The Power of Collaboration:
Ge Wang, Duke Prof. at Stanford

• Duke 2000: Music and Computer Science
  • https://www.stanforddaily.com/2016/03/09/qa-with-ge-wang-
    father-of-stanford-laptop-orchestra/
  • http://www.youtube.com/watch?v=ADEHmkL3HBg

• About Design in Compsci 308

  Our investment into a huge and meticulous design process was a huge
  factor in making later progress. 35000+ lines of code / design / documentation
  gave us a project we were all very happy and proud to be a part of.

Announcements

• Assign 6 Recommender due TODAY!
• APT-7, due Tuesday
• Assign 7 due April 26
  • Can be turned in by April 30 with NO PENALTY
• APT Quiz 2 posted on APT page – for practice
• Lab 11 Friday – due prelab before going

• Final Exam – Thurs, May 4, 9am
Interested in being a UTA?

• Enjoy Compsci101?
• Would like to help others learn it?

• Consider applying to join the team!
• https://www.cs.duke.edu/undergrad/uta

• Apply soon

Assignment 7:
More samples from previous semesters

A Story – One Eternity Later

APT Due
Haiku – From Previous Semester

Turtles and Pythons
But We Are Not at the Zoo
We Are in CompSci

PFTD

• Review Recursion
• Modules and exceptions
• An APT

Haiku – From Previous Semester

Ugh Syntax Error
Did I Forget a Colon?
Nope. Parentheses.

Review: Recursion Summary

• Make Simpler or smaller calls
  • Call a clone of itself with different input
• Must have a base case when no recursive call can be made
  • Example - The last folder in the folder hierarchy will not have any subfolders. It can only have files. That forms the base case
  • This is the way out of recursion!
Problem: is a number in a list?

- Is 5 in [7, 5, 6, 8]?
- Is 8 in [5, [ [7,4], 9, [3, 4]], [4, [5, [2, [8, 1], 4 ], ]], 5 ]?

Possible solution

```python
def isItInList(alist, num):
    for item in alist:
        if type(item) == type([]): # is a list
            return isItInList(item, num)
        else: # type is number
            if item == num:
                return 'yes'
    return 'no'
```

- Doesn’t work! Consider 2 and [3, [6,7], 8, [2, 7]]
Possible solution

```python
def isItInList(alist, num):
    for item in alist:
        if type(item) == type([]):  # is a list
            return isItInList(item, num)
        else:  # type is number
            if item == num:
                return 'yes'
    return 'no'
```

• Doesn’t work! Consider 2 and [3, [6,7], 8, [2, 7] ]

Possible Solution 2

```python
def isItInList2(alist, num):
    for item in alist:
        if type(item) == type([]):  # is a list
            if isItInList2(item, num) == 'yes':
                return 'yes'
        else:  # type is number
            if item == num:
                return 'yes'
    return 'no'
```

• Works! Consider 2 and [3, [6,7], 8, [2, 7] ]
Possible Solution 2

```python
def isItInList2(alist, num):
    for item in alist:
        if type(item) == type([]):  # is a list
            if isItInList2(item, num) == 'yes':
                return 'yes'
        else:  # type is number
            if item == num:
                return 'yes'
    return 'no'
```

- Works! Consider 2 and [3, [6,7], 8, [2, 7] ]

Problem: is a number in a list?

- Is 5 in [7, 5, 6, 8] ?
- Is 8 in [5, [ [7,4], 9, [3, 4]], [4, [5, [2, [8, 1], 4], ]], 5 ] ?

Returns 'yes'
Revisit the APT Bagels Recursively

filename: Bagels.py

def bagelCount(orders):
   """
   return number of bagels needed to fulfill the orders in integer list parameter orders
   """

1. orders = [1, 3, 5, 7]
   Returns: 16
   No order is for more than a dozen, return the total of all orders.

2. orders = [11, 22, 33, 44, 55]
   Returns: 175 since 11 + (22+1) + (33+2) + (44+3) + (55+4) = 175

Why use modules?

- Module – Python file (.py file)
- Can have several modules work together
- Easier to organize code
- Easier to reuse code
- Easier to change code
  - As long as the “what” is the same, the “how” can change
  - Ex: sorted(...), one function many sorting algorithms
Modules for Creating

- **“MadLibs” → Tag-a-Story**
  - User chooses template
  - Computer fills everything in

From <noun> to story

In lecture I saw a <color> <noun>
For lunch I had a <adjective> <food>
The day ended with seeing a <animal> <verb> in <place>

Demo

- Run storyline.py
- Show Lecture template
- Show Haiku’s
- Make modifications

Let's create/modify a story

- Choose a template or make a new one
  - We'll choose lecturetemplate.txt first
- Add a new category/replacement
  - We'll choose number and list some choices
- Run the program and test our modifications
  - Randomized, hard to test, but doable
Main Parts (3 modules) for tag-a-story

• Put everything together, the template and words
  • Storyline.py

• Loading and handling user choosing templates
  • TemplateChooser.py

• Loading and picking the word for a given tag
  • Replacements.py

Creating a story

• Main steps in Storyline.py
  • Get template – use module TemplateChooser
  • Go through template
    • Get words for a tag – use module Replacements
    • Replace tag with word

• Using modules
  • Assume they work
  • Only care **what** they do, not **how** (abstraction!)

Main Parts (3 modules) for tag-a-story

• Put everything together, the template and words
  • Storyline.py

• Loading and handling user choosing templates
  • TemplateChooser.py

• Loading and picking the word for a given tag
  • Replacements.py

Modules in Action: makeStory() is in Storyline.py

• How can we access TemplateChooser functions?
  • import and access as shown

```python
41 def makeStory():
42     """
43     let user make a choice of
44     available templates and print
45     the story from the chosen template
46     """
47     lines = TemplateChooser.getTemplateLines("templates")
48     st = linesToStory(lines)
49     print(st)
```
Modules in Action:
makeStory() is in Storyline.py

• How can we access TemplateChooser functions?
  • import and access as shown

```python
def makeStory():
    
    let user make a choice of
    available templates and print
    the story from the chosen template

    lines = TemplateChooser.getTemplateLines("templates")
    st = linesToStory(lines)
    print(st)
```

Modules in Action:
linesToStory() is in Storyline.py

• We call doWord() – does replacements for words

```python
def linesToStory(lines):
    
    lines is a list of strings, each a line from a template file
    Return a string based on substituting
    for each <tag> in each line

    story = 
    for line in lines:
        st = 
        for word in line.split():
            st += doWord(word) + " 
        story += st.strip() + " 

    return story
```

1. How can we access TemplateChooser functions?
2. import and access as shown

```python
def makeStory():
    
    let user make a choice of
    available templates and print
    the story from the chosen template

    lines = TemplateChooser.getTemplateLines("templates")
    st = linesToStory(lines)
    print(st)
```

A function in the file: TemplateChooser.py

A function in this file: doWord
no dot before it
• What does getReplacement do?
  • How does getReplacement do it?

A function in the file: Replacements.py

• Put everything together, the template and words
  • Storyline.py

• Loading and handling user choosing templates
  • TemplateChooser.py

• Loading and picking the word for a given tag
  • Replacements.py
Another module TemplateChooser.py

• Get template
  • TemplateChooser.getTemplateLines(DIR)
  • What:
    • From the templates in the directory DIR (type: str)
    • Return a list of strings, where each element is a line from one of the templates in DIR

• Word for a tag
  • Replacements.getReplacement(TAG)
  • What:
    • Return a random word that matches TAG (type: str)

Where is it called from?

• In module Storyline.py, function makestory

```python
lines = TemplateChooser.getTemplateLines("templates")
```

• Where templates is a folder with three templates:

TemplateChooser.py Steps

• List all templates in the folder
• Get user input that chooses one
• Load that template
• Return as list of strings

TemplateChooser.py Steps

• List all templates in the folder
  • pathlib Library
• Get user input that chooses one
  • Handle bad input → try...except
• Load that template
  • Open file, .readlines()
• Return as list of strings
These Steps in Code
getTemplateLines in TemplateChooser.py
• Read directory of templates, convert to dictionary
• Let user choose one, open and return it

```python
def getTemplateLines(dirname):
    """
    dirname is a string that's the name of a folder
    Prompt user for files in folder, allow user to choose, and return the lines read from file
    """
    d = dirToDictionary(dirname)
    lines = chooseOne(d)
    return lines
```

Creating User Menu
dirToDictionary in TemplateChooser.py
• What does this function return? What type?

```python
def dirToDictionary(dirname):
    """
    d = {}
    index = 0
    for one in pathlib.Path(dirname).iterdir():
        d[index] = one
        # print(type(one))
        index += 1
    return d
```

Creating User Menu
dirToDictionary in TemplateChooser.py
• What does this function return? What type?

Folder in Pycharm

Output:

```
D: C:\Users\Susan\Py
  \tagreplacements
  \templates
    \haiku.txt
    \labtemplate.txt
    \lecturetemplate.txt
    \Replacements.py
```

```
C:\Users\Susan\AppData\Local\Microsoft\Windows\AppData\Local\Packages\Microsoft.Windows.ContentDeliveryManager_cw5n1h2txf7cn\LocalState\ApplicationData\Microsoft\Edge\Application\default\Downloads\210408 haiku.txt
0 haiku.txt
the slimy bathtub
reminded them of Africa
chartreuse squeaky brown
```
pathlib Library

- **Path:**
  "rodger/Pycharm/cps101/lab11/temp/haiku.txt"

- The **pathlib** library is more recent/Python3
  - Simpler, easier to use than functions from `os`

- Handles **domain specifics**!
  - Doesn’t matter if on Windows, Mac, etc.
  - We worry about the `what`, it handles the `how`

Understanding the Unknown chooseOne in TemplateChooser.py

- **We will return to this, but analyze parts now**
  - What's familiar? What's not familiar …

```python
def chooseOne(d):
    """
    ..."
    while True:
        for key in sorted(d.keys()):
            print(f"{key}\s" % (key, d[key].parts[-1]))
        print("-----")
        st = input("choose one> ")
        try:
            val = int(st)
            if 0 <= val and val < len(d):
                return reader(d[val])
        except ValueError:
            print("please enter a number")
```

Python exceptions

- **What should you do if you prompt user for a number and they enter "one"**
  - Test to see if it has digits?

- **Use exceptions with `try`: and `except`:**
  - See code in function `chooseOne` from `TemplateChooser.py`

• Path:
  "rodger/Pycharm/cps101/lab11/temp/haiku.txt"

• `pathlib.Path(DIR).iterdir()`
  - Returns iterable of `Path` objects representing each “thing” in the directory `DIR`

• Path object’s `.parts` – tuple of strings, each element is a piece of a filename’s path

• Pathlib Library cont.
Handling Exceptions

- **What happens:**
  
  ```python
  st = input("choose one> ")
  try:
      val = int(st)
      if 0 <= val and val < len(d):
          return reader(d[val])
  except ValueError:
      print("please enter a number")
  ```

APT WordPlay

**APT: WordPlay**

**Problem Statement**

Given a phrase of words, your task is to return a string of the unique words from the phrase, with the words sorted using the following rules:

1. First the unique words should be sorted in reverse order based on their length (number of characters in the word)
2. For words the same length, they should be sorted in alphabetical order based on only the first letter of each such word
3. If there are ties after 1) and 2) criteria, then sort those words in reverse alphabetical order based on the last letter of each such word
4. If there are ties after 1), 2) and 3) criteria, then sort those words in alphabetical order based on the sub-word between the first and last letter of each such word.

**Example**

"mouse elephant moth zebra mole tiger moose moth mule"

Returns:

"elephant moose mouse tiger zebra moth mole mule"
APT WordPlay example

"mouse elephant moth zebra mole tiger moose moth mule"
Returns:

"elephant moose mouse tiger zebra moth mole mule"
• No duplicates

APT WordPlay example

"mouse elephant moth zebra mole tiger moose moth mule"
Returns:

"elephant moose mouse tiger zebra moth mole mule"
• No duplicates
• Reverse order by length
• Ties: alphabetical by first letter

APT WordPlay example

"mouse elephant moth zebra mole tiger moose moth mule"
Returns:

"elephant moose mouse tiger zebra moth mole mule"
• No duplicates
• Reverse order by length
• Ties: alphabetical by first letter
• 2\textsuperscript{nd} Ties: reverse alphabetical by last letter
**APT WordPlay example**

"mouse elephant moth zebra mole tiger moose moth mule"

Returns:

"elephant moose mouse tiger zebra moth mole mule"

- No duplicates
- Reverse order by length
- Ties: alphabetical by first letter
- 2\textsuperscript{nd} Ties: reverse alphabetical by last letter
- 3\textsuperscript{rd} Ties: alphabetical sub-word between first and last letter

**Problem Statement**

Given a phrase of words, your task is to return a string of the unique words from the phrase, with the words sorted using the following rules:

1. First the unique words should be sorted in reverse order based on their length (number of characters in the word).
2. For words the same length, they should be sorted in alphabetical order based on only the first letter of each such word.
3. If there are ties after 1) and 2) criteria, then sort those words in reverse alphabetical order based on the last letter of each such word.
4. If there are ties after 1), 2) and 3) criteria, then sort those words in alphabetical order based on the sub-word between the first and last letter of each such word.
def sortinorder(phrase):
    alist = list(set(phrase.split()))
    blist = sorted(alist, key=lambda f: f[1:-1])
    clist = sorted(blist, key=lambda f: f[-1], reverse=True)
    dlist = sorted(clist, key=lambda f: f[0])
    elist = sorted(dlist, key=lambda x: len(x), reverse=True)
    return " ".join([x for x in elist])