Discussion 7

Mutable Trees and Linked Lists

Antonio Kam

anto [at] berkeley [dot] edu

All slides can be found on

`teaching.rouxl.es`

Announcements

- Ants
  - Phase 1: Friday!
  - Phase 2: Tuesday (next week)
  - Due Date: next Friday

Notes from last section

- hi
  - hi! 👀
- do you play any rhythm games
  - not very often, but have played some
    - rhythm heaven/rhythm doctor
    - crypt of the necrodancer
- Do you play League of Legends 😄
  - nope!
- OOP review
  - to an extent we'll get something today!

Notes from last section

- Multiple inheritance
  - won't need to worry too much about this
  - it is in scope, but you generally don't need to know too much about it other than the fact that you can inherit methods from different parents
- i think the mini lectures could have been a bit slower and we could have spent less time on the second problem.
  - honestly i agree yeah oops

Notes from last section

- where were you on monday?
  - at a cubing competition
  - i went to vancouver the past weekend for canadian championships
  - i got pretty good results (load a video i guess)
- In the Game Pokemon Mystery Dungeon, the is a useful item known as the Pass Scarf, which, at the expense of 2 hunger per pass, can pass off any physical attack. It tends to pass off to the physical attack to the lower right, so proper positioning of your partner Pokemon can lead to strategies built of reflecting damage. There is also a surprising variety of moves that can be passed off, though no thrown items are susceptible to this ability
  - meta moves in pmd are very different to regular pokemon

Temperature Check 🌡️

- Mutable Trees
- Linked lists (Covered in today's lecture)

Trees

Trees (construction)

- The `Tree` constructor takes in a `label`, and an *optional* `list` of `branches`. If `branches` isn't given, it defaults to an empty list `[]`

```
class Tree: # simplified version compared to the 61A implementation
    def __init__(self, label, branches = []):
        assert type(branches) == list
        for item in branches:
            assert isinstance(item, Tree)
        self.label = label
        self.branches = branches
```

Construction: `Tree(2)`; `Tree(2, [Tree(3), Tree(4)])`

Note that the branches **must** be in a list.

Trees (access)

```
t = Tree(3, [Tree(4, [Tree(5)]), Tree(6)])
>>> t.label
3
>>> t.branches
[Tree(4, [Tree(5)]), Tree(6)]
```

Worksheet!

Linked Lists

Linked Lists

- Linked lists are recursive data structures
- You can use linked lists to create your own version of a sequence
- They are generally useful when you want to have an infinitely-sized list, or want to dynamically change the size of the list (more useful in 61B if you do end up taking it)
- In general, linked lists problems can be solved using both iteration and recursion
  - Like trees, they are recursive data structures, but unlike trees, you can use both recursion and iteration to solve them.

Linked Lists (Anatomy)

```
class Link:
    empty = ()

    def __init__(self, first, rest = Link.empty):
        assert rest is Link.empty or isinstance(rest, Link)
        self.first = first
        self.rest = rest
```

- Linked lists have a `first` (similar to `label`) attribute and a `rest` (similar-ish to `branches`) attribute.

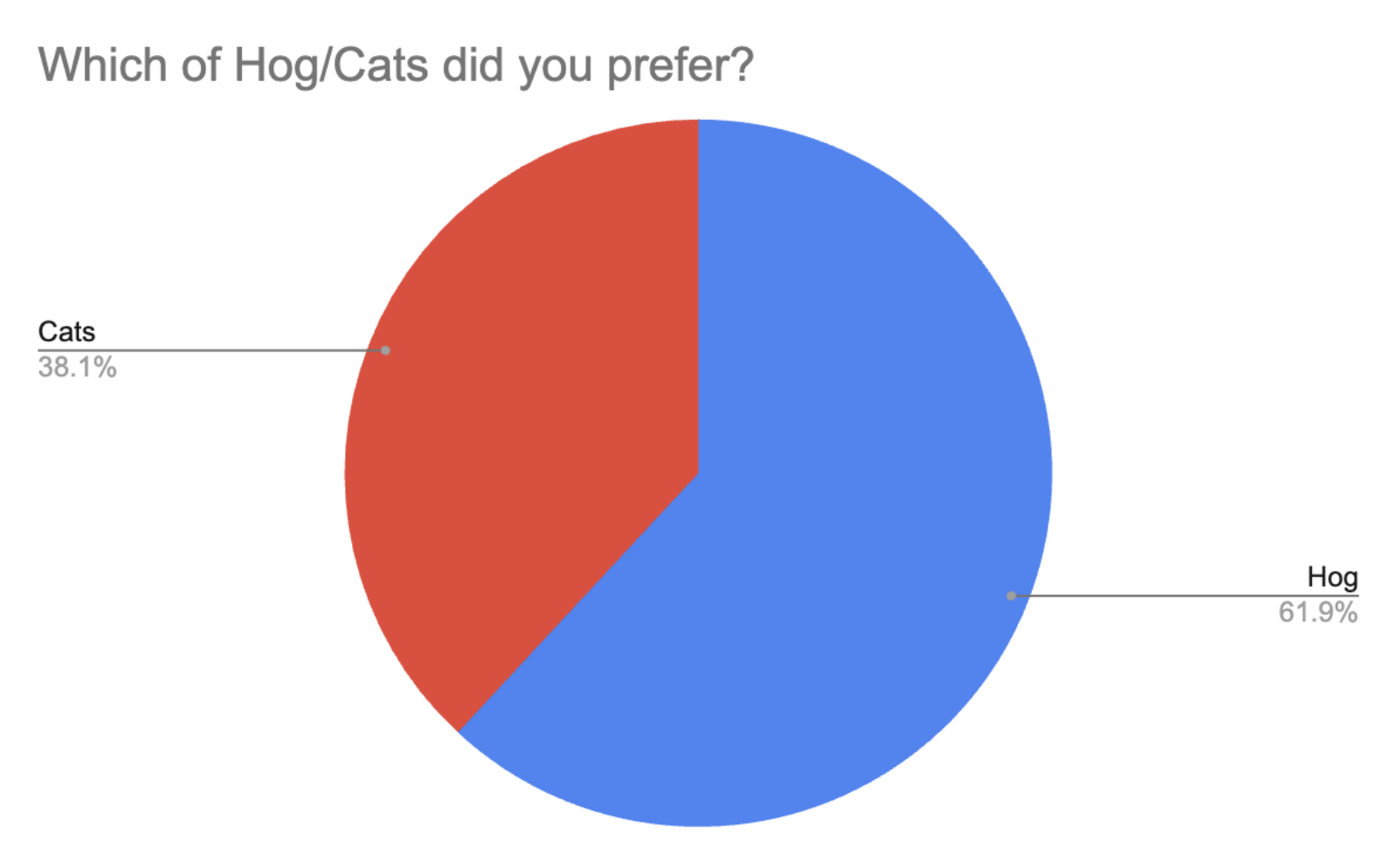
Linked Lists (Construction)

Note to Anto: Draw box-and-pointer diagram

```
s = Link(1, Link(2, Link(3)))
>>> s.first
1
>>> s.rest
Link(2, Link(3))
>>> s.rest.first
2
s2 = Link(1, 2) # This will error because rest is not a linked list
>>> s.rest.rest.first
3
```

Worksheet!

Results from last section (`links.rouxl.es/disc`)



Mental Health Resources

- CAPS:
  - If you need to talk to a professional, please call CAPS at 510-642-9494.
- After Hours Assistance
  - For any assistance after hours, details on what to do can be found at this link

Anonymous Feedback Form

`links.rouxl.es/feedback`

Thanks for coming! 🥳

Please give me feedback on what to improve!