Discussion 5

Sequences, Data Abstraction, Trees

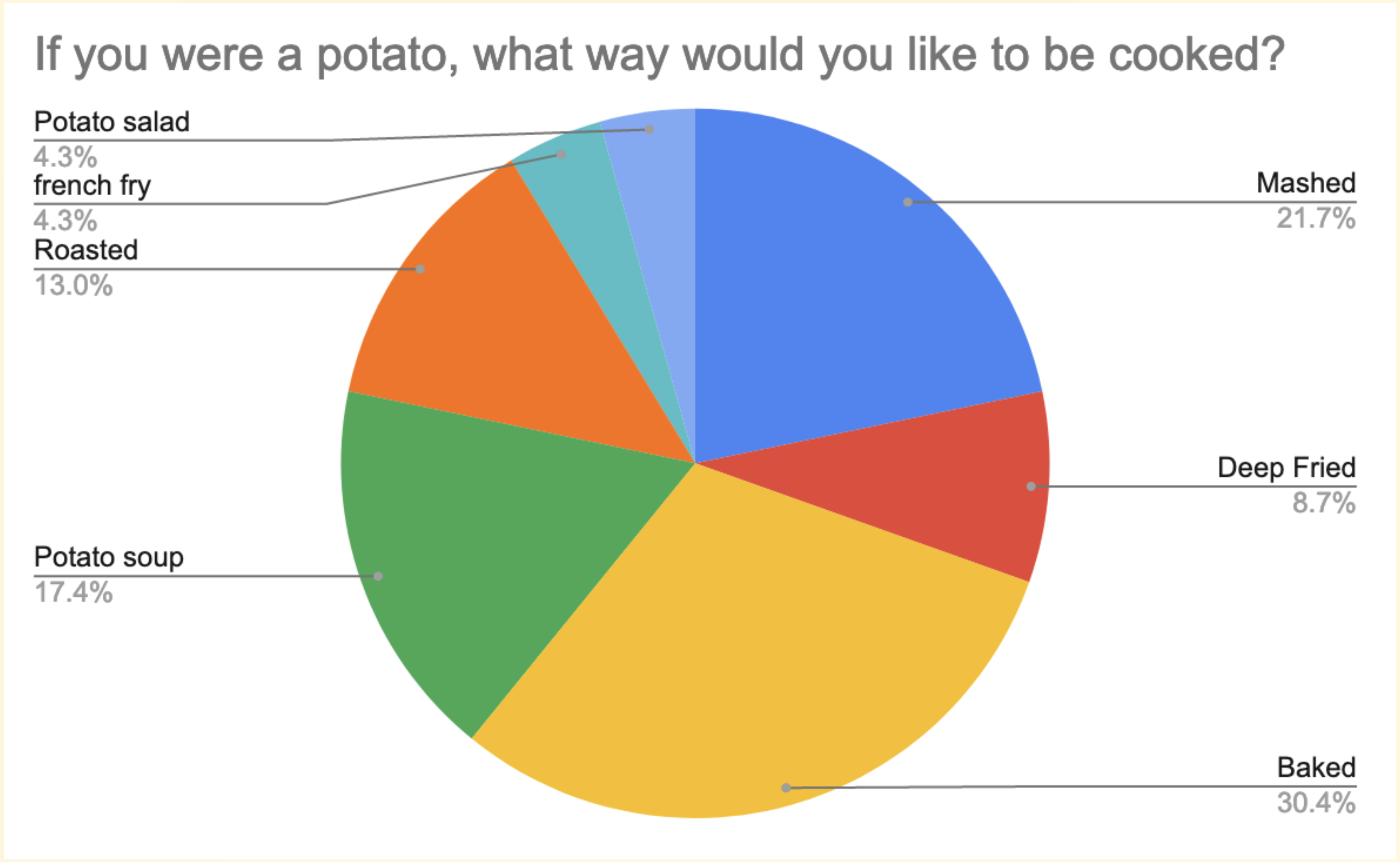
Antonio Kam

`anto [at] berkeley [dot] edu`

Announcements

- My voice is still fried 🥳 I really hope it gets better by next lab, but who knows at this point
    - This also means I won't be as energetic today, and also will sometimes take deep breaths before speaking
    - This is not your fault
    - oops
- Please finish cats! Deadline is today! 🐈
- HW4 Released!
- I won't be here for discussion next week
    - I'm doing something pretty exciting 👀

Results from last section



Comments from last section

- Do I play valorant
    - nah
- Recursion practice! (With helper methods)
- Efficiency (CS 61A's stance on efficiency)
- cs61a.rouxl.es

Temperature Check 🌡️

- Recursion
- Tree Recursion
- `map`, `filter`, `reduce`
- Trees! 🎄

All slides can be found on `teaching.rouxl.es`

Sequences

Lists

- So far, we've only really been able to store one piece of data at a time
- A list allows you to store multiple pieces of data in 1 variable
- Lists can store any data type, and can be a *mix* of different data types
  - For example: `[1, "hello", [2, "hi"]]`
- Very useful for storing data/information

(List) Slicing; Can also be applied with strings

Syntax is `lst[<start index>:<end index>:<step>]`; this creates a *copy* of all or part of your list (will be important later)

- `start index` is inclusive (if not included, it defaults to the first value)
- `end index` is exclusive (if not included, it defaults to the end of the list)
- `step` can be negative!

```
lst = [3, 4, 5]
lst[:] # Makes a copy; returns [3, 4, 5]
lst[1:] # [4, 5]
lst[::-1] # [5, 4, 3]
lst[::2] # [3, 5]
```

For loops

```
for <variable> in <sequence>:
  [body of for loop]
```

Example:

```
lst = [3, 4, 5]
for elem in lst:
  print(elem)
```

List Comprehensions

Very similar to `for` loops, but can be done in 1 line!

```
[<expression> for <variable> in <sequence> [if <condition>]]

lst = [3, 2, 1]
[x * 2 for x in lst if x < 3] # [4, 2]
[x * 2 for x in [3, 2, 1]] # [6, 4, 2]
```

`map`, `filter`, `reduce`

- Used to manipulate sequences
- Makes copies of lists
- `map` applies a *function* to each element in a sequence
- `filter` chooses elements in a sequence based on a *predicate*
- `reduce` combines elements together based on a *function*

Worksheet!

Data Abstractions

What are Data Abstractions?

- Data abstractions are a super powerful way to let people treat code as objects, rather than knowing how the thing works itself
- Allows you to worry about how something works, rather than how something is implemented
- You'll see a lot of abstractions in other courses (Data 8, Data 100 are filled with abstractions of some sort)

What are Data Abstractions?

- Data abstractions have the following:
    - Constructors: Used to build the abstract data type
        - IMPORTANT: You do not need to know how the programmer decided to implement this!
    - Selectors: Used to interact with the data type

Example: Tree Data Abstraction

- Trees are recursive data structures (as in, trees contain more trees)
- Important terms:
  - Root Node
  - Branch(es)
    - This will be a list!
  - Leaf Node
  - Children
- Sort of looks like an upside-down tree compared to the real world
- Questions are generally solved using tree recursions

Tree ADT Implementation:

```
def tree(label, branches=[]):
    """Construct a tree with the given label value and a list of branches."""
    return [label] + list(branches) # All items in branches must be trees!

def label(tree):
    """Return the label value of a tree."""
    return tree[0]

def branches(tree):
    """Return the list of branches of the given tree."""
    return tree[1:]

def is_leaf(tree):
    return not branches(tree)
```

Tree Example:

```
t = tree(1,
      [tree(3,
          [tree(4),
           tree(5),
           tree(6)]),
      tree(2)])
```

Attendance

`links.rouxl.es/disc`

Worksheet!

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!