VIDHI ROHIRA

SY BTECH

COMPUTER ENGINEERING

OST LAB 5

231071052

BATCH-C

LABORATORY 5

AIM:- Collaboration and Contribution- open source project workflows, contributing to open source projects, case studies of successful open source projects.

THEORY:-

Case Study: Scaling Open-Source Contribution with Git and GitHub

BACKGROUND:
Organization: OpenAI DevOps Team
Industry: Artificial Intelligence (AI) & Machine Learning (ML)
Project Size: 50 core contributors, 200+ external contributors
Location: Global (remote, distributed team)

Challenge
The OpenAI DevOps team faced challenges in scaling its code management and collaboration as more contributors, both internal and external, joined the

open-source development. Some of the key issues included:

- Inefficient workflows leading to slow integration of new features.
- Frequent merge conflicts, especially with contributions from external developers.
- Lack of clarity in project management and code review processes, resulting in delays and duplication of efforts.
- Maintaining code quality with the growing number of pull requests (PRs).

Objectives

- Streamline the collaboration process to handle a larger volume of contributions.
- Ensure smoother integration of code with minimal conflicts.
- Increase visibility into project progress and facilitate faster releases.
- Maintain high standards of code quality while reducing bottlenecks in the review process.

Solution

The OpenAI team adopted **Git** and **GitHub** to address these challenges, focusing on optimizing workflows and leveraging automation for better collaboration.

1. Adoption of Git Workflow:

- **Feature Branches**: Contributors, both internal and external, worked on dedicated branches for each feature, bug fix, or enhancement. This allowed them to develop independently and minimized the risk of breaking the main codebase.
- **Pull Requests (PRs)**: Every contribution, whether internal or external, was submitted as a PR. The GitHub PR system ensured that code was peer-reviewed before being merged into the main repository. External contributors especially benefited from the feedback loop within the PR discussions.

2. Implementing GitHub Actions for Automation:

- **Automated Testing**: GitHub Actions was used to set up automated testing for every PR. This ensured that any new code adhered to existing standards and did not break any functionalities. Contributors received immediate feedback when their code failed tests, expediting the review process.
- **Continuous Integration/Continuous Deployment (CI/CD)**: A CI/CD pipeline was set up using GitHub Actions to automate building, testing, and deploying code. This significantly reduced the time

between development and deployment, allowing new features and fixes to go live faster.

3. Project Management with GitHub Projects:

- **Kanban Boards**: The team utilized GitHub Projects' Kanban boards to track ongoing tasks, including features in development, bugs, and upcoming releases. This helped with better task visibility and prioritization, especially in managing contributions from external developers.
- **Milestones and Issues**: Milestones were set up to plan sprints and releases. GitHub's issue tracker enabled clear assignment of tasks, allowing contributors to easily pick up tasks they could work on.

4. Structured Communication & Code Review:

- **Weekly Syncs and Code Reviews**: The team held regular sync meetings and virtual code reviews to discuss ongoing issues and upcoming features. This provided a platform for contributors to raise concerns, ask questions, and clarify any changes before submitting PRs.
- **Mentorship for New Contributors**: A mentorship program was set up to guide new contributors in understanding the workflow, setting up their development environment, and following best practices.

Implementation

1. Training and Onboarding Sessions:
Workshops were organized to train internal developers and external contributors on using Git, GitHub, and the team's specific workflows. This reduced friction during the transition and ensured everyone followed the same processes.

2. Documentation:
The team created clear and comprehensive documentation on:

- **Branching Strategies**: Guidelines on creating feature branches, naming conventions, and merging practices.
- **Contribution Guidelines**: A well-defined contribution guide to help external contributors get involved easily. This included information on code standards, testing requirements, and submitting PRs.
- **Automated Testing**: Instructions on how to run tests locally before submitting code, helping to avoid unnecessary back-and-forth during the PR process.

Results

- **Increased Contribution Efficiency**:
  The structured Git workflow, combined with automated testing and CI/CD pipelines, enabled the team to integrate new features and fixes faster. External contributions, in particular, became smoother as developers received quick feedback on their PRs.
- **Reduced Merge Conflicts**:
  By utilizing feature branches and regular PR reviews, the team significantly reduced the number of merge conflicts, even with a large number of concurrent contributions. The CI/CD pipeline ensured that only passing code was merged, preventing issues from surfacing in the main branch.
- **Improved Project Management and Visibility**:
  The use of GitHub Projects and milestones gave both internal developers and external contributors clear visibility into the project's progress. This also helped prioritize urgent fixes and features, reducing delays in releases.
- **Enhanced Code Quality**:
  GitHub Actions' automated testing and the robust code review process ensured that the project maintained high code quality standards. PR reviews became more collaborative, with senior

developers providing mentorship and constructive feedback to newer contributors.

- **Higher Contributor Satisfaction**:
  The streamlined contribution process, clear communication channels, and faster feedback loops improved overall satisfaction among contributors, leading to a more engaged and productive open-source community.

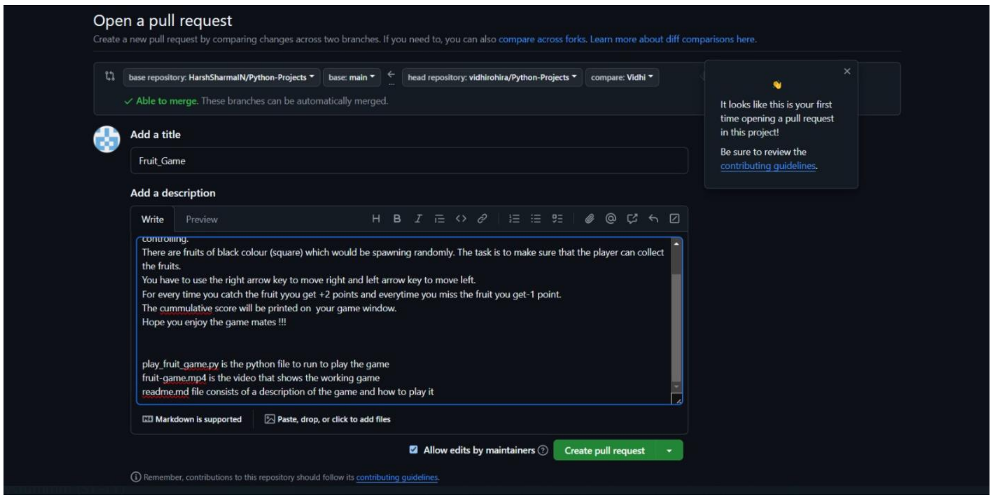
Conclusion
The adoption of Git and GitHub transformed the OpenAI DevOps team's collaboration model. By integrating automated workflows, clear project management tools, and fostering a culture of open communication, the team was able to scale their development efforts efficiently. This allowed them to handle a larger number of external contributions while maintaining high code quality and fast-paced development. The improved processes have set a strong foundation for future growth and success in the open-source community.

PULL REQUESTS

Q] WHAT IS A PULL REQUEST ON GITHUB?

A pull request on GitHub is a feature that allows developers to propose changes to a codebase within a repository. When a developer creates a pull request, they can compare their changes (often from a separate branch) against the main branch, enabling others to review the modifications before merging. This process facilitates collaboration, as team members can discuss, comment on, and suggest improvements to the proposed changes. Pull requests also serve as a record of discussions and decisions made during the development process, enhancing transparency and accountability. Once reviewed and approved, the changes can be merged into the main codebase.

EXAMPLE



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. Learn more about diff comparisons here.

base repository: HarshSharmalN/Python-Projects · base: main ← head repository: vidhirohira/Python-Projects · compare: Vidhi

✓ Able to merge. These branches can be automatically merged.

Add a title

Fruit_Game

Add a description

```
controlling.
There are fruits of black colour (square) which would be spawning randomly. The task is to make sure that the player can collect
the fruits.
You have to use the right arrow key to move right and left arrow key to move left.
For every time you catch the fruit yyou get +2 points and everytime you miss the fruit you get-1 point.
The cummulative score will be printed on  your game window.
Hope you enjoy the game mates !!!


play_fruit_game.py is the python file to run to play the game
fruit-game.mp4 is the video that shows the working game
readme.md file consists of a description of the game and how to play it
```

It looks like this is your first time opening a pull request in this project!

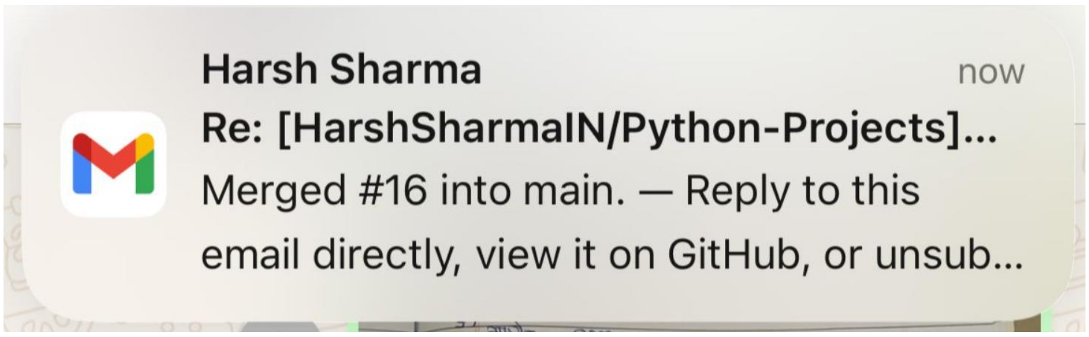
Be sure to review the contributing guidelines.

Remember, contributions to this repository should follow its contributing guidelines.

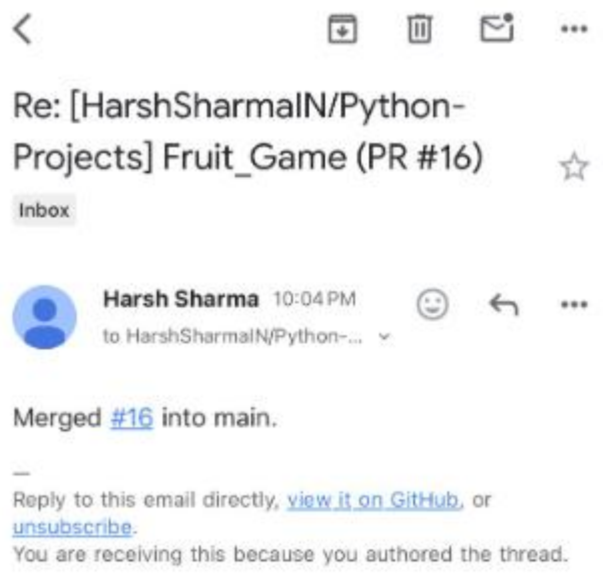
Write
Preview
Markdown is supported
Paste, drop, or click to add files
Allow edits by maintainers
Create pull request

I added a file in a repository of pythonprojects. I pushed a code file of a simple game in python. The above image shows the description that I wrote while creating the pull request.

Soon my pull request was merged. Here are some evidences.



Harsh Sharma now
Re: [HarshSharmaIN/Python-Projects]...
Merged #16 into main. — Reply to this email directly, view it on GitHub, or unsub...



Re: [HarshSharmaIN/Python-Projects] Fruit_Game (PR #16)

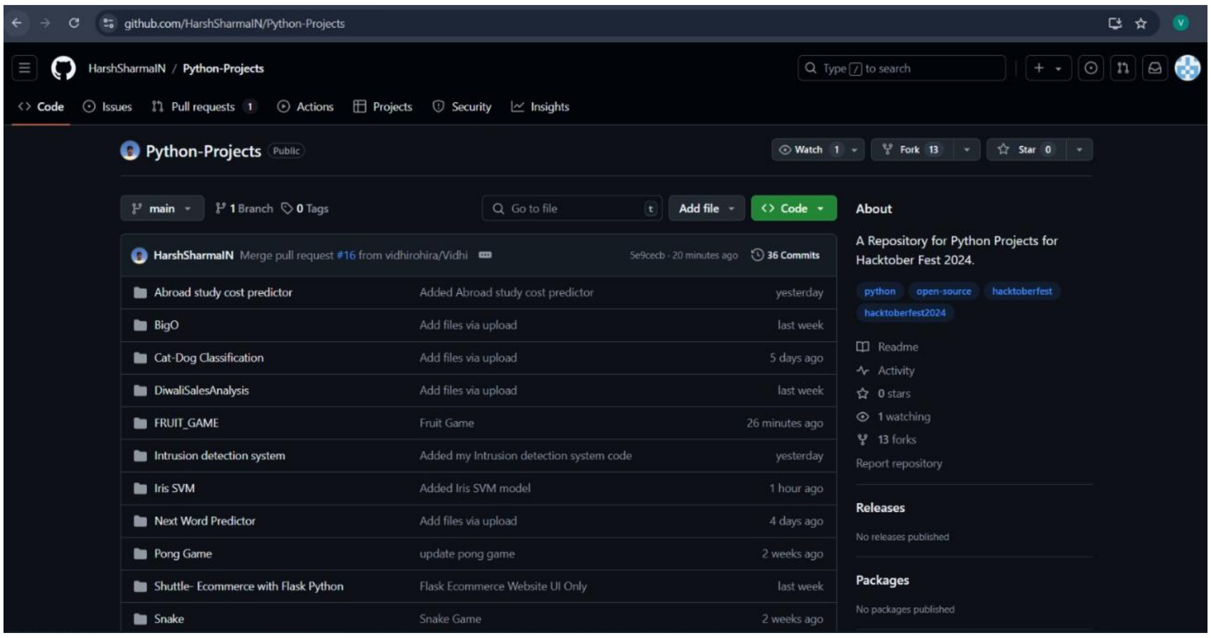
Inbox

Harsh Sharma 10:04 PM
to HarshSharmaIN/Python-...

Merged #16 into main.

—
Reply to this email directly, view it on GitHub, or unsubscribe.
You are receiving this because you authored the thread.

Mail to verify that my pull request has been merged.



github.com/HarshSharmaIN/Python-Projects

HarshSharmaIN / Python-Projects

Python-Projects Public

main · 1 Branch · 0 Tags

HarshSharmaIN Merge pull request #16 from vidhirohira/Vidhi · 5e9cecb · 20 minutes ago · 36 Commits

Abroad study cost predictor	Added Abroad study cost predictor	yesterday
BigO	Add files via upload	last week
Cat-Dog Classification	Add files via upload	5 days ago
DiwaliSalesAnalysis	Add files via upload	last week
FRUIT_GAME	Fruit Game	26 minutes ago
Intrusion detection system	Added my Intrusion detection system code	yesterday
Iris SVM	Added Iris SVM model	1 hour ago
Next Word Predictor	Add files via upload	4 days ago
Pong Game	update pong game	2 weeks ago
Shuttle- Ecommerce with Flask Python	Flask Ecommerce Website UI Only	last week
Snake	Snake Game	2 weeks ago

About

A Repository for Python Projects for Hacktober Fest 2024.

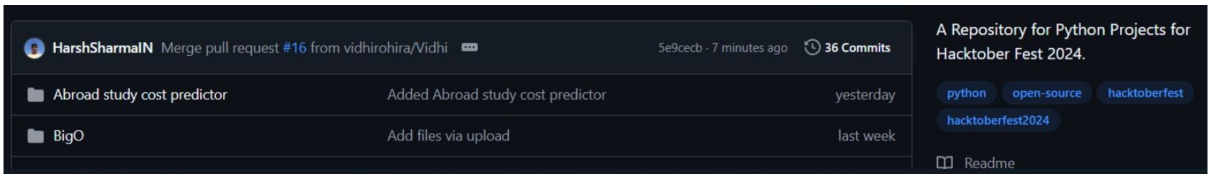
python · open-source · hacktoberfest · hacktoberfest2024

Readme · Activity · 0 stars · 1 watching · 13 forks · Report repository

Releases: No releases published

Packages: No packages published

Pull request merged.



HarshSharmaIN Merge pull request #16 from vidhirohira/Vidhi 5e9cecb · 7 minutes ago 36 Commits

Abroad study cost predictor	Added Abroad study cost predictor	yesterday
BigO	Add files via upload	last week

A Repository for Python Projects for Hacktober Fest 2024.

python open-source hacktoberfest hacktoberfest2024

Readme

Pull request merged.

GitHub repo link:-

- https://github.com/HarshSharmaIN/Python-Projects link of repo
- https://github.com/HarshSharmaIN/Python-Projects/tree/main/FRUIT_GAME

link of my folder in repo

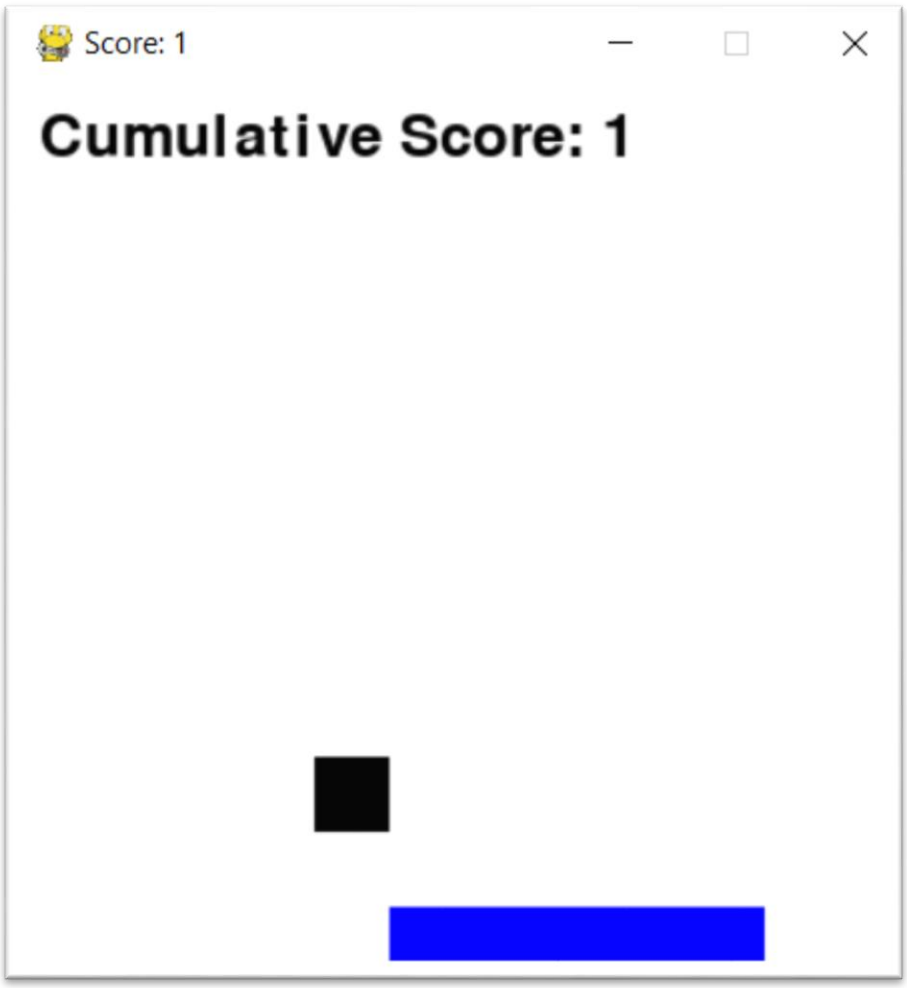
CODE:-

```
'''
Hello Mates!!
This is a fruit game that you can enjoy and play. This game consists of a blue
player (rectangle shape) which you would be controlling.
There are fruits of black colour (square) which would be spawning randomly.
The task is to make sure that the player can collect the fruits.
You have to use the right arrow key to move right and left arrow key to move
left.
For every time you catch the fruit yyou get +2 points and everytime you miss
the fruit you get-1 point.
The cummulative score will be printed on  your game window.
Hope you enjoy the game mates !!!
'''






import pygame
import numpy as np

# --- Game Environment ---
class Field:
    def __init__(self, height=10, width=5):
        self.width = width
        self.height = height
        self.clear_field()

    def clear_field(self):
        self.body = np.zeros(shape=(self.height, self.width))

    def update_field(self, fruits, player):
        self.clear_field()
        for fruit in fruits:
            if not fruit.out_of_field:
                for y in range(fruit.y, min(fruit.y + fruit.height,
self.height)):
                    for x in range(fruit.x, min(fruit.x + fruit.width,
self.width)):
                        self.body[y][x] = 1
        for i in range(player.width):
            self.body[player.y][player.x + i] = 2

class Fruit:
    def __init__(self, height=1, width=1, x=None, y=0, speed=1, field=None):
        self.field = field
        self.height = height
        self.width = width
        self.x = self.generate_x() if x is None else x
        self.y = y
        self.speed = speed
        self.out_of_field = False
        self.is_caught = 0

    def generate_x(self):
        return np.random.randint(0, self.field.width - self.width)

    def set_out_of_field(self):
        self.out_of_field = self.y > self.field.height - 1

    def move(self):
        self.y += self.speed
        self.set_out_of_field()

    def set_is_caught(self, player):
        if self.y != player.y:
            self.is_caught = 0
        else:
            if self.x + self.width > player.x and self.x < player.x +
player.width:
                self.is_caught = 1
            else:
                self.is_caught = -1

class Player:
    def __init__(self, height=1, width=1, field=None):
        self.field = field
        self.height = height
        self.width = 5
        self.x = int(self.field.width / 2 - width / 2)
        self.last_x = self.x
        self.y = self.field.height - 1
        self.dir = 0
        self.colour = "blue"

    def move(self):
        self.last_x = self.x
        self.x += self.dir
        self.dir = 0
        self.constrain()

    def action(self, action):
        if action == 1:
            self.dir = -1
        elif action == 2:
            self.dir = 1
        else:
            self.dir = 0

    def constrain(self):
        if self.x < 0:
            self.x = self.field.width - self.width
        elif (self.x + self.width) > self.field.width:
            self.x = 0

class Environment:
    F_HEIGHT = 12
    F_WIDTH = 12
    PLAYER_WIDTH = 2
    FRUIT_WIDTH = 1

    ENVIRONMENT_SHAPE = (F_HEIGHT, F_WIDTH, 1)
    ACTION_SPACE = [0, 1, 2]
    ACTION_SPACE_SIZE = len(ACTION_SPACE)
    ACTION_SHAPE = (ACTION_SPACE_SIZE,)
    PUNISHMENT = -1  # Penalty for missing fruit
    REWARD = 2       # Reward for catching fruit
    score = 0
    MAX_VAL = 2

    LOSS_SCORE = -5
    WIN_SCORE = 5

    DRAW_MUL = 30
    WINDOW_HEIGHT = F_HEIGHT * DRAW_MUL
    WINDOW_WIDTH = F_WIDTH * DRAW_MUL

    game_tick = 0
    FPS = 12
    MOVE_FRUIT_EVERY = 1
    MOVE_PLAYER_EVERY = 1
    MAX_FRUIT = 1
    INCREASE_MAX_FRUIT_EVERY = 100
    SPAWN_FRUIT_EVERY_MIN = 60
    SPAWN_FRUIT_EVERY_MAX = 61
    next_spawn_tick = 0

    FRUIT_COLOURS = {-1: "red", 0: "black", 1: "green"}

    def __init__(self):
        self.reset()
    def get_state(self):
        return self.field.body / self.MAX_VAL

    def reset(self):
        self.game_tick = 0
        self.game_over = False
        self.game_won = False
        self.field = Field(height=self.F_HEIGHT, width=self.F_WIDTH)
        self.player = Player(field=self.field, width=self.PLAYER_WIDTH)
        self.score = 0
        self.fruits = []
        self.spawn_fruit()
        self.field.update_field(self.fruits, self.player)
        return self.get_state()

    def spawn_fruit(self):
        if len(self.fruits) < self.MAX_FRUIT:
            self.fruits.append(Fruit(field=self.field,
height=self.FRUIT_WIDTH, width=self.FRUIT_WIDTH))
            self.set_next_spawn_tick()

    def set_next_spawn_tick(self):
        self.next_spawn_tick = self.game_tick +
np.random.randint(self.SPAWN_FRUIT_EVERY_MIN, self.SPAWN_FRUIT_EVERY_MAX)

    def step(self, action=None):
        self.game_tick += 1
        if self.game_tick % self.INCREASE_MAX_FRUIT_EVERY == 0:
            self.MAX_FRUIT += 1
        if self.game_tick >= self.next_spawn_tick or len(self.fruits) == 0:
            self.spawn_fruit()
        if action is not None:
            self.player.action(action)
        self.player.move()

        reward = 0
        if self.game_tick % self.MOVE_FRUIT_EVERY == 0:
            in_field_fruits = []
            for fruit in self.fruits:
                fruit.move()
                fruit.set_is_caught(self.player)
                if fruit.is_caught == 1:
                    self.update_score(self.REWARD)
                    reward = self.REWARD
                elif fruit.is_caught == -1:
                    self.update_score(self.PUNISHMENT)
                    reward = self.PUNISHMENT
                if not fruit.out_of_field:
                    in_field_fruits.append(fruit)
            self.fruits = in_field_fruits

        self.field.update_field(fruits=self.fruits, player=self.player)

        if self.score <= self.LOSS_SCORE:
            self.game_over = True
        if self.score >= self.WIN_SCORE:
            self.game_won = True

        return self.get_state(), reward, self.game_over or self.game_won,
self.score

    def update_score(self, delta):
        self.score += delta

    def render(self, screen, cumulative_score, solo=True, x_offset=0,
y_offset=0):
        if solo:
            screen.fill((255, 255, 255))  # White background
            pygame.display.set_caption(f"Score: {self.score}")

        # Draw player
        pygame.draw.rect(
            screen,
            pygame.Color(self.player.colour),
            pygame.Rect(self.player.x * self.DRAW_MUL + x_offset,
self.player.y * self.DRAW_MUL + y_offset, self.player.width * self.DRAW_MUL,
self.player.height * self.DRAW_MUL)
        )

        # Draw fruits
        for fruit in self.fruits:
            pygame.draw.rect(
                screen,
                pygame.Color(self.FRUIT_COLOURS[fruit.is_caught]),
                pygame.Rect(fruit.x * self.DRAW_MUL + x_offset, fruit.y *
self.DRAW_MUL + y_offset, fruit.width * self.DRAW_MUL, fruit.height *
self.DRAW_MUL)
            )

        # Display cumulative score
        font = pygame.font.Font(None, 36)
        score_text = font.render(f'Cumulative Score: {cumulative_score}',
True, (0, 0, 0))
        screen.blit(score_text, (10, 10))

        pygame.display.flip()
```

```
# --- Main Loop with Manual Play ---
def main():
    pygame.init()
    screen = pygame.display.set_mode((Environment.WINDOW_WIDTH,
Environment.WINDOW_HEIGHT))
    clock = pygame.time.Clock()
    env = Environment()

    cumulative_score = 0  # Initialize cumulative score tracker
    running = True
    episode = 0

    while running:
        state = env.reset()
        done = False
        episode_score = 0  # Track score for each episode

        while not done:
            action = 0
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    running = False
                    done = True
                elif event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_LEFT:
                        action = 1  # Move left
                    elif event.key == pygame.K_RIGHT:
                        action = 2  # Move right
                elif event.type == pygame.KEYUP:
                    action = 0  # Stop moving when key is released

            next_state, reward, done, score = env.step(action)
            episode_score += reward  # Update episode score

            # Update cumulative score after each step
            cumulative_score += reward

            env.render(screen, cumulative_score)

            clock.tick(env.FPS)

        # Print scores
        episode += 1
        print(f"\nEpisode {episode} Finished. Episode Score: {episode_score}.
Cumulative Score: {cumulative_score}")

    pygame.quit()
```

```python
if __name__ == "__main__":
    main()
```

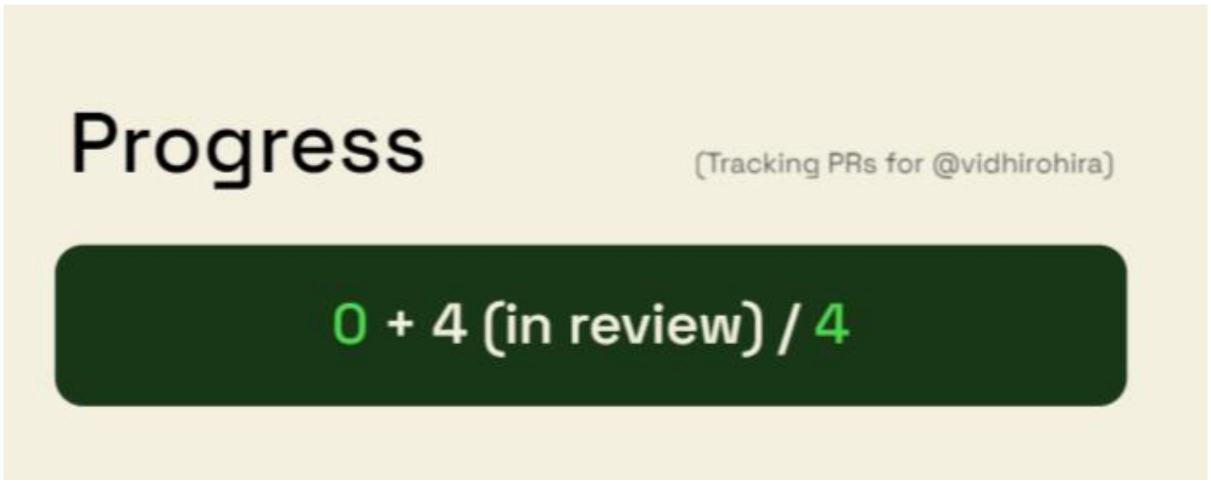
OUTPUT:-



Score: 1

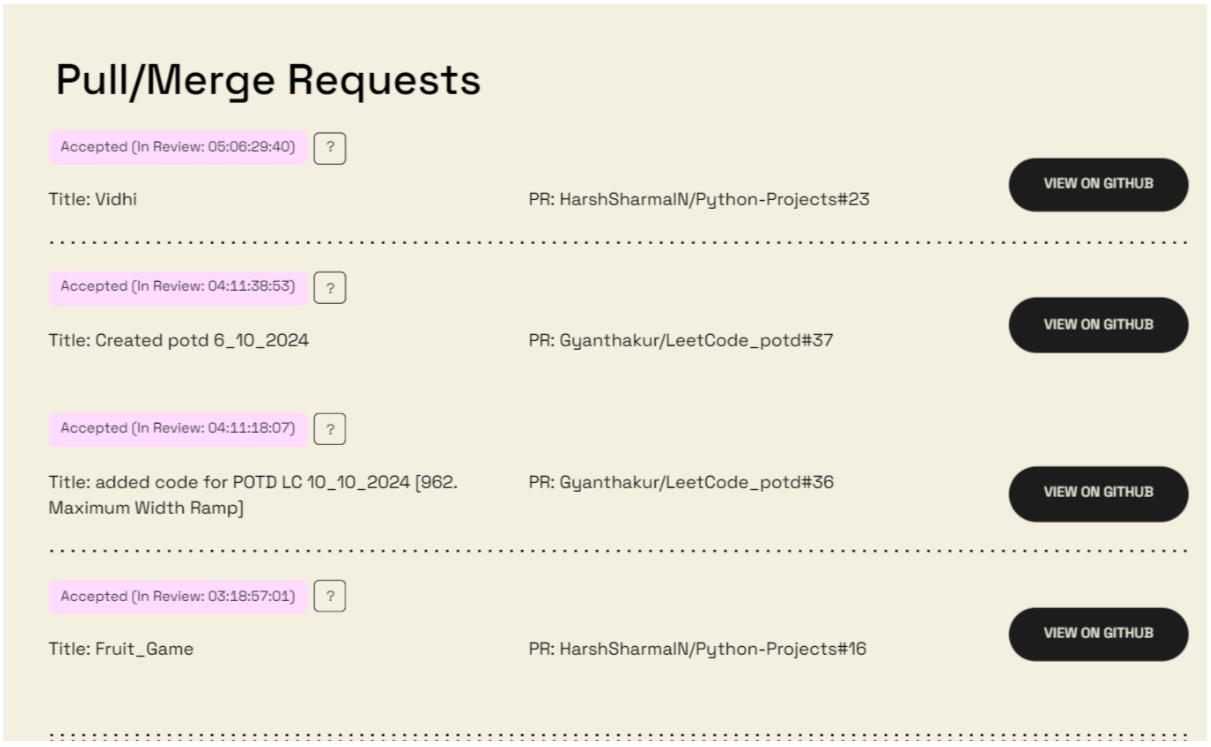
Cumulative Score: 1

HACKTOBER PROGRESS


Progress

(Tracking PRs for @vidhirohira)

0 + 4 (in review) / 4


Pull/Merge Requests

Accepted (In Review: 05:06:29:40) ?

Title: Vidhi PR: HarshSharmalN/Python-Projects#23 VIEW ON GITHUB

Accepted (In Review: 04:11:38:53) ?

Title: Created potd 6_10_2024 PR: Gyanthakur/LeetCode_potd#37 VIEW ON GITHUB

Accepted (In Review: 04:11:18:07) ?

Title: added code for POTD LC 10_10_2024 [962. Maximum Width Ramp] PR: Gyanthakur/LeetCode_potd#36 VIEW ON GITHUB

Accepted (In Review: 03:18:57:01) ?

Title: Fruit_Game PR: HarshSharmalN/Python-Projects#16 VIEW ON GITHUB

CONCLUSION:- Hence in this lab we have studied a successful open source case study and pulled a pull request on GitHub which got merged and shown the code and screenshots too.