GAME#1

“SHOOT THE FRUIT”

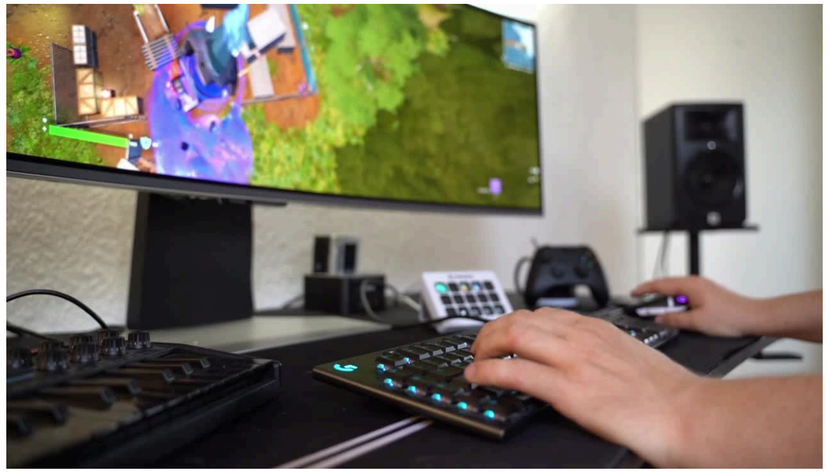
VORTEX ACADEMY
THE ART OF THINKING

WHAT DO YOU THINK PYTHON LIBRARIES CAN DO?

VORTEX ACADEMY
THE ART OF THINKING

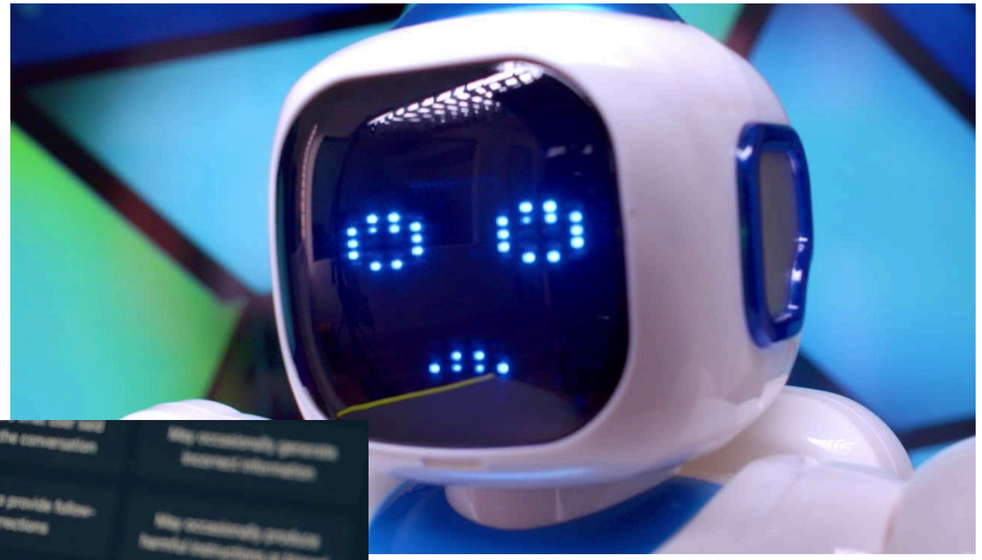
WHAT ARE PYTHON LIBRARIES?

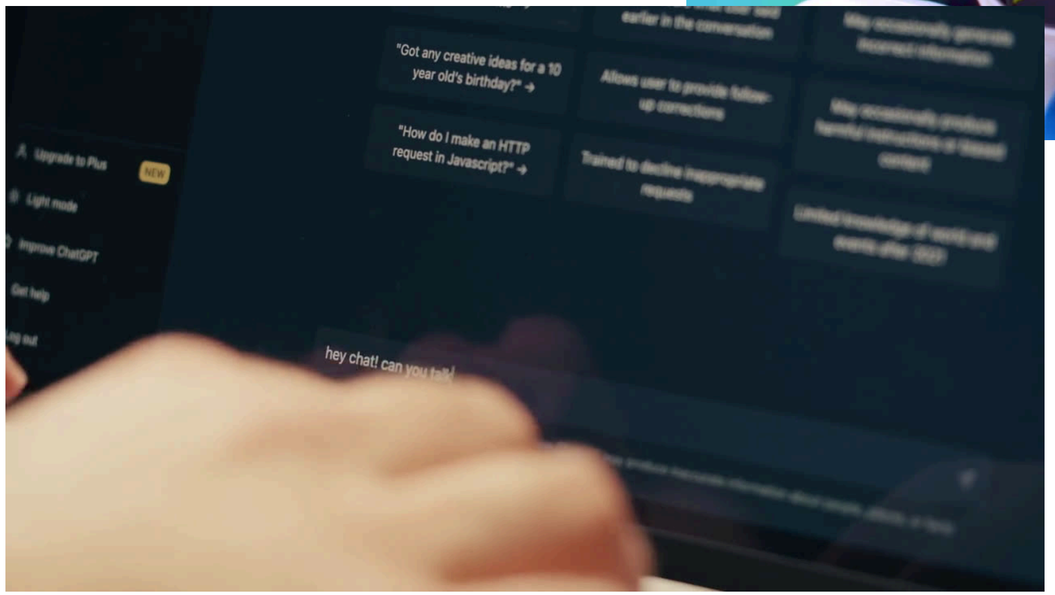
VORTEX ACADEMY
THE ART OF THINKING



WHAT DO YOU THINK PYTHON LIBRARIES CAN DO?

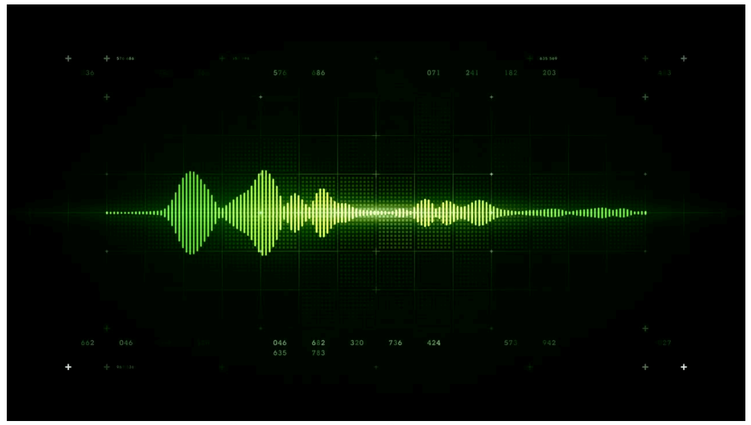
VORTEX ACADEMY
THE ART OF THINKING





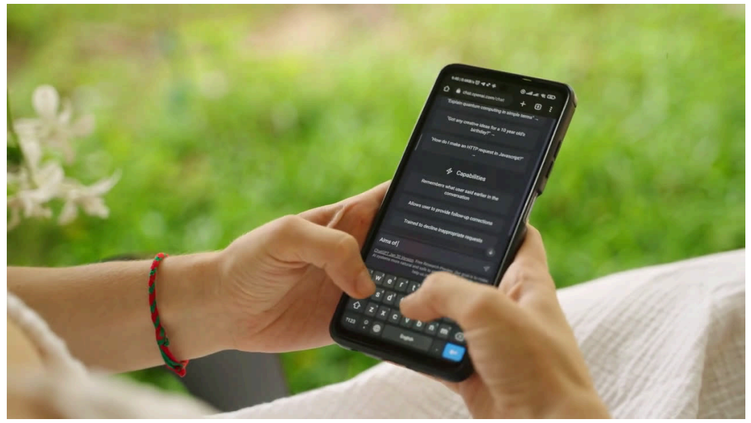
WHAT ARE PYTHON LIBRARIES?

VORTEX ACADEMY
THE ART OF THINKING



AI Music and voice generator

Autonomous driving

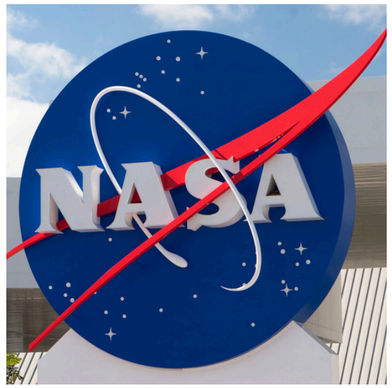


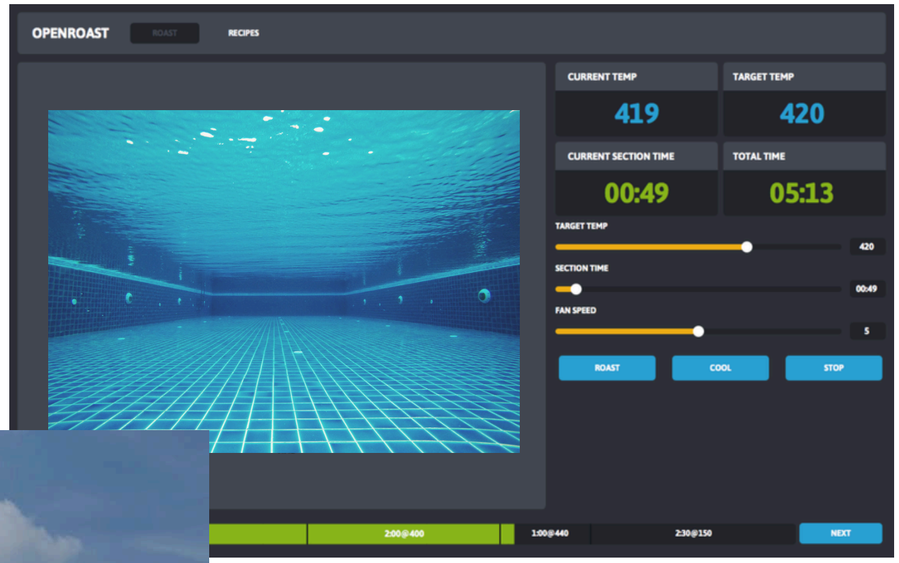
ChatGPT & LLMs

Photo and video generator

WHAT DO YOU THINK PYTHON LIBRARIES CAN DO?

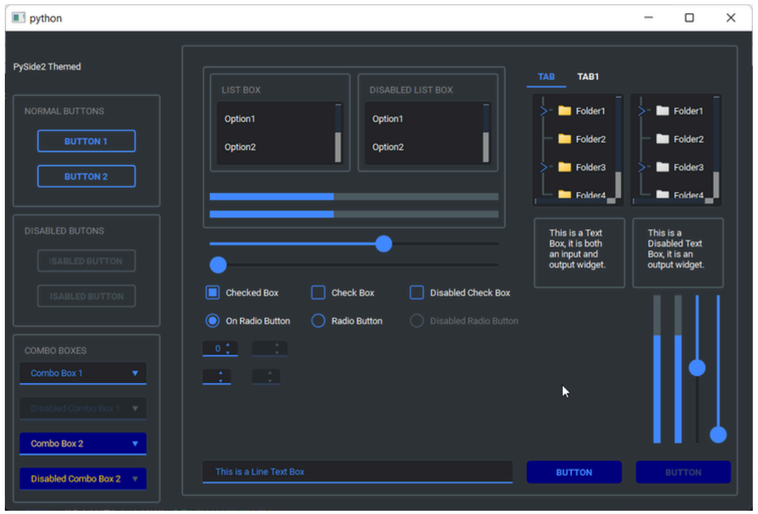
VORTEX ACADEMY
THE ART OF THINKING

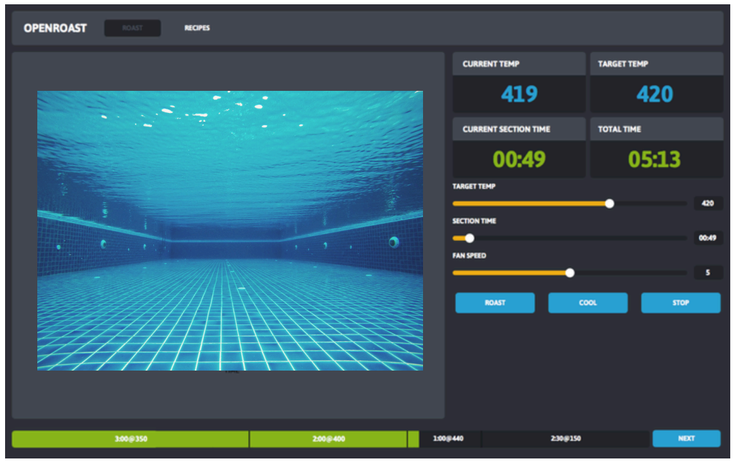




WHAT ARE PYTHON LIBRARIES?

VORTEX ACADEMY
THE ART OF THINKING





WHAT DO YOU THINK PYTHON LIBRARIES CAN DO?

VORTEX ACADEMY
THE ART OF THINKING

amazon

WHAT ARE PYTHON LIBRARIES?

VORTEX ACADEMY
THE ART OF THINKING

TOPICS

VORTEX ACADEMY
THE ART OF THINKING

01

Introduction to Game Development

- How do games work?
- What is pygame and pygame zero?
- Installing pygame zero.

02

Drawing to the Screen

- How do we draw our game to the screen?
- How do we write text in our game?

03

Using the Mouse

- How do we know when the player clicks the mouse and where?
- Building Shoot the fruit.

VORTEX ACADEMY
THE ART OF THINKING

HOW DO GAMES WORK?

WHY PYGAME?

VORTEX ACADEMY
THE ART OF THINKING

WHY PYGAME?

Pygame is a python library used for game development, it enables us to develop 2D games similar to what we saw in the previous video and can be used with other libraries to make 3D games.

Itch.io pygame games

Steam pygame games

Indie Pygame developer

Games on Pygame site.

Pygame is used to create games that can be sold on steam or itch.io.
Some developers host their games on their own websites, while others work at game studios.
You can check the games on pygame site to see exampls of finished games and thier source codes.

VORTEX ACADEMY
THE ART OF THINKING

PYGAME ZERO

Pygame zero is a version of pygame that is easier to use, it helps us by implementing the game loop and providing the functions we need to make our game.
Pygame provides the basic tools, pygame zero helps us build the game.
In order to develop our game we modify the functions in pygame zero game loop to run our game.

To install pygame zero open a terminal in VS code and run the command:

pip install pgzero

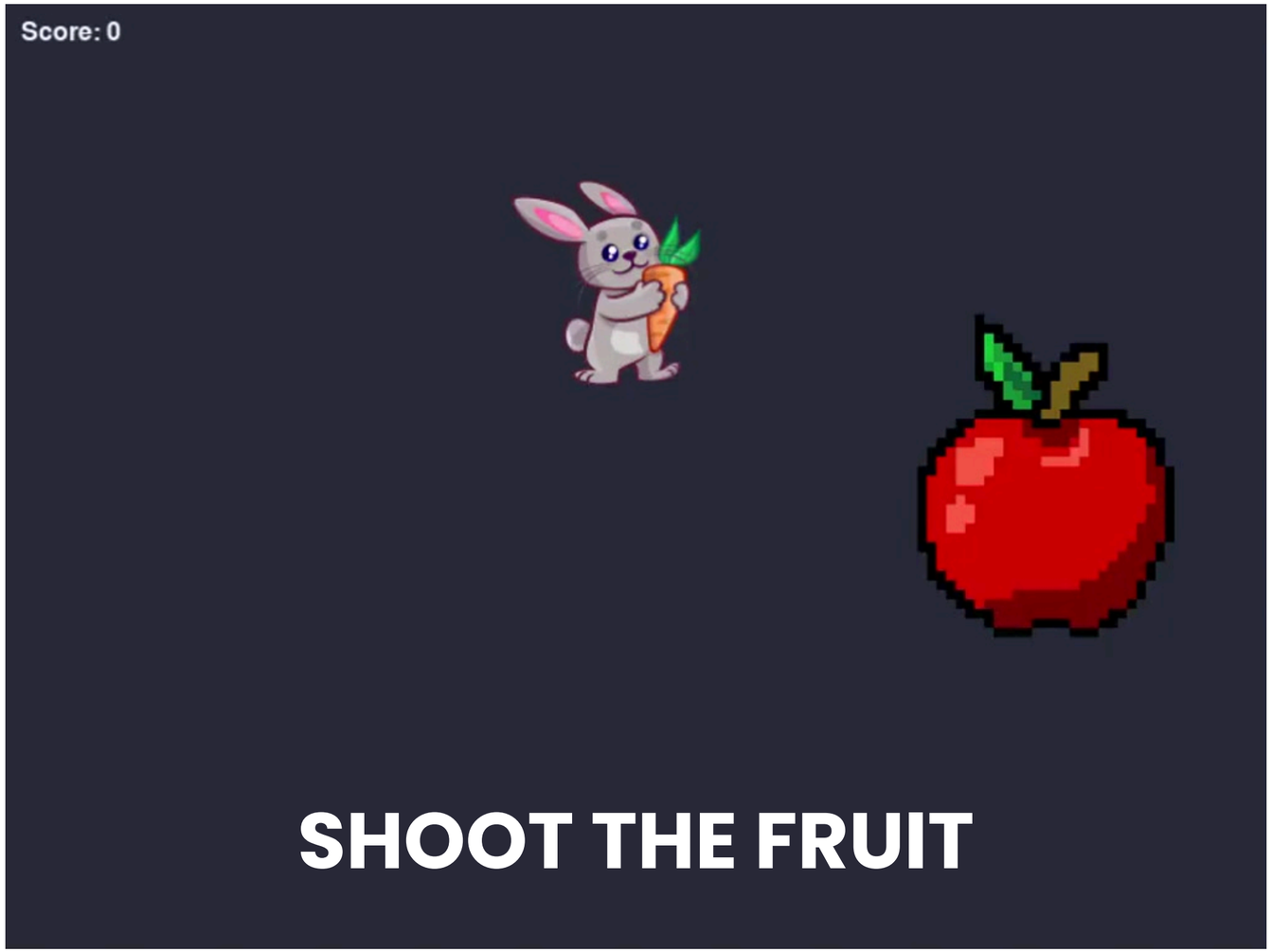
Use the following code in a python script to test your installation:

```
import pgzrun

def draw():
  screen.fill('teal')

pgzrun.go()
```

Pygame Zero Documentation

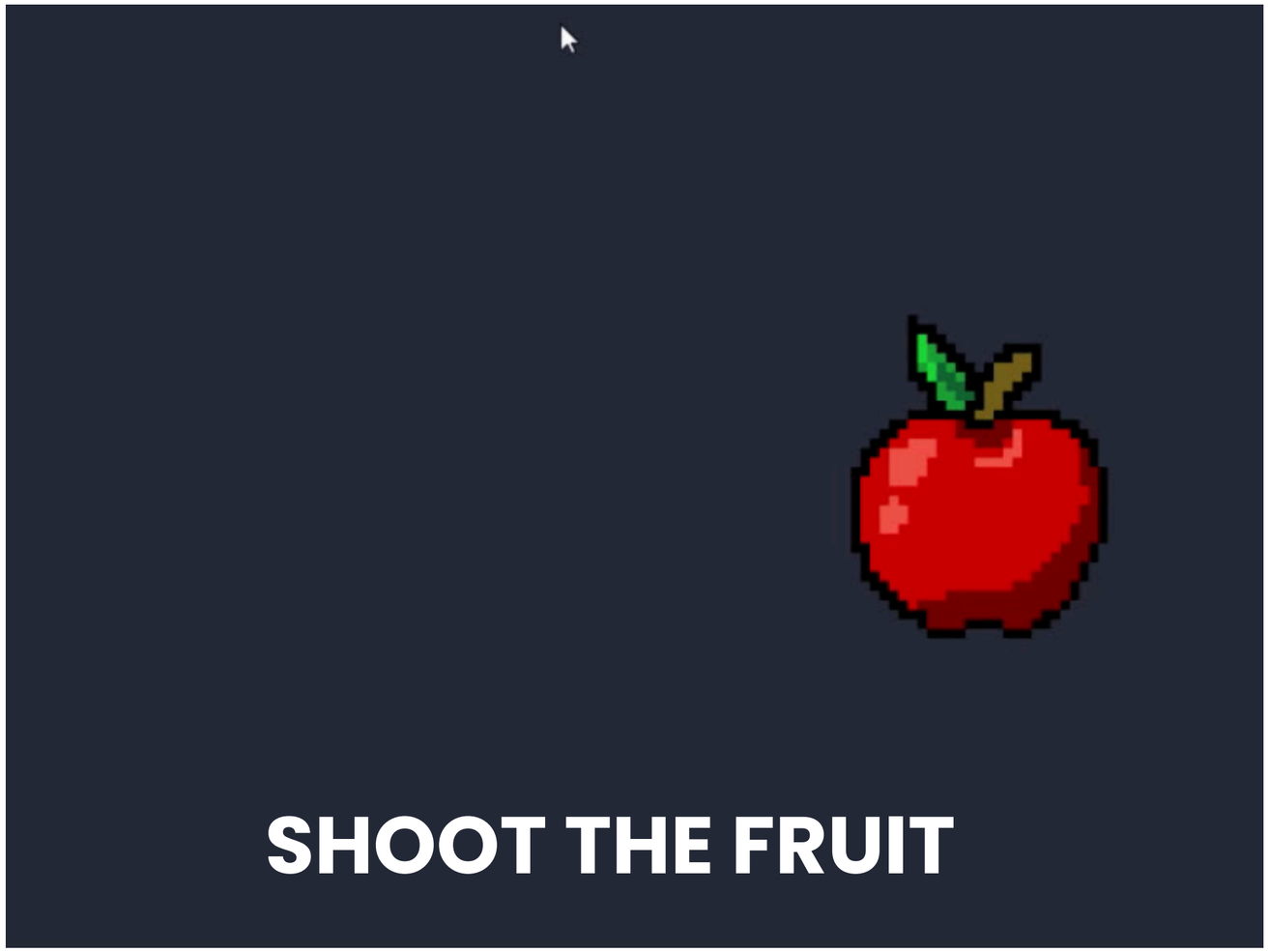
CAN YOU GUESS WHAT WE'LL CREATE TODAY?

VORTEX ACADEMY
THE ART OF THINKING



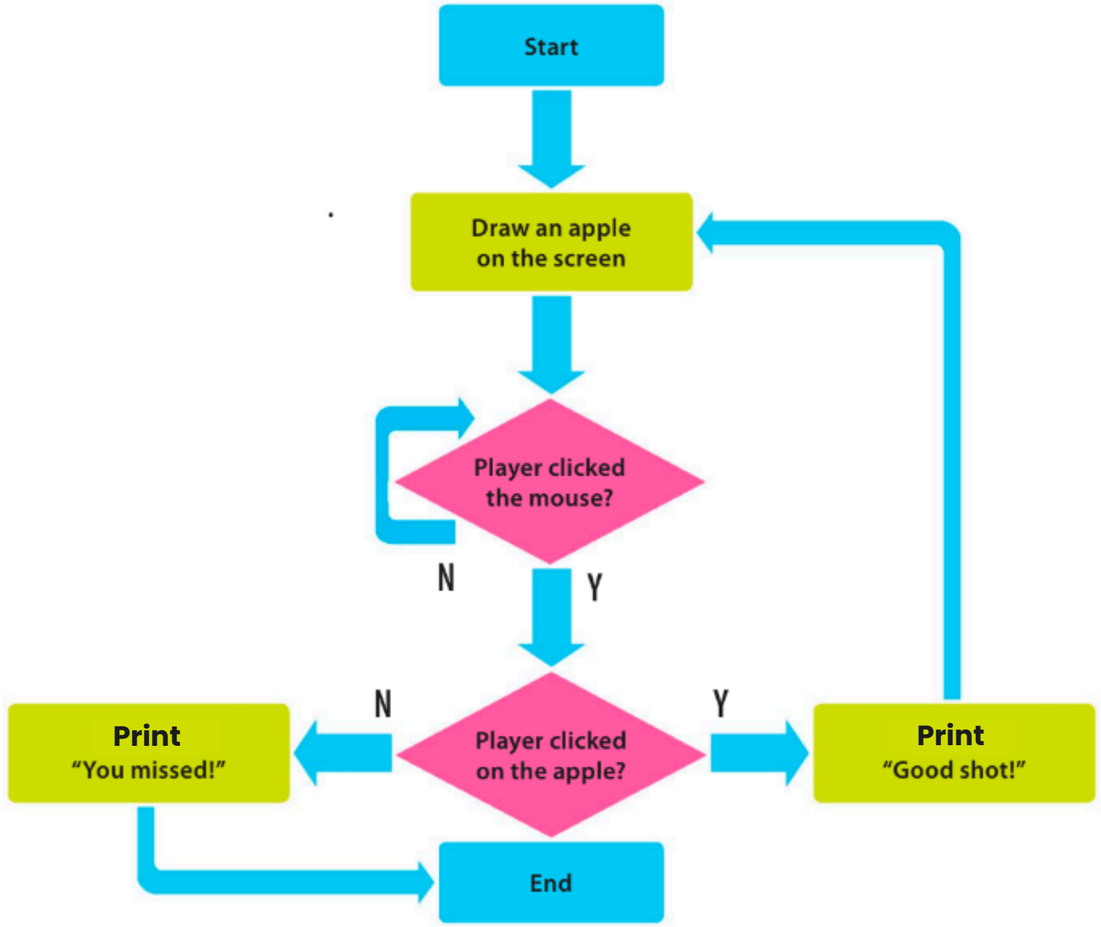
HOW DO WE MAKE SHOOT THE FRUIT?

VORTEX ACADEMY
THE ART OF THINKING



HOW DO WE MAKE SHOOT THE FRUIT?

VORTEX ACADEMY
THE ART OF THINKING

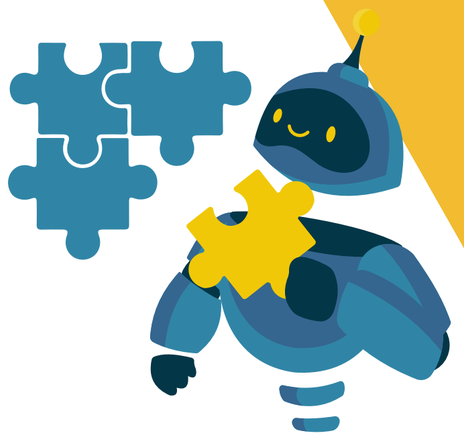


ACTIVITY#1

Try the following code

```
import pgzrun

color = input("Enter the color: ")

def draw():
  screen.clear()
  screen.fill(color)

def update():
  #This allows the window to update when we move it,
  #even if it's empty.
  pass

pgzrun.go()
```

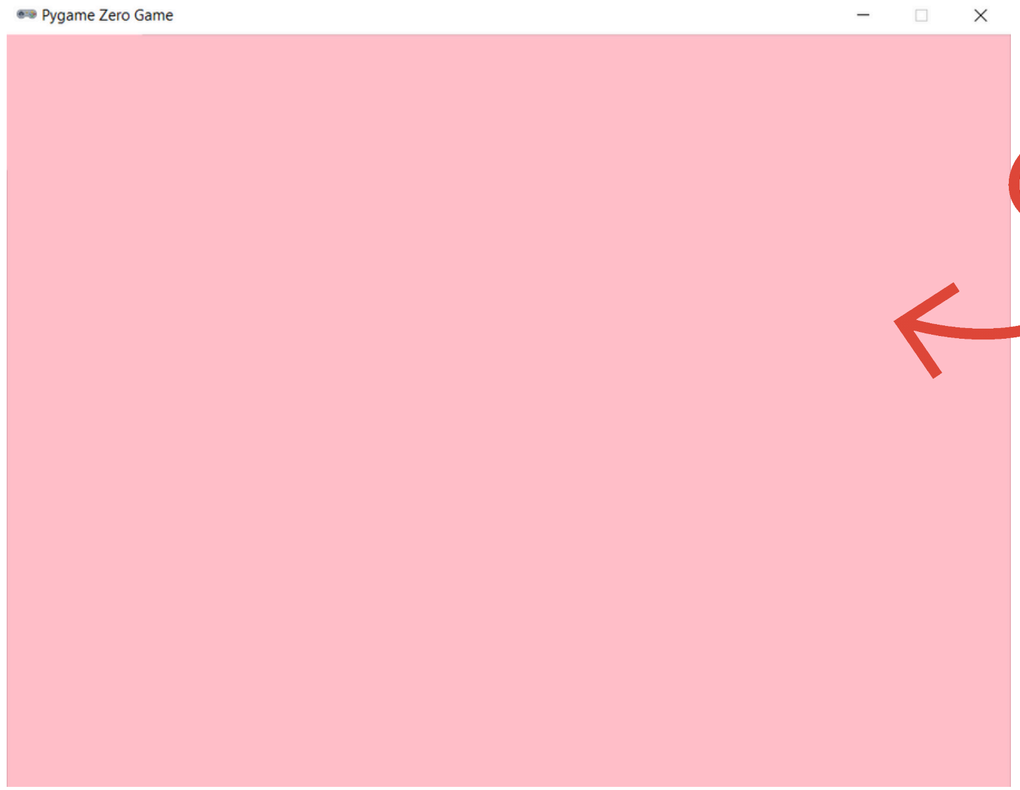
The colors available in pygame zero



VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#1

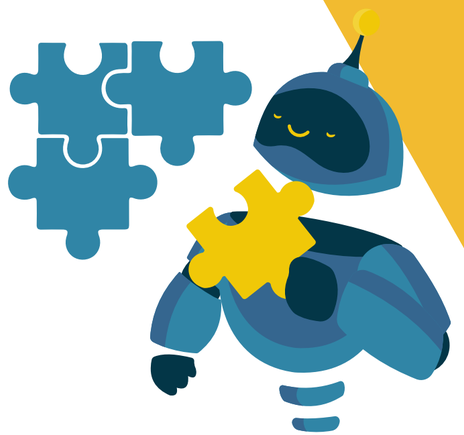
OUTPUT:



Background color should match the color you picked

#HINT: you can try different colors using this website.

The colors available in pygame zero

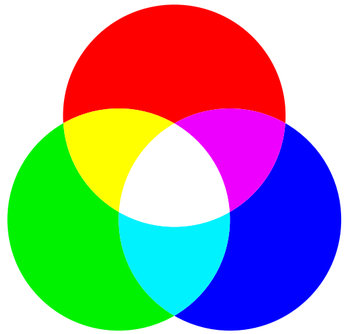


VORTEX ACADEMY
THE ART OF THINKING

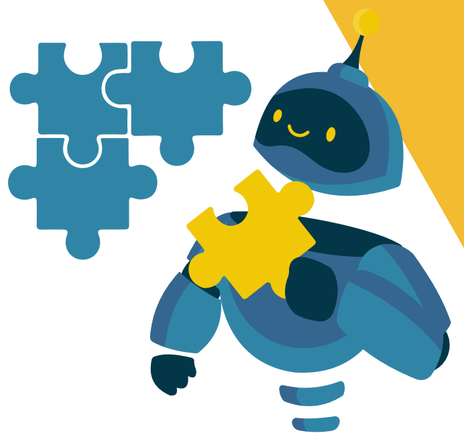
ACTIVITY#2

Modify your code and try

```
import pgzrun
#Change the lines in yellow
color = eval(input("Enter the color as a tuple (e.g., (255, 0, 0)): "))
#The eval() function transforms the input from a string to a tuple.
def draw():
  screen.clear()
  screen.fill(color)

def update():
  pass

pgzrun.go()
```



Use this website to determine the RGB values of colors

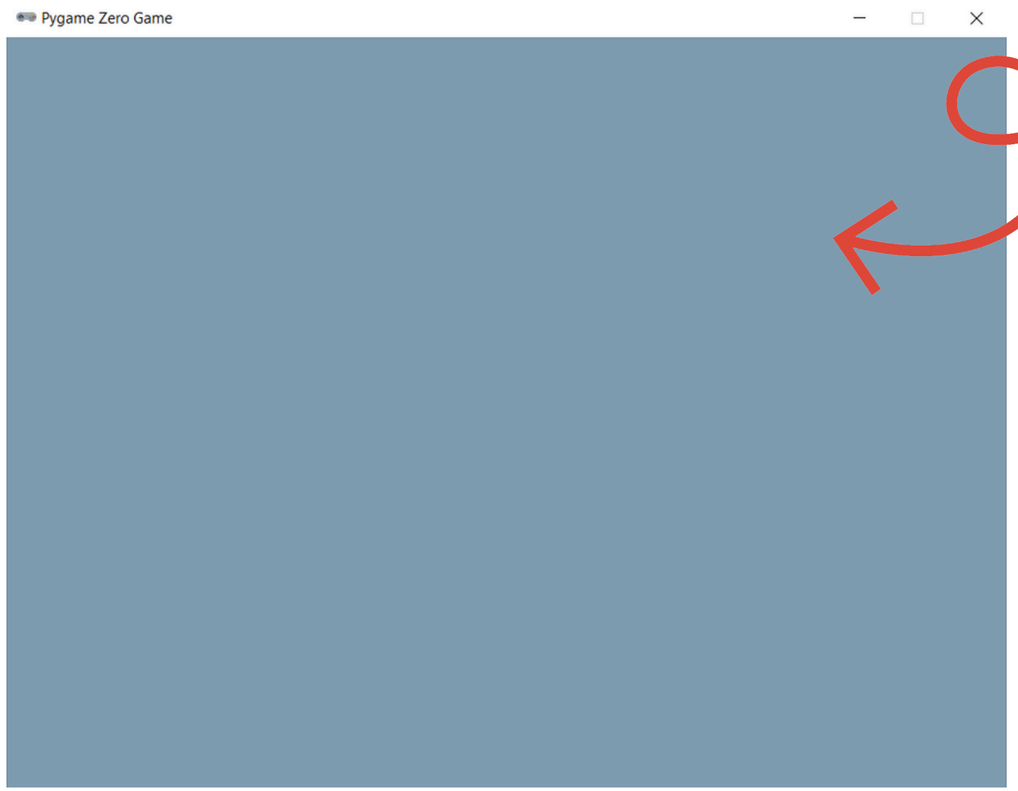


VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#2

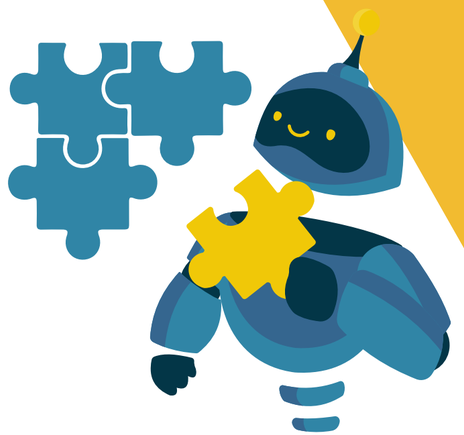
OUTPUT:

Background color should match the color you picked



#HINT: you can try different colors using this website.

Use this website to determine the RGB values of colors



DRAW()

VORTEX ACADEMY
THE ART OF THINKING

The **draw()** function runs in our main game loop to draw all the elements in our game.
Its is responsible for filling the background color in the previous example and any function that is meant to draw anything to our game screen should go inside the **draw()** function.

Syntax:

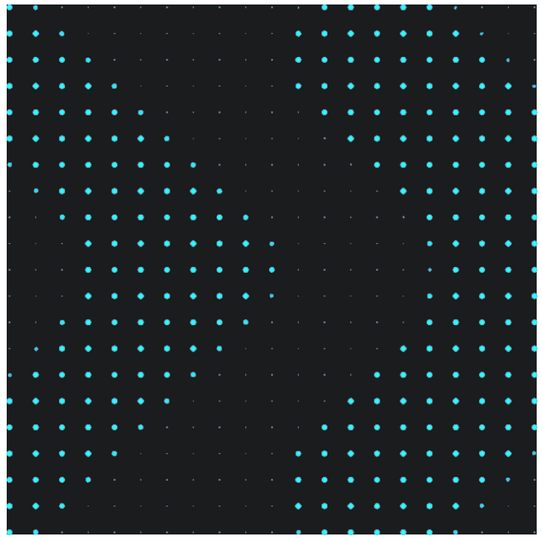
```
def draw():
    #The different functions used to draw anything to our screen
```

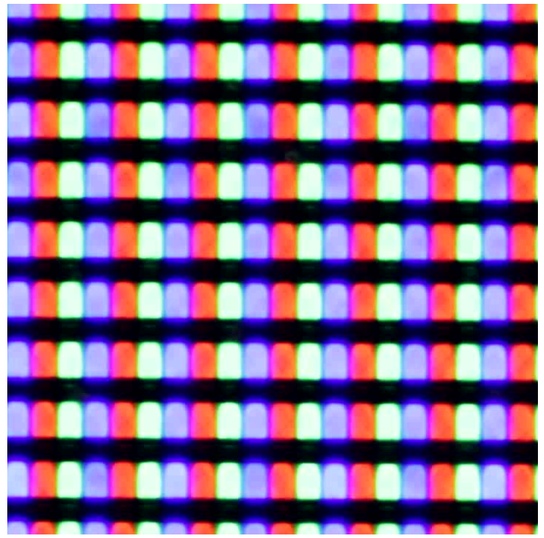
ACTIVITY#3

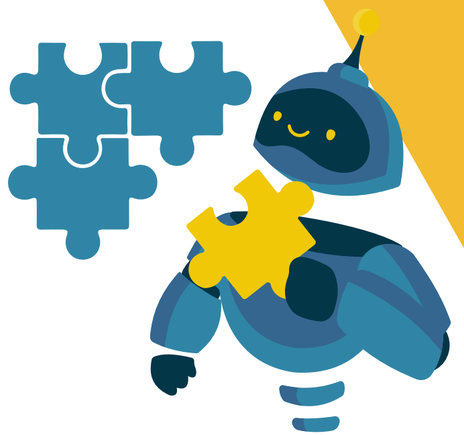
VORTEX ACADEMY
THE ART OF THINKING

Modify your code and try

```
import pgzrun
#Change the lines in yellow
WIDTH = int(input("Enter the screen width: "))
HEIGHT = int(input("Enter the screen height: "))

# You can choose any color you want. We are using slate gray for the game.
color = "slate gray"

def draw():
  screen.clear()
  screen.fill(color)

def update():
  pass

pgzrun.go()
```







ACTIVITY#3

OUTPUT:

Window size should match the size you picked

#HINT: try these resolutions to make the window fill your screen:

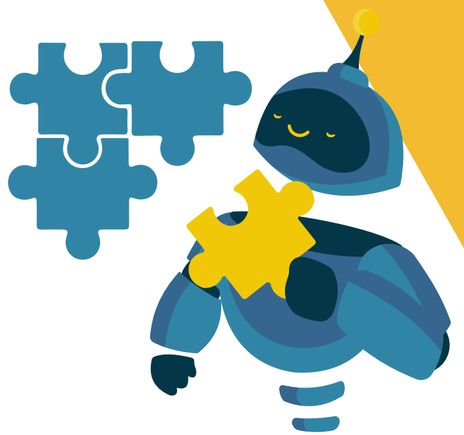
WIDTH: **1920** HEIGHT: **1080**

WIDTH: **1366** HEIGHT: **768**

WIDTH: **1280** HEIGHT: **720**

WIDTH: **2560** HEIGHT: **1440**

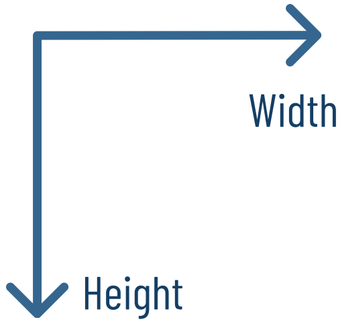
Note that windows scales apps according to display settings.



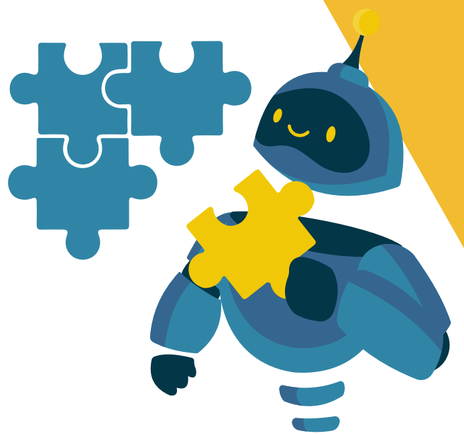
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#4

Modify your code and try




```
#Change the lines in yellow
import pgzrun
# This is the resolution we will use for the game.
WIDTH = 800
HEIGHT = 600

color = “slate gray”

def draw():
  screen.clear()
  screen.fill(color)
  screen.draw.text("Now you are a game developer", (WIDTH/2, HEIGHT/2))

def update():
  pass

pgzrun.go()
```

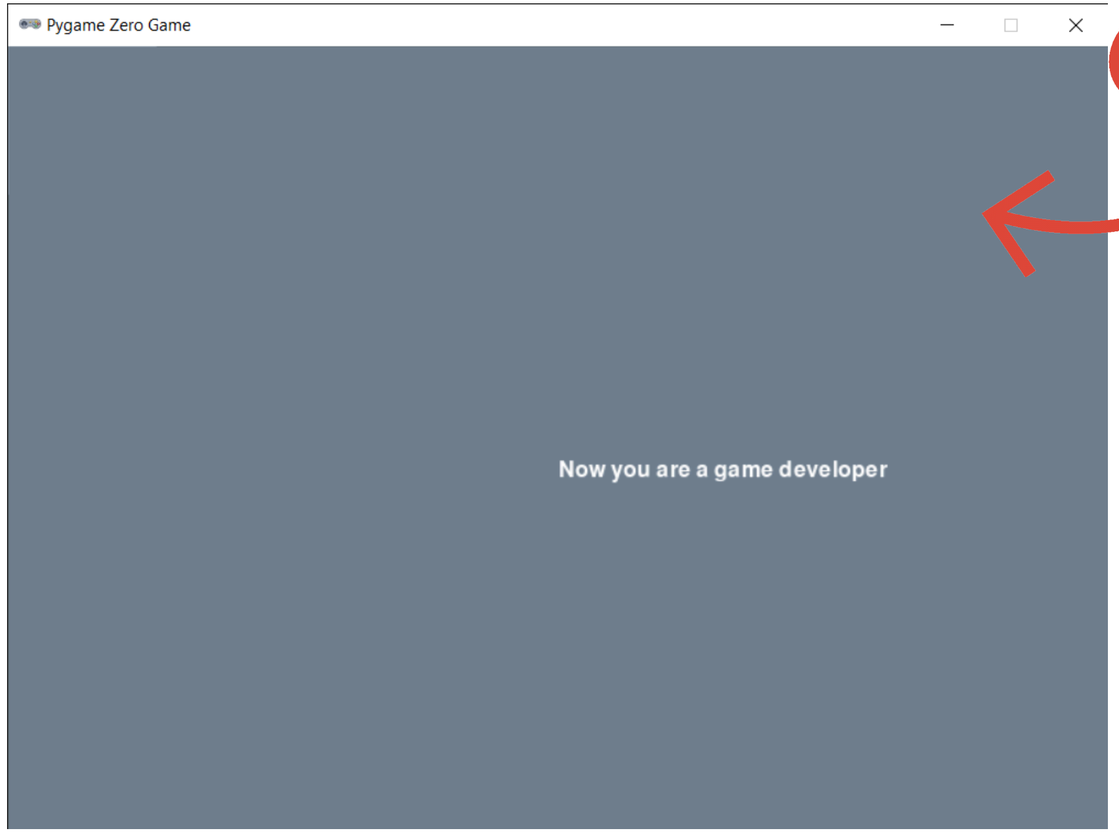


VORTEX ACADEMY
THE ART OF THINKING

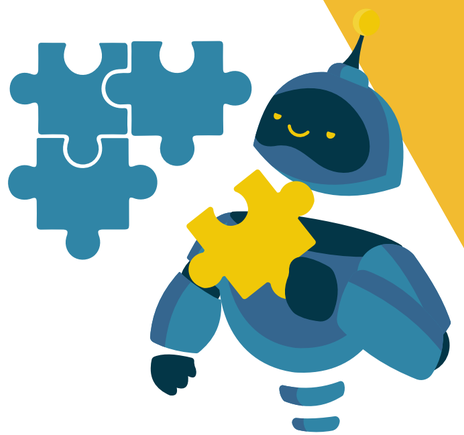
ACTIVITY#4

OUTPUT:

Try to change the text location



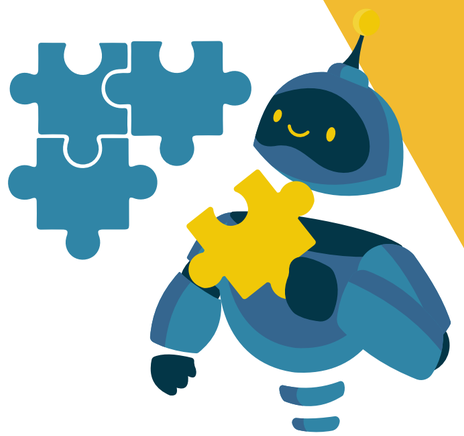
#HINT: in the previous slide the coordinates of the top left corner of the text are determined to be the center of the screen from (WIDTH/2, HEIGHT/2), try passing any other coordinates to the function draw.text() as (100,100) or (523,478) and see what happens.



ACTIVITY#5

Modify your code and try

```
#Change the lines in yellow
import pgzrun

WIDTH = 800
HEIGHT = 600
TITLE = input("Enter the title of the game: ")

color = “slate gray”

def draw():
  screen.clear()
  screen.fill(color)
  screen.draw.text("SHOOT THE FRUIT", (WIDTH/2-90, 10))
# We’ve used (WIDTH/2-90, 10) to position the text at the top of the screen at the center.

def update():
  pass

pgzrun.go()
```

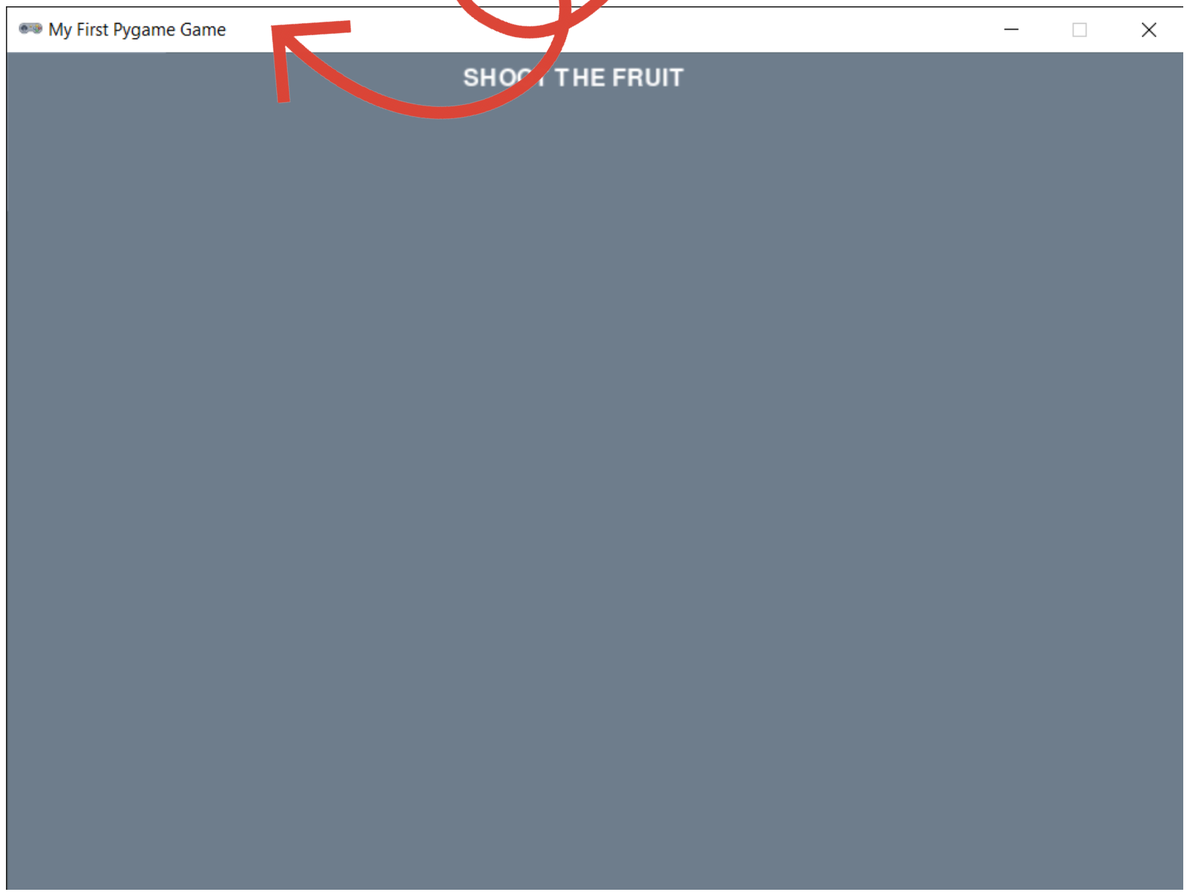


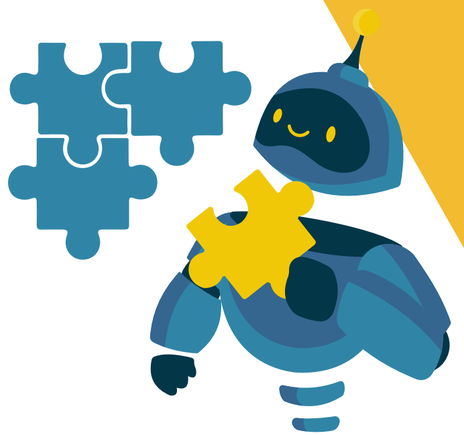
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#5

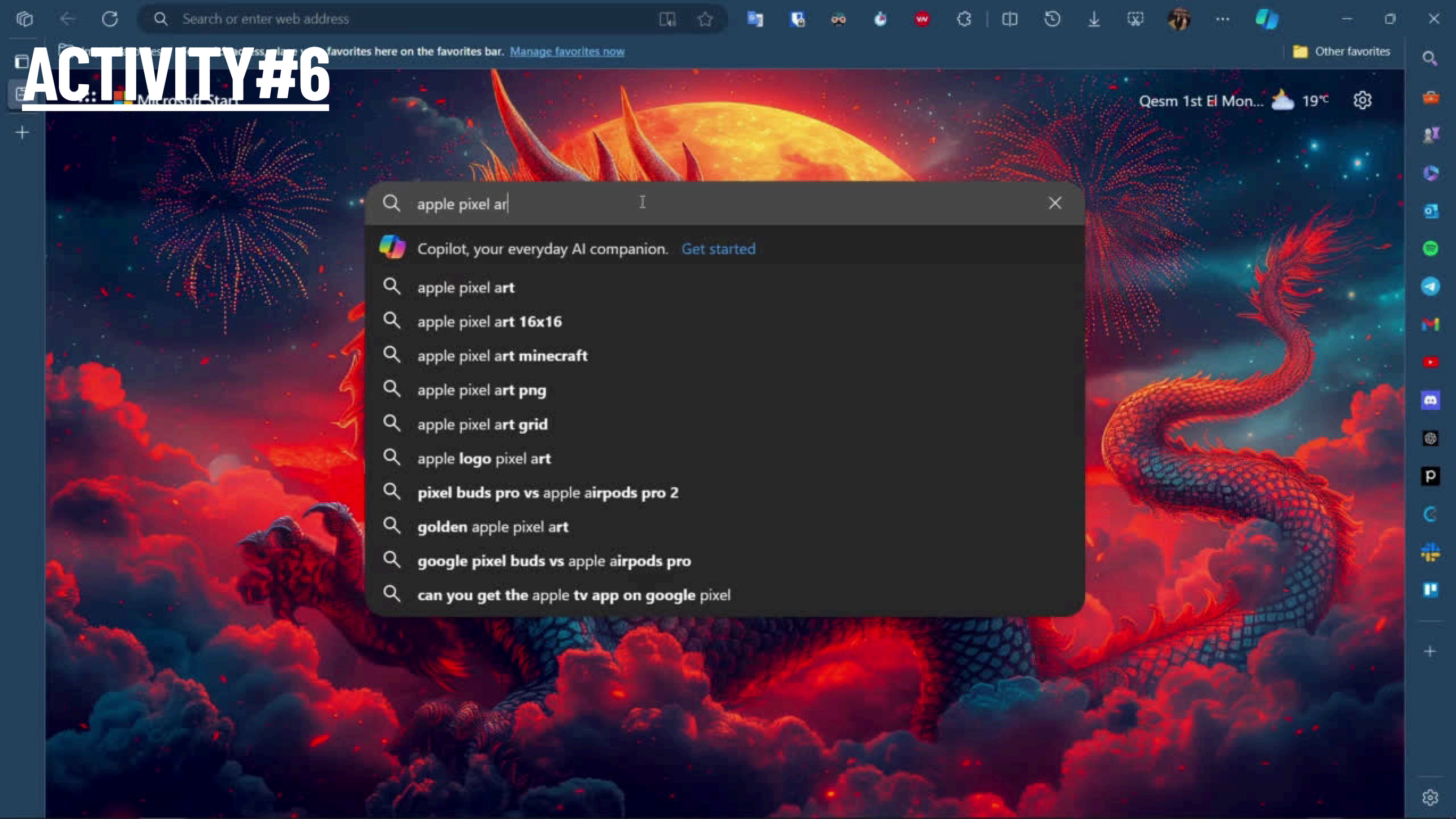
Notice how the window title changed

OUTPUT:





ACTIVITY#6

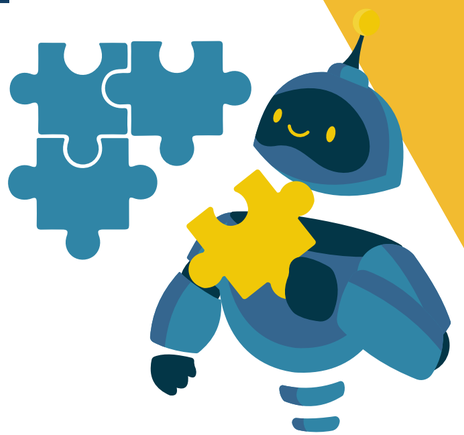

Search or enter web address
favorites here on the favorites bar. Manage favorites now
Other favorites
Qesm 1st El Mon...
19°C
apple pixel ar
Copilot, your everyday AI companion. Get started
apple pixel art
apple pixel art 16x16
apple pixel art minecraft
apple pixel art png
apple pixel art grid
apple logo pixel art
pixel buds pro vs apple airpods pro 2
golden apple pixel art
google pixel buds vs apple airpods pro
can you get the apple tv app on google pixel

ACTIVITY#6

Modify your code and try

```
#Change the lines in yellow
import pgzrun

WIDTH = 800
HEIGHT = 600
TITLE = "MY FIRST GAME"  # We will use this title for the game.
color = "slate gray"

actor = Actor('apple')# Replace ‘apple’ with the name of your png.
actor.pos = WIDTH/2, HEIGHT/2

def draw():
  screen.clear()
  screen.fill(color)
  screen.draw.text("SHOOT THE FRUIT", (WIDTH/2-90, 10))
  actor.draw()

def update():
  pass

pgzrun.go()
```

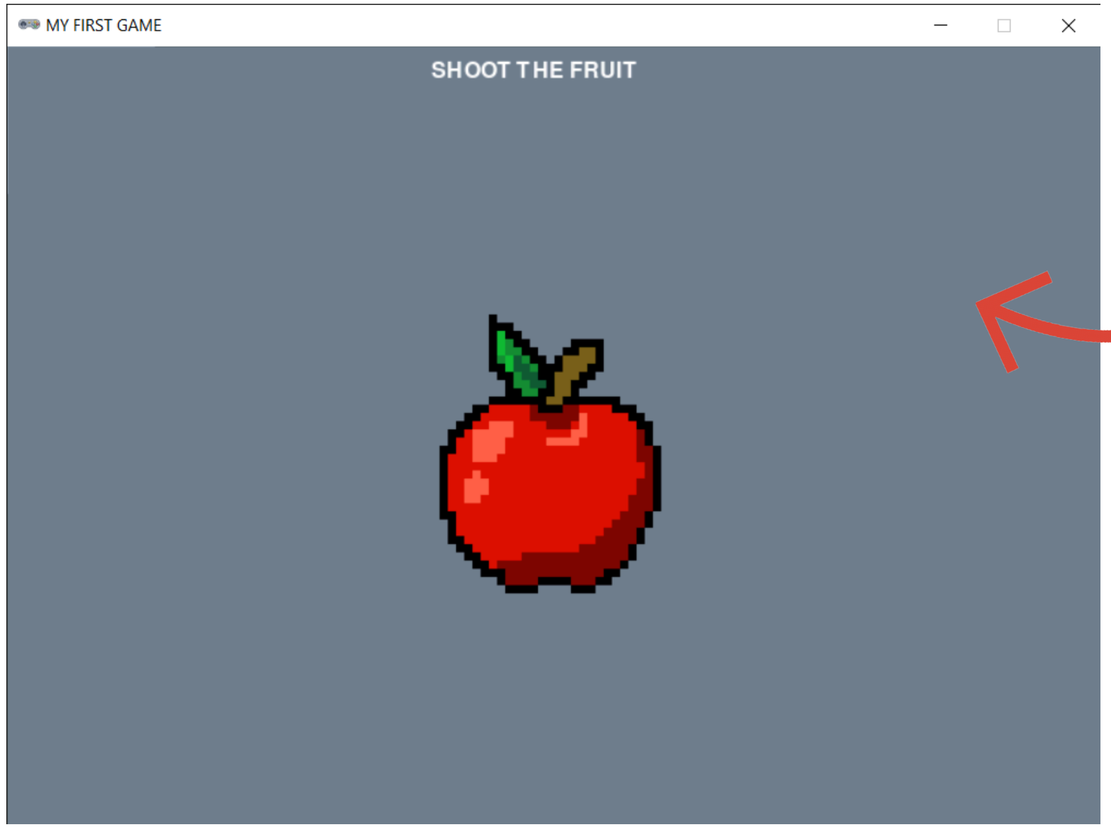
You Can Use This Apple



VORTEX ACADEMY
THE ART OF THINKING

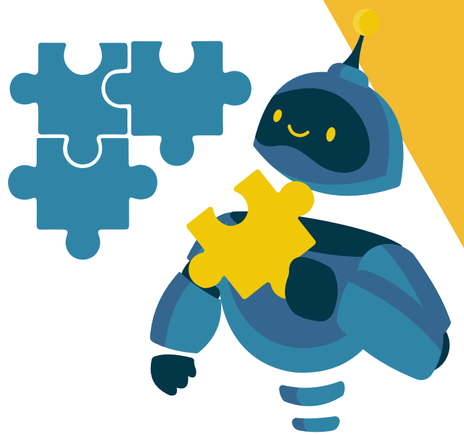
ACTIVITY#6

OUTPUT:



The apple is now in the middle of the screen

#HINT: in the previous slide the coordinates of the apple are determined to be the center of the screen from (WIDTH/2, HEIGHT/2), try giving it any other coordinates as (100,100) or (523,478) and see what happens.



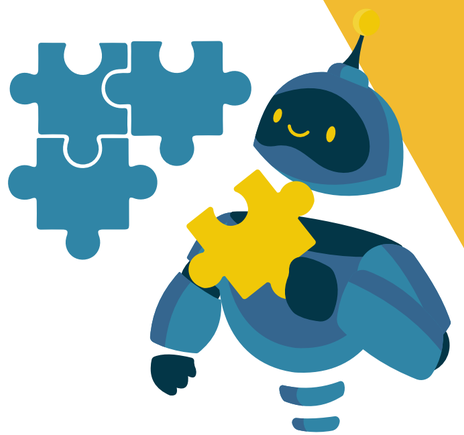
ACTIVITY#7

Modify your code and try

```
#Change the lines in yellow
import pgzrun

WIDTH = 800
HEIGHT = 600
TITLE = "MY FIRST GAME"
color = "slate gray"

actor = Actor('apple')# Replace 'apple' with the name of your png
actor.pos = WIDTH/2, HEIGHT/2

def draw():
  screen.clear()
  actor.draw()
  screen.fill(color)
  screen.draw.text("SHOOT THE FRUIT", (WIDTH/2-90, 10))

def update():
  pass

pgzrun.go()
```

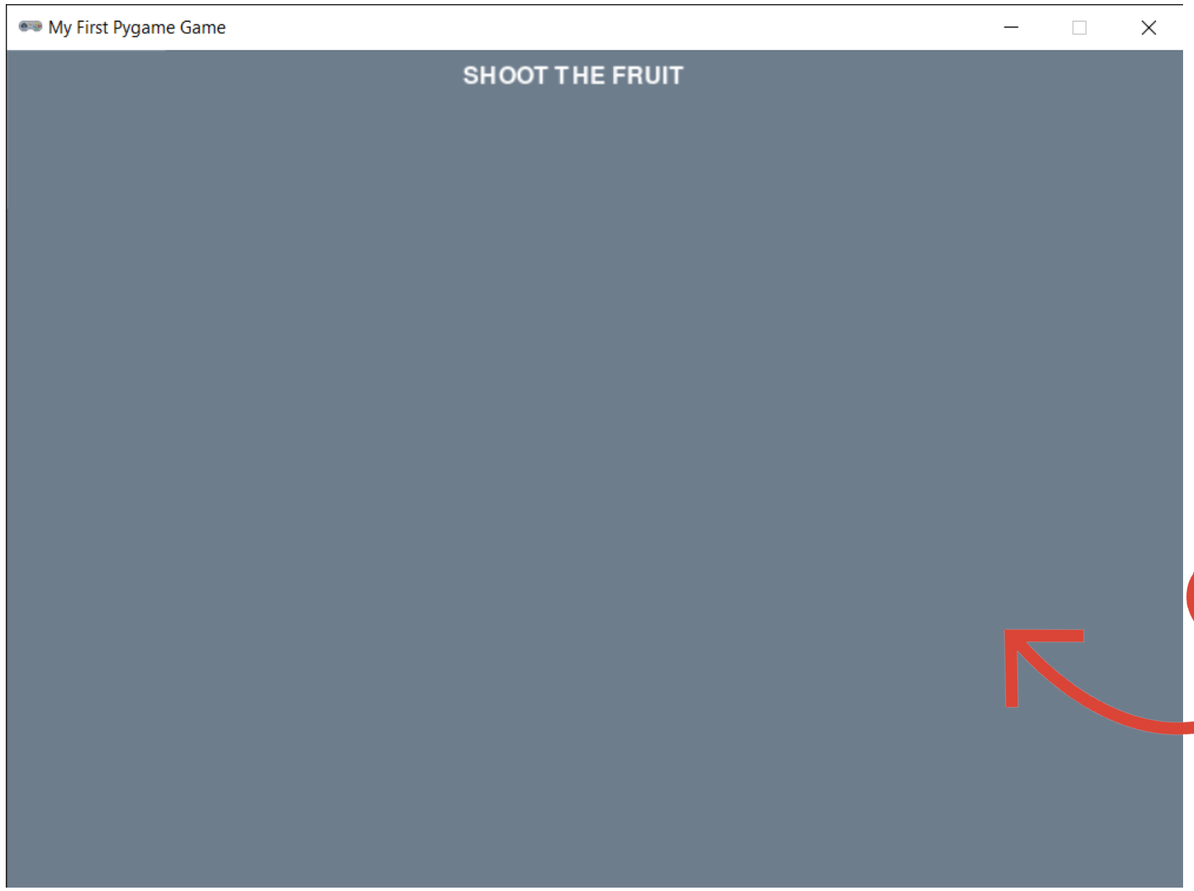
You Can Use This Apple

#NOTE:*The actor should be a .png in a folder named images in the same folder as your python script.*

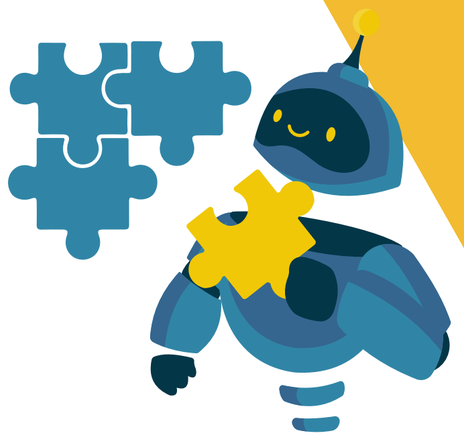


ACTIVITY#7

OUTPUT:



Our apple is hidden behind the background fill, Always watch out for the Drawing order of your game screen.



VORTEX ACADEMY
THE ART OF THINKING

ACTORS

VORTEX ACADEMY
THE ART OF THINKING

Actors are the building blocks of the game.
They allow us to add characters to the game, move them around and interact with them using the mouse and keyboard to complete the game.

To add an actor to the game, we should :

a. Find or create an image to represent the actor.
->We use png files when we want the actor to not have a background

b. Add the image to a folder named images in the same folder as the python script.

c. Add the actor to the script using *"actor_name"* = Actor(*"actor_file_name_in_images"*) and determining their starting position using "actor_name".pos = *starting_width*, *starting_height*

d. Drawing the actor to the screen inside the draw function, while making sure to watch out for drawing order.

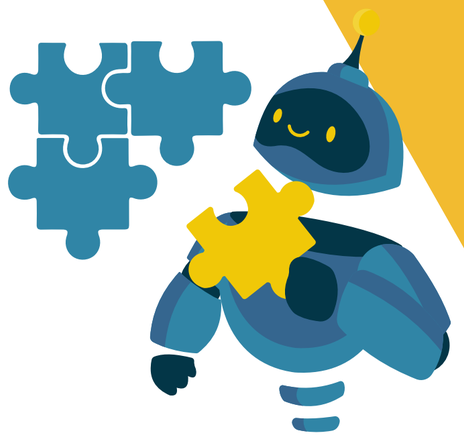
ACTIVITY#8

Modify your code and try

```
#Change the lines in yellow
import pgzrun

WIDTH = 800
HEIGHT = 600
TITLE = "MY FIRST GAME"
color = "slate gray"

actor = Actor('apple')# Replace 'apple' with the name of your png
actor.pos = WIDTH/2, HEIGHT/2

def draw():
 screen.clear()
 screen.fill(color)
 screen.draw.text("SHOOT THE FRUIT", (WIDTH/2-90, 10))
 actor.draw()

def update():
 actor.x += 2
 if actor.x > WIDTH:
  actor.x = 0

pgzrun.go()
```

You Can Use This Apple

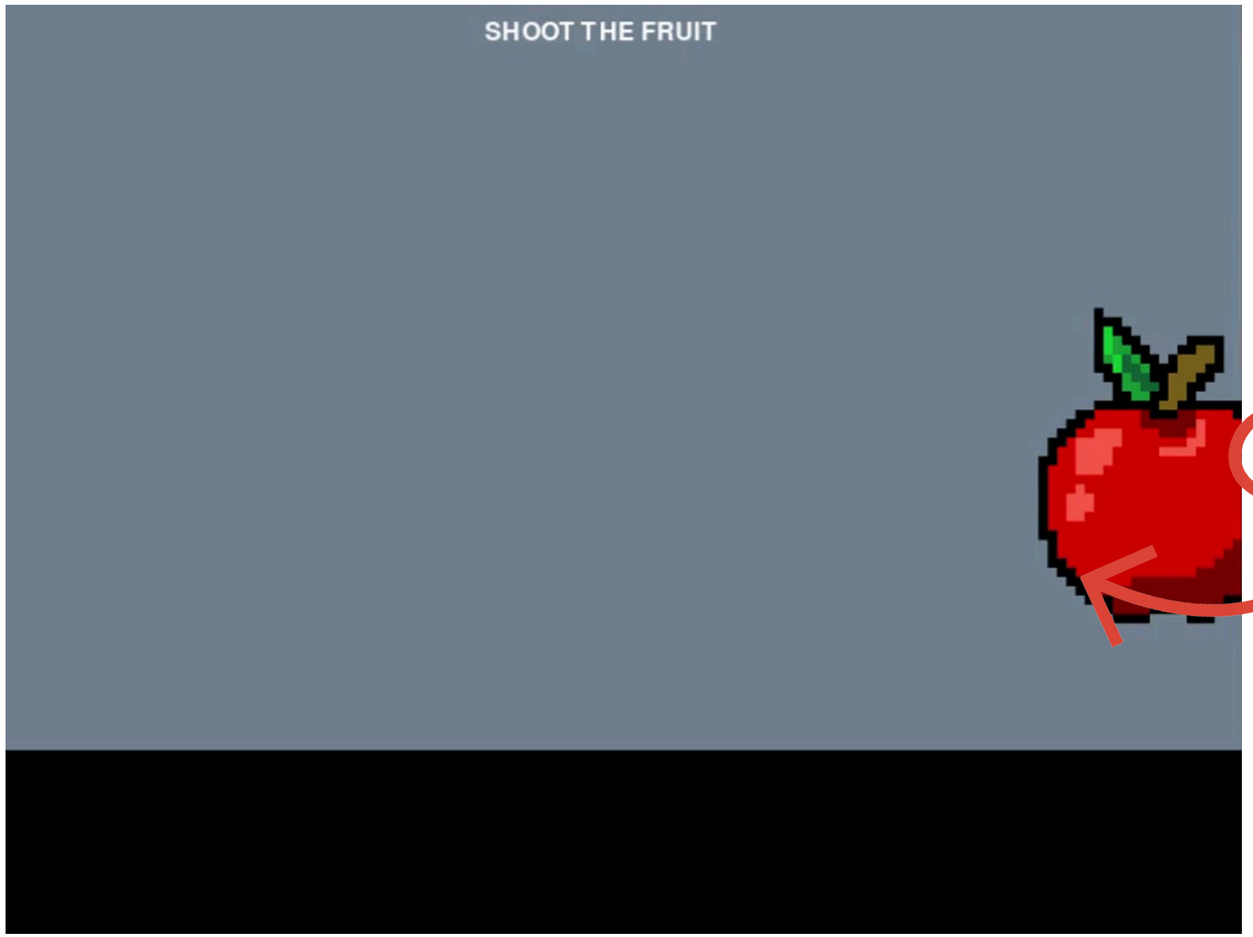
#NOTE:*The actor should be a .png in a folder named images in the same folder as your python script.*



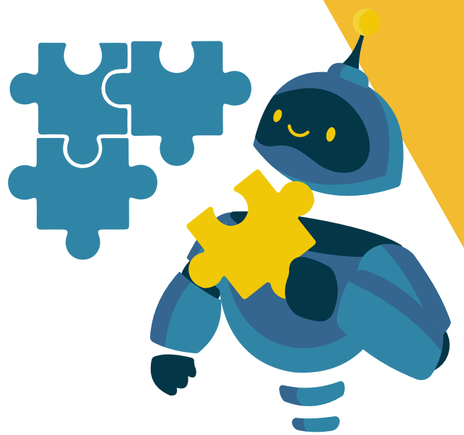
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#8

OUTPUT:



Our apple is now moving across the screen



UPDATE()

VORTEX ACADEMY
THE ART OF THINKING

The **update()** function runs after the **draw()** function to update our game.
It's where our game logic resides to do different functions.
In the previous example it moves our actor by 2 pixels before the draw functions draws it again, so that our actor moves across the screen.

Syntax:

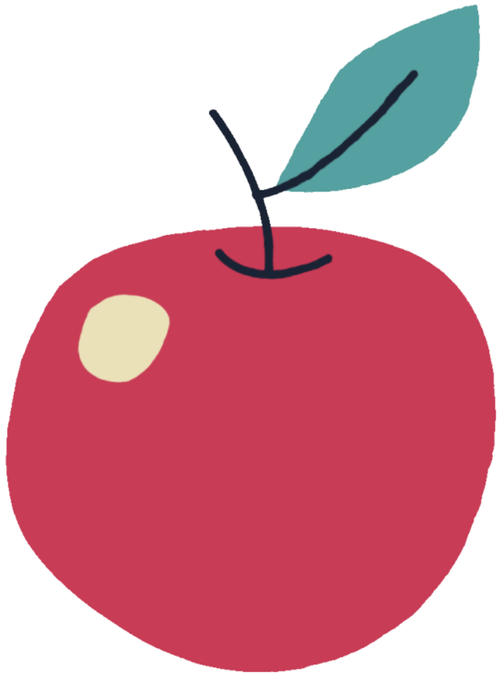
```
def update():
 #Our game logic to perform different functions
```

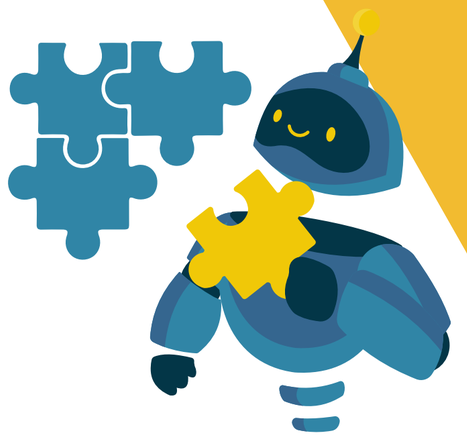
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#9

Add To Your Previous Code & Try It

```
# Add this functon after def draw():

def on_mouse_down(pos):
 if actor.collidepoint(pos):
   actor.pos = WIDTH/2, HEIGHT/2
   print("Good shot!")
 else:
   print("You missed me!")


# And before def update():
```

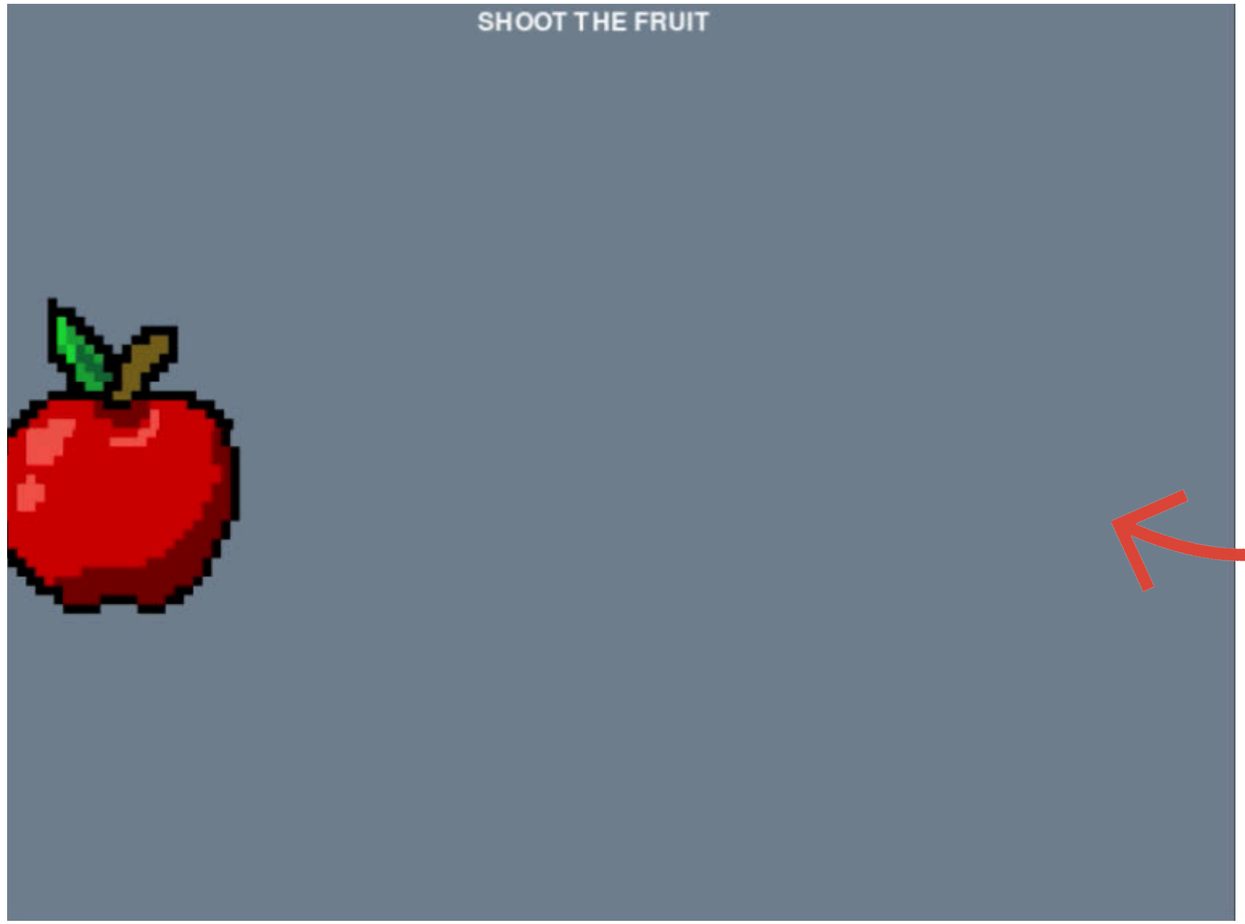




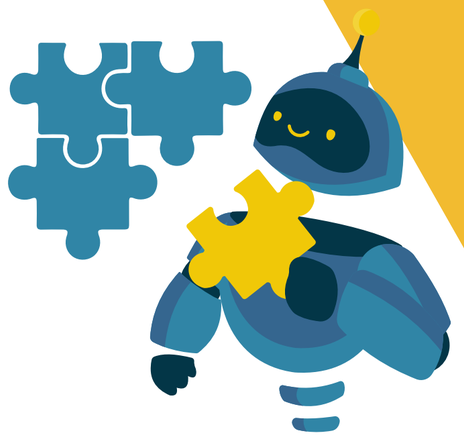
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#9

OUTPUT:



Now when we click on the apple it returns to it's start position



MOUSE CLICKS

VORTEX ACADEMY
THE ART OF THINKING

One of the ways we can interact with our game is mouse clicks.

We use the function **on_mouse_click**(*pixel_coordinates*): to tell pygame what should happen when we click anywhere on our screen.

It receives the pixel coordinates at which we clicked so that we can determine what to do with the click.

In the previous example it recieved ***pos*** which is where we clicked and used the function actor.**collidepoint**(*pos*) to determine if we clicked on our apple or not.

Syntax:
def **on_mouse_down**(*pixel_coordinates*):
  # The code to be executed after mouse click.

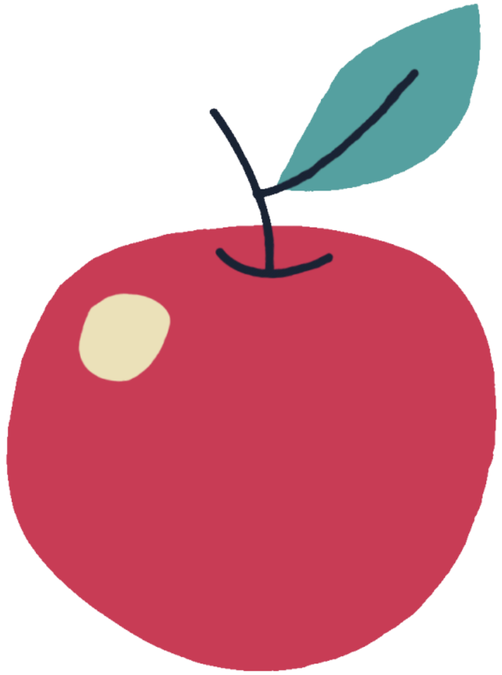
The function "actor_name".**collidepoint**(*pixel_coordinates*) can be used to detect if the pixel coordinates of our click are on our actor "actor_name"

pixel_coordinates: are the pixel coordinates of where we clicked.

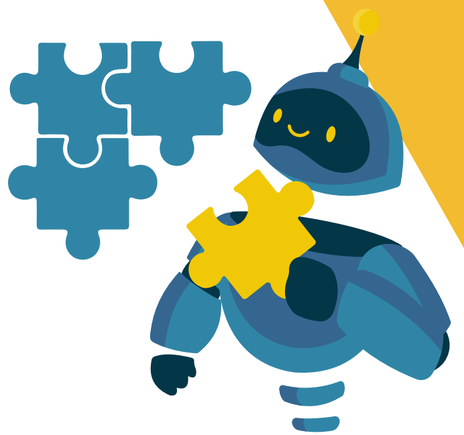
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#10

Add To Your Previous Code & Try It



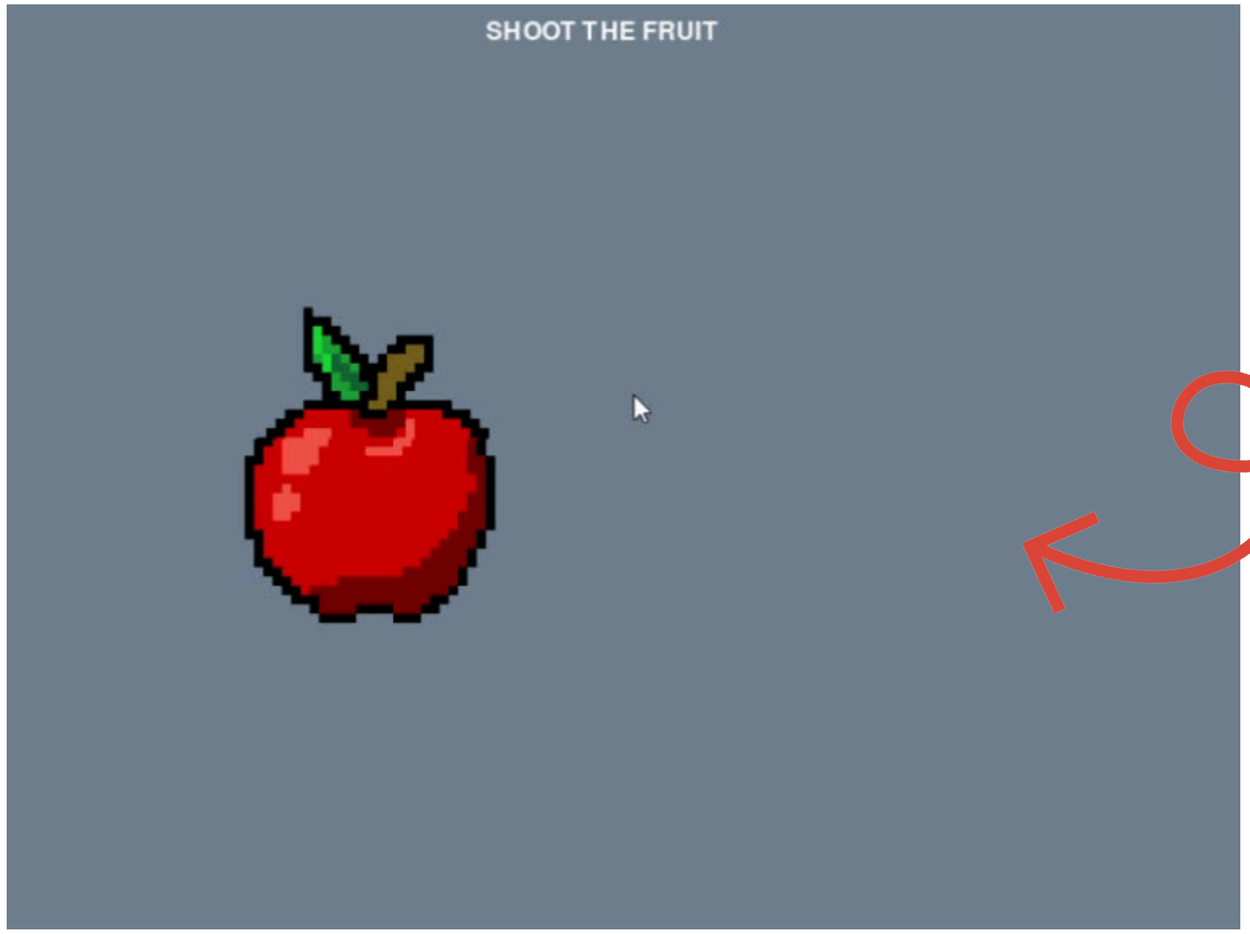
```
# Add this import
import random

# Modify the following function.

def on_mouse_down(pos):
  if actor.collidepoint(pos):
    actor.pos = random.randint(100, 700), random.randint(100, 500)
    print("Good shot!")
  else:
    print("You missed me!")
    quit()
```



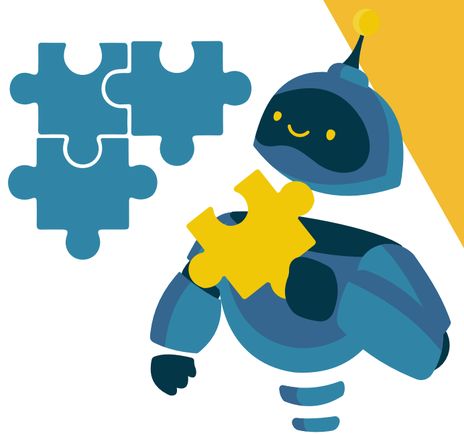
VORTEX ACADEMY
THE ART OF THINKING

ACTIVITY#10

OUTPUT:



Now we have completed SHOOT THE FRUIT



SHOOT THE FRUIT

VORTEX ACADEMY
THE ART OF THINKING

Now we have finished our first game and you are officially a game developer.
What's next? what additions do you want to add to the game?
How can we increase the game speed?
Can we use different Fruits?

HINT: To match the game in the second slide use actor.**x** += *6*
background color = (*40, 43, 59*)
To change fruits you should change the actor image

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 it has a game speed of 6 and make your background color (40, 43, 59), with the apple image as your actor.

Task2: Add a score counter to the top left corner of your screen that increases every time you shoot the fruit, also print the final score to the terminal at the end of the game.

Task3: Make the game only decrease your score when you miss instead of quitting.

Task4: Add another Actor to your game that is an animal that moves to the left with a rate of 4 and jumps randomly to another location when it leaves the screen, shooting this animal will quit the game and make you lose.

VORTEX ACADEMY
THE ART OF THINKING

SKILL MASTERY CHALLENGE

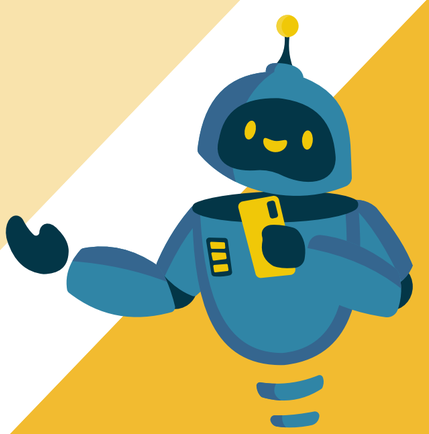
LOCKED & LOADED

LOCKED & LOADED

OUTPUT:

```
PS M:\Vortex Fury Code> python -u "m:\Vortex Fury Code\Skill
Welcome to the Mystery Box Treasure Hunt!
Pick a box between 1 and 3. You have 3 attempts.
Attempt 1: Choose a box:
```

VORTEX ACADEMY
THE ART OF THINKING



LOCKED & LOADED

VORTEX ACADEMY
THE ART OF THINKING

Requirments:

1. Import Required Libraries:

1.1. Import the random module to enable random number generation.

2. Define the Main Game Function:

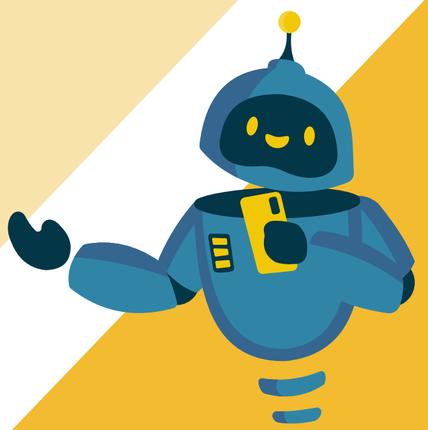
2.1. Create a function named play_round.

3. Initialize Game Variables within play_round:

3.1. Randomly select the number of treasure boxes (num_boxes) between 3 and 7.
3.2. Randomly select a treasure box number (treasure_box) between 1 and num_boxes.
3.3. Initialize an attempts counter to 0.
3.4. Initialize a score variable to 0.

4. Provide Instructions to the Player:

4.1. Display a message informing the player to pick a box between 1 and num_boxes and that they have 3 attempts.



LOCKED & LOADED

VORTEX ACADEMY
THE ART OF THINKING

Requirments:

5. Implement the Guessing Loop:

5.1. Create a loop that allows the player up to 3 attempts to guess the correct box.

- Input Handling:

5.1.1. Prompt the player to input their guess, displaying the current attempt number.

5.1.2. Use a try-except block to handle non-integer inputs:

- If the input is not an integer, display an error message and prompt again without consuming an attempt.

5.1.3. Check if the guessed number is within the valid range (1 to num_boxes):

- If not, display an error message and prompt again without consuming an attempt.

- Input Validation:
  - Ensure that the player's input is a valid integer within the specified range.
  - Use a try-except block to catch ValueError exceptions when converting input to an integer.
  - If the input is invalid or out of range, prompt the player again without incrementing the attempts counter.

LOCKED & LOADED

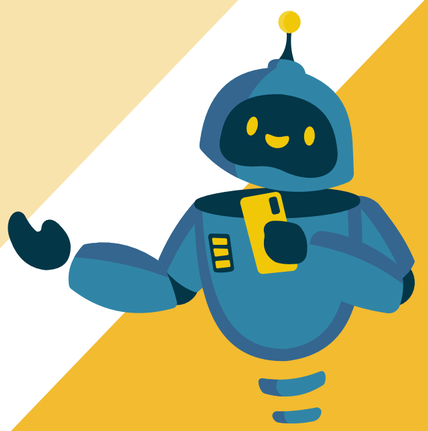
VORTEX ACADEMY
THE ART OF THINKING

Requirments:

- Processing the Guess:
  5.1.4. Increment the attempts counter by 1.
  5.1.5. Compare the player's guess to the treasure_box:
  - If the guess is correct:
    - Display a congratulatory message.
    - Calculate the score as 4 - attempts (the score decreases with each wrong attempt).
    - Scoring Mechanism:
      - The player starts with a maximum possible score of 3.
      - The score decreases by 1 with each incorrect guess.
      - If the player finds the treasure on the first attempt, they receive a score of 3.
      - If they find it on the second attempt, the score is 2, and so on.
    - Exit the loop.

  - If the guess is incorrect:
    - If there are remaining attempts (less than 3 total attempts):
      - Provide a hint indicating whether the treasure is in a higher or lower numbered box compared to the guessed box.
      - Hints to the Player:
        - After an incorrect guess (and if attempts remain), inform the player whether the treasure is in a higher or lower numbered box to assist in their next guess.

    - If there are no remaining attempts:
      - Inform the player that they are out of attempts and reveal the correct treasure box number.



LOCKED & LOADED

Requirments:

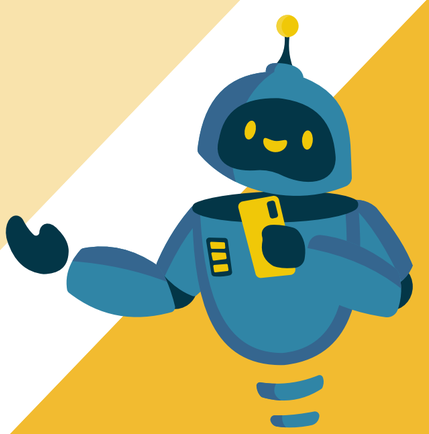
VORTEX ACADEMY
THE ART OF THINKING

6. Return the Player's Score:

6.1. After the loop concludes, return the score from the play_round function.

7. Execute the Game:

7.1. Display a welcome message: "Welcome to the Mystery Box Treasure Hunt!"
7.2. Call the play_round function.
7.3. Store the returned score in a variable round_score.
7.4. Display the player's score for the round.
7.5. Thank the player for playing: "Thanks for playing!"

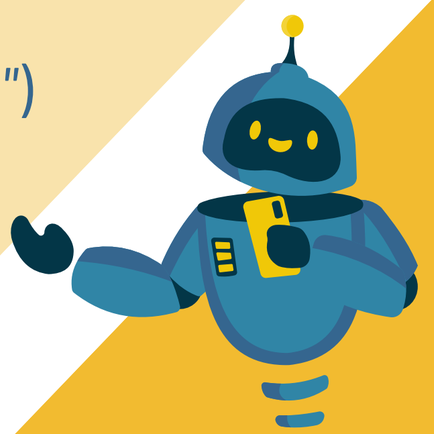


LOCKED & LOADED

Solution:

VORTEX ACADEMY
THE ART OF THINKING

```
import random
# Function to play the game
def play_round():
  num_boxes = random.randint(3, 7)
  treasure_box = random.randint(1, num_boxes)
  attempts = 0
  score = 0

  print(f"Pick a box between 1 and {num_boxes}. You have 3 attempts.")
```

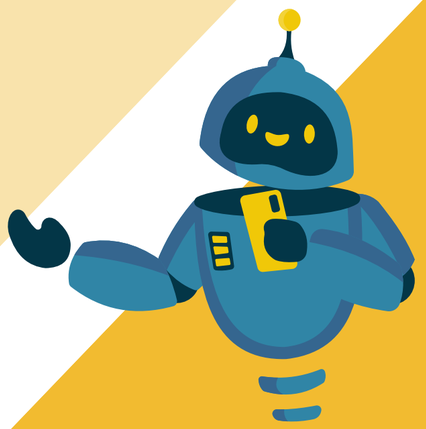


LOCKED & LOADED

VORTEX ACADEMY
THE ART OF THINKING

Solution:

```
# Continue in play_round function.
 while attempts < 3:
  try:
   guess = int(input(f"Attempt {attempts + 1}: Choose a box: "))
  except ValueError:
   print("Please enter a valid number.")
   continue

  if guess < 1 or guess > num_boxes:
   print(f"Please choose a box between 1 and {num_boxes}.")
   continue

  attempts += 1

  if guess == treasure_box:
   print("Congratulations! You found the treasure!")
   score = 4 - attempts # Score decreases with each wrong attempt
   break
  else:
   if attempts < 3:
    hint = "higher" if guess < treasure_box else "lower"
    print(f"Wrong guess! Try a {hint} box.")
   else:
    print(f"Sorry, you're out of attempts! The treasure was in box {treasure_box}.")

 return score
```

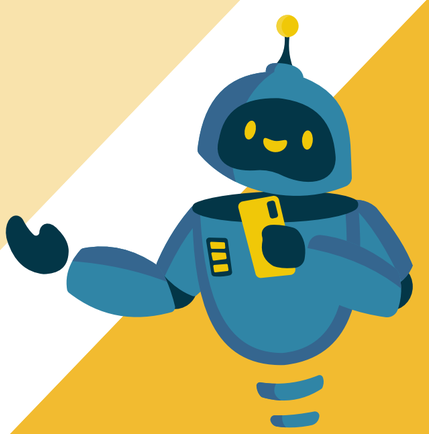


LOCKED & LOADED

VORTEX ACADEMY
THE ART OF THINKING

Solution:

```
print("Welcome to the Mystery Box Treasure Hunt!")
round_score = play_round()
print(f"Your score for this round: {round_score}")
print("Thanks for playing!")
```



VORTEX ACADEMY
THE ART OF THINKING

THANK YOU

FOR YOUR ATTENTION