COMP 110

More Practice with Recursive Functions

Reminders:

- Tutoring @5-7PM today and tomorrow
- Virtual review session tomorrow (11/21) at 7pm
    - Link on the site’s agenda!

Welcome to Dog110!

The COMP110 dogs went to daycare and each dog's behavior was scored on a scale of 1-10. If all 3 dogs scored at least an 8, we'll pet them 110 times. Let's write a recursive function to see if all dogs in the list were good today!

Nelli

Ada

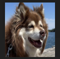
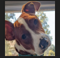
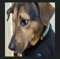
Pip

Welcome to Dog110!

The COMP110 dogs went to daycare and each dog's behavior was scored on a scale of 1-10. If all 3 dogs scored at least an 8, we'll pet them 110 times. Let's write a recursive function to see if all dogs in the list were good today!

3 parameters:

- `scores: list[dict[str, str]]`
    - list of dictionaries of dogs' names and scores
- `thresh: int`
    - Threshold we're using to determine if a dog was good
- `idx: int`
    - Index of dog of interest for the function call

```
pack: list[dict[str, str]] = [
    {"name": "Nelli", "score": "10"},
    {"name": "Ada", "score": "9"},
    {"name": "Pip", "score": "7"},
]
```

Example usage:

`print(all_good(scores=pack, thresh=8, idx=0))` would return `False`

`print(all_good(scores=pack, thresh=7, idx=0))` would return `True`

`all_good` Algorithm

Let’s write a recursive function to see if all dogs in the list were good today!

Example usage:

`print(all_good(scores=pack, thresh=8, idx=0))` would return `False`

`print(all_good(scores=pack, thresh=7, idx=0))` would return `True`

Conceptually, what will our **base case** be?

What will our **recursive case** be?

What is an **edge case** for this function? How could we account for it?

Visualizing recursive calls to `all_good`

```
all_good(scores=pack, thresh=8, idx=0) returns False
```

`scores[0]["score"] >= thresh`. Good dog, Nelli!
Now, let's check the next dict in the list…

```
return all_good(scores, thresh, idx + 1)
return all_good(scores, thresh, 1)
return False
```

`scores[1]["score"] >= thresh`. Good dog, Ada!
Now, let's check the next dict in the list…

```
return all_good(scores, thresh, idx + 1)
return all_good(scores, thresh, 2)
return False
```

`scores[2]["score"] < thresh`…
not all dogs were good!

```
return False
```

Values

```
thresh = 8
idx = 2
pack: list[dict[str, str]] = [
    {"name": "Nelli", "score": "10"},
    {"name": "Ada", "score": "9"},
    {"name": "Pip", "score": "7"},
]
```

Let’s write the `all_good` function together!

Memory diagram

```python
"""Reviewing dogs' performance in daycare."""

def all_good(scores: list[dict[str, str]], thresh: int, idx: int) -> bool:
    """Determine if all dogs were good in daycare."""
    is_good: bool = int(scores[idx]["score"]) >= thresh
    is_last: bool = len(scores) == idx + 1

    # (Let's let Python deal with the edge case(s))
    if is_good:
        if is_last:
            return True
        else:
            return all_good(scores, thresh, idx + 1)
    else:
        return False

pack: list[dict[str, str]] = [
    {"name": "Nelli", "score": "10"},
    {"name": "Ada", "score": "9"},
    {"name": "Pip", "score": "7"},
]
print(all_good(pack, 8, 0))
```

Visualizing recursive calls to `all_good`

When developing a recursive function:

Base case:

- ❏ Does the function have a clear base case?
    - ❏ Ensure the base case returns a result directly (without calling the function again).
- ❏ Will the base case *always* be reached?

Recursive case:

- ❏ Ensure the function moves closer to the base case with each recursive call.
- ❏ Combine returned results from recursive calls where necessary.
- ❏ Test the function with edge cases (e.g., empty inputs, smallest and largest valid inputs, etc.). Does the function account for these cases?