Announcements

• Reading and RQ 11 due next time
• APT 4 out due next Thursday

• Today:
  – More on Lists
  – Solving APTs with: while True
  – Coming – more ways to process data
  – Exam 1 back next time

Problem: Find the location of first adjacent duplicate word

• “This is a story story about a girl with a red hood…”

• Return 3 as the location of the first word that has a duplicate adjacent word (start at 0)

Seven Steps – Step 1 work example by hand

• This is a story story about a girl ...
• This is a story story about a girl ...
• This is a story story about a girl ...
• This is a story story about a girl ...

• Step 2 – write down what you did
• Step 3 – generalize, special cases
• Step 4 – work another example
Seven Steps – Step 1 work example by hand

• *This is a story story about a girl* ...
• *This is a story story about a girl* ...
• *This is a story story about a girl* ...
• *This is a story story about a girl* ...

Step 2 – write down what you did
Step 3 – generalize, special cases
Step 4 – work another example

Bitly/101f17-1012-1

```python
def positionDuplicate(phrase):
    words = phrase.split()
    if len(words) < 2:
        return -1
    pos = 0
    while True:
        pos = pos + 1
        return pos
```

APTs solved in a similar way

• Pikachu
• NameGroup

APTs solved in a similar way

• NameGroup
• Pikachu
APT: Pikachu

**Problem Statement**

Pikachu is a well-known character in the Pokemon anime series. Pikachu can speak, but only 3 syllables: "pi", "ka", and "chu". Therefore, Pikachu can only pronounce strings that can be created as a concatenation of one or more syllables he can pronounce. For example, he can pronounce the words "pikapi" and "pikachu".

You are given a String word. Your task is to check whether Pikachu can pronounce the string. If the string can be produced by concatenating copies of the strings "pi", "ka", and "chu", return "yes" (quotes for clarity). Otherwise, return "no".

<table>
<thead>
<tr>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename: Pikachu.py</td>
</tr>
<tr>
<td>def check(word):</td>
</tr>
<tr>
<td>***</td>
</tr>
<tr>
<td>return String based on parameter</td>
</tr>
<tr>
<td>word, a String</td>
</tr>
<tr>
<td>***</td>
</tr>
<tr>
<td># you write code here</td>
</tr>
</tbody>
</table>

**Good:**

Try: pikapi

**NOT VALID**

Try: pichukaru

Good: pichuka

---

APT: Pikachu

- What is the iteration?
- What are the choices: *pi ka chu*

**pichukarunkapi**

Try:

**Good:**

---

APT: Pikachu

- What is the iteration?
- What are the choices: *pi ka chu*

**pichukarunkapi**

Try:

**Good:**

---

APT NameGroup

**Problem Statement**

You are given a string of names called `names`, two additional names, called `one` and `two`, and a number called `space`. Your task is to figure out if the two names are in the string names with one first and then two and exactly "space" names between them. Neither name one or two can be in the names between them, and they must be different names. A `namegroup` is then a string of the name one, followed by the names in the space and then name two, all separated by a space. If this is true, return the first such namegroup string found. Otherwise return an empty string.

<table>
<thead>
<tr>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename: NameGroup.py</td>
</tr>
<tr>
<td>def calculate(names, one, two, space):</td>
</tr>
<tr>
<td>***</td>
</tr>
<tr>
<td>names is a string of names separated by blanks, one and two are names, returns a string of names that starts with one and ends with two and has &quot;space&quot; names between them. Neither one nor two can be in the names between them. Neither one nor two can be in the spacing. one cannot equal two. Otherwise returns &quot;&quot;</td>
</tr>
<tr>
<td>***</td>
</tr>
<tr>
<td># you write code here</td>
</tr>
</tbody>
</table>
7 steps – Step 1 work example calculate(names, “joe”, “bo”, 2)

- moe joe sue bo joe po fa bo sue
- moe joe sue bo joe po fa bo sue

YES NO

- moe joe sue bo joe po fa bo sue

YES

- moe joe sue bo joe po fa bo sue

YES

- moe joe sue bo joe po fa bo sue

YES

- moe joe sue bo joe po fa bo sue

YES

Found!
Alice programming language
alice.org, Alice version 2.4

Nested Loop

Fair Ride – Octopus

Wac-A-Mole
More on lists

```python
sounds = ['fa', 'la', 'ti']
sounds2 = sounds * 2
sounds[1] = 'so'

sounds = ['fa', 'la', 'ti']
sounds2 = [sounds] * 2
sounds[1] = 'so'

x = ['a', 'b', 'c']
y = x
z = y
w = y[:]
```

Creating a list

- Given a list of numbers, create a second list of every number squared.

```python
nums = [8, 3, 5, 4, 1]
sqnums = []
for v in nums:
    sqnums.append(v*v)
print sqnums

[64, 9, 25, 16, 1]
```

More on List operations

- Previous page
  - nameOfList “dot” function (parameter)
    ```python
    sqnums.append(v*v)
    ```
  - See list operations on next page
- Mutator vs hybrid vs return
  - Mutator changes the list (no return value)
  - Hybrid changes list and returns value
  - Return – returns value, no change to list

```python
a = [5, 6, 7]
b = [2, 3]
c = a + b
d = [a] + [b]
```
List operations from book

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>append</td>
<td>item</td>
<td>mutator</td>
<td>Adds a new item to the end of a list</td>
</tr>
<tr>
<td>insert</td>
<td>position, item</td>
<td>mutator</td>
<td>Inserts a new item at the position given</td>
</tr>
<tr>
<td>pop</td>
<td>none</td>
<td>hybrid</td>
<td>Removes and returns the last item</td>
</tr>
<tr>
<td>pop</td>
<td>position</td>
<td>hybrid</td>
<td>Removes and returns the item at position</td>
</tr>
<tr>
<td>sort</td>
<td>none</td>
<td>mutator</td>
<td>Modifies a list to be sorted</td>
</tr>
<tr>
<td>reverse</td>
<td>none</td>
<td>mutator</td>
<td>Modifies a list to be in reverse order</td>
</tr>
<tr>
<td>index</td>
<td>item</td>
<td>return idx</td>
<td>Returns the position of first occurrence of item</td>
</tr>
<tr>
<td>count</td>
<td>item</td>
<td>return ct</td>
<td>Returns the number of occurrences of item</td>
</tr>
<tr>
<td>remove</td>
<td>item</td>
<td>mutator</td>
<td>Removes the first occurrence of item</td>
</tr>
</tbody>
</table>

Problem

- Remove all negative numbers from list
  \([4, -2, 5, 6, -3]\)  \(\rightarrow [4, 5, 6]\)

- Two ways
  1) return a new list with all negative numbers removed
  2) Modify a list to remove negative numbers

```
def removeNegatives(numberlist):
    answer = []
    for num in numberlist:
        if num >= 0:
            answer.append(num)
    return answer

somenums = [3, -1, 8, -5, -2, 6, 7]
onegs = removeNegatives(somenums)
```

```
def removeNegatives2(numberlist):
    for x in range(len(numberlist)):
        value = numberlist[x]
        if value < 0:
            numberlist.pop(x)

somenums = [3, -1, 8, -5, -2, 6, 7]
removeNegatives2(somenums)
```
def removeNegatives3(numberlist):
    pos = 0;
    while (True):
        if pos >= len(numberlist):
            break
        value = numberlist[pos]
        if value < 0:
            numberlist.pop(pos)
        pos = pos + 1

somenums = [3, -1, 8, -5, -2, 6, 7]
removeNegatives3(somenums)

Solving problems – APT MorseLikeCode

• Compare find vs index
  – find with string – returns -1 when not found
  – index with list – CRASHES if not there!
  – You can’t say: pos = alist.index(“…”)
  – Instead: if “…” in alist:
    pos = alist.index(“…”)

• How to get started?