Discussion 6

Mutability, Iterators, Generators

Antonio Kam

anto [at] berkeley [dot] edu

Announcements

- GL with the midterm 🥲
- Y'all will do great!
- Cats checkpoint due today
- Cats due on next Tuesday

Comments from last section

- Cover more stuff on discussion worksheets 🙄
- How do you reverse a tree
  - 💀
  - depends on what implementation of trees you're using (the 61a version is a bit harder, but still possible with the `reverse` function)
    - ask me over email, i'll send over something
  - Do you have a favorite cat breed?
    - I like all cat breeds (even the hairless ones are cute 🙄)
  - how tall are you ? recursion and leap of faith and tuples!!!
    - im 180cm (i think that's 5'11")
    - we can get to the other stuff once we hit midterm review

Comments from last section

- how many pushups can u do
  - 💀
- Definitely some extra review on trees
  - got some during lab, might get some more today!
- Dr Pepper >> Coca cola
  - still haven't tried dr pepper (maybe)
- In the game Enter the Gungeon, there is one NPC you can only understand if you have a specific item. He is very eleoquent
  - bullet hell games!! hard

Temperature Check 🌡️

- Iterators
- Generators

Iterators

Iterators

- Iterable
  - Old View: Something we can go through one at a time
  - New View: Something we can call `iter` on - this will return an iterator.
- Iterator
  - A type of object that lets us keep track of which element is next in the sequence

Functions to Interact with Iterators

- `iter(iterable)`
  - Takes in an iterable and returns an iterator.
    - Calling `iter` on an iterable makes a brand new iterator
    - Calling `iter` on an iterator returns that same iterator (not a copy!) and does not change the state of that iterator
- `next(iterator)`
  - Takes in an iterator and outputs the next element
    - Raises `StopIteration` when it has no more elements to go through
    - Cannot call `next` on an *iterable*

Analogy

- Iterable = Book
- Iterator = Bookmark
- Calling `iter` on an iterable (book) will create a new bookmark
- Calling `iter` on an iterator (bookmark) will just give you the same bookmark (no reason to mark where a bookmark is)
- Calling `next` on an iterator moves the bookmark to the next chapter

do some terminal example pls thx

Worksheet!

Generators

Generators

- Generators are more specific versions of iterators that you can create yourself!
- When a generator function is called, it returns a generator object!
- The body of a generator function is not evaluated until `next` is called on the returned generator object.
- Generator functions look like normal functions, but use `yield` instead of `return`. Python will automatically see `yield` and determine a function a generator function if necessary.

Generators (continued)

- When `next` is called on a generator object, the body of a function is executed until `yield` is reached.
- From there, the yield statement will return a value, and then pauses that specific function at that moment (by saving the frame, and the line that it's on)
- When `next` is called again, we continue where we left off, until `yield` is reached again.
- `StopIteration` will be raised at the end of a generator function

Example

```
def countdown_generator(n):
    assert n >= 0
    while n >= 0:
        yield n
        n -= 1
```

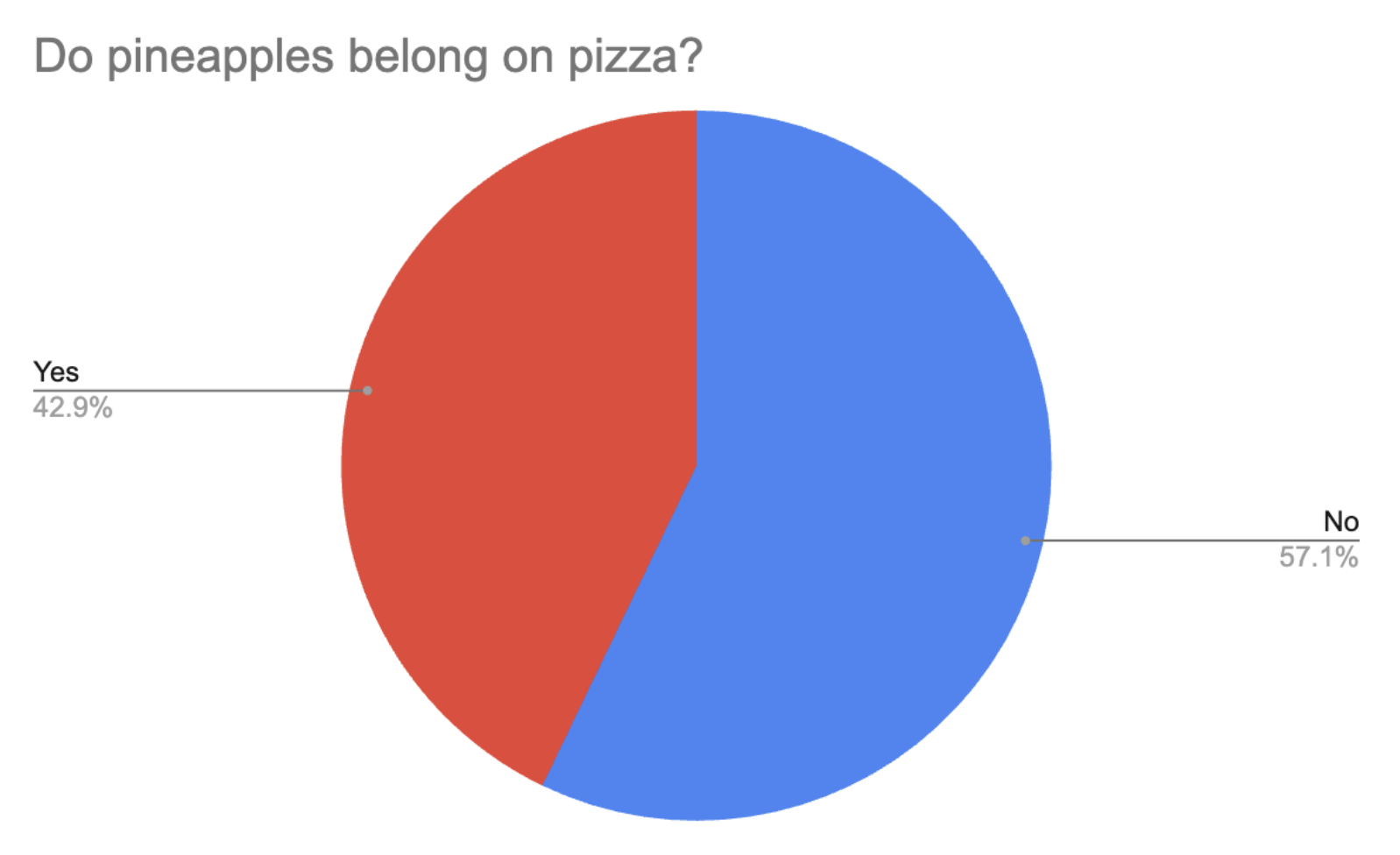
`yield from`

- You can use `yield from <iterable>` if you want to yield every value from an iterable. These are equivalent:

```
lst = [1, 2, 3]
# Version 1
yield from lst

for item in lst:
    yield item
```

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



Midterm Review

Midterm tips

- Studying
- Question annotation
- Example Question (Fall 2021 MT2 `amounts`)

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!