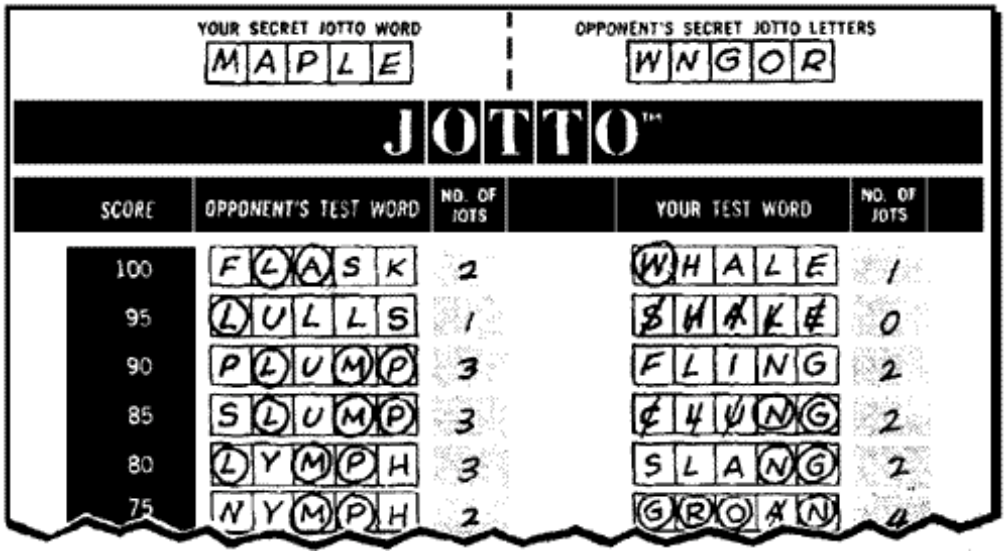
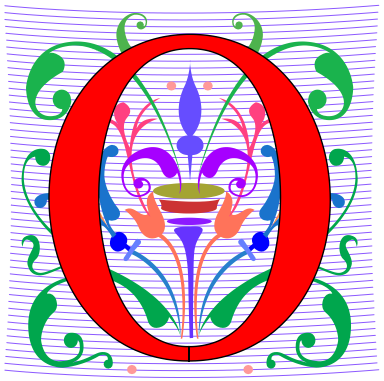
CompSci 101
Problem Solving



Susan Rodger

March 9, 2023

O is for …



- **Open Source**
  - Copyright meets the Creative Commons
- **Object Oriented**
  - Using classes and more in programming
- **Occam's Razor**
  - Not just compsci. Simple is good

The Tech Twins

- **Troy and Travis Nunnally**
- **Between them: 2 master's and 1 doctorate from Georgia Tech**
- **Cofounders of Brain Rain Solutions**
  - Augmented-reality
  - Internet-of-things
- **Applied machine learning**

Troy: "My advice would be to stay consistent. Always think persistently and consistently about learning a particular craft."

Travis: "I think that you have to be passionate and find something that you simply love and enjoy. Not only find that thing — but actually be a lifelong learner around that."

https://www.wired.com/story/what-atlanta-can-teach-tech-about-cultivating-black-talent/

Announcements

- **Assign 4 GuessWord due Thursday, March 23**
  - Sakai Assignment Quiz due **TUESDAY, March 21**
- **APT-5 and Assign 5 out, March 23**
  - Will talk about Thursday after spring break
- **No Lab Friday!**
- **APT Quiz 2 on March 30-Apr 3**

- **Do not discuss Exam 2 until handed back!**

PFTD

- **Problem Solving**
- **Jotto game**

Problem Solving – What to use

- **Do you need to loop over anything?**
  - Do you need the index of the item?

For loop or while loop?

index loop?

for item in range(len(alist))

Example: do you need to know the index position of the longest word in the list

Problem Solving – What to use

- **Do you need to loop over anything?**
  - Do you need the index of the item?
- **Do you need to make a decision?**
- **Do you need unique elements?**

Use an if?

Do you need elif or a second if?

Use a set?

Do you need to keep the order of the elements? Do you need the original list AND a separate set of the elements just to know how many unique elements?

Problem Solving – What to use

- **Do you need to loop over anything?**
  - Do you need the index of the item?
- **Do you need to make a decision?**
- **Do you need unique elements?**
- **Are you working with two groups of things?**
  - Are they parallel lists?
  - Are you comparing elements in some way with two groups of elements

Need an index loop to access two elements in different lists at the same position

Do you want to put both into sets and use a set operation?

Problem Solving – What to use

- **Do you need to loop over anything?**
  - Do you need the index of the item?
- **Do you need to make a decision?**
- **Do you need unique elements?**
- **Are you working with two groups of things?**
  - Are they parallel lists?
  - Are you comparing elements in some way with two groups of elements

Now let's look at an APT from APT-5

Sandwich Bar

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.

Class

```
filename: SandwichBar.py

def whichOrder(available, orders):
    """
    return zero-based index of first
    sandwich in orders, list of strings
    that can be made from ingredients
    in available, list of strings
    """

    # you write code here
```

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-1 SandwichBar
http://bit.ly/101s23-0309-1

Another Trip to the SandwichBar

- **Use sets to solve this!**

- **Idea**

```
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?**

```
def canMake(order, available):
    sandwichSet = set(order.split())
    availableSet = set(available)
    intersection = availableSet & sandwichSet
    if len(intersection) == len(sandwichSet):
        return True
    return False
```

order is "cheese tomato cheese"

["cheese", "tomato", "cheese"]

sandwichSet is {"cheese", "tomato"}

available is ["cheese", "cheese","onion", "cheese", "tomato"]

availableSet is {"cheese", "onion", "tomato" }

intersection is ["cheese", "tomato"]

True

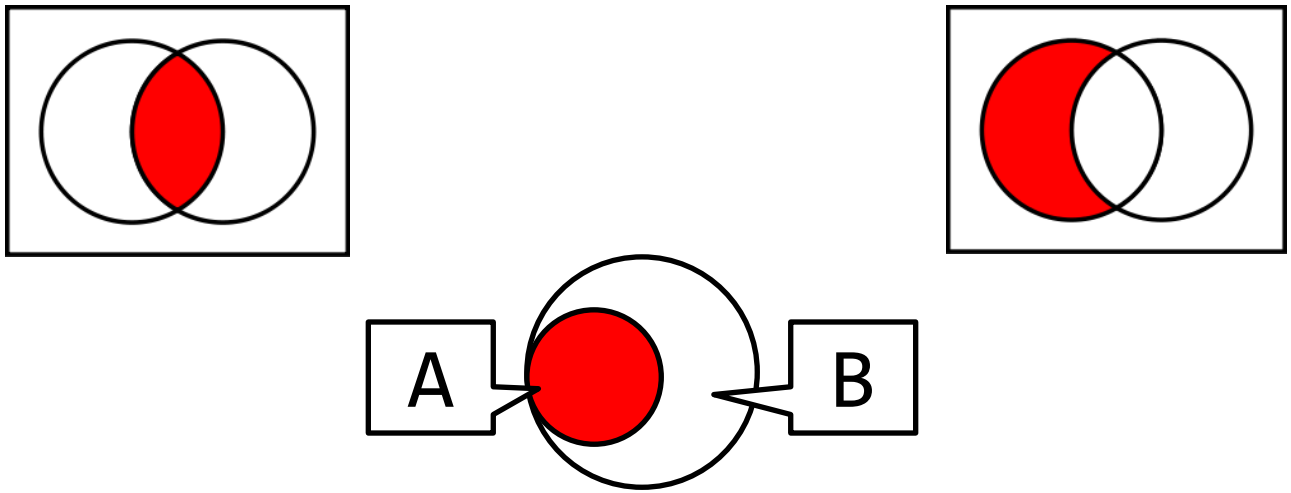
Given two lists A and B

- **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?**

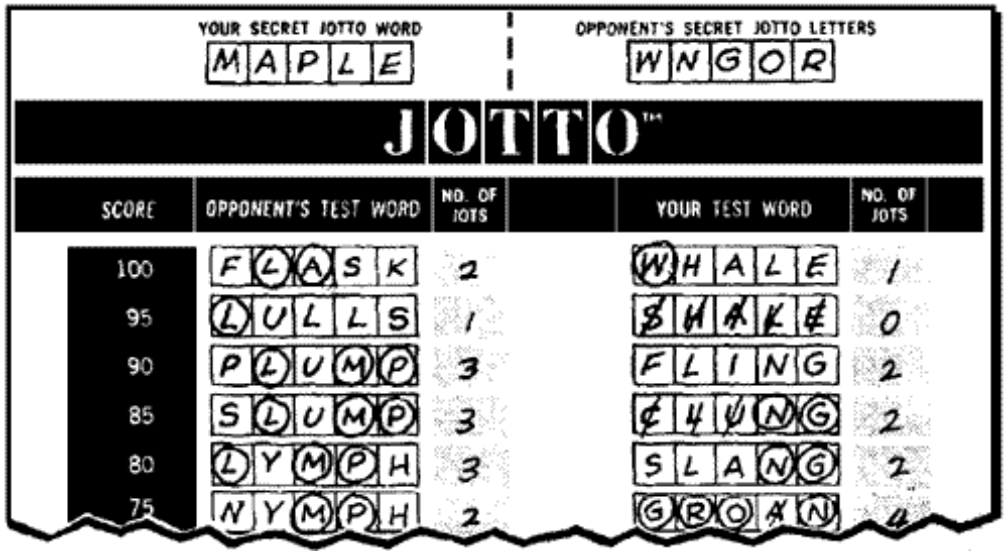
Given two sets A and B

- **Determine if all elements in A are also in B**
  - `if len(A & B) == len(A)`
  - `if len(A - B) == 0`



Jotto: Game similar to GuessWord

- https://en.wikipedia.org/wiki/Jotto
- http://jotto.augiehill.com/single.jsp
- **No letters repeat – have to agree on this**
- **Shall we play a game?**



Write program where Computer Guesses Your Word

- **You give the computer a word to guess, called wordToGuess**
- **Computer does brute force, no thinking or eliminating letters**
  - It picks a word at random
  - Calculates how many letters in common with wordToGuess, say x letters
  - Only keep words with x letters in common
  - Repeats until guesses the word

We will build useful functions to use to build the game

WOTO-2 Approaching Implementation http://bit.ly/101s23-0309-2

- **What is needed?**
- **What order should the code do things?**

Start with Blank Screen

1. **Computer gets a list of words**
2. **Computer chooses a word at random**
3. **User/player enters # letters in common**
4. **Only keep words with that # in common**

Initialization

Loop

Iterative Programming!

- **Start with a task**
- **Implement only that task**
- **Write some code to check that the code works**
  - Run and debug until it works
- **Repeat**

Should you write all the code first and then run it?

"Debugging is twice as hard as writing the code in the first place. Therefore, If you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."
- Brian Kernighan (original Unix contributor)

SimpleJotto.py

- **We have a file of five letter words: `kwords5.txt`**
  - Would you like to play a game?

- **Let's start! Simple version that sort of works ☺**

Jotto Step 1

- **Read the file kwords5.txt**
  - Read the file kwords5.txt
  - Don't continue until we know this works

- **Let’s go code up!**

Jotto Step 2

- **Pick a word at random**
  - Pick a word at random, show the user
  - Don't continue until we know this works

Jotto Step 3

- **Get number of letters in common**
  - Get # letters in common, *do something?*
  - Don't continue until we know this works

WOTO-3 Jotto Two Functions
http://bit.ly/101s23-0309-3

Let's code those up!

- **chooseAWord**
- **commonCount**

updateWordList

- **The next function updateWordList can be done with a list comprehension**

Try writing updateWordList

- **Given:**
  - words – a list of words
  - nextWord – the word the computer just guessed
  - numInCommon – the number of letters in common between nextWord and wordToGuess, which is the word the user wants the computer to guess
- **Return**
  - A new list that has only the words from the list words that have the same number of letters in common that nextWord has with wordToGuess

Now put together game

- **wordToGuess is #Ask user for word to guess**
- **Read in list of word from file**
- **While not found word**
  - nextWord is #choose a word at random
  - Get number of letters in common between nextWord and wordToGuess
  - Update word list to keep only those with same number of letters in common as nextWord
  - Check to see if nextWord is wordToGuess

Think about how to put the game together with all these pieces

Next slides show how to put the game together

Code up getWordList and chooseAWord

```
 9  def getWordList(filename):
10      ret = []
11      f = open(filename)
12      for line in f:
13          ret.append(line.strip())
14      f.close()
15      return ret
16
17  def chooseAWord(words):
18      return random.choice(words)
```

Code up updateWordList and commonCount

```
20  def handleUserInput():
21      wordToGuess = input("Guess a word with 5 letters: ")
22      return wordToGuess
23
24  def updateWordList(words, numInCommon, userword):
25      return [w for w in words if numInCommon == commonCount(w, userword)]
26
27  def commonCount(word1, word2):
28      # assumes words don't have duplicate letters
29      set1 = set(list(word1))
30      set2 = set(list(word2))
31      #print(set1, set2)
32      return len(set1 & set2)
```

playGame First part

```
34  def playGame():
35      won = False
36      wordToGuess = handleUserInput()
37      numTries = 0
38
39      # Initialize
40      ## Read the file kwords5.txt
41      words = getWordList('kwords5.txt')
```

playGame loop

```
42      while (not won and numTries < 30 ):
43      # Loop
44          ## Pick a word at random
45          nextWord = chooseAWord(words)
46          ## Get number of letters in common
47          numInCommon = commonCount(wordToGuess, nextWord)
48          ## Only keep words with that number in common
49          words = updateWordList(words, numInCommon, nextWord)
50          print("next word is: ", nextWord)
51          print("num letters in common is: ", numInCommon)
52          print("words remaining: ", len(words))
53
54          #check stopping condition
55          if nextWord == wordToGuess:
56              won = True
57          numTries += 1
58      return won
```