Cell 1: Simple Explanations and Basic Visualization Setup

This cell defines a `show_progress_bar` function that creates a simple progress bar visualization in the console.

- **Inputs**: Current progress, total steps, and a description.
- **Output**: A progress bar visualization.

```
import random
import time
import matplotlib.pyplot as plt
from IPython.display import clear_output

def show_progress_bar(progress, total, description=""):
    """Make a simple visual progress bar"""
    bar_length = 20
    filled = int(bar_length * progress / total)
    bar = "🟦" * filled + "⬜" * (bar_length - filled)
    print(f"{description} [{bar}] {progress}/{total}")
```

Cell 2: Learning Like a Child (DQN Explained)

This cell introduces the concept of reinforcement learning through a toy analogy.

- A `ToyLearner` class mimics how a child learns from interactions with their environment.
- **Methods**:
    - `try_action`: Assigns and retrieves scores for situations and actions.
    - `learn_from_result`: Updates the scores based on feedback.
    - `show_learning_progress`: Plots the happiness level over time.

```
import random
import time
import matplotlib.pyplot as plt
from IPython.display import clear_output
```

Explain code NameError •••

You

Please explain the error:

```
NameError: name 'random' is not
```

Gemini

Hypothesis

The error message `NameError: name 'random' is not defined` indicates that the `random` module has not been imported or is not accessible in the current scope. While the user has included `import random` in the first cell, it's possible that the kernel or the way the cells are being executed might be causing the `random` module to not be availa the scope where it's being used.

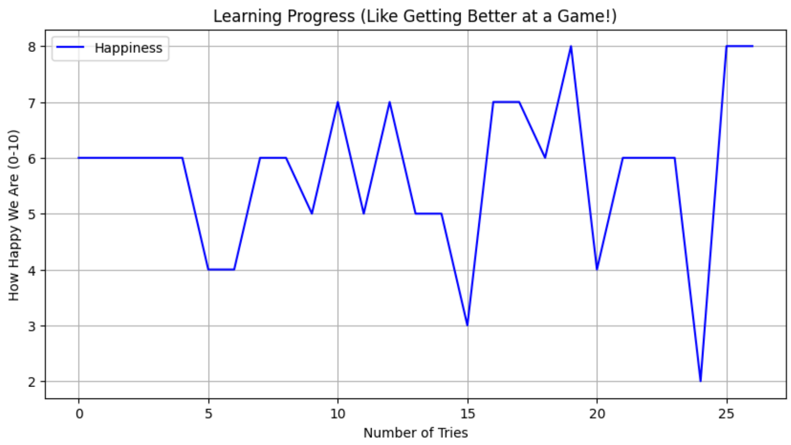
Suggested Changes

```
import random
import time
import matplotlib.pyplot as pl
from IPython.display import cl

def show_progress_bar(progress
    """Make a simple visu
    bar_length = 20
    filled = int(bar_len
    bar = "🟦" * filled +
    print(f"{description} [{ba

class ToyLearner:
    """
    Imagine a child learning t
    - The child tries differen
```

```
def show_progress_bar(progress, total, description=""):
    """Make a simple visual progress bar"""
    bar_length = 20
    filled = int(bar_length * progress / total)
    bar = "🟦" * filled + "⬜" * (bar_length - filled)
    print(f"{description} [{bar}] {progress}/{total}")

class ToyLearner:
    """
    Imagine a child learning to play with toys!
    - The child tries different actions
    - Remembers what worked well
    - Gets better over time
    """
    def __init__(self):
        # This is like the child's memory
        self.memory = {}
        self.happiness = []  # Track how happy the child i

    def try_action(self, situation, action):
        """
        Just like a child trying something new:
        - What's the situation? (Like "toy on table")
        - What to do? (Like "reach for toy")
        """
        memory_key = f"{situation}_{action}"

        # If we've never tried this before, guess it's oka
        if memory_key not in self.memory:
            self.memory[memory_key] = 5  # Score out of 10

        return self.memory[memory_key]

    def learn_from_result(self, situation, action, was_it_
        """
        Learning from what happened:
        - If it worked well, remember to do it more
        - If it didn't work, remember to try something els
        """
        memory_key = f"{situation}_{action}"
        old_score = self.memory.get(memory_key, 5)

        # Adjust our memory based on what happened
        if was_it_good:
            new_score = old_score + 1  # That was good!
        else:
            new_score = old_score - 1  # Oops, not so good

        # Keep scores between 0 and 10
        self.memory[memory_key] = max(0, min(10, new_score
```

```
- Remembers what worked we
- Gets better over time
"""
def __init__(self):
    # This is like the chi
    self.memory = {}
    self.happiness = []  #

def try_action(self, situa
    """
    Just like a child tryi
    - What's the situation
    - What to do? (Like "r
    """
    memory_key = f"{situat

    # If we've never tried
    if memory_key not in s
        self.memory[memory

    return self.memory[mem

def learn_from_result(sel
    """
    Learning from what hap
    - If it worked well, r
    - If it didn't work, r
    """
    memory_key = f"{situat
    old_score = self.memor

    # Adjust our memory ba
    if was_it_good:
        new_score = ol
    else:
        new_score = (

    # Keep scores between
    self.memory[memory_key
    self.happiness.append(

def show_learning_progress
    """Show how we're lear
```

```
        self.happiness.append(new_score)

    def show_learning_progress(self):
        """Show how we're learning over time with pictures
        plt.figure(figsize=(10, 5))
        plt.plot(self.happiness, color='blue', label='Happ
        plt.title('Learning Progress (Like Getting Better 
        plt.xlabel('Number of Tries')
        plt.ylabel('How Happy We Are (0-10)')
        plt.legend()
        plt.grid(True)
        plt.show()

learner = ToyLearner()

# Example learning process
situations = ["toy_on_table", "toy_far_away", "toy_in_box'
actions = ["reach", "ask_help", "jump"]
for day in range(3):  # Shortened loop for demonstration
    for situation in situations:
        for action in actions:
            # The random module is already imported at the
            result = random.choice([True, False])
            learner.learn_from_result(situation, action, r
learner.show_learning_progress()
```



```
        plt.figure(figsize=(10
        plt.plot(self.happines
        plt.title('Learning Pr
        plt.xlabel('Number of 
        plt.ylabel('How Happy 
        plt.legend()
        plt.grid(True)
        plt.show()

learner = ToyLearner()

# Example learning process
situations = ["toy_on_table",
actions = ["reach", "ask_help"
for day in range(3):  # Shorte
    for situation in situation
        for action in actions:
            # The random modul
            result = random.ch
            learner.learn_from_
learner.show_learning_progress
```

Use code with caution

Explanation of Changes:

1. **Combined Cells**: I have merged the content of the first two cells into a single cell. This ensures that the `import random` statement is in the same scope as the code that uses the `random` module, specifically the `random.choice` function. should resolve the `Name`

▶ Sources

New

Rate this answer

Enter a prompt here

0 / 2000

Responses may display inaccurate or offensive information that doesn't represent Google's views. Learn more

New