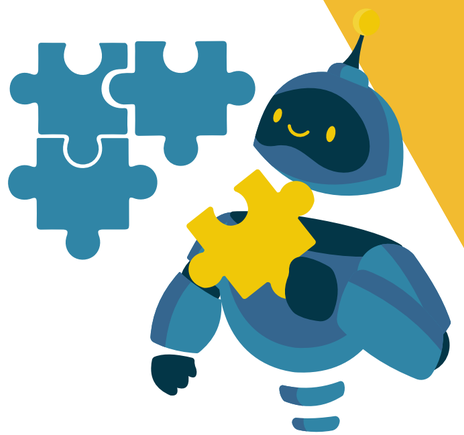
To Continue Our Game:

How can we use random.**shuffle()** **to change the order of the red star among the other stars randomly? How do we layout our stars correctly?**



In the function **createStars**() reorder the stars randomly using random.**shuffle**()then divide the **WIDTH** of the screen by the number of the stars+1 and use that to space the stars evenly across the screen.

#Hint: **WIDTH**/number of stars + 1 gives you the gap width between stars on the screen.
Create a variable distance = gapWidth
Give the first star an x value equal to distance then increase distance by one gap width for every star that follows

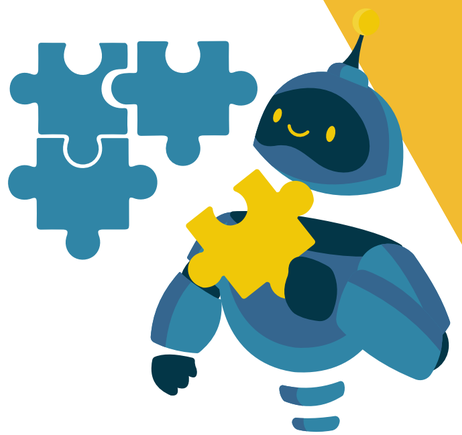


ACTIVITY#3.3

VORTEX ACADEMY
THE ART OF THINKING

Modify Your Previous Code & Try:

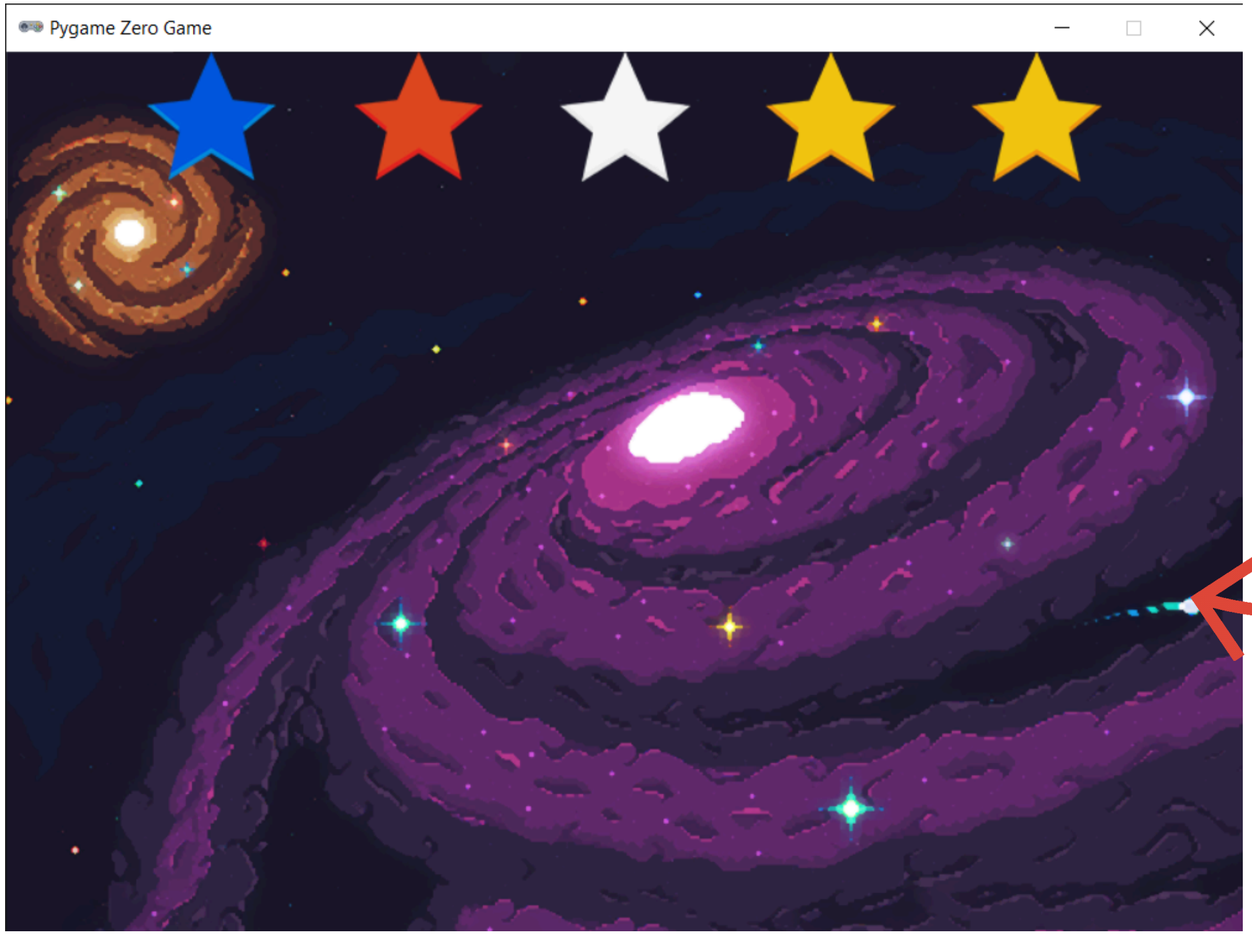
```
# Remove the lines in red and add the lines in yellow.
def createStars():
    global stars, colors, currentLevel, WIDTH
    stars.append(Actor("red"))
    for _ in range(currentLevel):
        star = Actor(random.choice(colors))
        star.x = random.randint(50, WIDTH - 50)
        stars.append(star)

    gapSize = WIDTH/(len(stars) + 1 )
    distance = gapSize
    random.shuffle(stars)
    for star in stars:
        star.x = distance
        distance += gapSize
```



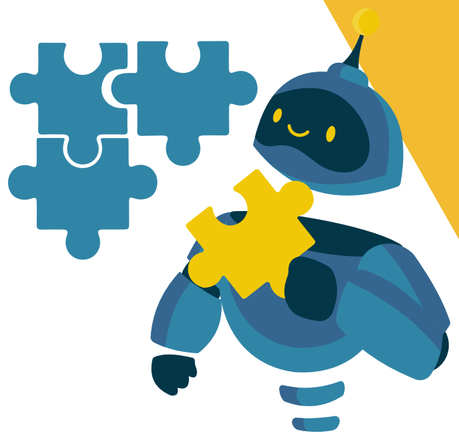
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#3.3

OUTPUT:



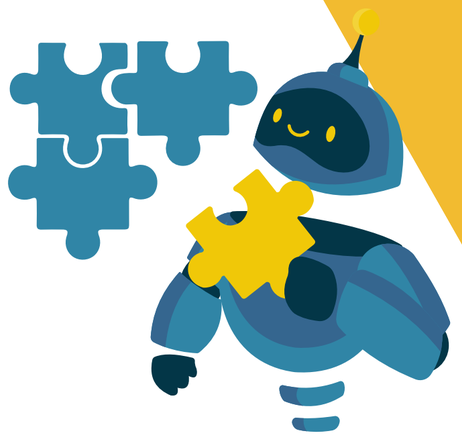
Now our stars are spaced evenly across the screen



ACTIVITY#4.1

Try in another script the following code

```
import pgzrun
import random

WIDTH = 800
HEIGHT = 600

redStar = Actor("red")

def draw():
    screen.clear()
    screen.fill("slate Grey")
    redStar.draw()

def update():
    redStar.x += 1
    if redStar.x > WIDTH:
        redStar.x = 0

pgzrun.go()
```



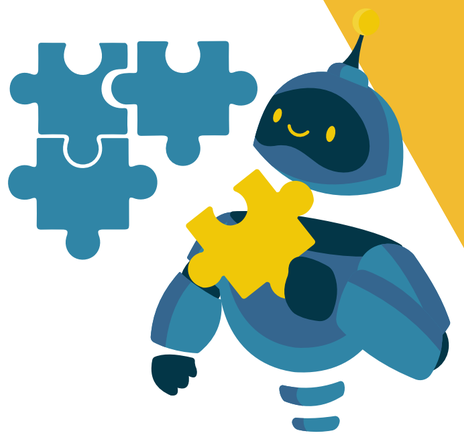
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#4.1

OUTPUT:

Now the red star moves across the screen.

#HINT: try changing the speed at which the star moves by changing the increase in x coordinate. Can you determine how long the star will take to get to a certain position?

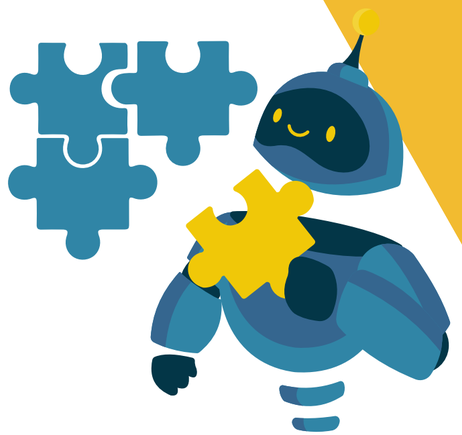


ACTIVITY#4.2

VORTEX ACADEMY
THE ART OF THINKING

Try in another script the following code

```
# You can use the code from the previous script and add the lines in yellow
import pgzrun
import random

WIDTH = 800
HEIGHT = 600

redStar = Actor("red")

def draw():
    screen.clear()
    screen.fill("slate Grey")
    redStar.draw()

def update():
    redStar.x += 1
    if redStar.x > WIDTH:
        redStar.x = 0
    redStar.y += 1
    if redStar.y > HEIGHT:
        redStar.y = 0

pgzrun.go()
```



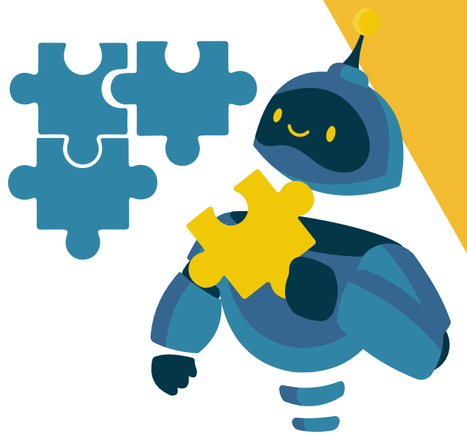
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#4.2

OUTPUT:

Now the star moves diagonally.

#HINT: try changing the direction in which the star moves by changing the increase in X or Y coordinates.
Can you determine where the star will be after a certain amount of time?

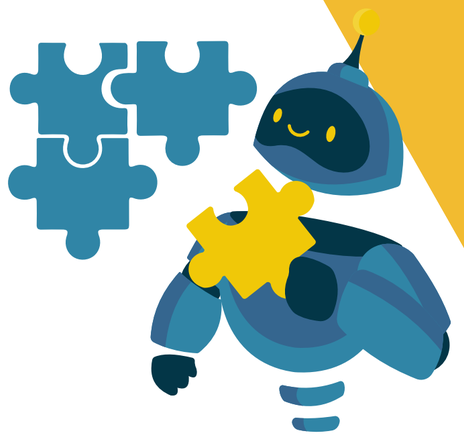


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#4.3

Try in another script the following code

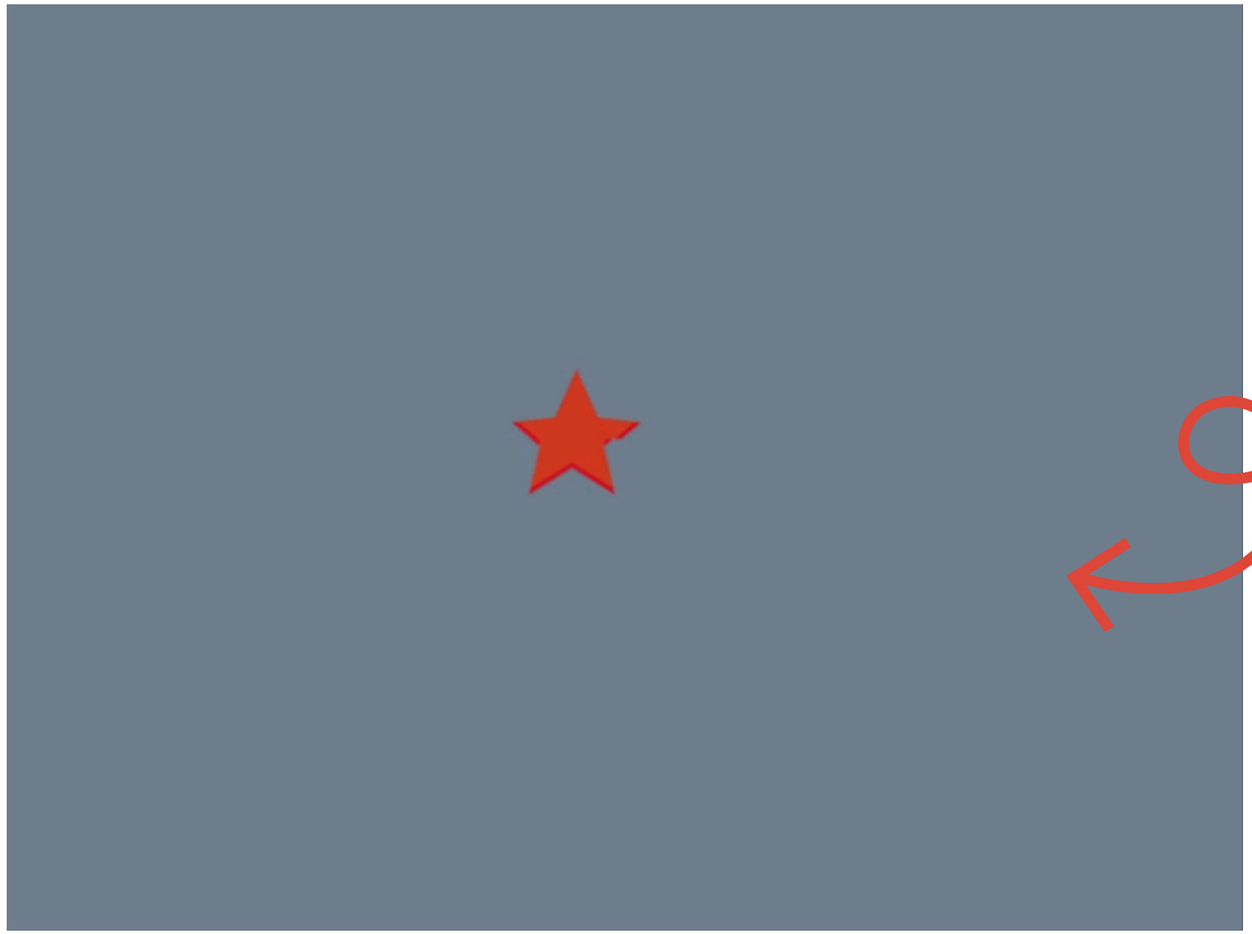
```
# You can use the code from the previous script and modify the lines in yellow.
import pgzrun
import random

WIDTH = 800
HEIGHT = 600

redStar = Actor("red")
starAnimation = animate(redStar, pos=(400, 300), duration=2, tween="linear")

def draw():
    screen.clear()
    screen.fill("slate Grey")
    redStar.draw()

def update():
    pass

pgzrun.go()
```



VORTEX ACADEMY
THE ART OF THINKING

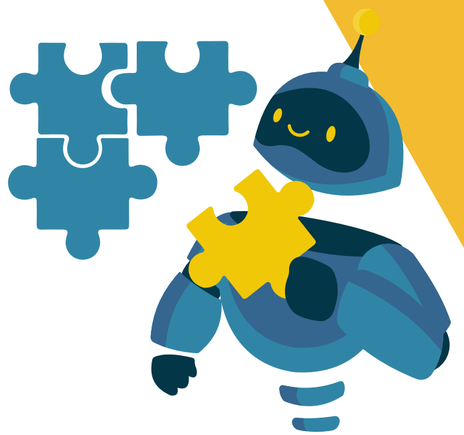
ACTIVITY#4.3

OUTPUT:



Animate() makes our animations easier and gives us more control over our animations

#HINT: try changing final location of the star (which is (400,300) in the previous example) and the time it takes to get there (which is 2 in the previous example)



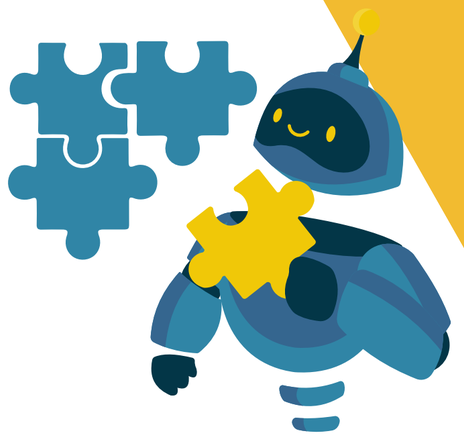
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#4.4

Try in another script the following code

```
# You can use the code from the previous script and modify the lines in yellow
import pgzrun
import random

WIDTH = 800
HEIGHT = 600

redStar = Actor("red")
starAnimation = animate(redStar, pos=(400, 300), duration=2, tween="accelerate")

def draw():
    screen.clear()
    screen.fill("slate Grey")
    redStar.draw()

def update():
    pass

pgzrun.go()
```

Tween Options



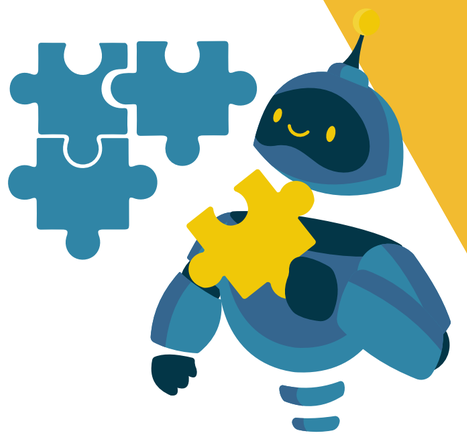
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#4.4

OUTPUT:

Tween controls how the animations look. In this case the star keeps getting faster until it reaches it's final position

#HINT: try changing the way the star moves in the previous slide by changing *tween* from the options in next slide



animate() Tween Options

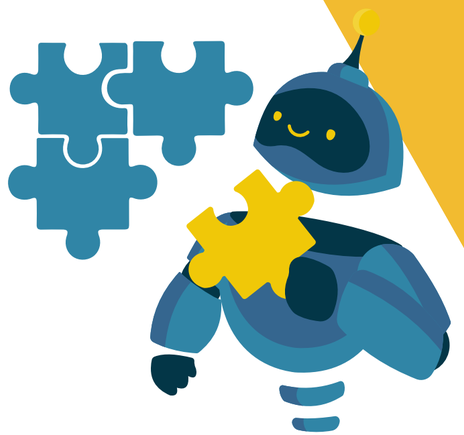
VORTEX ACADEMY
THE ART OF THINKING

‘linear’	*Animate at a constant speed from start to finish.*
‘accelerate’	*Start slower and accelerate to finish.*
‘decelerate’	*Start fast and decelerate to finish.*
‘accel_decel’	*Accelerate to mid point and decelerate to finish.*
‘end_elastic’	*Give a little wobble at the end.*
‘start_elastic’	*Have a little wobble at the start.*
‘both_elastic’	*Have a wobble at both ends.*
‘bounce_end’	*Accelerate to the finish and bounce there.*
‘bounce_start’	*Bounce at the start.*
‘bounce_start_end’	*Bounce at both ends.*

ACTIVITY#4.5

Try in another script the following code

```
# You can use the code from the previous script and modify
the lines in yellow
import pgzrun
import random

WIDTH = 800
HEIGHT = 600
def animation_finished():
    print("Animation Finished!")

redStar = Actor("red")
starAnimation = animate(redStar, pos=(400, 300), duration=2,
tween="bounce_end",on_finished=animation_finished)
def draw():
    screen.clear()
    screen.fill("slate Grey")
    redStar.draw()

def update():
    pass
pgzrun.go()
```



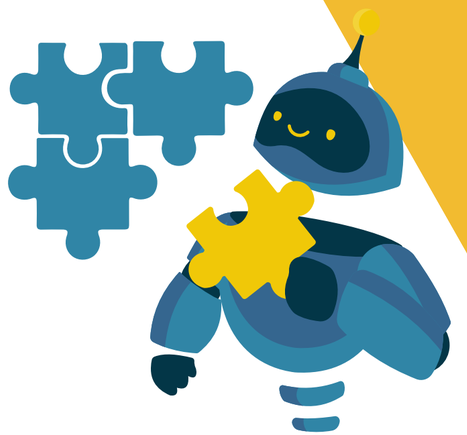
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#4.5

OUTPUT:

When the animation is finished “Animation Finished!” is printed to the terminal, Also notice the different tween.

#HINT: try changing what happens when the animation is finished, print something else or restart the animation.



VORTEX ACADEMY
THE ART OF THINKING

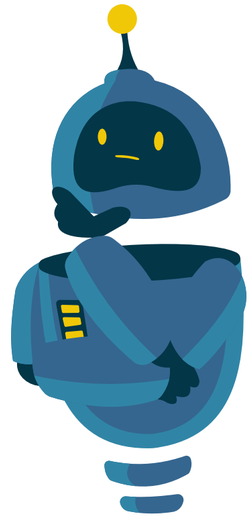
animate()

animation = animate(ACTOR_NAME, pos = (FINAL_X,FINAL_Y), duration = DURATION, tween = OPTION, on_finished = FUNCTION_TO_RUN_WHEN_DONE) allows us to animate the game actors much easier by specifying their end position or end x or y coordinate with many more options to control the animation, as tween and on_finished.
It returns an object that we can use to control the animation later.

In the previous example we used **starAnimation** = **animate**(*redStar, pos=(400, 300), duration=2, tween="bounce_end",on_finished=animation_finished*) to animate our red star from it's starting position to (400,300) with a duration of 2 seconds and a tween of bounce_end, when the animation was finished "Animation Finished!" was printed to the screen.

Syntax:

animation = **animate**(*ACTOR_NAME, pos = (FINAL_X,FINAL_Y), duration = DURATION, tween = OPTION, on_finished = FUNCTION_TO_RUN_WHEN_DONE*)

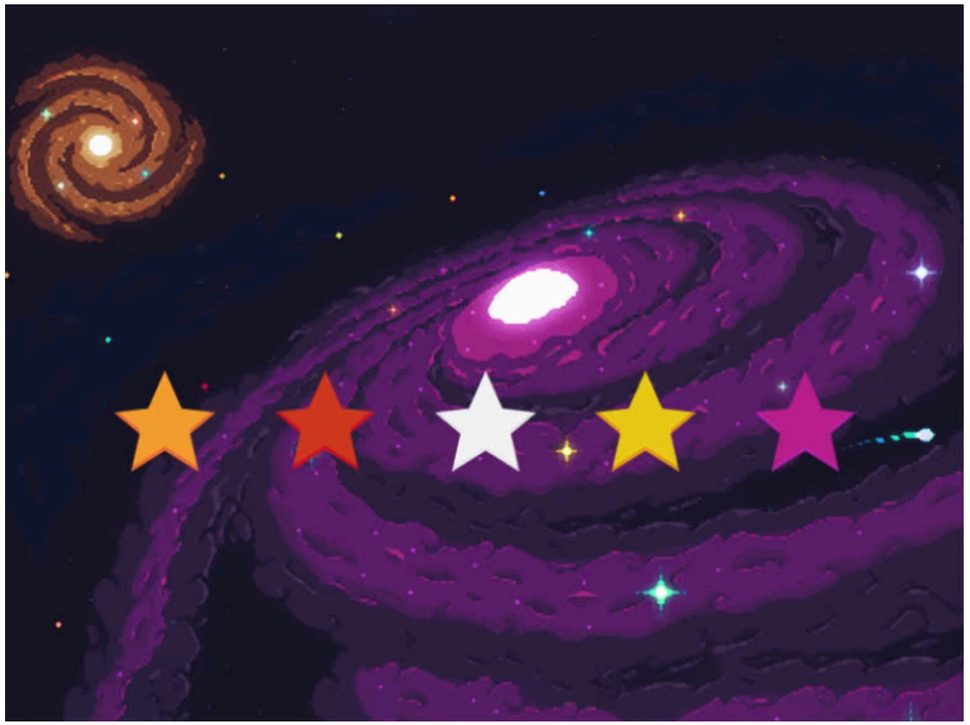


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#4.6

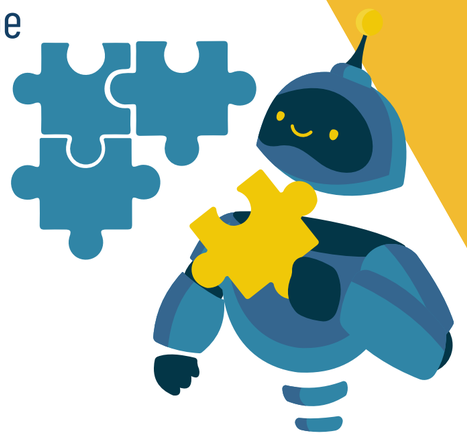
To Continue Our Game:

How can we use animate() to animate our stars motion down the screen?



Create a list for the animations and In the function **createStars**() animate the stars and add their animations to the list for later use.
The duration of the the animation should decrease with each level.
Use a placeholder function **endGame**() to be called when the animation ends.

#Hint: make the duration equal the final level number (10) - current level.

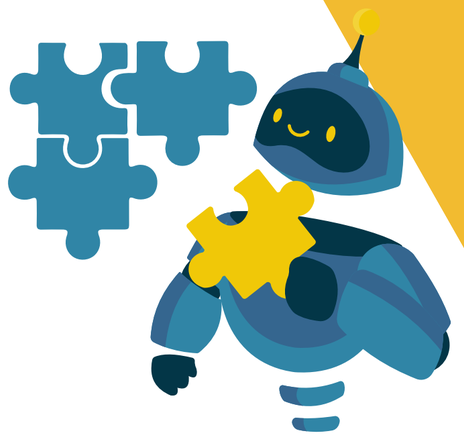


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#4.6

Modify Your Previous Code & Try:

```
# Add to the game variables
FINAL_LEVEL = 10
animations = [ ]

def createStars():

    # Add this section to animate the stars
    duration = FINAL_LEVEL - currentLevel
    for star in stars:
        animation = animate(star, y = HEIGHT,
duration=duration, on_finished=endGame)
        animations.append(animation)
# Add this function.
def endGame():
    pass
```

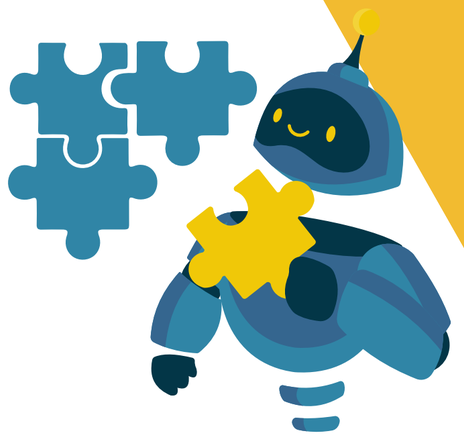


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#4.6

OUTPUT:

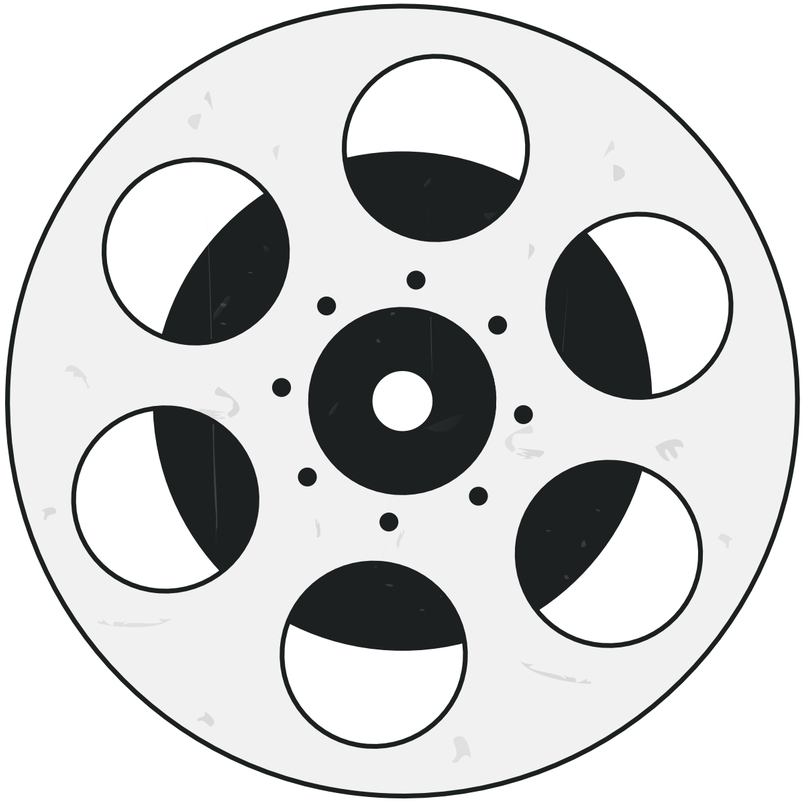
Now our stars move down the screen according to our animation.

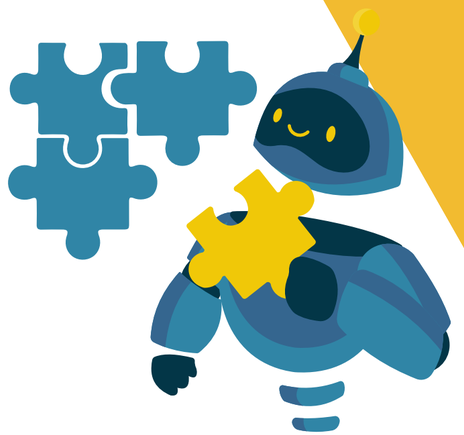


ACTIVITY#5.1

Try in another script the following code

```
# You can use the code from the previous script and modify the
lines in yellow
import pgzrun
import random

WIDTH = 800
HEIGHT = 600

def animation_finished():
    print("Animation Finished!")

redStar = Actor("red")
starAnimation = animate(redStar, pos=(400, 300), duration=2,
tween="bounce_end",on_finished=animation_finished)

def draw():
    screen.clear()
    screen.fill("slate Grey")
    redStar.draw()

def on_mouse_down(pos):
    global starAnimation
    if starAnimation.running:
        starAnimation.stop()

def update():
    pass
pgzrun.go()
```

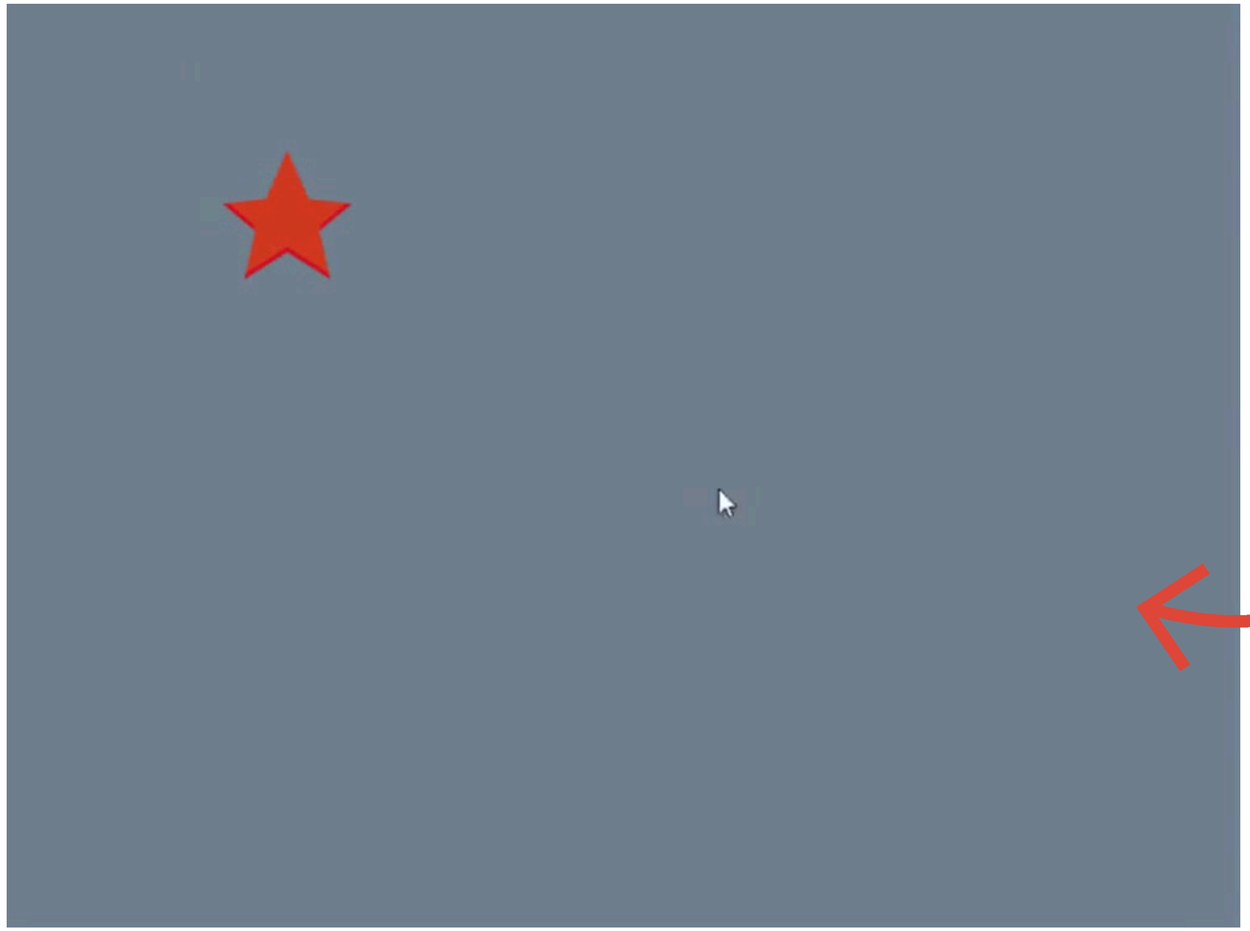




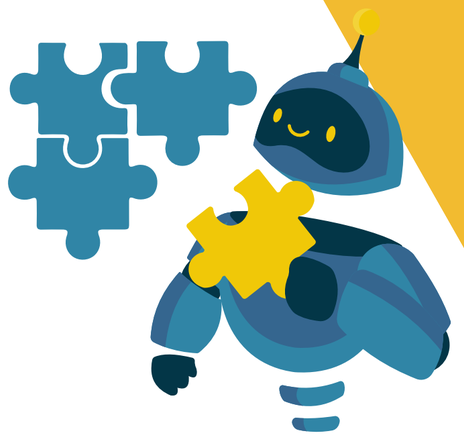
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#5.1

OUTPUT:



Now our animation continues until we click on it.



VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#5.2

To Continue Our Game:

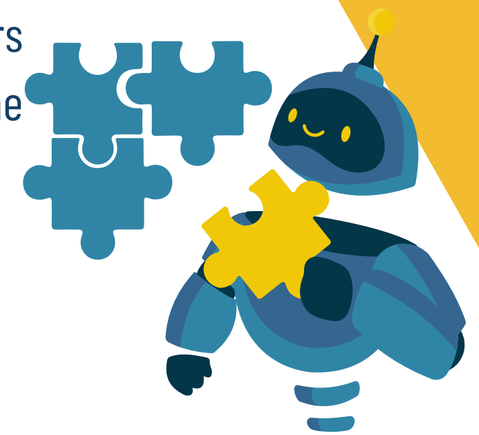
How can we stop the animations? How can we make the stars disappear when one of them reaches the bottom of the screen?

Create a function **removeStars**() that empties the list of stars and stops all animations.

Create a global variable **gameOver** that stores the game state.

Make the game end when one of the stars reachs the bottom of the screen using the **endGame**() function.

#Hint: use **endgame**() to set **gameOver** so the update function can remove the stars.

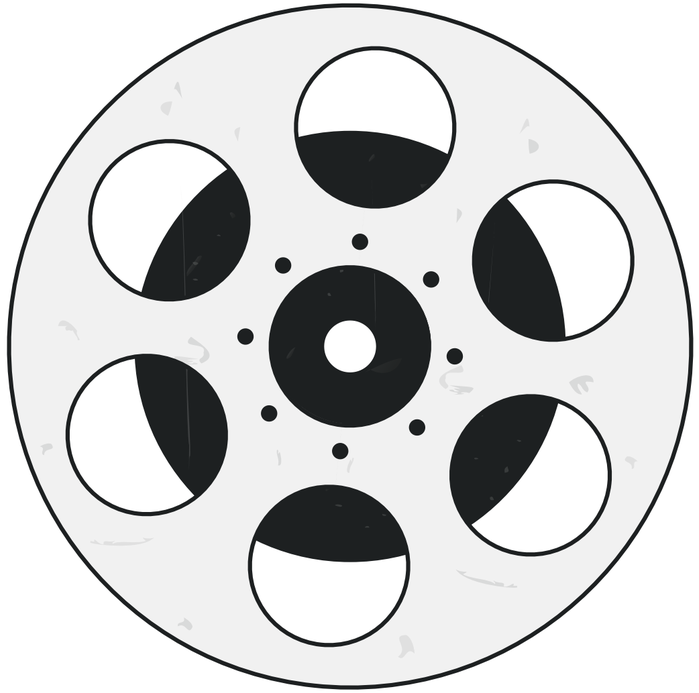


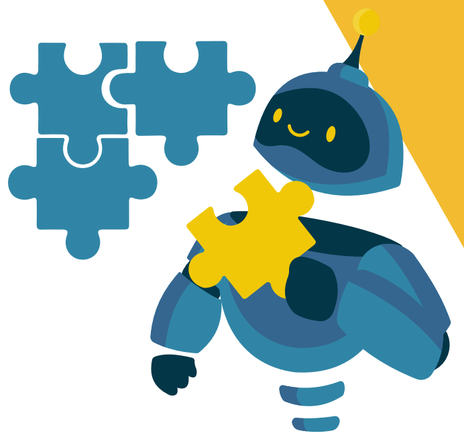
ACTIVITY#5.2

VORTEX ACADEMY
THE ART OF THINKING

Modify Your Previous Code & Try:

```
# Add this to your game variables.
gameOver = False
# Add this function.
def removeStars():
    global stars,animations
    stars = [ ]
    for animation in animations:
        if animation.running:
            animation.stop()
#Modify this function with the lines in yellow.
def endGame():
  global gameOver
  gameOver = True
#Modify this function with the lines in yellow.
def update():
  if len(stars) == 0:
    createStars()
  if gameOver:
    removeStars()
```



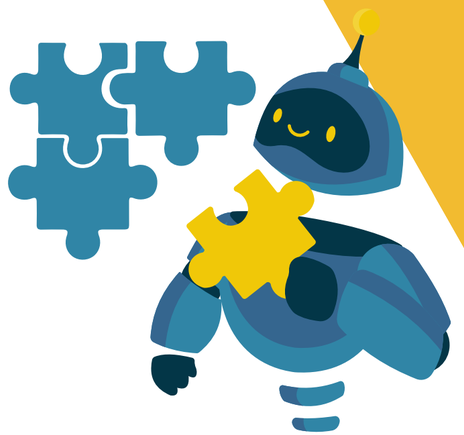


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#5.2

OUTPUT:

Now the stars dissappear when they reach the bottom of the screen.

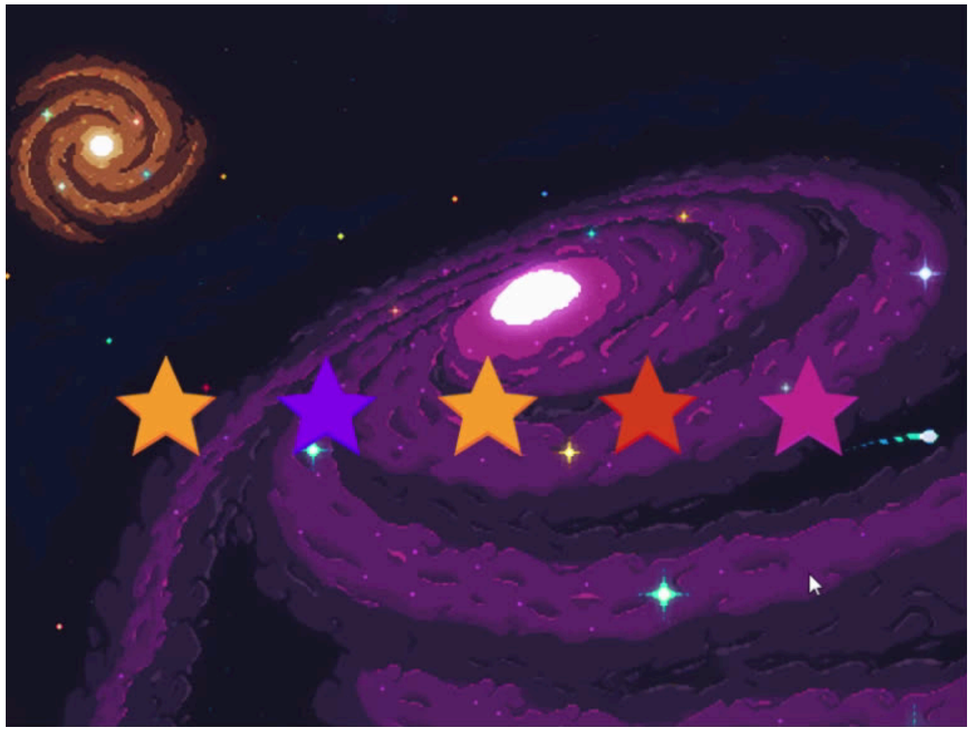


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#6

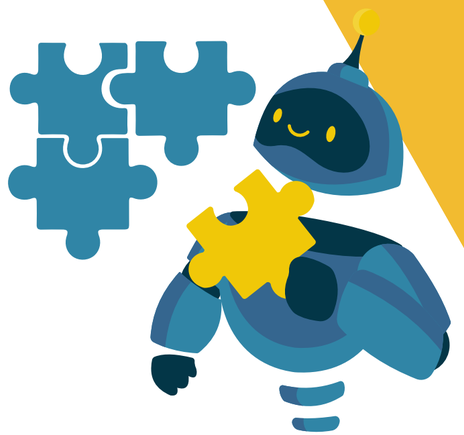
To Continue Our Game:

How can we make our mouse clicks move us to the next level if we click on the red star? How can we make them end the game if we click on any other star?



Implement the **on_mouse_click**() function to move us to the next level when we click on a red star, or call **endGame**() when we click on any other star.

#Hint: star.**image** is the name of the image that the actor uses, so for red star, star.**image** = *"red"*

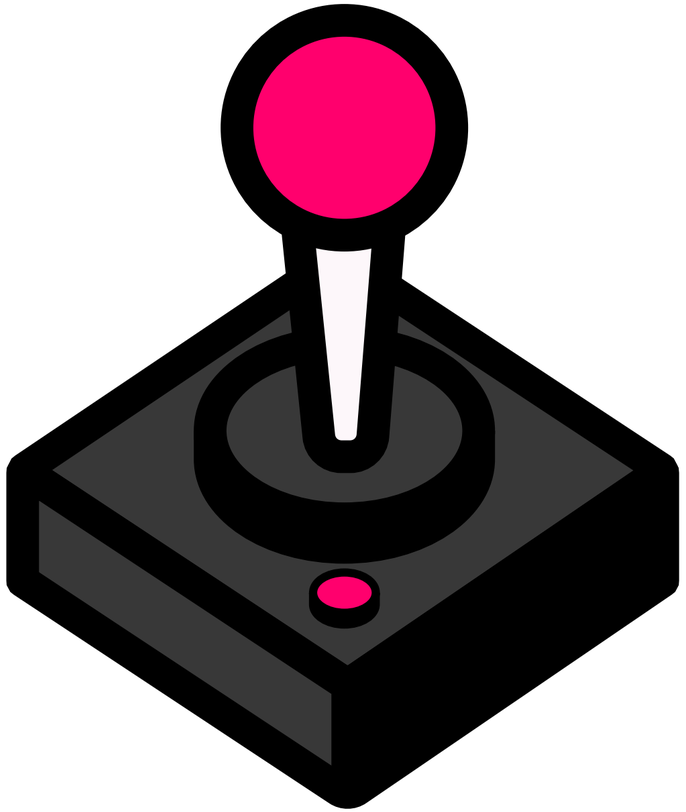


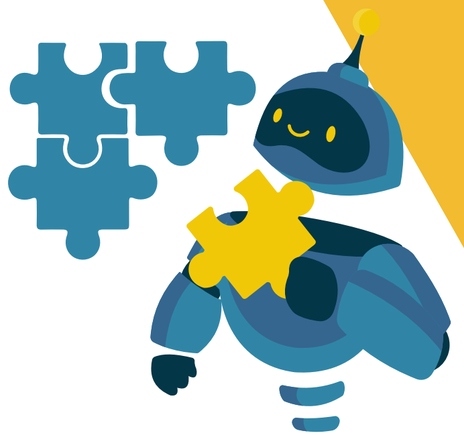
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#6

Modify Your Previous Code & Try:

```
# Add this function.
def on_mouse_down(pos):
    global currentLevel, stars
    for star in stars:
        if star.collidepoint(pos):
            # If the player clicks on the red star
            if star.image == "red":
                currentLevel += 1
            # If the player clicks on any other star
            else:
                endGame()
```



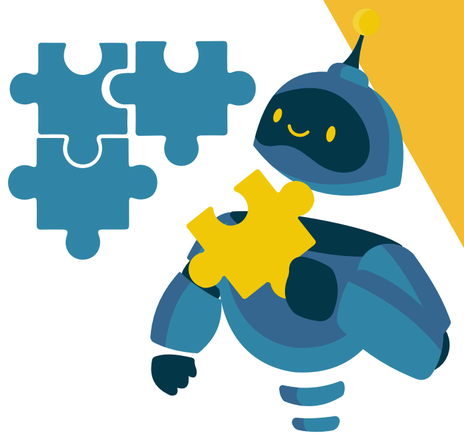


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#6

OUTPUT:

Now our level increases when we click on the red star and the stars dissappear when we click on any other star



ACTIVITY#7

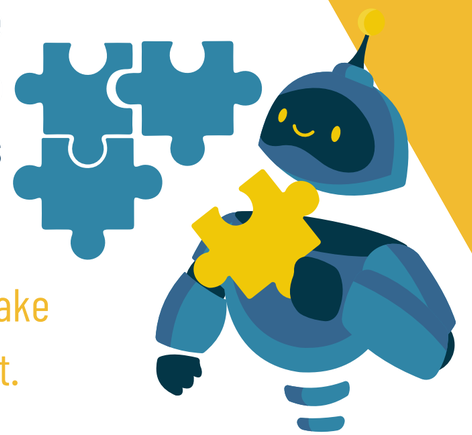
VORTEX ACADEMY
THE ART OF THINKING

To Continue Our Game:

How do we implement the leveling method in our game? How do we determine if the player has lost or won? How do we make the number of stars equal our current level?

Make the **createStars**() function create a number of stars equal to our current level.
Use the **draw**() function to draw the screen while not **gameOver**, or draw the end screen ("YOU WON" or "YOU LOST") when the game is over.
Use the **update**() function to create new stars when the number of stars isn't the same as the current level i.e the player has leveled up, also make it end the game when the player has reached the final level.

#Hint: the for loop in create stars should take **current_level-1** as a red star is already in the list.

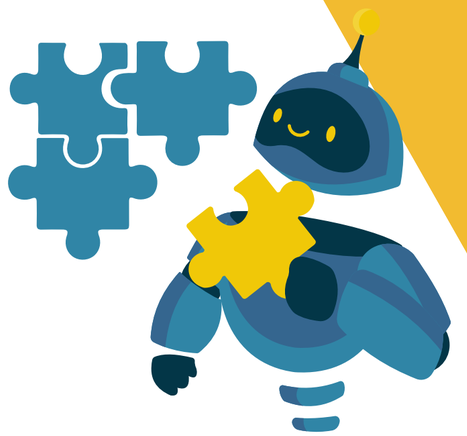


ACTIVITY#7

Modify Your Previous Code & Try:

```
currentLevel = 1  # Change the value to 1.
# In the createStars() function modify the following.
  stars.append(Actor("red"))
  for _ in range(currentLevel-1): # Change this value.
    star = Actor(random.choice(colors))
    stars.append(star)
# Add the lines in yellow to the draw function.
def draw():
    global stars, gameOver
    screen.clear() # Drawing the screen while not gameOver
    screen.blit("space", (0, 0))
    if not gameOver:
        for star in stars:
            star.draw()
    else:   # Displaying the final screen
        if currentLevel == FINAL_LEVEL:     # Player has won
            screen.draw.text("YOU WON!", (WIDTH/2,HEIGHT/2))
            screen.draw.text("Well done.", (WIDTH/2,HEIGHT/2 + 30))
        else:   # Player has lost
            screen.draw.text("GAME OVER!", (WIDTH/2,HEIGHT/2))
            screen.draw.text("Try again.", (WIDTH/2,HEIGHT/2 + 30))
```

```
# Replace the update function with this.
def update():
  global stars, gameOver, currentLevel
   # If the player has clicked on the red star i.e
moved to the next level
  if len(stars) != currentLevel and not gameOver :
    removeStars()
    createStars()
  # If the player has reached the final level
  if currentLevel == FINAL_LEVEL:
    endGame()
```

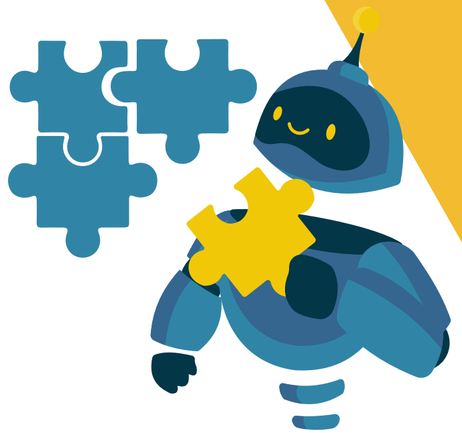


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#7

OUTPUT:

Now we have the basic functionality of RED ALERT!

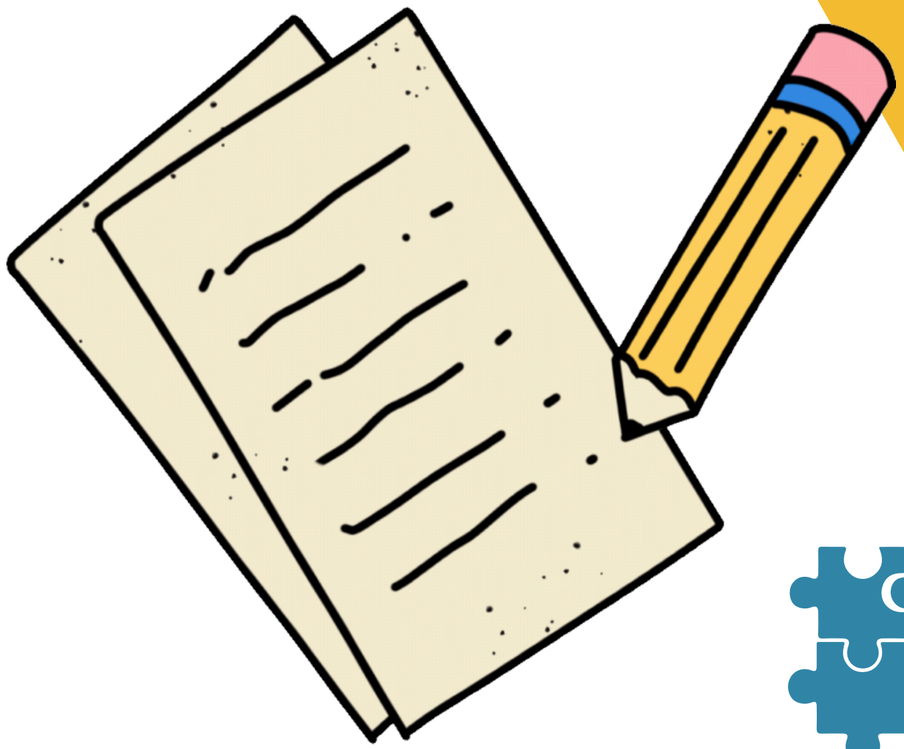


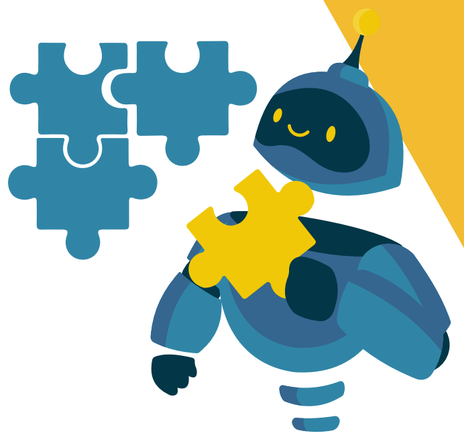
ACTIVITY#8.1

VORTEX ACADEMY
THE ART OF THINKING

Try in another script the following code

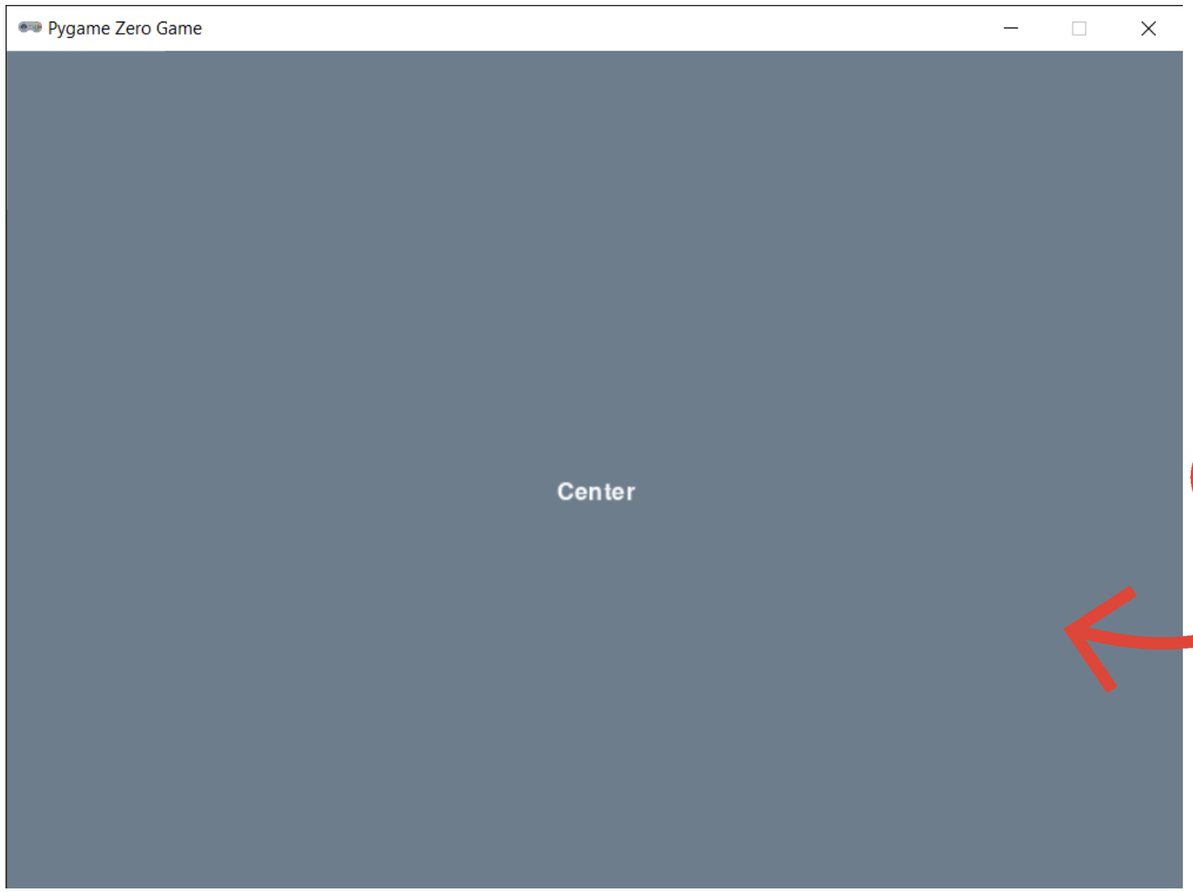
```
import pgzrun

WIDTH = 800
HEIGHT = 600
center = (WIDTH/2, HEIGHT/2)

def draw():
  screen.clear()
  screen.fill("slate Grey")
  # screen.draw.text("Center", center = center)
  # screen.draw.text("Midtop", midtop = center)
  # screen.draw.text("Midbottom", midbottom = center)
  # screen.draw.text("Midleft", midleft = center)
  # screen.draw.text("Midright", midright = center)

def update():
  pass
pgzrun.go()
```





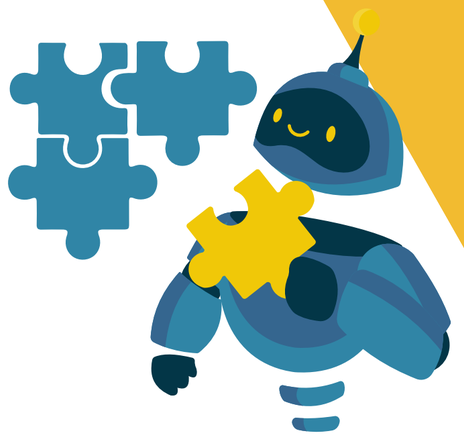
ACTIVITY#8.1

OUTPUT:



The center of the text is now at the coordinates we specified.

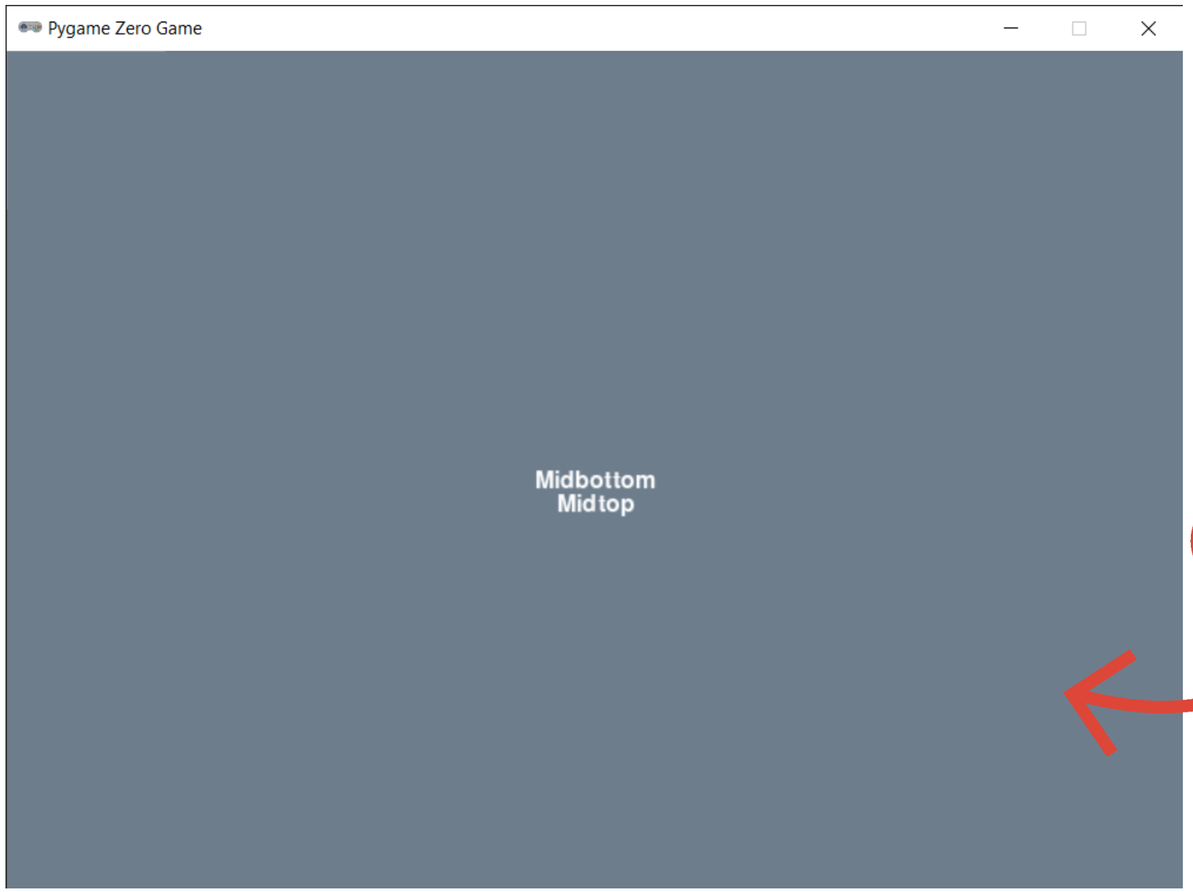
#HINT: try changing the text being displayed.



VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#8.1

OUTPUT:

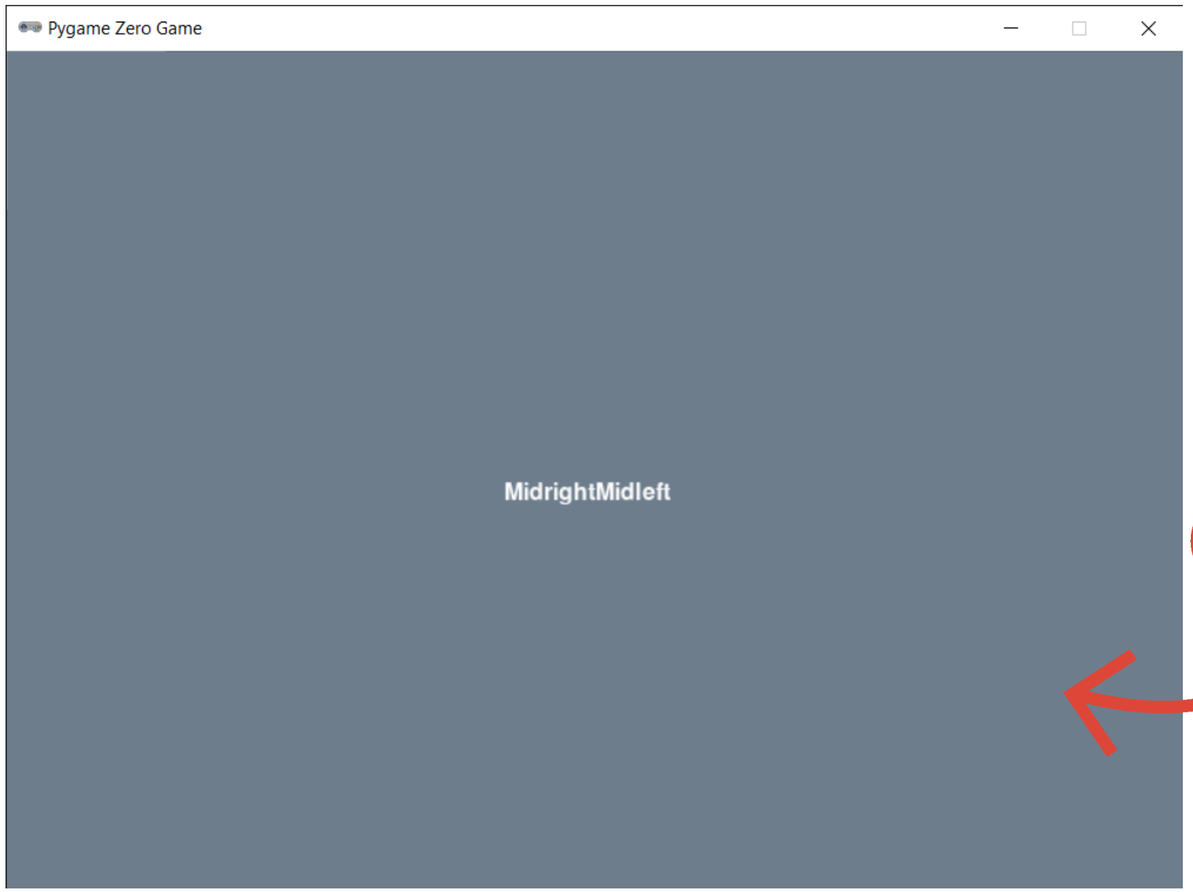


The middle of the top and the middle of the bottom of the texts is now at the coordinates we specified.

#HINT: try changing the text being displayed.

ACTIVITY#8.1

OUTPUT:



The middle of the left and the middle of the right of the texts is now at the coordinates we specified.

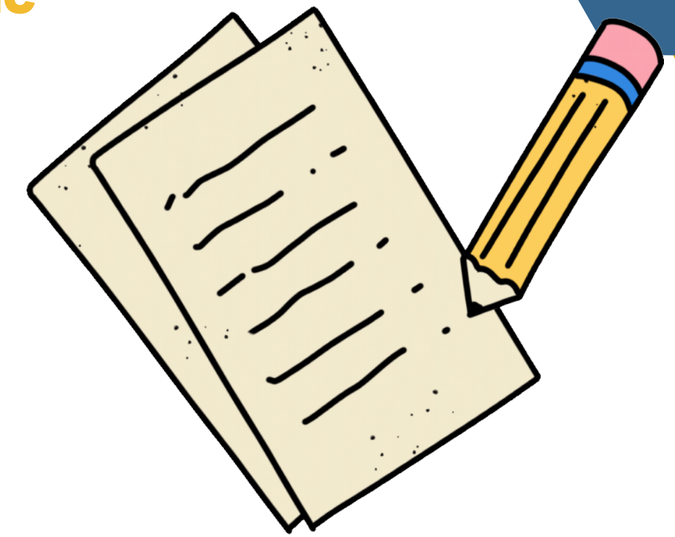
#HINT: try changing the text being displayed.

VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#8.2

VORTEX ACADEMY
THE ART OF THINKING

Try in another script the following code

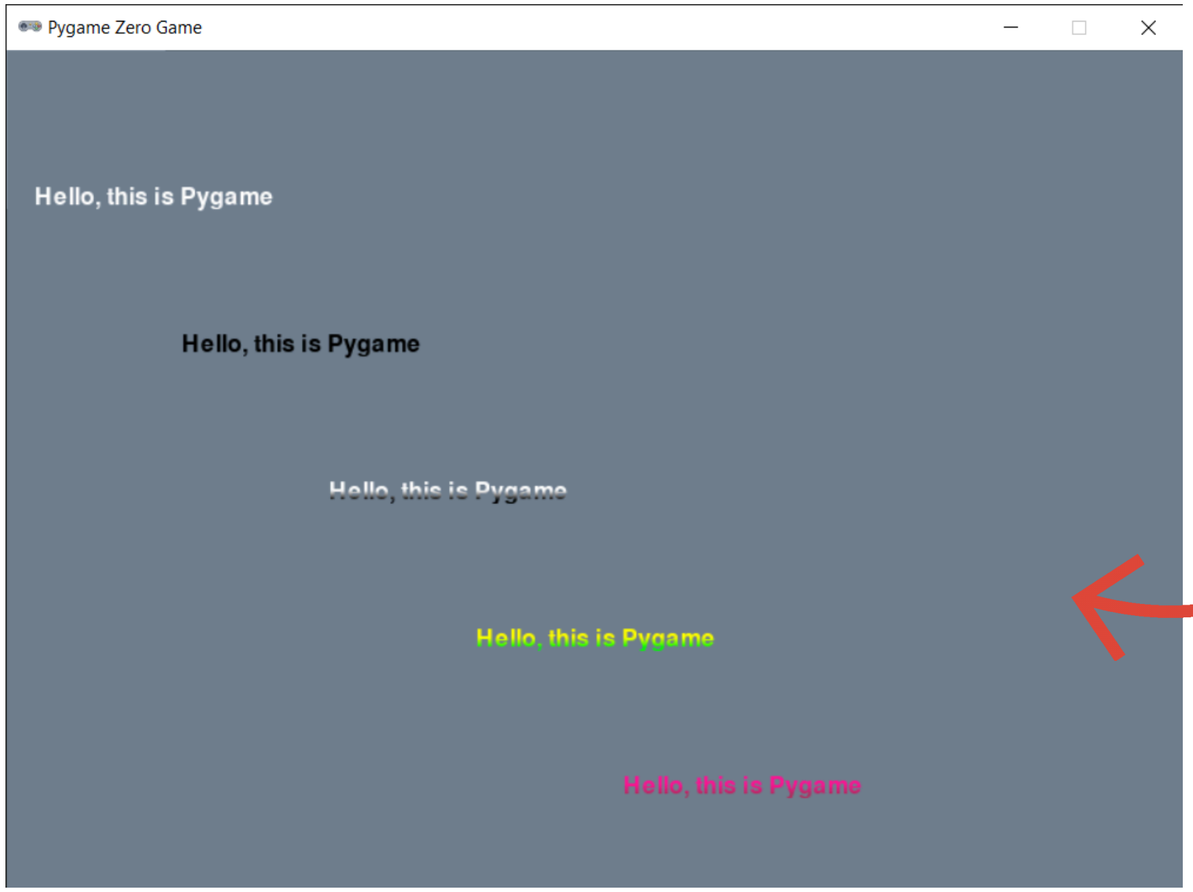


```
import pgzrun
import random

WIDTH = 800
HEIGHT = 600

def draw():
    screen.clear()
    screen.fill("slate Grey")
    screen.draw.text("Hello, this is Pygame", center = (100,100), color = "white")
    screen.draw.text("Hello, this is Pygame", center = (200,200), color = "black")
    screen.draw.text("Hello, this is Pygame", center = (300,300), color = "white", gcolor = "black")
    screen.draw.text("Hello, this is Pygame", center = (400,400), color = "yellow", gcolor = "green")
    screen.draw.text("Hello, this is Pygame", center = (500,500), color = "deep pink", gcolor = "maroon")

def update():
    pass
pgzrun.go()
```

VORTEX ACADEMY
THE ART OF THINKING

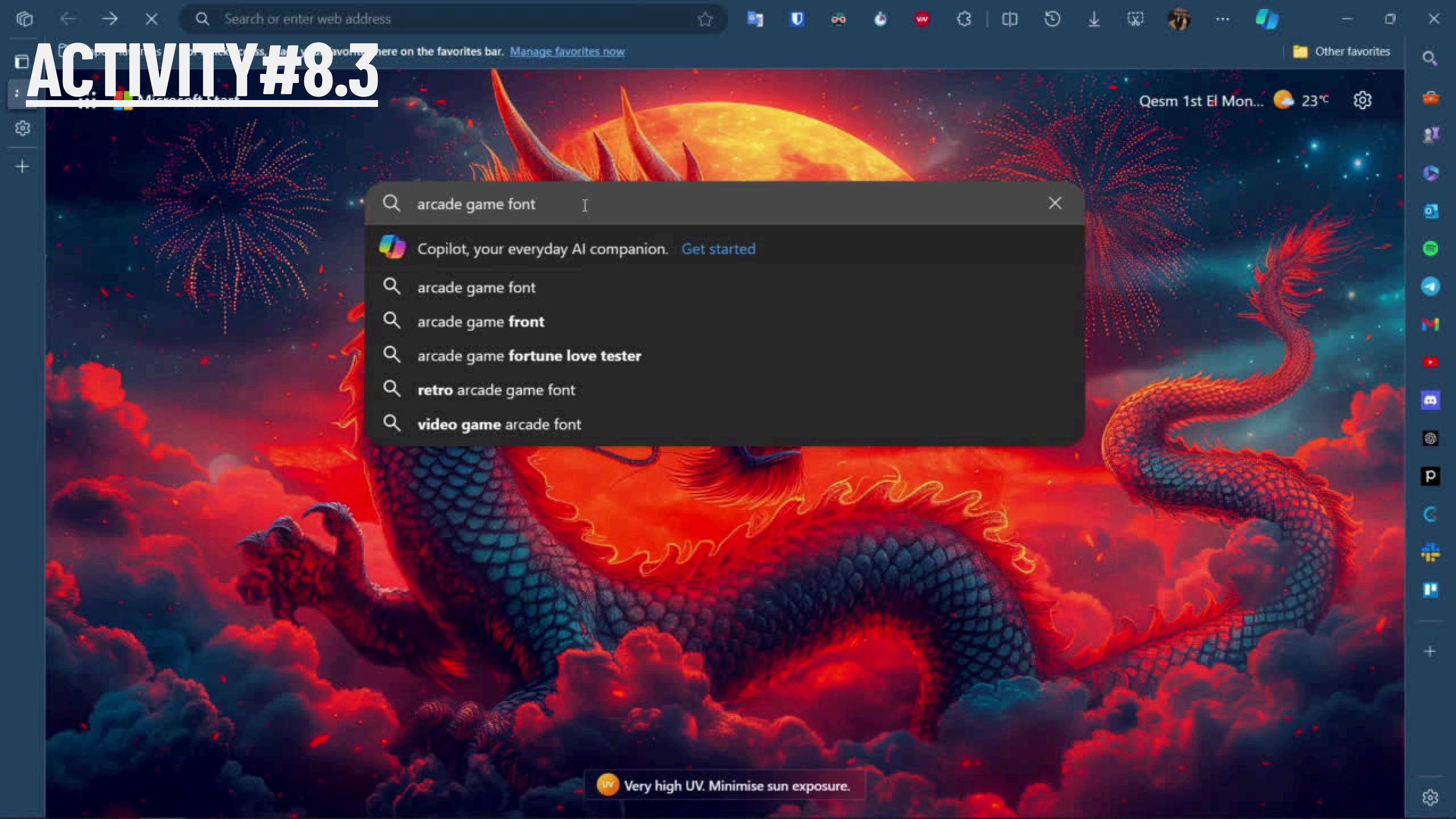
ACTIVITY#8.2

OUTPUT:



The text has different colors and gradients as we specified.

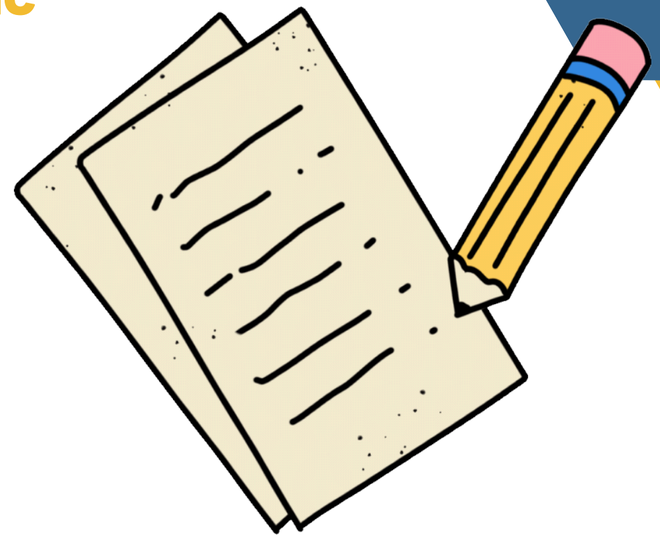
#HINT: try changing the colors being displayed.


ACTIVITY#8.3
Search or enter web address
here on the favorites bar. Manage favorites now
Other favorites
Qesm 1st El Mon...
23°C
arcade game font
Copilot, your everyday AI companion. Get started
arcade game font
arcade game front
arcade game fortune love tester
retro arcade game font
video game arcade font
Very high UV. Minimise sun exposure.

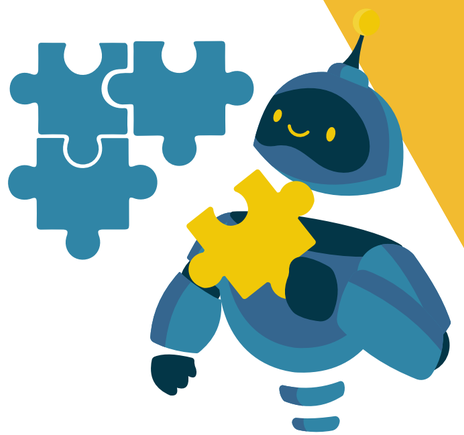
ACTIVITY#8.3

VORTEX ACADEMY
THE ART OF THINKING

Try in another script the following code

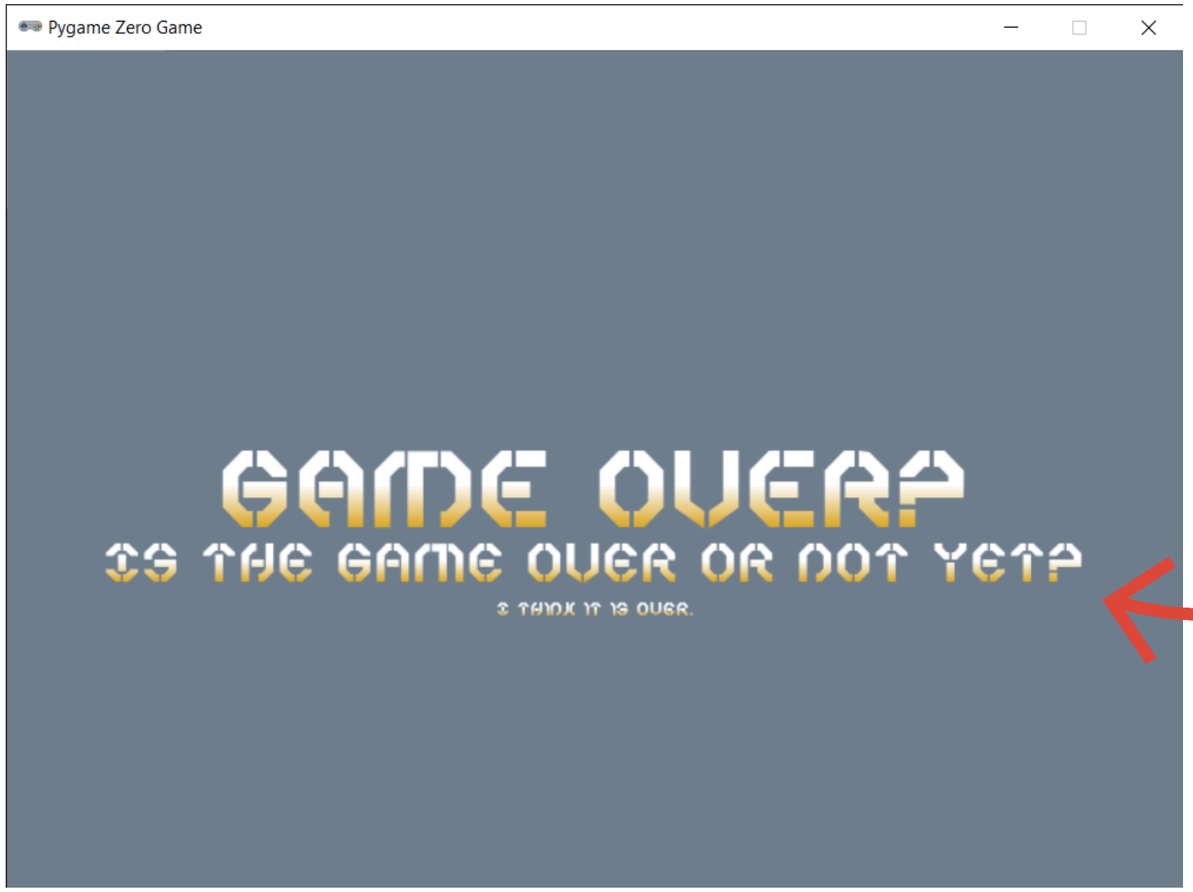


```
import pgzrun
import random

WIDTH = 800
HEIGHT = 600

def draw():
    screen.clear()
    screen.fill("slate Grey")
    screen.draw.text("GAME OVER?", fontname = "font", fontsize = 60, center = (WIDTH/2, HEIGHT/2), color = "white", gcolor = "golden rod")
    screen.draw.text("Is the game over or not yet?", fontname = "font", fontsize = 30, center = (WIDTH/2, HEIGHT/2 + 50), color = "white", gcolor = "golden rod")
    screen.draw.text("I think it is over.", fontname = "font", fontsize = 10, center = (WIDTH/2, HEIGHT/2 + 80), color = "white", gcolor = "golden rod")

def update():
    pass
pgzrun.go()
```

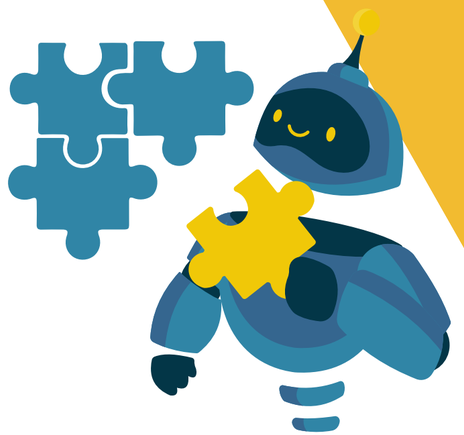


ACTIVITY#8.3

OUTPUT:



Now we have changed the font of our text.



More Text Options

VORTEX ACADEMY
THE ART OF THINKING

We can change the attributes of our text from its size to it's font, color, gradient color, and positioning method by changing the arguments we pass to the screen.draw.**text**("TEXT", fontname = "FONT_NAME", fontsize = FONT_SIZE, center = (X,Y), color = "COLOR", gcolor = "GRADIENT_COLOR") function.

Syntax:

screen.draw.**text**("TEXT", fontname = "FONT_NAME", fontsize = FONT_SIZE, center = (X,Y), color = "COLOR", gcolor = "GRADIENT_COLOR")

Positioning parameters: *center, midtop, midbottom, midleft, midright, topleft, topright, bottom left, bottom right.*

Color parameters: *color = COLOR_IN_ANY_OF_THE_FORMATS_WE_KNOW*
gcolor = COLOR_IN_ANY_OF_THE_FORMATS_WE_KNOW

Font parameters: *fontname = THE_NAME_OF_OUR_FONT_IN_OUR_FONTS_FOLDER*
fontsize = THE_SIZE_OF_THE_TEXT

VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#8.4

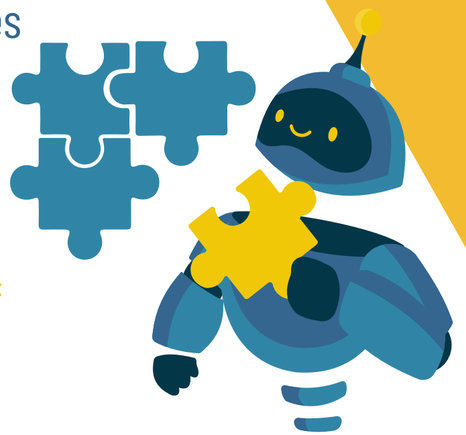
To Continue Our Game:

How can we use the new text options to improve our end screen? How can we center it in the middle of the screen? change it's font? give it a gradient?

In the function **draw**() modify the end screen so that it uses the font in our fonts folder with a size of 60 for the end screen title and a size of 30 for the subtitle with a white color and a "golden rod" gradient color, and increase the distance between the two lines to 50.

#Hint: Syntax:

screen.draw.**text**("TEXT", fontname = "FONT_NAME", fontsize = FONT_SIZE, center = (X,Y), color = "COLOR", gcolor = "GRADIENT_COLOR")

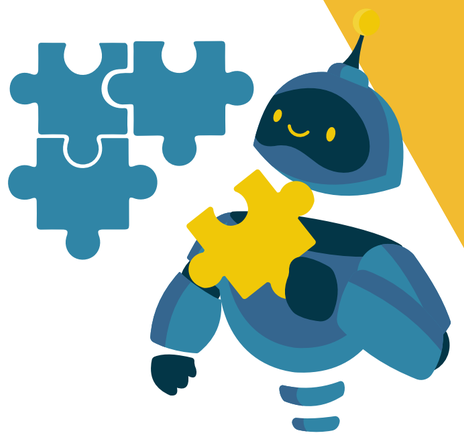


ACTIVITY#8.4

VORTEX ACADEMY
THE ART OF THINKING

Modify Your Previous Code & Try:

```
# Modify the lines in yellow.
def draw():
 global stars, gameOver   # Drawing the screen while not gameOver
 screen.clear()
 screen.blit("space", (0, 0))
 if not gameOver:
  for star in stars:
   star.draw()
 else: # Displaying the final screen
  # Player has won
  if currentLevel == FINAL_LEVEL:
   screen.draw.text("YOU WON!", fontsize=60, fontname = "font", center=(WIDTH/2,HEIGHT/2), color="white",
gcolor= "golden rod")
   screen.draw.text("Well done.", fontsize=30, fontname = "font", center=(WIDTH/2,HEIGHT/2 + 50),
color="white", gcolor= "golden rod")
  # Player has lost
  else:
   screen.draw.text("GAME OVER!", fontsize=60, fontname = "font", center=(WIDTH/2,HEIGHT/2),
color="white", gcolor= "golden rod")
   screen.draw.text("Try again.", fontsize=30, fontname = "font", center=(WIDTH/2,HEIGHT/2 + 50),
color="white", gcolor= "golden rod")
```

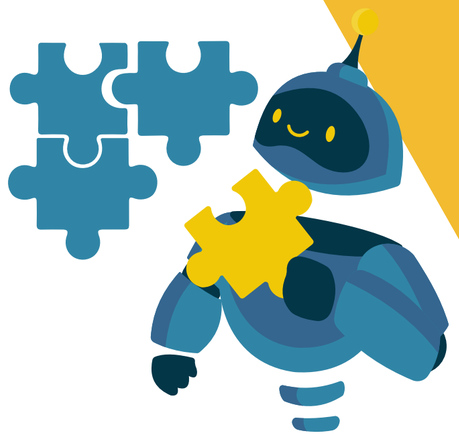


VORTEX ACADEMY
THE ART OF THINKING

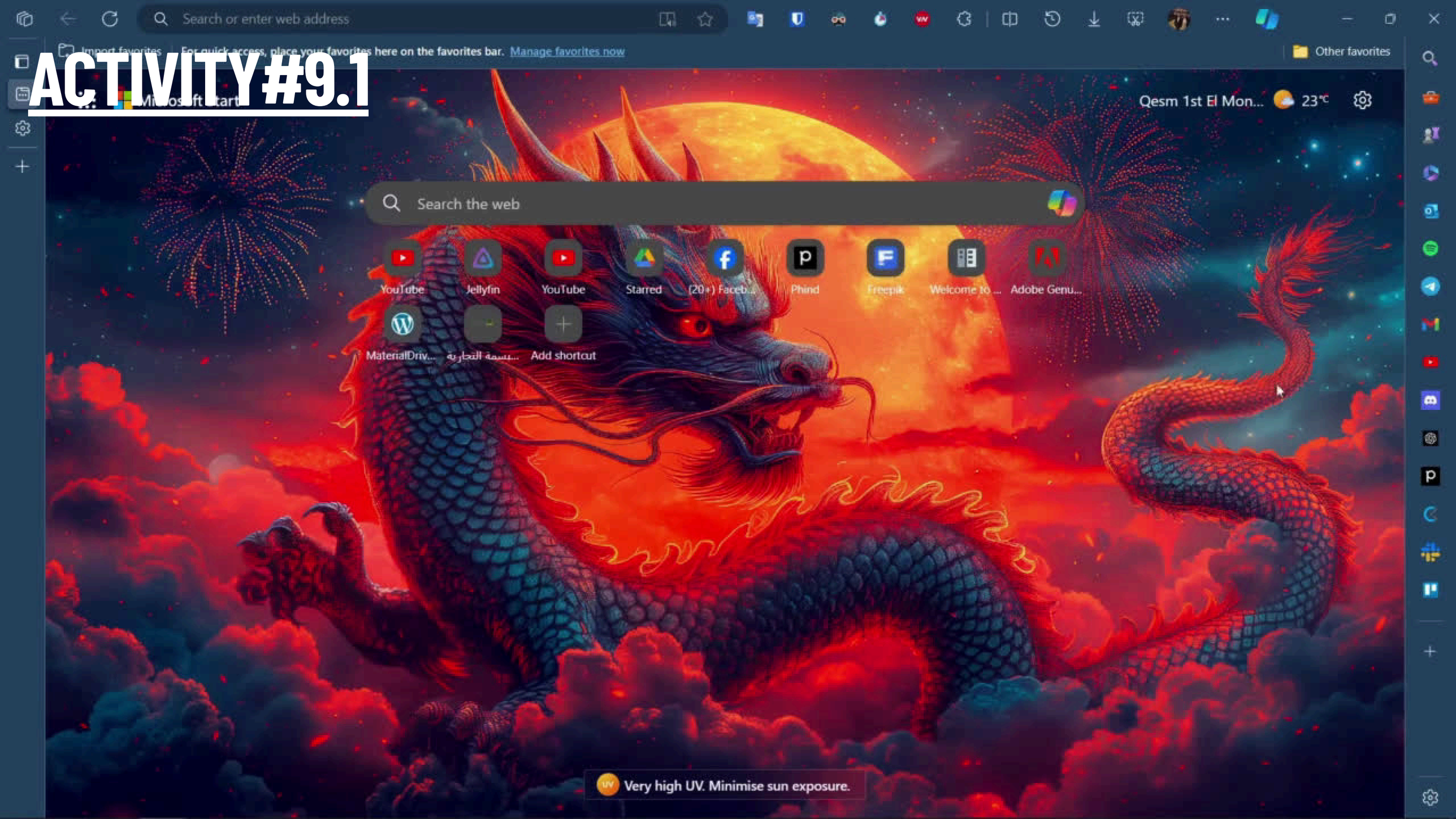
ACTIVITY#8.4

OUTPUT:

Now the end screen has the font and gradient applied to the text.



ACTIVITY#9.1

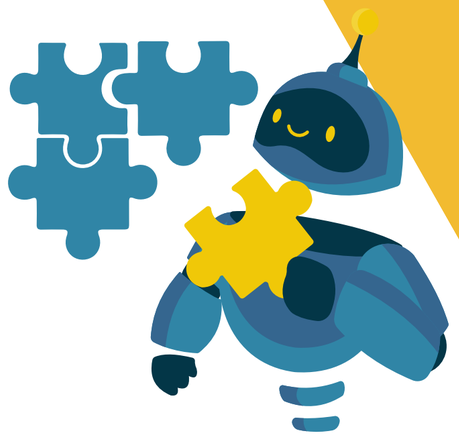

Search or enter web address
Import favorites
For quick access, place your favorites here on the favorites bar. Manage favorites now
Other favorites
Qesm 1st El Mon...
23°C
Search the web
YouTube
Jellyfin
YouTube
Starred
(20+) Faceb...
Phind
Freepik
Welcome to ...
Adobe Genu...
MaterialDriv...
...بسمة التجارية
Add shortcut
Very high UV. Minimise sun exposure.

ACTIVITY#9.1

VORTEX ACADEMY
THE ART OF THINKING

Try in another script the following code

```
import pgzrun

WIDTH = 800
HEIGHT = 600

def draw():
    screen.clear()
    screen.fill("slate Grey")

def update():
    pass

music.play("background")
pgzrun.go()
```

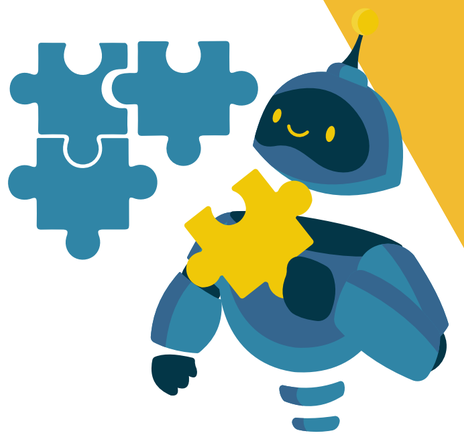


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#9.1

OUTPUT:

The game plays the background music when it starts

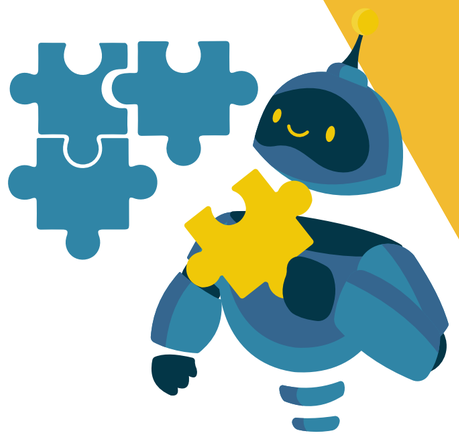


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#9.2

Modify Your Previous Code & Try:

```
# Add the music to music folder then add:
music.play("background")
# Before pgzrun.go()
```

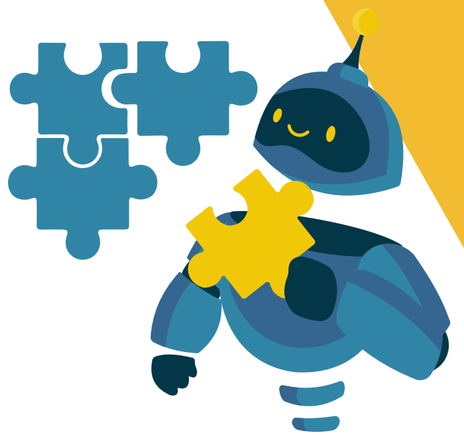


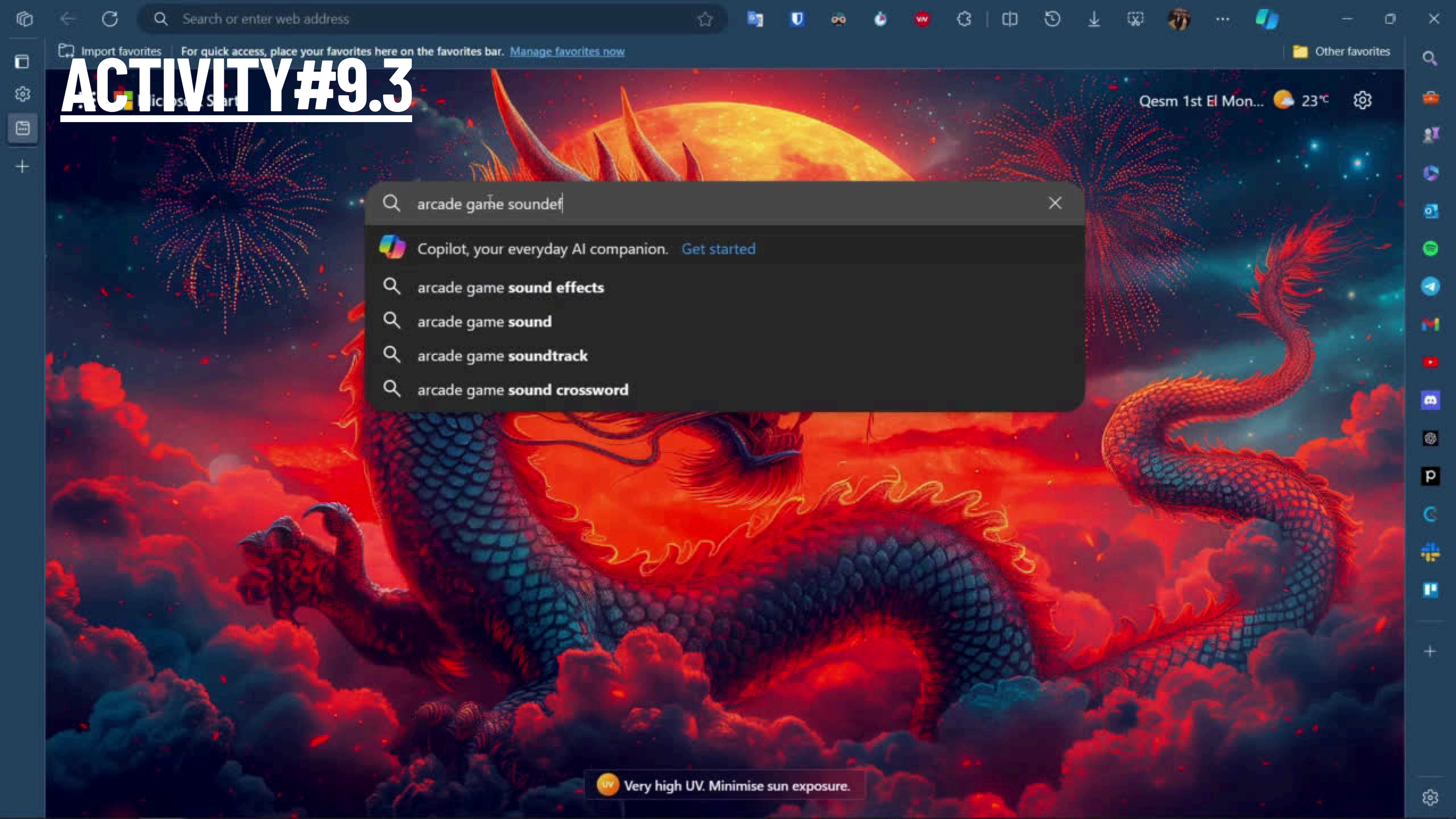
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#9.2

OUTPUT:

Now the game has background music

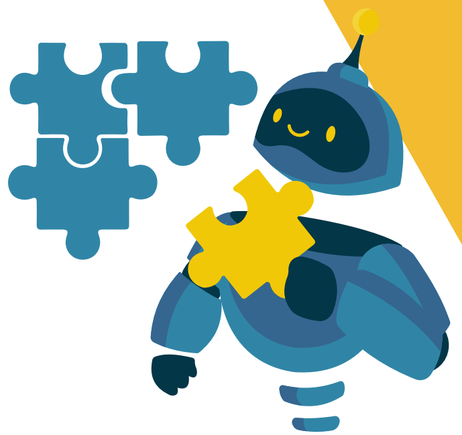



Search or enter web address
Import favorites
For quick access, place your favorites here on the favorites bar. Manage favorites now
Other favorites
ACTIVITY#9.3
Qesm 1st El Mon...
23°C
arcade game soundef
Copilot, your everyday AI companion. Get started
arcade game sound effects
arcade game sound
arcade game soundtrack
arcade game sound crossword
Very high UV. Minimise sun exposure.

ACTIVITY#9.3

Try in another script the following code

```
import pgzrun

WIDTH = 800
HEIGHT = 600

def draw():
    screen.clear()
    screen.fill("slate Grey")

def update():
    pass

def on_mouse_down(pos):
    sounds.level_win.play()

music.play("background")
pgzrun.go()
```



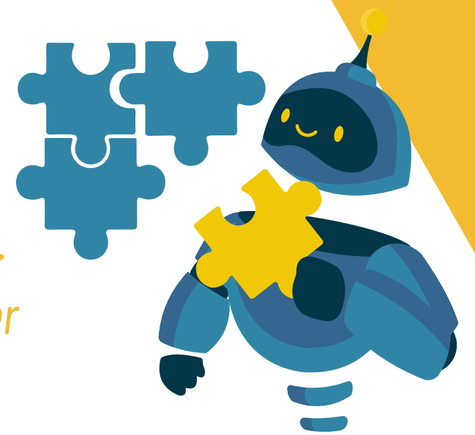
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#9.3

OUTPUT:

The game plays a level_win sound everytime we click the screen.

#HINT: try changing the sound effect being played.
use game_win or game_over



Music & Audio

VORTEX ACADEMY
THE ART OF THINKING

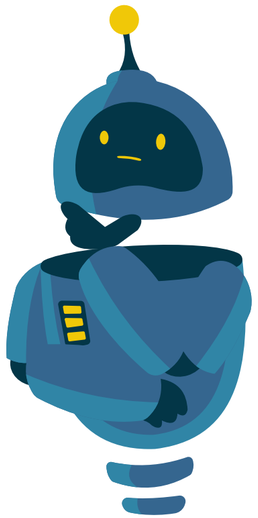
When we want to add music to our game we add it to the music folder in the same folder as our python script then use music.**play**(*"THE_MUSIC"*) to play it within our game, THE_MUSIC should be an mp3 file.

When we want to add sound effects to our game we add it to the sounds folder in the same folder as our python script then use sounds.*SOUND_EFFECT_NAME*.**play**() to play it within our game, the SOUND_EFFECT should be a wav file and be small.

In the previous example we used music.**play**(*"background"*) and sounds.*level_win*.**play**() to play our background music and sound effect when clicking on the screen.

Syntax:

```
# For music
music.play("THE_MUSIC")

# For sound effects
sounds.SOUND_EFFECT_NAME.play()
```

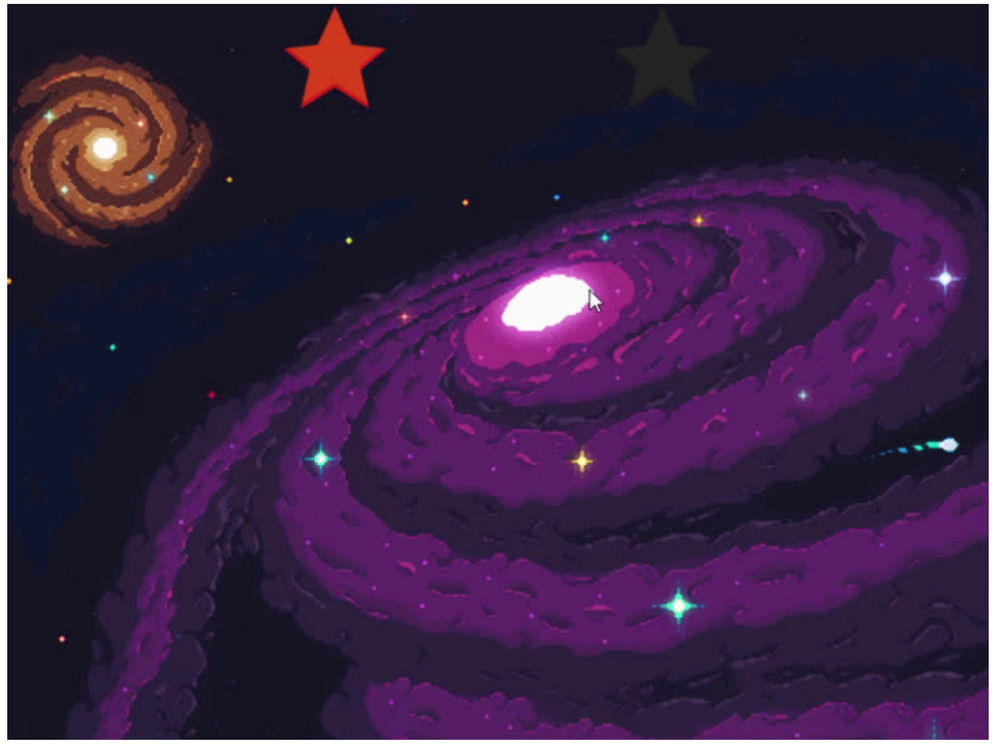


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#9.4

To Continue Our Game:

How can we add sound effects to the game when the player wins a level or wins the game? when the player loses the game?

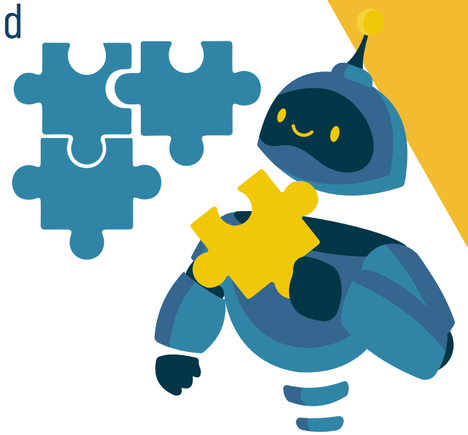


Add a sound effect when the player presses on the red star of the level win.

Add a check in **update**() to play a sound effect when the player loses the game.

Add to the check that ends the game when the player wins to play the winning sound effect.

#Hint: you can check gameOver to know if the player has lost

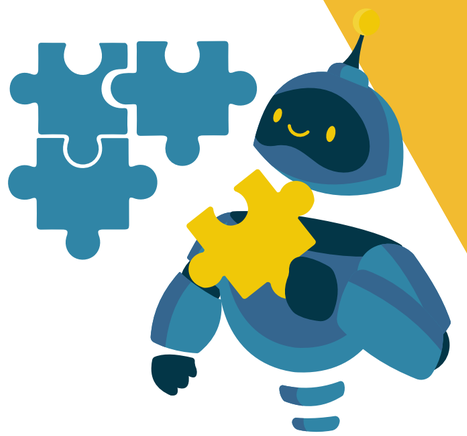


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#9.4

Modify Your Previous Code & Try:

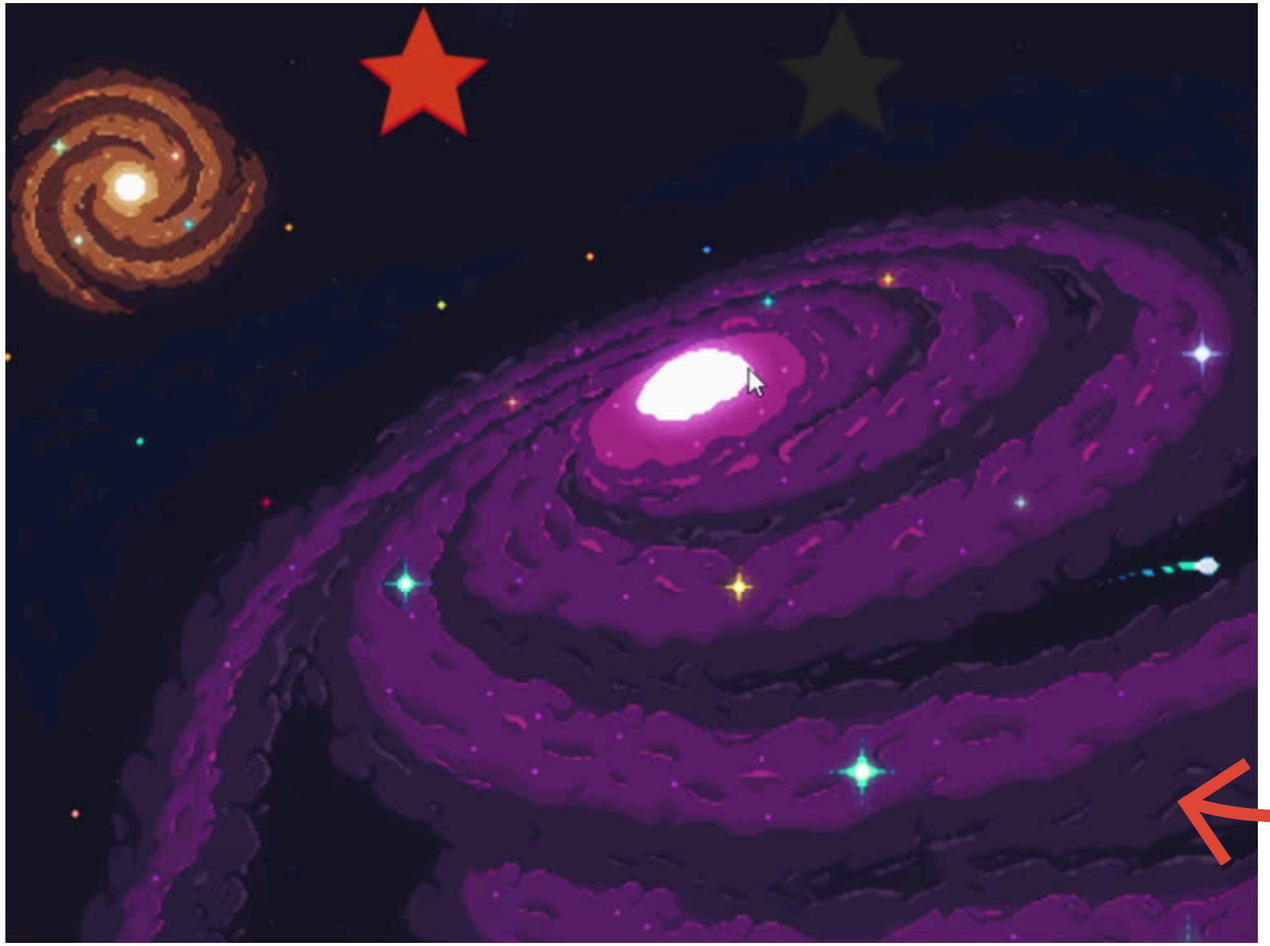
```
# Modify in the function def on_mouse_down()
# Add the lines in Yellow
if star.image == "red":
        currentLevel += 1
        sounds.level_win.play()

# Modify def update()
  # If the player has reached the final level
  if currentLevel == FINAL_LEVEL:
    sounds.game_win.play()
    endGame()
  # If the player has lost the game
  elif gameOver:
    sounds.game_over.play()
    endGame()
```



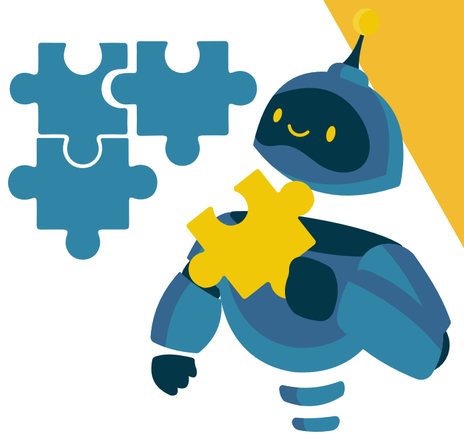
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#9.4

OUTPUT:



Now the game plays the sound effects appropriately, however There is a bug, did you notice it?

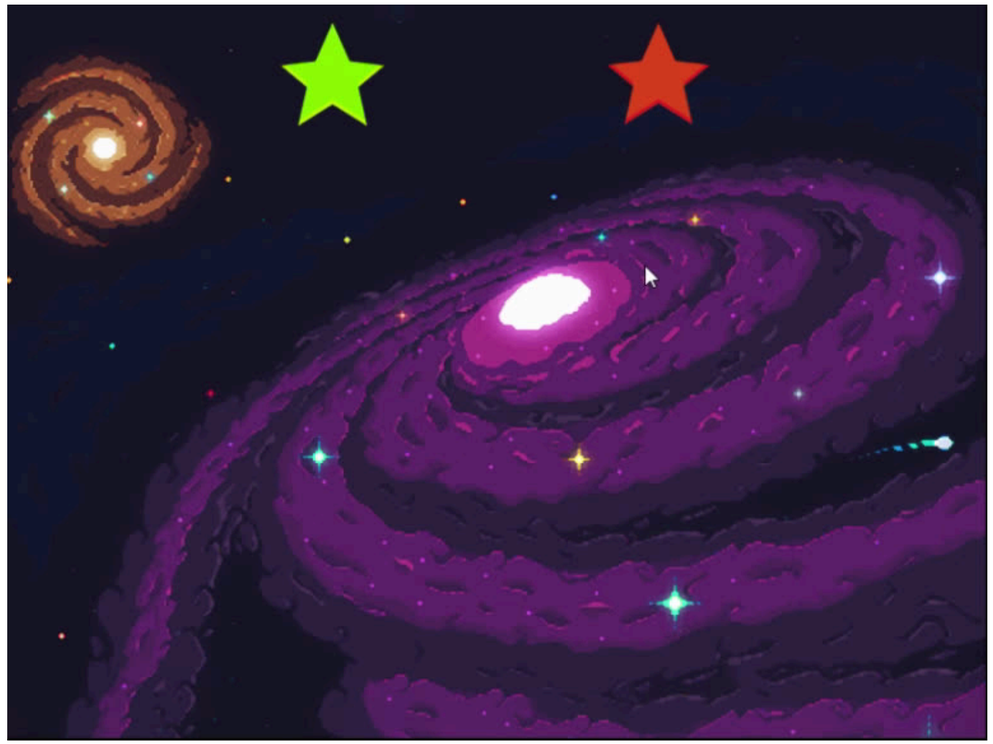


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#10

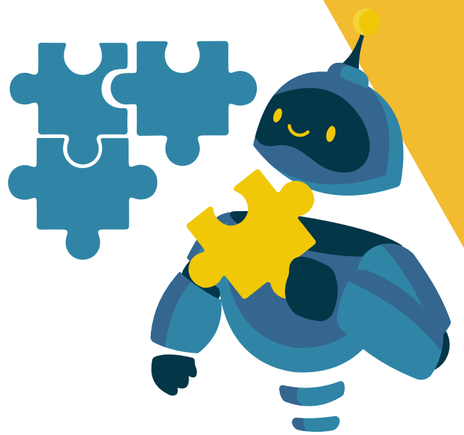
To Continue Our Game:

What bug does The game have? How can we fix it?



The game currently keeps looping the ending sounds, so we need to add a check so that if they were already played once they're not played again.

#Hint: add a bool endSoundPlayed to check if the end sound is played. if it isn't and the game is over (winning or losing) play the appropriate sound, otherwise set it to true and don't play the sound again.

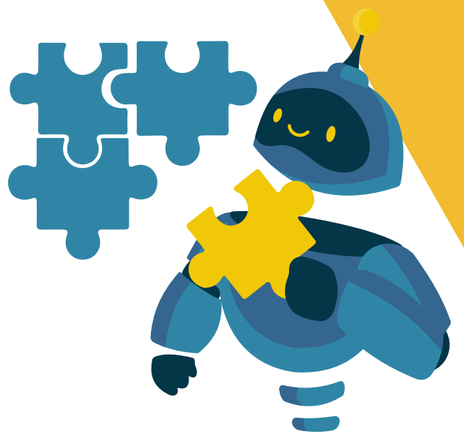


ACTIVITY#10

VORTEX ACADEMY
THE ART OF THINKING

Modify Your Previous Code & Try:

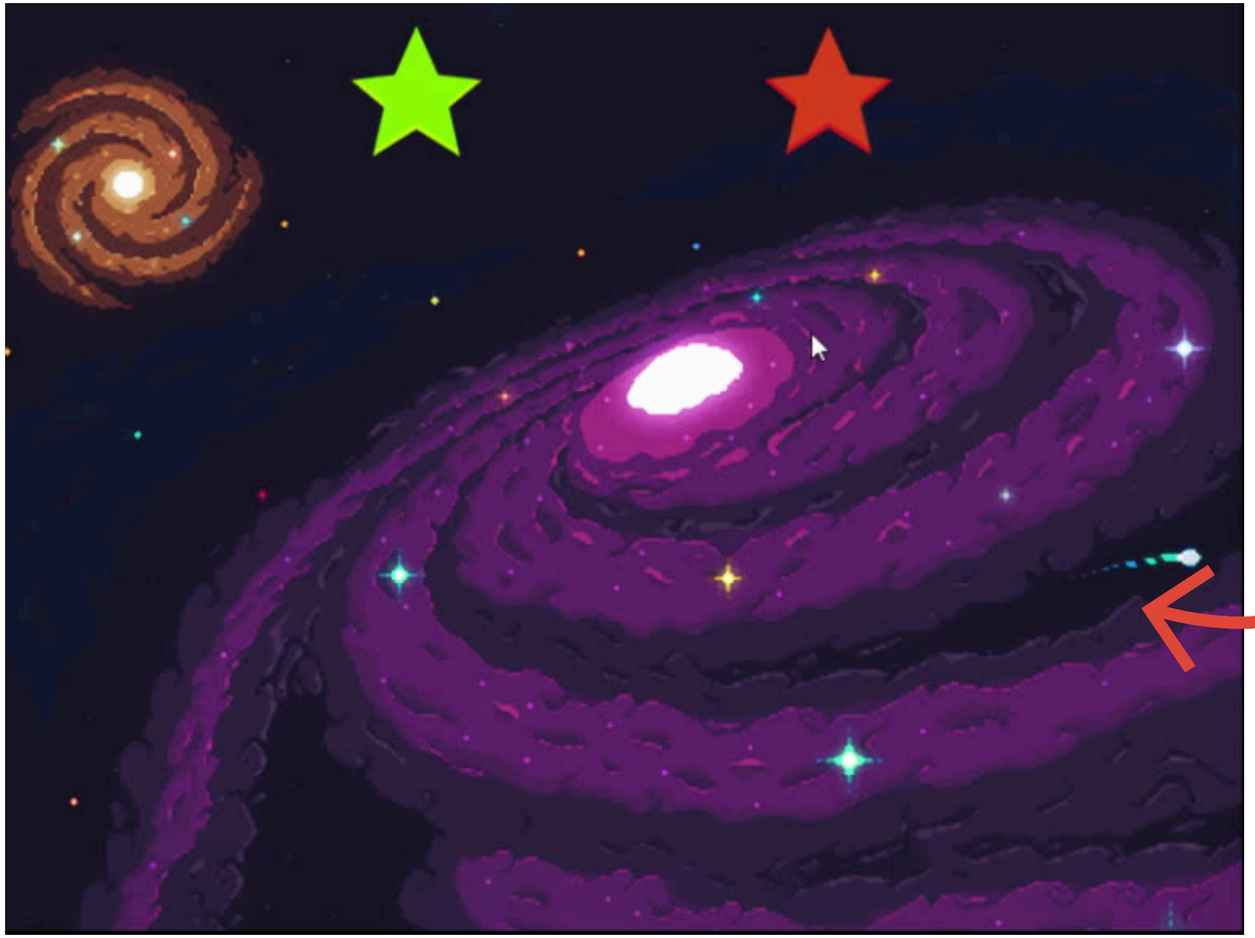
```
# Add to the game variables
endSoundPlayed = False

# Modify def update() to add the lines in yellow.
  global endSoundPlayed
   if currentLevel == FINAL_LEVEL:
      if not endSoundPlayed:
         sounds.game_win.play()
         endSoundPlayed = True
      endGame()
   elif gameOver:
      if not endSoundPlayed:
         sounds.game_over.play()
         endSoundPlayed = True
      endGame()
```



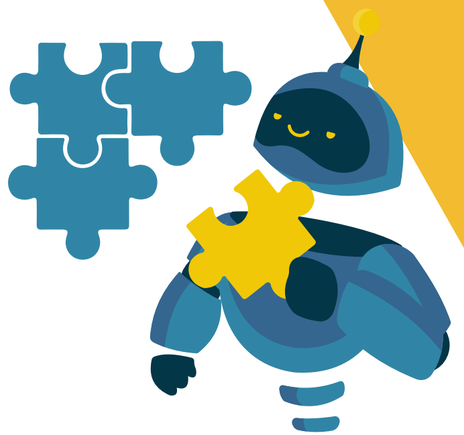
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#10

OUTPUT:



Now we have built RED ALERT!



ASSIGNMENT

1	NOTES	Write session notes
2	SOLVE ASSIGNMENT	

ASSIGNMENT

Continuing on the code that you wrote during the session.

Task1: Modify your code so that at the end of the game the user can click the screen to try again and add text to the end screen to ask user to click to try again.

#HINT the game state is saved in the three variables currentLevel, gameOver and endSoundPlayed.

Task2: Continue on your code so that it has a score counter on the top left of the screen that counts how many levels the user has passed even if they lose and try again.

Task3: Continue on your code so that it makes each star randomly faster or slower than the others.

#HINT use random.**randint**() to choose a random duration divisior for each star between (*1, currentLevel*2*) then divide the FINAL_LEVEL by that divisor and use that as the duration.

Task4: Continue on your code so that it makes some stars move from to down to up instead of from up to down randomly.

#HINT use random.**choice**() to determine which stars move up or down and modify the starting position and animation direction of the stars that will go down

VORTEX ACADEMY
THE ART OF THINKING

THANK YOU

FOR YOUR ATTENTION