

Welcome!

Please sit with your group members!

Data Structures

4/5/2022

Announcements

Lists

- Collection of objects
- Allows for duplicates and multiple types
- Adding to a list
- Removing from a list
- Parsing into a list

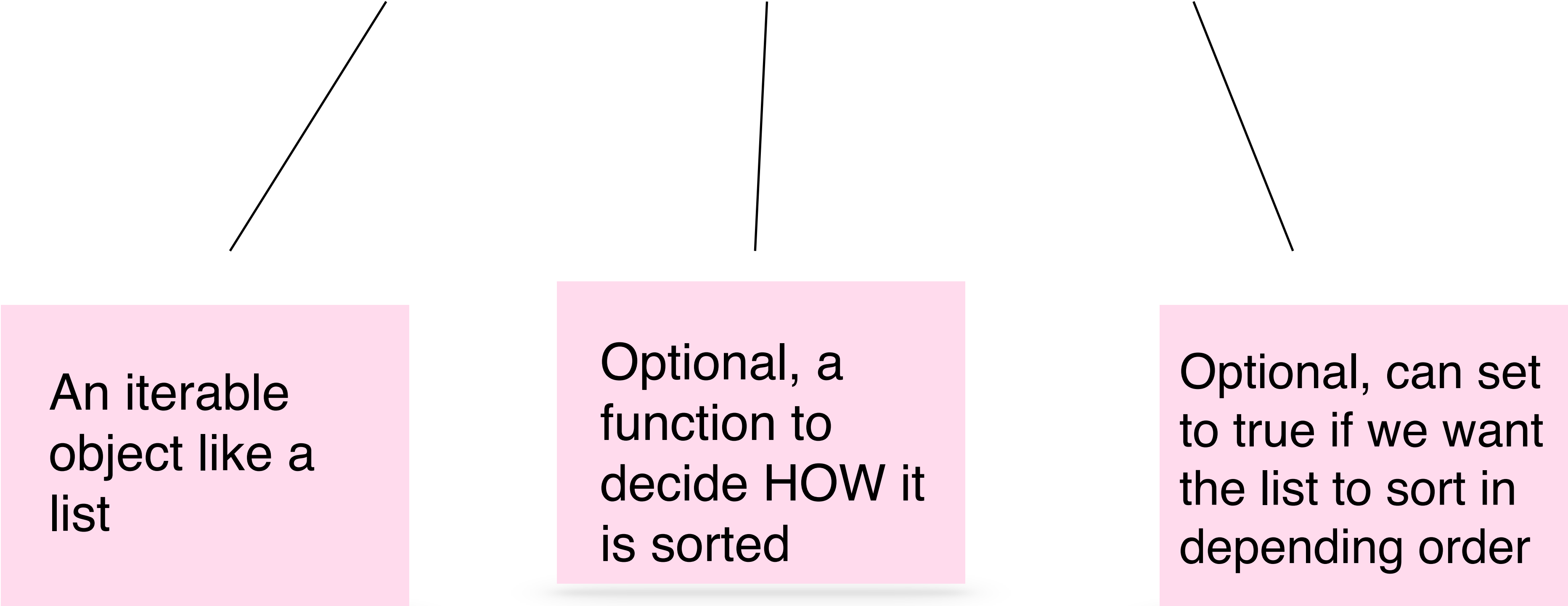
Lists

```
my_list = []  
a = my_list
```

- Doing the following does not make a copy of the list, it instead just points to the same memory address
- Can check this with the `id()` function

Sorted

`sorted(iterable, key=key, reverse=reverse)`



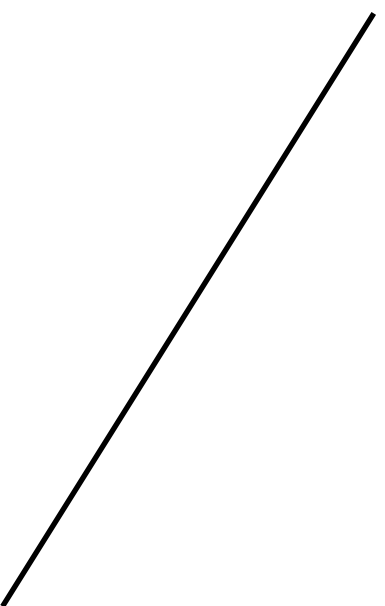
An iterable
object like a
list

Optional, a
function to
decide HOW it
is sorted

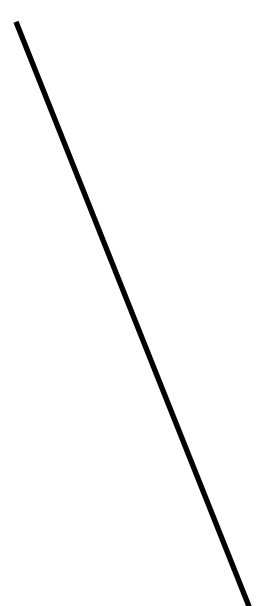
Optional, can set
to true if we want
the list to sort in
depending order

Enumerate

```
enumerate(iterable, start=0)
```



An iterable
object like a
list



Optional, we can
set the starting
index value

Tuple

- Basically same thing as a list, just immutable and defined by () instead of []
- Why is this helpful?
 - Permanence
 - Hashable (when objects inside are hashable)

Tuple Packing and Unpacking

- If we set a variable to a list of objects separated by commas, this will automatically create a tuple. This is called packing.
- If we do the opposite, and set multiple variable names separated by commas to a tuple, it will assign the corresponding values from the tuple to the variables. This is unpacking.
- We can also use these together to set multiple variables at once like so:

```
first, last = "Tara", "Jones"
```

Sets

- A mutable object similar to a list, but can only contain hashable objects and cannot contain duplicates
- Unlike a list or a tuple, this is unordered
- Adding, removing
- Support math operations

```
my_set = set()
```

Dictionaries

Key, Value

```
graph TD; A[Key, Value] --- B[Must be hashable, unique]; A --- C[Can be anything!]
```

The diagram illustrates the components of a dictionary. At the top, the text "Key, Value" is centered. Two diagonal lines extend downwards from this text to two separate pink rectangular boxes. The left box contains the text "Must be hashable, unique", and the right box contains the text "Can be anything!".

Must be hashable,
unique

Can be
anything!

Dictionaries

```
grades = {}
```

```
grades["CS41"] = "Credit"
```

```
grades["Math51"] = "B-"
```

Dictionaries

- Removing
- Getting keys, values
- Looping through

Activity: Data Structures

Let's take a content break and
talk logistics

Discussion Norms

Engage

- Take space when it's your turn to speak, then step down and make space for others.
- Be willing to talk and engage—especially with your team and your peers.
- Be open to being wrong and being challenged.
- No cold-calling.
- The course staff should strive to allow everyone to participate in the discussion so that it isn't monopolized.

Have Empathy

- Don't judge people for asking questions and respect their ideas.
- Be conscious of different experiences and backgrounds in CS and life more generally.
- When being critical, make sure to be respectful.

In groups

- Everyone should introduce themselves with their name and pronouns (if comfortable) first.
- Group members should check in on each other at the beginning of meetings.

Logistical

- Use Ed to ask and answer questions of each other and of the course staff.
- Maintain open, honest, and frequent communication between students and staff.

And lastly... Take breaks

- Have breaks or low-energy transitions between topics.
- Be easy on yourself! :)

Sign up for sections

≡

Sp22-CS-41-01 > People > Groups

Spring 2022

Home

Ed Discussion

People

EveryoneGroups

Search Groups or People

Arpit's Section WEDNESDAY 1:30–3PM – 460-429 Lab Sign Up0 students🔒

Elizabeth's Section (Thornton 209) Lab Sign Up0 students🔒

Shounak's Section (50–52E) Lab Sign Up0 students🔒

Theo's Section (120–314) Lab Sign Up0 students🔒

+ Group

Assignment 1

- Will be released this Thursday
- Focuses on group dynamics
- We want to know how your group is going!

Dear Data Video



Reading!

Comprehensions

- Very very cool python feature!
- Can “flatten” a loop
- Let's try to double everything in a list

Comprehensions

- General pattern here is:

```
[fn(elem) for elem in collection]
```

Dictionary Comprehensions

```
counts = {"dog": 4, "cat": 4, "the": 8, "grass": 4}  
#double keys  
counts = {key: value*2 for key,value in counts.items()}  
print(counts)
```

Activity: List Comprehensions

Lecture Code

Lists, Sorted

```
my_list = []

my_list.append("CS41")
my_list.append(6)
my_list.append("CS41")
my_list.insert(1, "Horses")

# ['CS41', 'Horses', 6, 'CS41']

my_list.append(5)
my_list.remove("CS41")
my_list.pop(3)

# ['Horses', 6, 'CS41']

my_list[1] += 1
# ['Horses', 7, 'CS41']
```

```
star_wars = ["R2D2", "C3P0", "Luke", "Vader"]

print(sorted(star_wars))
#['C3P0', 'Luke', 'R2D2', 'Vader']

print(star_wars)
#notice it did not change the actual list
#['R2D2', 'C3P0', 'Luke', 'Vader']

star_wars.sort() #but this does
print(star_wars)
#['C3P0', 'Luke', 'R2D2', 'Vader']

def second_letter(s):
    return s[1]

star_wars = sorted(star_wars, key=second_letter)
print(star_wars)
#['R2D2', 'C3P0', 'Vader', 'Luke']

star_wars= sorted(star_wars, reverse=True)
print(star_wars)
#['Vader', 'R2D2', 'Luke', 'C3P0']
```

Enumerate

```
students_2022 = ["Tara", "Parth", "Theo", "Elizabeth"]  
  
#every students ID will be their graduation year plus a unique int  
  
print((list(enumerate(students_2022, 20220))))  
#[(20220, 'Tara'), (20221, 'Parth'), (20222, 'Theo'), (20223, 'Elizabeth')]
```


Tuple

```
address = ("680 Lomita", "Stanford", "CA")  
address.append("USA") #will throw an Attribute Error
```

```
#packing
```

```
a = 1  
b = 2  
c = 3
```

```
nums = a,b,c #packing  
a+=1 #will not affect the tuple
```

```
my_tup = (4,5,6)  
a,b,c = my_tup #unpacking
```

```
print(a,b,c)  
first, last = "Tara", "Jones"
```

Dictionaries

```
grades = {}

grades["CS41"] = "Credit"
grades["Math51"] = "B-"

print(grades["CS41"])
#print(grades["Phsyics43"]) #will give an error
print(grades.get("Phsyics43")) #will give None

del grades["Math51"]

print(grades.keys())
print(grades.values())
print(grades.items())

for key,value in grades.items():
    print(key,value)
```

Activity 1

```
def create_counts_dict(s):  
    """  
    Returns a dictionary that maps a word to how many times it showed up in the string.  
    """  
    words = s.split(" ")  
    d = {}  
    for word in words:  
        word = word.lower()  
        if word not in d:  
            d[word] = 0  
        d[word] += 1  
    return d  
  
def mix_things_up(counts):  
    new_counts = {}  
    for key, value in counts.items():  
        if value not in new_counts:  
            new_counts[value] = []  
        new_counts[value].append(key)  
    return new_counts
```

Comprehensions

```
l = [1, 2, 3, 4, 5]
```

```
for i in range(len(l)):  
    l[i] *= 2
```

```
print(l)
```

```
l = [1, 2, 3, 4, 5]
```

```
l = [n*2 for n in l]
```

```
print(l)
```