Discussion 4

Sequences

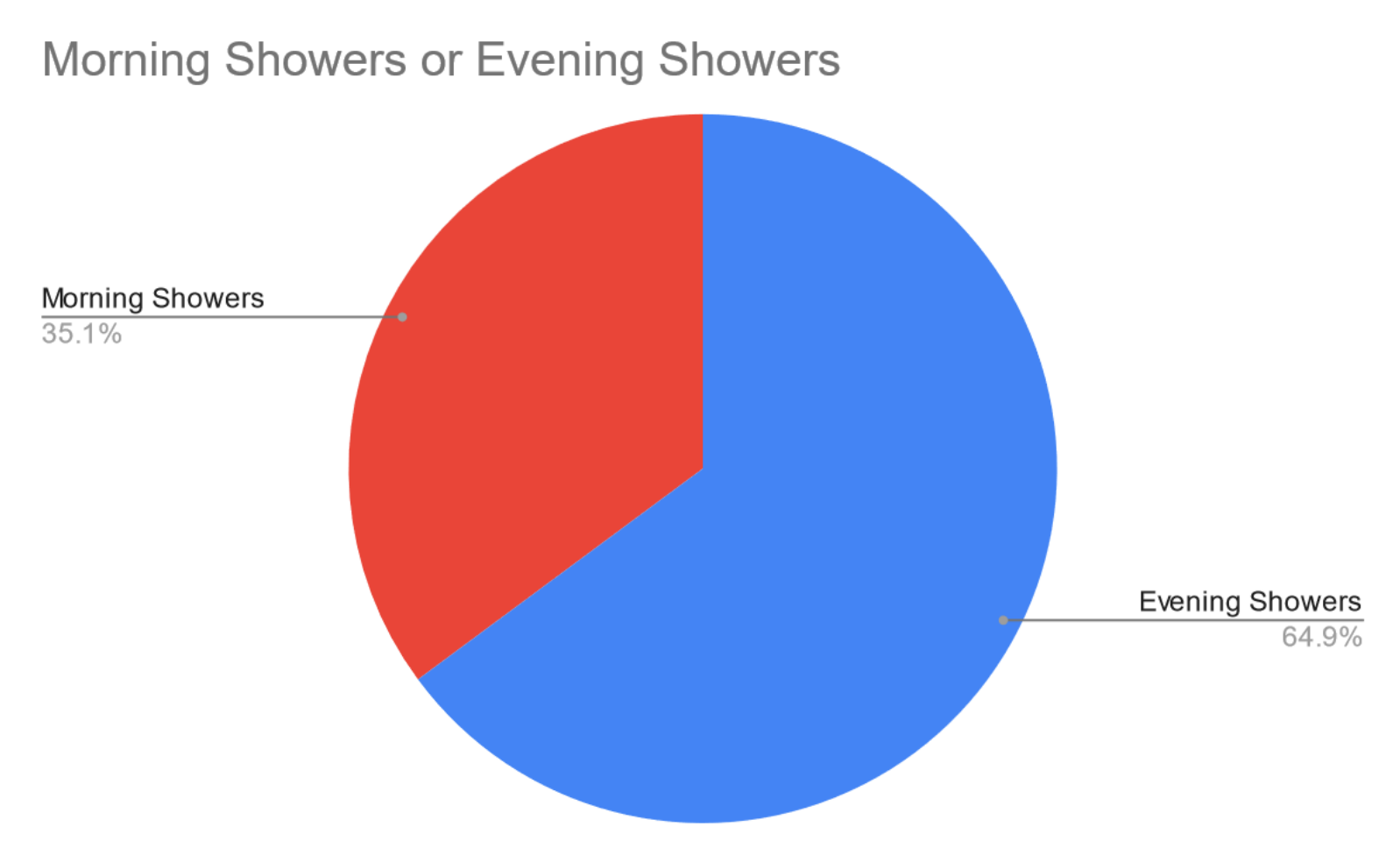
Antonio Kam

anto [at] berkeley [dot] edu

Announcements

- Hog due on Wednesday! (July 6th)
  - Submit by Tuesday to get EC points (Today! July 5th)
  - Extensions can't be used for EC unless you have specific accommodations
- HW2 due Thursday (July 7th)
- Cats (Project 2!) begins Thursday
  - Checkpoint next Tuesday (July 12th)
  - Due following Tuesday (July 19th)
- I'm out of town 😭 from Wednesday to Monday
  - People will cover labs and discussions; attendance will work the same way
  - Please still come - show them some love and make them feel welcomed 👀
- Instructor OH - conceptual help; super useful!

Result from last poll



Questions/Comments from last time

- Starting `is_prime`'s helper function from `n - 1`

```
def is_prime(n):
    def helper(i):
        if i == 1:
            return True
        elif i < 1 or n % i == 0: # i < 1 necessary for if you enter n = 1
            return False
        return helper(i - 1)
    return helper(n - 1)
```

Questions/Comments from last time

- Study Groups - will be done on the attendance form (sorry I forgot about this last time 😭)
  - There is something on Piazza to search for teammates FYI, but this is more for projects
- Recursion Resources
  - Current Resources - can also be found on my website
  - My CS 61A Notes from a while ago

Temperature Check 🌡️

- Recursion
- Tree Recursion
- `map`, `filter`, `reduce`
- `for` loops
- Lists
  - List Comprehensions
- Dictionaries

All slides can be found on

`teaching.rouxl.es`

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

Creating Lists

In general, to create a list, you wrap something in square brackets

- For example: `[1, 2, 3]` creates a list.

(List) Slicing

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!

Attendance

`links.rouxl.es/disc`

Dictionaries 📖

- Maps `keys` to `values`
- Doesn't really have an order
- Access elements using `keys` rather than indices

Dictionaries 📖

- Maps `keys` to `values`
- Doesn't really have an order
- Access elements using `keys` rather than indices
- Defined with curly braces (`{}`)
  - `{key: value}`

Demo:

```
pokemon = {'pikachu': 25, 'dragonair': 148, 25: 'hello'}
pokemon['pikachu'] # 25
pokemon['hello'] = 'hi'
pokemon # {'pikachu': 25, 'dragonair': 148, 25: 'hello', 'hello': 'hi'}
```

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!