Day 3: Jack and the Giant

By Peng Wang

(Inspired by Andy, so I will try to start with a story too :)

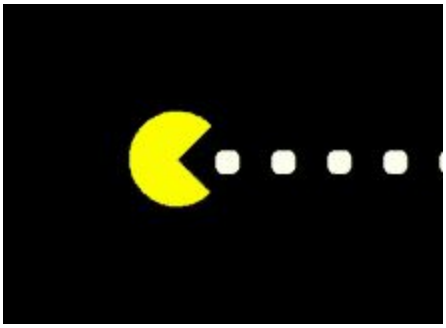
Once upon a time, there was a Giant. The Giant squeezed Jack and said, “TELL ME A STORY OR I WILL GRIND YOUR BONES TO MAKE MY BREAD. AND THEN YOUR STORY IS FINISHED, I WILL GRIND YOUR BONES TO MAKE MY BREAD ANYWAY! HOHOHO”. The Giant laughed an ugly laugh. Jack thought, “He’ll kill me if I do. He’ll kill me if I don’t. There is only one way to get out of this.” Jack cleared his throat, and then began his story:

> Once upon a time, there was a Giant. The Giant squeezed Jack and said, “TELL ME A STORY OR I WILL GRIND YOUR BONES TO MAKE MY BREAD. AND THEN YOUR STORY IS FINISHED, I WILL GRIND YOUR BONES TO MAKE MY BREAD ANYWAY! HOHOHO. " The Giant laughed an ugly laugh. Jack thought, “He’ll kill me if I do. He’ll kill me if I don’t. There is only one way to get out of this.” Jack cleared his throat, and then began his story:
>
> > Once upon a time, there was a Giant. The Giant squeezed Jack and said, “TELL ME A STORY OR I WILL GRIND YOUR BONES TO MAKE MY BREAD. AND THEN YOUR STORY IS FINISHED, I WILL GRIND YOUR BONES TO MAKE MY BREAD ANYWAY! HOHOHO. " The Giant laughed an ugly laugh. Jack thought, “He’ll kill me if I do. He’ll kill me if I don’t. There is only one way to get out of this.” Jack cleared his throat, and then began his story:
> >
> > > Once upon a time, there was a Giant. The Giant squeezed Jack and said, “TELL ME A STORY OR I WILL GRIND YOUR BONES TO MAKE MY BREAD. AND THEN YOUR STORY IS FINISHED, I WILL GRIND YOUR BONES TO MAKE MY BREAD ANYWAY! HOHOHO. " The Giant laughed an ugly laugh. Jack thought, “He’ll kill me if I do. He’ll kill me if I don’t. There is only one way to get out of this.” Jack cleared his throat, and then began his story
> > > Once upon a time, there was a Giant. The Giant squeezed Jack and said, “TELL ME A STORY OR I WILL GRIND YOUR BONES TO MAKE MY BREAD. AND THEN YOUR STORY IS FINISHED, I WILL GRIND YOUR BONES TO MAKE MY BREAD ANYWAY! HOHOHO. " The Giant laughed an ugly laugh. Jack thought, “He’ll kill me if I do. He’ll kill me if I don’t. There is only one way to get out of this.” Jack cleared his throat, and then began his story
> > > > Once upon a time, there was a Giant. The Giant squeezed Jack and said, “TELL ME A STORY OR I WILL GRIND YOUR BONES TO MAKE MY BREAD. AND THEN YOUR STORY IS FINISHED, I WILL GRIND YOUR BONES TO MAKE MY BREAD ANYWAY! HOHOHO”. The Giant laughed an ugly laugh. Jack though, “He’ll kill me if I do. He’ll kill me if I don’t. There is only one way to get out of this.” Jack cleared his throat, and then began his story
> > > > Once upon a time, there was a Giant. The Giant squeezed Jack and said, “TELL ME A STORY OR I WILL GRIND YOUR BONES TO MAKE MY BREAD. AND THEN YOUR STORY IS FINISHED, I WILL GRIND YOUR BONES TO MAKE MY BREAD ANYWAY! HOHOHO”. The Giant laughed an ugly laugh. Jack though, “He’ll kill me if I do. He’ll kill me if I don’t. There is only one way to get out of this.” Jack cleared his throat, and then began his story
> > > > > Come on, you know how this ends, right?
> > > > > Come on, you know how this ends, right?
> > > > > > Come on, you know how this ends, right?

I’m always fascinated by how recursion can help us solve problems. Jack would be a fantastic software engineer if he’d be born today.

In computer science, the gist of recursion is:

- If you can bite off a small chunk of the problem, and the remaining portion is still the same kind of problem, then congratulations, you are close to solve the whole thing with recursion! All you have to do is to focus on how to bite off the first small chunk, and then you know how to finish it to the end (usually there is an end we hope). Almost as simple as that!



Python doesn’t support recursion in a full-fledged way. Say by default you can only take 1000 bites for solving a single problem. But, it doesn’t discourage our enthusiasm about it at all!

Python problem

Here is a nested dictionary about Pac Man

```
d2 = {"name": "Pac Man",
      "nickname": "Mr Puck",
      "activity": {
          'day': 'sleep',
          'night': {'eat': "pellets",
                    'fight': "monsters"}
      }
      }
```

Flatten it into the following fashion so that we can easily ingest it into a SQL database in a tabular format.

```
{'activity.night.fight': 'monsters',
'activity.night.eat': 'pellets',
'activity.day': 'sleep',
'nickname': 'Mr Puck',
'name': 'Pac Man'}
```

Solving the above is sufficient. But if it's simply not enough exercise for your mind, try adding a little twist:

-- what if

- We have empty dictionaries like
  `'hobby':{}`
- We also have nested lists and tuples, like
  `"madness": [1, ['a', {"name": 1, "gender": 2}, ['x', 'z']], 3]`

---- Just provides Peng's full solution here for reference:

```
def flatten(x: dict, prefix: str = ''):
    output = {}

    if x:                           # Final exit condition when x is depleted
        temp = x.popitem()                                                  # pop one out

        key = str(temp[0]) if prefix == '' else prefix + "." + str(temp[0])  # Combine previous key and current key

        if not temp[1]:
            output.update({key: None})
```

```
    elif isinstance(temp[1], dict):
        output.update(flatten(temp[1], prefix = key))                    # If a child is another dictionary
    elif isinstance(temp[1], list) or isinstance(temp[1], tuple):
        output.update(flatten(dict(enumerate(temp[1])), prefix = key))   #  Child recursion too
    else:
        output.update({key: temp[1]})                                    # Otherwise, update the output

    output.update(flatten(x, prefix = prefix))       # Process other items

return output
```