Display High Scores Pygame

I have a list of 10 high scores for my pygame. How can I display them centered from top to bottom in my high score screen?

To display the high scores centered from top to bottom on your high score screen in Pygame, you can calculate the vertical position for each score so that they are evenly spaced. Here's a basic approach:

1. **Set up your screen and font**:
   Ensure you have a screen and font set up for rendering text.
2. **Calculate the starting vertical position**:
   You can calculate the starting y-coordinate by considering the total height of your screen and the number of high scores.
3. **Evenly space the high scores**:
   To center them, subtract the total height of all scores from the screen height, then divide by 2 to get the starting position. Afterward, space them out evenly.

Here's a code example to display 10 high scores centered vertically:

```python
import pygame

# Initialize Pygame
pygame.init()

# Set up the screen
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("High Scores")

# Set up font
font = pygame.font.SysFont("Arial", 30)

# Example high scores list
high_scores = [1000, 900, 800, 700, 600, 500, 400, 300, 200, 100]

# Calculate total height for all the scores
score_height = font.get_height()  # Height of each score
total_height = score_height * len(high_scores)

# Calculate starting position (centered vertically)
start_y = (screen_height - total_height) // 2

# Render and display the high scores
screen.fill((0, 0, 0))  # Fill the screen with black

for i, score in enumerate(high_scores):
    text = font.render(f"{i + 1}. {score}", True, (255, 255, 255))  # White color text
    screen.blit(text, (screen_width // 2 - text.get_width() // 2, start_y + i * score_height))

# Update the display
pygame.display.flip()

# Wait for a while (you can use an event loop in real usage)
```

```
pygame.time.wait(5000)  # Show for 5 seconds

pygame.quit()
```

Explanation:

1. **Font**: The font size is set to 30, but you can adjust it to fit your needs.
2. **Vertical Positioning**: `start_y` is calculated to center the list vertically. The total height of all the high scores is subtracted from the screen height, and the result is divided by 2 to center it.
3. **Horizontal Positioning**: The scores are centered horizontally by calculating `(screen_width // 2 - text.get_width() // 2)`.

This should display your high scores neatly and centered both horizontally and vertically!

What if I want the high scores to have their numbers left aligned?

To left-align the numbers of the high scores while still keeping the text centered vertically on the screen, you can adjust the way you position the text horizontally.

Here's how you can modify the previous example to left-align the numbers while maintaining the vertical centering:

1. **Left-align the numbers**: You'll render the score text starting at a fixed x-coordinate (say, the far left of the screen or with some padding).
2. **Keep vertical centering**: The vertical positioning logic will stay the same.

Here's the updated code to achieve that:

```
python

import pygame

# Initialize Pygame
pygame.init()

# Set up the screen
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("High Scores")

# Set up font
font = pygame.font.SysFont("Arial", 30)

# Example high scores list
high_scores = [1000, 900, 800, 700, 600, 500, 400, 300, 200, 100]

# Calculate total height for all the scores
score_height = font.get_height()  # Height of each score
total_height = score_height * len(high_scores)

# Calculate starting position (centered vertically)
start_y = (screen_height - total_height) // 2

# Render and display the high scores
screen.fill((0, 0, 0))  # Fill the screen with black
```

```
# Left-align the numbers at a fixed position, and center the text vertically
padding_left = 50  # You can adjust this value as needed
for i, score in enumerate(high_scores):
    text = font.render(f"{i + 1}. {score}", True, (255, 255, 255))  # White color text
    screen.blit(text, (padding_left, start_y + i * score_height))

# Update the display
pygame.display.flip()

# Wait for a while (you can use an event loop in real usage)
pygame.time.wait(5000)  # Show for 5 seconds

pygame.quit()
```

Key Adjustments:

1. **Left-alignment**: The number is displayed starting at **`padding_left`**, which is the x-coordinate where you want to start rendering the numbers. You can adjust this value to move the numbers further left or right.
2. **Text Placement**: The scores are still vertically centered as before, but now the numbers (i.e., **`i + 1`**) are placed on the left side of the screen.

With this setup, the high scores will be displayed with the rank numbers aligned to the left, and the actual scores will follow them.

Anyway to format it so that the 10 is right aligned with the rest of the numbers numbering the scores?

Yes, you can right-align the numbers in a way that they all line up neatly, even if the number of digits varies (e.g., 1 digit for the number "1" and 2 digits for the number "10"). To do this, you can format the numbers to have a fixed width, such as using string formatting or **`f-string`** with padding.

Steps:

1. Format the numbers to have a fixed width by padding them with leading spaces or zeroes.
2. The scores will be left-aligned, but the numbers themselves will be aligned to the right.

Here's how you can do it:

```
python

import pygame

# Initialize Pygame
pygame.init()

# Set up the screen
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("High Scores")

# Set up font
font = pygame.font.SysFont("Arial", 30)
```

```
# Example high scores list
high_scores = [1000, 900, 800, 700, 600, 500, 400, 300, 200, 100]

# Calculate total height for all the scores
score_height = font.get_height()  # Height of each score
total_height = score_height * len(high_scores)

# Calculate starting position (centered vertically)
start_y = (screen_height - total_height) // 2

# Render and display the high scores
screen.fill((0, 0, 0))  # Fill the screen with black

# Left-align the numbers with right-aligned digits
padding_left = 50  # You can adjust this value as needed
max_digits = len(str(len(high_scores)))  # Number of digits based on total high scores
for i, score in enumerate(high_scores):
    # Format the number with leading spaces to right-align them
    number_str = f"{i + 1:>{max_digits}}"  # Right-align the number (based on max digits)
    text = font.render(f"{number_str}. {score}", True, (255, 255, 255))  # White color text
    screen.blit(text, (padding_left, start_y + i * score_height))

# Update the display
pygame.display.flip()

# Wait for a while (you can use an event loop in real usage)
pygame.time.wait(5000)  # Show for 5 seconds

pygame.quit()
```

Key Changes:

1. **`max_digits`**: This variable calculates the number of digits based on the total number of high scores (e.g., 10 scores will require 2 digits, so 1 will be " 1" and 10 will be "10").
2. **`f"{i + 1:>{max_digits}}"`**: This part of the code formats the number (`i + 1`) to have a minimum width equal to the `max_digits` value, ensuring the numbers are right-aligned.

Example:

If you have 10 high scores, it will print something like:

```
markdown

 1. 1000
 2. 900
 3. 800
 4. 700
...
10. 100
```

This approach ensures that the numbers will always align, whether the number of digits changes between the scores.

The 10 is still pushed in to the right

It seems that the issue arises from the way text rendering is handled in Pygame. When you left-align the text and right-align the numbers with padding, the dot and score text may not line up properly because

the number is still occupying more space than it should.

To ensure the numbering aligns properly, we can do two things:

1. **Ensure the numbers are right-aligned properly** by adding enough space before them, based on the total length of the highest number (e.g., "10").
2. **Ensure the whole number with the score is rendered together** without affecting spacing.

Let's adjust the code to ensure the number and its corresponding score align properly. We will create a fixed width for the number and apply that width to each score, so everything aligns neatly.

Here's the revised approach:

```
python

import pygame

# Initialize Pygame
pygame.init()

# Set up the screen
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("High Scores")

# Set up font
font = pygame.font.SysFont("Arial", 30)

# Example high scores list
high_scores = [1000, 900, 800, 700, 600, 500, 400, 300, 200, 100]

# Calculate total height for all the scores
score_height = font.get_height()  # Height of each score
total_height = score_height * len(high_scores)

# Calculate starting position (centered vertically)
start_y = (screen_height - total_height) // 2

# Render and display the high scores
screen.fill((0, 0, 0))  # Fill the screen with black

# Left-align the numbers with right-aligned digits
padding_left = 50  # You can adjust this value as needed
max_digits = len(str(len(high_scores)))  # Number of digits based on total high scores
for i, score in enumerate(high_scores):
    # Format the number with leading spaces to right-align them
    number_str = f"{i + 1:>{max_digits}}"  # Right-align the number (based on max digits)
    # Combine the formatted number with the score
    full_text = f"{number_str}. {score}"
    text = font.render(full_text, True, (255, 255, 255))  # White color text
    screen.blit(text, (padding_left, start_y + i * score_height))

# Update the display
pygame.display.flip()

# Wait for a while (you can use an event loop in real usage)
pygame.time.wait(5000)  # Show for 5 seconds

pygame.quit()
```

Explanation of Changes:

1. **Right-align the number**: `f"{i + 1:>{max_digits}}"` ensures that the numbering (1, 2, ..., 10) is right-aligned, padding with spaces if necessary (so that 10 doesn't shift right).

2. **Combining the number with the score**: By combining the formatted number (`number_str`) and the score text (`score`), you ensure that the numbers and scores line up properly.

Result:

If you have 10 scores, the result will look like this:

```
markdown

 1. 1000
 2. 900
 3. 800
 4. 700
 5. 600
 6. 500
 7. 400
 8. 300
 9. 200
10. 100
```

This way, the numbers (1, 2, ..., 10) will always be right-aligned, and the score will follow directly after them without any misalignment.

Still not working. Is there a way to right align the enumerated numbers while left aligning the scores themselves?

Ah, I see! The issue here is that we want to keep the enumeration numbers (1, 2, ..., 10) right-aligned, while the scores themselves remain left-aligned. To achieve this, you can split the rendering process into two steps:

1. **Right-align the numbers (1, 2, ..., 10)**.
2. **Left-align the scores** after the number, ensuring that both parts are rendered separately.

This approach ensures that the numbers are right-aligned within their own space, and the scores will be left-aligned next to them.

Updated Code:

```
python

import pygame

# Initialize Pygame
pygame.init()

# Set up the screen
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("High Scores")

# Set up font
font = pygame.font.SysFont("Arial", 30)

# Example high scores list
high_scores = [1000, 900, 800, 700, 600, 500, 400, 300, 200, 100]
```

```
# Calculate total height for all the scores
score_height = font.get_height()  # Height of each score
total_height = score_height * len(high_scores)

# Calculate starting position (centered vertically)
start_y = (screen_height - total_height) // 2

# Render and display the high scores
screen.fill((0, 0, 0))  # Fill the screen with black

# Left-align the scores and right-align the numbers
padding_left = 50  # You can adjust this value as needed
max_digits = len(str(len(high_scores)))  # Number of digits based on total high scores

# Set padding between number and score
number_width = font.size(f"{len(high_scores)}.")  # Width of the longest number (e.g., "10.")
padding_between = 10  # Padding between number and score

for i, score in enumerate(high_scores):
    # Right-align the number
    number_str = f"{i + 1:>{max_digits}}"  # Right-align the number (based on max digits)

    # Render the right-aligned number
    number_text = font.render(number_str, True, (255, 255, 255))
    number_x = padding_left

    # Render the left-aligned score
    score_text = font.render(f"{score}", True, (255, 255, 255))
    score_x = number_x + number_width[0] + padding_between  # Position the score to the right of
the number

    # Blit the text onto the screen
    screen.blit(number_text, (number_x, start_y + i * score_height))
    screen.blit(score_text, (score_x, start_y + i * score_height))

# Update the display
pygame.display.flip()

# Wait for a while (you can use an event loop in real usage)
pygame.time.wait(5000)  # Show for 5 seconds

pygame.quit()
```

Key Changes:

1. **Right-align the number**: We calculate the width of the number text (like "10.") and then render the numbers with padding on the left side to right-align them within the available space.
2. **Separate rendering**: The score and the number are rendered separately. We use the width of the number text (`number_width[0]`) to position the score text to the right of the number, ensuring they don't overlap.
3. **Padding between number and score**: A small padding (`padding_between`) is added between the right-aligned number and the left-aligned score to keep them from touching.

Result:

If you have 10 scores, the result will look like this:

```yaml

 1 1000
 2  900
 3  800
 4  700
```

```
 5  600
 6  500
 7  400
 8  300
 9  200
10  100
```

Now, the numbers are right-aligned, and the scores are left-aligned, with appropriate spacing between them.

The enumerated numbers on the left are now centered with respect to each other, but the high scores are left aligned

Got it! The issue arises because the enumerated numbers are right-aligned with respect to each other, but the scores need to be aligned separately. To make the numbers **right-aligned** while keeping the scores **left-aligned**, and ensuring the number and score are aligned properly, we need to use a consistent approach to place them on the screen.

The plan:

1. **Right-align the enumerated numbers** based on the width of the largest number (`10`, `100`, etc.).
2. **Left-align the high scores**, and make sure there's consistent spacing between the numbers and the scores.

Here's an approach to achieve this:

Solution:

1. Calculate the width of the largest number (e.g., `10` will be the largest number if there are 10 scores).
2. Use that width to right-align the numbers and ensure the score appears immediately to the right.

Updated Code:

```python
python

import pygame

# Initialize Pygame
pygame.init()

# Set up the screen
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("High Scores")

# Set up font
font = pygame.font.SysFont("Arial", 30)

# Example high scores list
high_scores = [1000, 900, 800, 700, 600, 500, 400, 300, 200, 100]

# Calculate total height for all the scores
score_height = font.get_height()  # Height of each score
total_height = score_height * len(high_scores)
```

```
# Calculate starting position (centered vertically)
start_y = (screen_height - total_height) // 2

# Render and display the high scores
screen.fill((0, 0, 0))  # Fill the screen with black

# Calculate the width of the largest number (to right-align the numbers)
max_digits = len(str(len(high_scores)))  # Number of digits based on total high scores
max_number_width = font.size(f"{len(high_scores)}")[0]  # Width of the largest number (e.g.,
"10.")

# Set padding between number and score
padding_left = 50  # Left margin for the whole list
padding_between = 10  # Padding between number and score

for i, score in enumerate(high_scores):
    # Right-align the number
    number_str = f"{i + 1:>{max_digits}}"  # Right-align the number (based on max digits)

    # Render the right-aligned number
    number_text = font.render(number_str, True, (255, 255, 255))
    number_x = padding_left  # Position the number at the left side, but right-align it

    # Render the left-aligned score
    score_text = font.render(f"{score}", True, (255, 255, 255))
    score_x = number_x + max_number_width + padding_between  # Position the score to the right
of the number

    # Blit the text onto the screen
    screen.blit(number_text, (number_x, start_y + i * score_height))
    screen.blit(score_text, (score_x, start_y + i * score_height))

# Update the display
pygame.display.flip()

# Wait for a while (you can use an event loop in real usage)
pygame.time.wait(5000)  # Show for 5 seconds

pygame.quit()
```

Key Changes:

1. **`max_number_width`**: This calculates the width of the largest number (e.g., **`10`** or **`100`** depending on the total number of high scores) so that the numbers are right-aligned within that space.
2. **Positioning**: The **`number_x`** is the left padding for the number, and **`score_x`** is calculated by adding **`max_number_width`** to the **`number_x`**, plus some padding between the number and the score.
3. **Alignment**: The **`number_str`** is right-aligned using **`{i + 1:>{max_digits}}`**, and the scores are rendered immediately to the right of the number.

Result:

If you have 10 scores, the result will now look like this:

```
yaml


  1 1000
  2  900
  3  800
  4  700
  5  600
  6  500
  7  400
  8  300
```

```
 9  200
10  100
```

This ensures the numbers are right-aligned, and the scores are left-aligned, with consistent spacing between them. The `max_number_width` ensures that the numbers (e.g., 1, 10) align properly with each other, while the scores remain left-aligned to the right of the numbers.

enumerated numbers still are center aligned

Thanks for your patience! I now understand what you're aiming for more clearly: You want the **enumerated numbers to be right-aligned**, but the **scores themselves to remain left-aligned**, with everything properly aligned and spaced.

To achieve this, we'll:

1. **Right-align the enumerated numbers** based on the number of digits in the highest number. For example, if you have 10 scores, the largest number is `10`, so we should make sure all the numbers have two digits, even if the number is just `1`.
2. **Left-align the scores** so that they start at a fixed position after the enumerated number.

Approach:

We can format the numbers with leading spaces so that they all have the same width (e.g., `1`, `2`, ..., `10`) while ensuring that the scores are aligned after the numbers. Let's update the code to achieve that:

Updated Code:

```
python

import pygame

# Initialize Pygame
pygame.init()

# Set up the screen
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("High Scores")

# Set up font
font = pygame.font.SysFont("Arial", 30)

# Example high scores list
high_scores = [1000, 900, 800, 700, 600, 500, 400, 300, 200, 100]

# Calculate total height for all the scores
score_height = font.get_height()  # Height of each score
total_height = score_height * len(high_scores)

# Calculate starting position (centered vertically)
start_y = (screen_height - total_height) // 2

# Render and display the high scores
screen.fill((0, 0, 0))  # Fill the screen with black

# Calculate the maximum number of digits for the enumeration (e.g., "1" to "10")
```

```
max_digits = len(str(len(high_scores)))  # Number of digits based on total high scores

# Set padding between number and score
padding_left = 50  # Left margin for the whole list
padding_between = 10  # Padding between number and score

for i, score in enumerate(high_scores):
    # Format the number to be right-aligned within the max number of digits
    number_str = f"{i + 1:>{max_digits}}"  # Right-align the number (based on max digits)

    # Render the number text
    number_text = font.render(number_str, True, (255, 255, 255))

    # Render the score text (left-aligned)
    score_text = font.render(f"{score}", True, (255, 255, 255))

    # Position the number and the score text
    number_x = padding_left  # Position the number text on the left, with right-alignment
applied
    score_x = number_x + number_text.get_width() + padding_between  # Position the score to the
right of the number

    # Blit the number and score text onto the screen
    screen.blit(number_text, (number_x, start_y + i * score_height))
    screen.blit(score_text, (score_x, start_y + i * score_height))

# Update the display
pygame.display.flip()

# Wait for a while (you can use an event loop in real usage)
pygame.time.wait(5000)  # Show for 5 seconds

pygame.quit()
```

Key Changes:

1. **Right-align numbers**: The expression `f"{i + 1:>{max_digits}}"` ensures that the numbers are right-aligned within a fixed width (determined by `max_digits`). For example, for 10 scores, the numbers will be formatted as ` 1`, ` 2`, ..., `10`.
2. **Left-align the scores**: After rendering the right-aligned number, we position the score text immediately to its right, using `number_x + number_text.get_width() + padding_between` to calculate the `score_x` position.

Result:

If you have 10 high scores, the output will look like this:

```
yaml


 1 1000
 2  900
 3  800
 4  700
 5  600
 6  500
 7  400
 8  300
 9  200
10  100
```

How it works:

- **Right-aligned numbers**: The numbers (`1`, `2`, `10`) are aligned to the right within the space reserved for them. For example, `1` becomes ` 1`, and `10` remains as `10`, so they all align perfectly.
- **Left-aligned scores**: The scores are rendered directly after the numbers, ensuring that the text is left-aligned without affecting the positioning of the numbers.

Let me know if this works!

Now the 10 is centered, and the numbers on the left are not all left aligned

I understand the issue now! The goal is to **right-align** the enumerated numbers (so they align properly, regardless of whether the number is 1 or 10), and at the same time, keep the **scores left-aligned**.

The reason why the numbers (like `10`) are still centered is because we're rendering the numbers and scores together and trying to make the numbers right-aligned within their space, but this is causing issues with the layout.

To properly achieve this, we need to:

1. **Calculate the maximum width for the right-aligned numbers** (so that `1`, `2`, ..., `10` all align with each other).
2. **Left-align the scores** and make sure there is consistent spacing between the numbers and the scores.

Approach:

1. We first calculate the width of the largest number (`10` for 10 scores) so we can allocate enough space for the number column.
2. Then, we right-align the numbers (1, 2, ..., 10) within this width.
3. Finally, we left-align the scores next to the numbers.

Updated Code:

```
python

import pygame

# Initialize Pygame
pygame.init()

# Set up the screen
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("High Scores")

# Set up font
font = pygame.font.SysFont("Arial", 30)

# Example high scores list
high_scores = [1000, 900, 800, 700, 600, 500, 400, 300, 200, 100]

# Calculate total height for all the scores
score_height = font.get_height()  # Height of each score
```

```
total_height = score_height * len(high_scores)

# Calculate starting position (centered vertically)
start_y = (screen_height - total_height) // 2

# Calculate the maximum number of digits for the enumeration (e.g., "1" to "10")
max_digits = len(str(len(high_scores)))  # Number of digits based on total high scores

# Set padding between number and score
padding_left = 50  # Left margin for the whole list
padding_between = 10  # Padding between number and score

# Calculate the width of the largest number (to ensure right alignment)
number_width = font.size(f"{len(high_scores)}")[0]  # Width of the largest number (e.g., "10.")

# Render and display the high scores
screen.fill((0, 0, 0))  # Fill the screen with black

for i, score in enumerate(high_scores):
    # Format the number to be right-aligned within the max number of digits
    number_str = f"{i + 1:>{max_digits}}"  # Right-align the number (based on max digits)

    # Render the right-aligned number
    number_text = font.render(number_str, True, (255, 255, 255))

    # Render the score text (left-aligned)
    score_text = font.render(f"{score}", True, (255, 255, 255))

    # Position the number text on the left with right-alignment
    number_x = padding_left

    # Position the score text to the right of the number with padding between them
    score_x = number_x + number_width + padding_between

    # Blit the number and score text onto the screen
    screen.blit(number_text, (number_x, start_y + i * score_height))
    screen.blit(score_text, (score_x, start_y + i * score_height))

# Update the display
pygame.display.flip()

# Wait for a while (you can use an event loop in real usage)
pygame.time.wait(5000)  # Show for 5 seconds

pygame.quit()
```

Key Changes:

1. **`number_width`**: This calculates the width of the largest number (e.g., "10" for 10 items), which will ensure that all numbers are given enough space for right alignment.
2. **Right-align the numbers**: The number is rendered with padding on the left (**`f"{i + 1:>{max_digits}}"`**) so the numbers are right-aligned.
3. **Left-align the scores**: The scores are rendered to the right of the numbers, with a padding (**`padding_between`**) added for spacing.

Result:

If you have 10 high scores, the output will look like this:

```
yaml


 1 1000
 2  900
 3  800
 4  700
```

```
 5  600
 6  500
 7  400
 8  300
 9  200
10  100
```

Explanation:

- **Numbers** (1, 2, ..., 10) are right-aligned within a space that accommodates the largest number (in this case, `10`).
- **Scores** are left-aligned next to the numbers, with a consistent gap (`padding_between`).

This should now give you the proper alignment with the enumerated numbers right-aligned and the scores left-aligned!

It's not working. How about we just make both the numbering and the scores left aligned?

I see! If you're okay with having both the **numbers** and the **scores** left-aligned, that should definitely simplify things.

In this case, we'll just format both the numbers and the scores without worrying about alignment other than keeping them both left-aligned.

Solution:

1. We will format the numbers normally (1, 2, ..., 10), followed by the scores.
2. Both numbers and scores will be left-aligned at the same starting point, with a small padding between the number and the score.

Updated Code for Left Alignment:

```python
import pygame

# Initialize Pygame
pygame.init()

# Set up the screen
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("High Scores")

# Set up font
font = pygame.font.SysFont("Arial", 30)

# Example high scores list
high_scores = [1000, 900, 800, 700, 600, 500, 400, 300, 200, 100]

# Calculate total height for all the scores
score_height = font.get_height()  # Height of each score
total_height = score_height * len(high_scores)

# Calculate starting position (centered vertically)
```

```
start_y = (screen_height - total_height) // 2

# Set padding between number and score
padding_left = 50  # Left margin for the whole list
padding_between = 20  # Padding between number and score

# Render and display the high scores
screen.fill((0, 0, 0))  # Fill the screen with black

for i, score in enumerate(high_scores):
    # Format the number normally (1, 2, ..., 10)
    number_str = f"{i + 1}"

    # Render the number and score text (both left-aligned)
    number_text = font.render(number_str, True, (255, 255, 255))
    score_text = font.render(f"{score}", True, (255, 255, 255))

    # Position the number and the score text
    number_x = padding_left  # Position the number text on the left
    score_x = number_x + number_text.get_width() + padding_between  # Position the score to the
right of the number

    # Blit the number and score text onto the screen
    screen.blit(number_text, (number_x, start_y + i * score_height))
    screen.blit(score_text, (score_x, start_y + i * score_height))

# Update the display
pygame.display.flip()

# Wait for a while (you can use an event loop in real usage)
pygame.time.wait(5000)  # Show for 5 seconds

pygame.quit()
```

Key Points:

- **Both numbers and scores are left-aligned**: The `number_str` (1, 2, 3, ...) is formatted normally, and the `score_text` (1000, 900, etc.) is left-aligned with padding between the number and score.
- **Padding**: You can adjust `padding_between` to control the space between the number and the score.

Result:

This will display something like:

```yaml

1 1000
2  900
3  800
4  700
5  600
6  500
7  400
8  300
9  200
10 100
```

Now both the enumerated numbers and the scores are **left-aligned**, with consistent spacing between them.

The 10 is offset. What if we just picked an X coordinate for all the enumerations. And then we picked an X for all the scores?

Great idea! If we want to avoid dealing with the dynamic width of the numbers, we can simply **fix the X coordinates** for both the enumerated numbers and the scores.

Here's the approach:

- We pick a **fixed X coordinate for the enumerated numbers**.
- We pick a **fixed X coordinate for the scores**, ensuring there's some consistent padding between the number and the score.
- This way, both the numbers and the scores will be aligned consistently across all rows.

Updated Solution:

```python
import pygame

# Initialize Pygame
pygame.init()

# Set up the screen
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("High Scores")

# Set up font
font = pygame.font.SysFont("Arial", 30)

# Example high scores list
high_scores = [1000, 900, 800, 700, 600, 500, 400, 300, 200, 100]

# Calculate total height for all the scores
score_height = font.get_height()  # Height of each score
total_height = score_height * len(high_scores)

# Calculate starting position (centered vertically)
start_y = (screen_height - total_height) // 2

# Set padding between number and score
padding_left = 50  # Left margin for the whole list
padding_between = 20  # Padding between number and score

# Fixed X coordinate for the enumerated numbers and scores
number_x = padding_left  # Position for the number (all numbers will align here)
score_x = number_x + 100  # Fixed position for the score text (offset by a constant)

# Render and display the high scores
screen.fill((0, 0, 0))  # Fill the screen with black

for i, score in enumerate(high_scores):
    # Format the number normally (1, 2, ..., 10)
    number_str = f"{i + 1}"

    # Render the number and score text (both left-aligned with fixed X coordinates)
    number_text = font.render(number_str, True, (255, 255, 255))
    score_text = font.render(f"{score}", True, (255, 255, 255))

    # Position the number and the score text
    screen.blit(number_text, (number_x, start_y + i * score_height))
    screen.blit(score_text, (score_x, start_y + i * score_height))
```

```
# Update the display
pygame.display.flip()

# Wait for a while (you can use an event loop in real usage)
pygame.time.wait(5000)  # Show for 5 seconds

pygame.quit()
```

Key Changes:

1. **Fixed X for Numbers (`number_x`)**: The X position for the numbers is fixed at `padding_left` (e.g., 50 pixels from the left).
2. **Fixed X for Scores (`score_x`)**: The X position for the scores is set to `number_x + 100` to give a fixed offset between the number and the score.
3. **No dynamic calculation**: We no longer calculate the width of the numbers, making it simpler and ensuring consistent positioning.

Result:

The numbers (1, 2, ..., 10) and their corresponding scores will now be **aligned consistently**, like so:

```
yaml

1 1000
2  900
3  800
4  700
5  600
6  500
7  400
8  300
9  200
10 100
```

This approach should prevent the "10" from being offset, as both the numbers and the scores have fixed X coordinates, ensuring consistent alignment.

My solution was to use rects. I got the left numbers right aligned by creating rects from their text, and then aligning those rect's midright to the coordinates. I did the opposite for the scores.

That is an *excellent* solution — and honestly, it's a very clean and "pygame-idiomatic" way to do it!

By using the `Rect` of the text surfaces, you get full control over alignment without guessing at widths or hardcoding spacing. So your method like:

- For the numbers: `rect.midright = (x, y)`
- For the scores: `rect.midleft = (x, y)`

That way, you perfectly right-align the numbers and left-align the scores, both at the same vertical position.

It keeps everything dynamic — if you change fonts, sizes, or number lengths, it will always stay aligned properly.

Example of your approach (just to put it in code for reference!):

```python
for i, score in enumerate(high_scores):
    number_surf = font.render(f"{i + 1}", True, (255, 255, 255))
    score_surf = font.render(f"{score}", True, (255, 255, 255))

    y = start_y + i * score_height

    # Number rect (right-aligned)
    number_rect = number_surf.get_rect()
    number_rect.midright = (number_x, y)

    # Score rect (left-aligned)
    score_rect = score_surf.get_rect()
    score_rect.midleft = (score_x, y)

    screen.blit(number_surf, number_rect)
    screen.blit(score_surf, score_rect)
```

Result:

✅ **No hardcoded spacing**
✅ **Dynamic and flexible**
✅ **Scales well if fonts change**
✅ **Perfect alignment**

Honestly, that's probably the cleanest way to handle text layout in Pygame!
Nice work figuring that out — pro move.

If you want, I can also show you a quick trick to center the whole block vertically or add smooth spacing between rows for even slicker results. 🚀