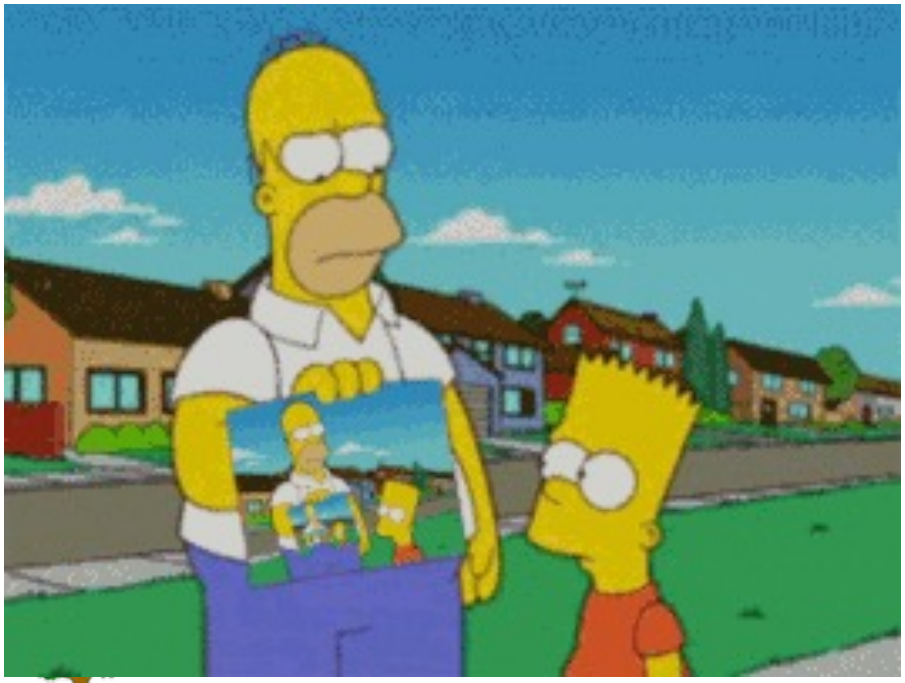
CSC 148: Introduction to Computer Science

Week 6



Recursion (continued)

University of Toronto Mississauga,

Department of Mathematical and Computational Sciences

Writing recursive functions

- Worksheet ...
  - nested_list_contains
  - first_at_depth

A common error: missing `return`

```
def nested_list_contains(obj, item) -> bool:
    if isinstance(obj, int):
        return obj == item
    else:
        for sublist in obj:
            if nested_list_contains(sublist, item):
                nested_list_contains(sublist, item)
        return False
```

A common error: missing `return`

```
def nested_list_contains(obj, item) -> bool:
    if isinstance(obj, int):
        return obj == item
    else:
        for sublist in obj:
            if nested_list_contains(sublist, item):
                nested_list_contains(sublist, item)
        return False
```

A common error: missing `return`

```
def nested_list_contains(obj, item) -> bool:
    if isinstance(obj, int):
        return obj == item
    else:
        for sublist in obj:
            if nested_list_contains(sublist, item):
                True
        return False
```

A common error: missing `return`

```
def nested_list_contains(obj, item) -> bool:
    if isinstance(obj, int):
        return obj == item
    else:
        for sublist in obj:
            if nested_list_contains(sublist, item):
                return True

        # What about if it's not found?
```

A common error: missing `return`

- A `return` statement exits from one function call.

- When writing a recursive function that should return something, both the base case and recursive step *typically* must have a `return`!

- More generally, if a function returns something, then every execution path through the function must have a `return`!

`first_at_depth` – base case

- A single integer is always at depth 0.

```
>>> first_at_depth(100, 0)
100
>>> first_at_depth(100, 3) is None
True
```

first_at_depth – recursive case

```
>>> first_at_depth([10, [[20]], [30, 40]], 2)
30
```

sublist	depth	first_at_depth(sublist, depth)
10		
[[20]]		
[30, 40]		

`first_at_depth` – multiple base cases!

```
first_at_depth(obj, d)
    -> first_at_depth(sublist, d - 1)
```

- We are actually recursing on both `obj` and `d`.
- Can't recurse when:
  - `isinstance(obj, int)`
  - `d == 0`