stuff is  {'color': 'black', 1: 2, 'cat': 100, (1, 1): 'yes', 1.5: 3}
N is for …

• Nested Loops
  • All pairs, all pixels, all 2D structures

• None
  • Default value for functions if no return

• Newline
  • The "\n" in a line
The 21 Most Important Googlers You've Never Heard Of

Georges Harik and Noam Shazeer created the underlying data that led to AdSense

Harik and Shazeer spent years analyzing data on webpages, trying to understand clusters of words and how they worked together. The data they gather wound up being used by Google for its AdSense product, which analyzed webpages for words, and then stuck ads on them.
Announcements

• Assign 3 Transform due Today!
• Assign 4 is out, due Thursday, March 17

• APT 4 due this Thursday
• APT-5 out Thursday, due March 24

• No lab this week
• A few consulting hours during spring break
• Do not discuss Exam 1 or APT Quiz 1 with anyone until they are handed back
PFTD

• Dictionaries
• Solving Problems with Dictionaries
Problem: Given a name, what is their favorite ice cream?

• Assume you have a lot of people, over 1 million.

• How is the data stored?

• Assume we have parallel lists
  • students is list of names
  • icecream is list of corresponding favorite ice cream
Code might be

1  if name in students:
2       pos = students.index(name)  # find position of name
3       answer = icecream[pos]     # answer in same pos

If a billion names, this is not efficient
How does this code work?
Code might be

1  if name in students:
2       pos = students.index(name)  # find position of name
3       answer = icecream[pos]     # answer in same pos

If a billion names, this is not efficient

How does this code work?

  line 1 search through a billion names to say yes
  line 2 search through a billion names again!
  line 3 just one step access it!
How does search with `.index` work?

- **Parallel Lists**
  - Search for name first in students list
  - Use index location of name to find favorite ice cream

```
students = ['Astrachan', 'Sun', 'Rodger', 'Forbes']
0         1        2        3

icecream = ['Chocolate', 'Chocolate Chip', 'Chocolate Chip', 'Strawberry']
0       1       2       3
```
How does search with .index work?

• Parallel Lists
  • Search for name first in students list
  • Use index location of name to find favorite ice cream

Find Rodger’s favorite ice cream

students =
['Astrachan', 'Sun', 'Rodger', 'Forbes']
0 1 2 3

icecream =
['Chocolate', 'Chocolate Chip', 'Chocolate Chip', 'Strawberry']
0 1 2 3
How does search with `.index` work?

- Parallel Lists
  - Search for name first in students list
  - Use index location of name to find favorite ice cream

Find Rodger’s favorite ice cream

```
students = ['Astrachan', 'Sun', 'Rodger', 'Forbes']
```

```
icecream = ['Chocolate', 'Chocolate Chip', 'Chocolate Chip', 'Strawberry']
```

`students[2]` has index `2`, so `icecream[students[2]]` is `icecream[2]` which is `'Chocolate Chip'`. Therefore, Rodger’s favorite ice cream is Chocolate Chip.
How does search with .index work?

- Parallel Lists
  - Search for name first in students list
  - Use index location of name to find favorite ice cream

Find Rodger’s favorite ice cream

students = ['Astrachan', 'Sun', 'Rodger', 'Forbes']

icecream = ['Chocolate', 'Chocolate Chip', 'Chocolate Chip', 'Strawberry']

Find Rodger’s favorite ice cream:

students[2] = 'Rodger'

icecream[1] = 'Chocolate Chip'

Compsci 101, Spring 2022
How does search with .index work?

- Parallel Lists
  - Search for name first in students list
  - Use index location of name to find favorite ice cream

Find Rodger’s favorite ice cream

students =
['Astrachan', 'Sun', 'Rodger', 'Forbes']
0       1       2       3

icecream =
['Chocolate', 'Chocolate Chip', 'Chocolate Chip', 'Strawberry']
0       1       2       3

YES!
Use index location in other list

- **Parallel Lists**
  - Search for name first in students list
  - Use index location of name to find favorite ice cream

**Find Rodger’s favorite ice cream**

students = ['Astrachan', 'Sun', 'Rodger', 'Forbes']  
0        1        2        3

icecream = ['Chocolate', 'Chocolate Chip', 'Chocolate Chip', 'Strawberry']  
0        1        2        3

FOUND!
Use index location in other list

- Parallel Lists
  - Search for name first in students list
  - Use index location of name to find favorite ice cream

Find Rodger’s favorite ice cream

```python
students = ['Astrachan', 'Sun', 'Rodger', 'Forbes']
icecream = ['Chocolate', 'Chocolate Chip', 'Chocolate Chip', 'Strawberry']

# In same index position
```

3/1/22
Code was easy

• But for a lot of data could take a long time.

• Let’s see another way, dictionaries
How the Dictionary is made

• Using a dictionary is reasonably straight-forward
  • We will be clients, not implementers
  • Efficiency not a large concern in 101
  • Our goal is to just get stuff done 😊
What is a Dictionary?

• A collection of (key, value) pairs (abstract view)
  • Look up key, find the value

• Very, very fast: essentially index by key
  • For list `a[3]` takes same time as `a[3000]`

• For Dictionary: `d["cake"]`
  • Finding the value associated with "cake"
Dictionaries/Maps

• Dictionaries are another way of organizing data
• Dictionaries are sometimes called maps
• Keys and Values
  • Each key maps to a value
  • Some keys can map to the same value
  • Can change the value a key maps to
Example

- Each student could be mapped to their favorite ice cream flavor

```
Students

- Astrachan
- Sun
- Rodger
- Forbes

Ice Cream Flavors

- Chocolate
- Chocolate Chip
- Strawberry
```
How is dictionary different than a list?

- List – have to search for name first
- Dictionary – each key maps to a value
- Getting name (or key) is automatic! Fast!

![Diagram showing students and ice cream flavors as keys and values, respectively.](image-url)
Implementing a Dictionary/Map

Keys map to values

• Create Empty dictionary
  somemap = {}

• Put in a key and its value
  somemap[“Forbes”] = “Strawberry”

• Get a value for a dictionary
  value = somemap[“Forbes”]

• Change a value for a dictionary
  somemap[“Forbes’”] = “Chocolate”
Change Astrachan’s value
somemap[“Astrachan”] = Coffee Mocha
Change Astrachan’s value
somemap[“Astrachan”] = Coffee Mocha

Students

Ice Cream Flavors

Astrachan
Sun
Rodger
Forbes

Coffee Mocha
Chocolate Chip
Strawberry
Value could be a set or list

Students
- Astrachan
- Sun
- Rodger
- Forbes

Ice Cream Flavors
- Coffee Mocha
- Chocolate
- Vanilla
- Blueberry
- Chocolate Chip
- Blueberry
- Banana
- Strawberry
- Coffee Mocha
How to use a Dictionary

• Create: `d = {}`
  • `d = {'a': 10, 'b': 100}`
  • `d = dict([(a, 10), (b, 100)])`  
• Insert: `d[KEY] = VALUE`
• Update/Reassign: `d[KEY] = VALUE`
• Get a value (like list indexing): `d[KEY]`
• Key membership (not values): `KEY in d`
  • No membership check for values
Examples

```python
stuff = {}
print(stuff)
print(type(stuff))
stuff['color'] = 'black'
stuff[1] = 2
stuff['cat'] = 100
stuff[(1, 1)] = 'yes'
stuff[1.5] = 3
print(stuff)
```
Examples

```
stuff={}
print(stuff)  # {}
print(type(stuff))  # <class 'dict'>
stuff['color'] = 'black'
stuff[1] = 2
stuff['cat'] = 100
stuff[(1,1)] = 'yes'
stuff[1.5] = 3
print(stuff)
  # {'color': 'black', 1: 2, 'cat': 100, (1, 1): 'yes', 1.5: 3}
```
Examples

stuff is {'color': 'black', 1: 2, 'cat': 100, (1, 1): 'yes', 1.5: 3}

print(len(stuff))
stuff[3] = [6, 3, 2]

stuff[[4,7]] = 'go'
Examples

```python
stuff is {'color': 'black', 1: 2, 'cat': 100, (1, 1): 'yes', 1.5: 3}
```

```python
print(len(stuff))  # 5
stuff[3] = [6, 3, 2]
stuff is {'color': 'black', 1: 2, 'cat': 100, (1, 1): 'yes', 1.5: 3, 3: [6, 3, 2]}
stuff[[4, 7]] = 'go'  # ERROR!!!
```

Keys can only be immutable types!
Examples

d = {}  
d[‘color’] = ‘black’

d[‘color’] = ‘red’

d[‘red’] = ‘color’

r = d[d[‘red’]]

r = d[‘monkey’]
Examples

d={}  
\textbf{d is \{\}}

d['color'] = 'black'
\textbf{d is \{\textquoteleft color\textquoteleft:\textquoteleft black\textquoteleft\} \}}

d['color'] = 'red'
\textbf{d is \{\textquoteleft color\textquoteleft:\textquoteleft red\textquoteleft\} \}}

d['red'] = 'color'
\textbf{d is \{\textquoteleft color\textquoteleft:\textquoteleft red\textquoteleft,  
\textquoteleft red\textquoteleft:\textquoteleft color\textquoteleft\} \}}

r = d[d['red']]
r is 'red'

r = d['monkey']
ERROR!!!!!!
Examples

d = {'a': 'cat', 'e': 'dog'}

'dog' in d
'a' in d
'pig' in d
Examples

d = {'a':'cat', 'e':'dog'}

'dog' in d     False
'a' in d       True
'pig' in d     False

in checks if ‘pig’ is a key in d
WOTO-1 Dictionaries
More on Dictionary

- Like lists, but with keys
- KEY – immutable type, unique within dictionary
- VALUE – any type, not unique within dictionary
- Dictionary is unordered collection of (KEY, VALUE) pairs
More on using a Dictionary/Map

• Get all the keys (as a list)
  • `listKeys = somemap.keys()`

• Get all the values (as a list)
  • `listValues = somemap.values()`

• Other methods
  • `clear` – empty dictionary
  • `items` – return (key,value) pairs
  • `iteritems` – return (key,value) pairs more efficiently, `iterator` – *must use with for*
  • `update` – update with another dictionary

*Oops, iteritems was only in Python 2, use items*
Examples

d = {'a':4, 'e':3, 'b':4 }

v = d.values()
k = d.keys()
p = d.items()

for t in d.items():
    print(t)
Examples

d = {'a':4, 'e': 3, 'b':4 }

v = d.values()  # v is [4, 3, 4]
k = d.keys()    # k is ['a','e','b']
p = d.items()   # p is [('a',4), ('e',3), ('b',4)]

for t in d.items():
    print(t)
    # ('a', 4)
    # ('e', 3)
    # ('b', 4)
Problem

• Given a list of names of people who ate at a restaurant, who ate there the most?
• A name appears more than once if they ate their more than once

• names = ['Sarah', 'Beth', 'Sarah', 'Purnima', 'Beth', 'Beth', 'Purnima']
WOTO-2 Problem Solving
d = {}

for word in names:
    if word not in d:
        d[word] = 1
    else:
        count = d[word]
        d[word] = count + 1

print("d:", d)
Finding Largest Value in d

```
val = 0
for key in d:
    if d[key] > val:
        val = d[key]
print("val:", val)
```

Alternative:

```
maxval = max(d.values())
print("maxval:", maxval)
```
Find key goes with largest value

```python
maxname = ""
for key in d:
    if d[key] == maxval:
        maxname = key
print("maxname", maxname)
```
**APT: SandwichBar Search**

**Problem Statement**

It's time to get something to eat and I've come across a sandwich bar. Like most people, I prefer certain types of sandwiches. In fact, I keep a list of the types of sandwiches I like.

The sandwich bar has certain ingredients available. I will list the types of sandwiches I like in order of preference and buy the first sandwich the bar can make for me. In order for the bar to make a sandwich for me, it must include all of the ingredients I desire.

Given `available`, a list of Strings/ingredients the sandwich bar can use, and a `orders`, a list of Strings that represent the types of sandwiches I like, in order of preference (most preferred first), return the 0-based index of the sandwich I will buy. Each element of `orders` represents one type of sandwich I like as a space-separated list of ingredients in the sandwich. If the bar can make no sandwiches I like, return -1.
Sandwich Bar Example

- available = [ "cheese", "cheese", "cheese", "tomato" ]
- orders = [ "ham ham ham", "water", "pork", "bread", "cheese tomato cheese", "beef" ]
Sandwich Bar Example

• available = [ "cheese", "cheese", "cheese", "tomato" ]
• orders = [ "ham ham ham", "water", "pork", "bread", "cheese tomato cheese", "beef" ]

• Returns 4
• Can make “cheese tomato cheese”

• Ignore any duplicates!
WOTO-3 SandwichBar
Another Trip to the SandwichBar

• Use sets to solve this!

• Idea

```python
for index in range(len(orders)):
    if canmake(orders[index], available):
        return index
```

• You would need to write the function canmake
• What type does it return?
• What set operation could you use?
Given two lists A and B

- Determine if all elements in A are also in B
  - Examine each element in A
    - If not in B? False
    - After examining all elements? True

- Think: Could we use sets instead?
Given two sets A and B

• Determine if all elements in A are also in B
  • $\text{if } \text{len} (A \cap B) == \text{len} (A)$
  • $\text{if } \text{len} (A - B) == 0$
Assignment 4: Guess Word

• We give you most of the functions to implement
  • Partially for testing, partially for guiding you
• But still more open ended than prior assignments
• If the doc does not tell you what to do:
  • Your chance to decide on your own!
    • Okay to get it wrong on the first try
  • Discuss with TAs and friends, brainstorm!