Programming Idioms and Ideas: PII

- Two kinds of loops: by-element, by-index
  - Underneath often by index, e.g., problems when removing from a list while iterating
- Two kinds of structured data: strings and lists
  - Soon to add sets, tuples, dictionaries
- Today: Strings, Lists, Sets, Oh My!

Solving Problems, Transforming Data

- Consider the Common APT, useful in the interactive game Jotto you’ll write
  - "seats", "tease" → 4
  - "seats", "meaty" → 3
  - "seats", "stats" → 4
- Ideas: loop over word1, cross out in word2
  - 's', "*tats" 1  does it matter which 's'? 
  - 'e', "*tats" 1  can you replace 's' with '*'? 
  - 'a', "*t*ts" 2

Ideas into code: thinking about loops

- As you loop over 's', 't' ... find and "mark"
  - You can look up the 's' in word2, find index
  - You can use index in word1 and in word2

```python
for ch in word1:
    dex = word2.find(ch)
    if dex != -1:
        for k in range(len(word1)):
            if ch in word1[k]:
                dex = word2.find(word1[k])
                if dex != -1:
                    for ch in list1:
                        if ch in list2:
                            dex = list2.index(ch)
                            list2[dex] = '*'
```

Using lists rather than strings

- Strings are immutable, can create new ones, but cannot change, lists are mutable!
  - Using a list instead makes code easier, unfortunately list has no find, only index

```python
for ch in word1:
    dex = word2.find(ch)
    if dex != -1:
        word2 = word2[:dex] + '*' + word2[dex+1:]
```
Which loop is right? Index or Element?

- It Depends! (always a good answer)
  - If you're going to always use one loop, to avoid having to make a choice, which one to use?
  - Can you go simply from index to element?
  - Can you go simply from element to index?

Eating Well or Good Eating: APT

- First think about solving this by hand...
  - In translating to Python, what's easy? Harder?
  - Can we find diners who eat at Elmo's easily?

- Structure
  - Strings and lists
  - Using .split(...)

Eliminating Duplicates

- Could process a list, avoid double counting by checking, but much easier solution: set!
  - Part of Python and many other languages
  - Typically implemented to be very efficient in determining membership

- Set – collection like list, but not indexable
  - Can .add(), .remove(),
  - Can iterate, cannot slice
  - Can if foo in coll: where coll is set or list

Thinking about sets

- Use list.append(x), use set.add(x)
  - If already in set, nothing happens
- Can create set from a list all at once
  
  uni = set([1,2,3,1,2,3,1,2,3,1,1,2,2,3,3])

- Later we'll see union |, intersection &,
  difference - and other operations ^ TBDiscussed
Question Interlude


Summary (from wikibooks)

- set1 = set()
- set1.add("cat")
- set1.update(["dog", "mouse"])  # Add several members
- set1.remove("cat")  # Remove a member – error not there
- for item in set1:  # Iteration or “for each element”
- len(set1)
- isempty = len(set1) == 0  # Test for emptiness
- set1 = set(["cat", "dog"])  # Initialize set from a list
- set3 = set1 & set2  # Intersection
- set4 = set1 | set2  # Union
- set5 = set1 - set3  # Set difference
- set6 = set1 ^ set2  # Symmetric difference (elements in either set but not both)
- Is Subset: set1 <= set2  # Subset test
- Is Superset: set1 >= set2  # Superset test
- set7 = set1.copy()  # Shallow copy (copies set, not elts)
- set8.clear()  # Clear, empty, erase

Indexes within indexes, loop in loops

- Very useful in solving two-dimensional and other problems
  - Lists are one-dimensional, for example

List in a list and loop in a loop

- z = [[1,2,3], [4,5,6], [7,8,9]]
  - for x in z: what is type of x?
- Use one loop inside another to access both
  - Could be list of student info as well

for x in z:
    for y in x:
        #what type is y?
Looping with Indexes

- How to understand a loop-in-a-loop?
  - What changes in the inner loop

```python
def doublenest(n):
    for i in range(n):
        for j in range(n):
            print i, j
```

```python
def doublenest2(n):
    for i in range(n):
        for j in range(i+1,n):
            print i, j
```

Create "couples"

- A name is fixed as the inner loop executes
  - See output to reinforce this idea

```python
A = ['sam', 'lou', 'chris']
B = ['terry', 'brook', 'val']
for aname in A:
    for bname in B:
        print aname,",", bname
```

Midterm and what it means

- Working to succeed can lead to success
  - Your score isn't as important as why and where you lost points
  - We will provide a path and approach for those who want to rethink approach to 101

- Is it better to get 30% of everything, or 70% of 50% of what we cover?