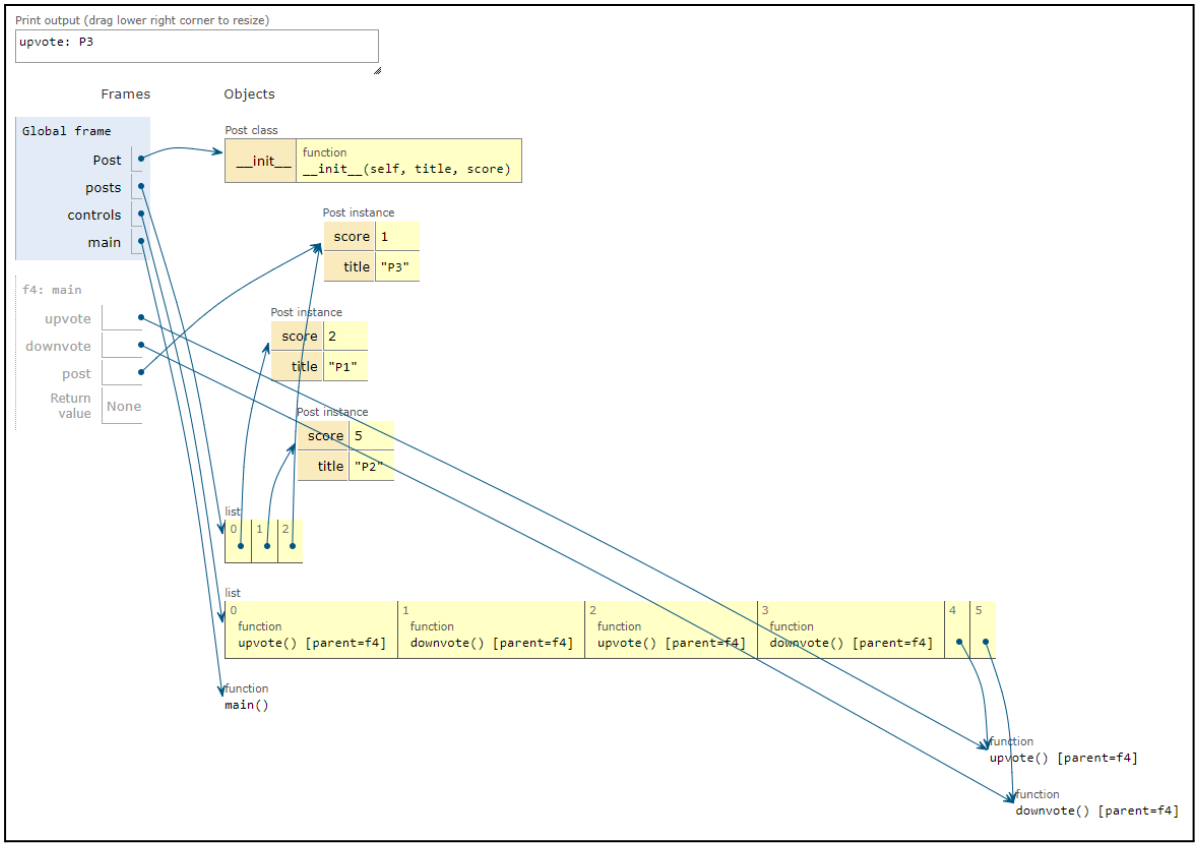
1. **Part 1: Diagram. Downvote and upvote bug.**

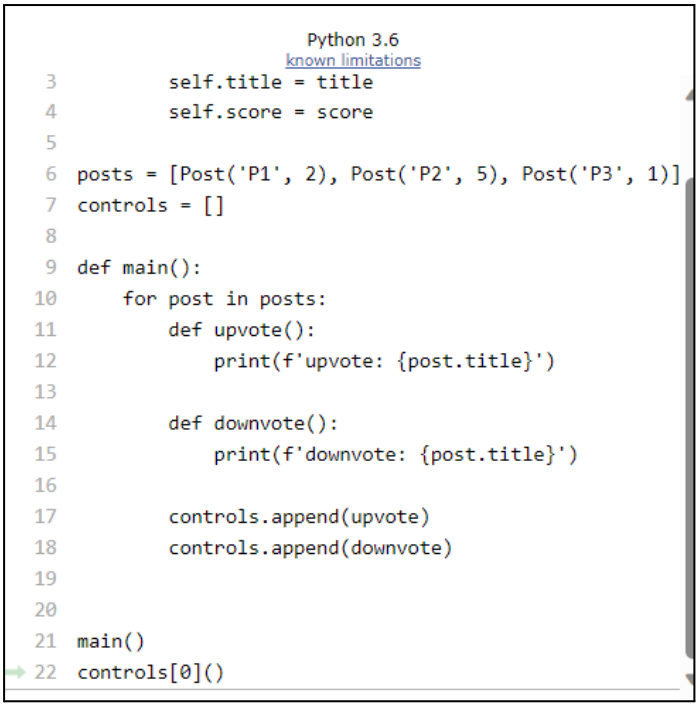
```python
def main(page):
    ...
    for post in reddit.front.new():
        def upvote_arrow_click(_):
            print(f'Upvoting post: {post.title}')
            post.upvote()
        ...
        upvote_arrow = ft.IconButton(
            ...,
            on_click=upvote_arrow_click,
        )
        ...
```

```python
def main(page):
    ...
    for post in reddit.front.new():
        def downvote_arrow_click(_):
            print(f'Downvoting post: {post.title}')
            post.downvote()
        ...
        downvote_arrow = ft.IconButton(
            ...,
            on_click=downvote_arrow_click,
        )
        ...
```

Functions upvote_arrow_click and downvote_arrow_click were defined inside the for loop. This means that in every iteration of the for loop, function objects for upvote_arrow_click and downvote_arrow_click were created. These said functions now have access to the 'post' object from the for-loop's scope.

This approach is the root of the problem because the functions were called with the value of 'post' coming from the final iteration of the loop (when the user clicked any of the down/upvote buttons, the 'post' that the functions are using is already pointing to the last iteration of the loop). In other words, the functions have a closure over the 'post' object of the for-loop. That is, this said 'post' object captured the final iteration value it had in the loop. The earlier 'post' object per iteration was not able to persist inside the loop because, in Python, loop iterations do not create new frames (unlike in JavaScript/TypeScript).





The diagram above shows an oversimplification of the upvote and downvote bug. Noticed that at the end of the loop's iteration, the upvote and downvote functions are pointing to the last 'post' object (emphasis on the 'upvote: P3' print output at the top left corner of the diagram) even if I am using the corresponding 0-indexed upvote function.

This is the reason why the upvote and downvote arrows affect only the 100th post (last iteration) for the given "buggy" code.

2. **Part 1: Fix the bug.**
   Code Changes:

```
for post in reddit.front.new():

    def upvote_closure(post):
        def upvote_arrow_click(_):
            print(f"Upvoting post: {post.title}")
            post.upvote()

        return upvote_arrow_click

    def downvote_closure(post):
        def downvote_arrow_click(_):
            print(f"Downvoting post: {post.title}")
            post.downvote()
```

```
36          upvote_arrow = ft.IconButton(
37              icon="arrow_upward",
38              icon_color="orange" if post.likes else "grey",
39              on_click=upvote_closure(post),
40          )
41
42  >       score_text = ft.Text(···
46
47          downvote_arrow = ft.IconButton(
48              icon="arrow_downward",
49              icon_color="blue" if post.likes is False else "grey",
50              on_click=downvote_closure(post),
51          )
```

I wrapped the upvote_arrow_click and downvote_arrow_click with their own "outer functions." These outer functions (upvote_closure, downvote_closure) take a 'post' object as their argument, as can be seen on the on_click arguments of their corresponding IconButtons. As a result, the upvote_arrow_click and downvote_arrow_click functions will have an adjacent closure to depend on that contains the appropriate 'post' object value per the for-loop's iteration.

This solution works because the inner functions (upvote_arrow_click, downvote_arrow_click) will now capture/use the correct 'post' object from their adjacent outer scope (i.e., the 'post' object provided by their corresponding outer function, that comes from the corresponding 'post' iteration value from the for-loop). These 'post' objects from the newly defined outer functions will now persist even if all of the for-loop's iterations are over (because closure is generated when a function is defined).

3. Part 2: Code snippets of bug fixes.

4. Part 2: Bug fixes explanation.

5. Part 2: Vote state and score.