Reminders

• Assignments
  • Assign 7 and APT 8 due Thursday (12/2)
  • GRACE PERIOD ENDS DEC 3! NOT ACCEPTED AFTER THIS!

• Lab 11
  • Pre-lab this week

• Ethics in AI talk

• Spring 2022 UTA applications open
  • See Ed announcement
Final Exam

• APT quiz style
  • Part A and B
    • 90 minutes each
  • Official schedule (12/10-9am-12pm)
    • May complete anytime between 12/10 (8am EST) and 12/12 (11pm EST)
  • Same rules apply of what can/cannot be used
    • Do not violate academic code of conduct!
Key instructions

• Input ✔
• Output ✔
• Assignments* ✔
• Math/Logic ✔
• Conditionals ✔
• Repetition ✔

*not listed in book
Python Data Types

• int, float, bool ✔
• Collections
  • Strings ✔
  • Lists ✔
  • Tuples ✔
  • Sets ✔
  • Dictionaries ✔
PFTD

• How do Dictionaries work so fast!
  • Access an element in constant time

• Recursion
  • Solving a problem by solving smaller problems
KISS Principle

• Think of the non-computing context for any word/terms
• KISS model
  • Work smarter, not harder!!
• “Good programmers are simply good designers.”
  • -Dr. Washington
• Design first and always!
• Importance of reusability
• USE PyCharm/PythonTutor IF YOU HAVE QUESTIONS!
People to Know: Dr. Rediet Abebe

• BS/MS/MA (Math/Applied Math)
  • Harvard, University of Cambridge
• Ph.D. (CS)
  • Cornell
• Co-founder
  • Black in AI
  • Mechanism Design for Social Good
Assignment 7: Create, Due 12/2
Grace period til 12/3, No late days!
Must be turned in by 12/3
This assignment is required!

Pick one:
Video: Green dance, advertisement for 101, song, other
Poem or Multiple Haikus
Story
Comic
One-pager
Feedback

Let's see some examples
A Story – One Eternity Later
APT Due

One night, 30 minutes before the APT is due...
Some Haikus

Haiku #1
A single red square
Defiant among all greens
APT: not done :(  

Haiku #2
My first day of class
Prof. Rodgers speaking while on mute
What a start to Zoom
Advice on Final Exam APTs

• Work an example, or 2 or 3 by hand
  • How are you solving it?
• What would the algorithm be?
• What tools do you need to implement?
  • Dictionary? Set? What do you need to loop over?
• What helper function could you write? Or two?
  • Debug it
• As you write code, print a lot!
• See debugging tips on APT main page
More on Grades

• Class Participation-WOTOs – ignore the first two weeks (drop/add period), plus drop 10 points

• Lab – drop 15 points (each lab is 5 pts)
  • Lab 11 covers recursion and debugging!
Recursion

• Solving a problem by solving similar but smaller problems
Recursion
Solving a problem by solving similar but smaller problems

**Question** - How many **rows** are there in this **classroom**?

**Similar but smaller question** - How many **rows** are there until your row?

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
</tbody>
</table>

I don’t know, let me ask
I don’t know, let me ask
I don’t know, let me ask
I don’t have anyone to ask. So I am in Row#1

Row count = 4+1 = 5

Return Value = 3+1 = 4
Return Value = 2+1 = 3
Return Value = 1+1 = 2
Return value = 1
Review: Recursion Summary

• Function calls smaller version of itself
  • Different input
  • Each call gets closer to base case
• Must have a base case (when no recursive call can be made)
  • Example: n=0 → Factorial
  • This is the way out of recursion! (Work backwards)
Example

```python
def Mystery(num):
    if num > 0:
        return 1 + Mystery(num//2)
    else:
        return 2 + num
```

- Mystery(7) is $1 + \text{Mystery}(3)$ = $1 + 4 = 5$
- Mystery(3) is $1 + \text{Mystery}(1)$ = $1 + 3 = 4$
- Mystery(1) is $1 + \text{Mystery}(0)$ = $1 + 2 = 3$
- Mystery(0) is $2 + 0$ = $2$
Something Recursion
https://bit.ly/101f21-12-02-1
Something Recursion

What is Something([3, 5, 1])?

def Something(data):
    # data is a list of integers
    if len(data) == 0:
        return 0
    if data[0] % 2 == 0:  # it is even
        return data[0] + Something(data[1:])
    else:
        return Something(data[1:])
def Something(data):
    # data is a list of integers
    if len(data) == 0:
        return 0
    if data[0] % 2 == 0:
        return data[0] + Something(data[1:])
    else:
        return Something(data[1:])

• Something([3, 5, 1]) is
• Something([5, 1]) = 0
• Something([1]) = 0
• Something([ ]) = 0

Something([3, 5, 1]) is 0
What is $\text{Something([5,4,2,3])}$?

def Something(data):
    # data is a list of integers
    if len(data) == 0:
        return 0
    if data[0] % 2 == 0:  # it is even
        return data[0] + Something(data[1:])
    else:
        return Something(data[1:])
def Something(data):
    # data is a list of integers
    if len(data) == 0:
        return 0
    if data[0] % 2 == 0:  # it is even
        return data[0] + Something(data[1:])
    else:
        return Something(data[1:]),

Something([5,4,2,3]) is 6

- Something([5,4,2,3]) is 6
- Something([4,2,3]) is 6
- Something([2,3]) is 2
- Something([3]) is 0
- Something([ ]) is 0
Revisit the APT Bagels Recursively

```python
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
APT Bagels Recursively
APT Bagels Recursively

A)
```python
def bagelCount(orders):
    if len(orders) > 0:
        return orders[0]//12 + orders[0] + bagelCount(orders[1:])
    else:
        return 0
```

B)
```python
def bagelCount(orders):
    if len(orders) > 0:
        return orders[-1]//12 + orders[-1] + bagelCount(orders[:-1])
    else:
        return 0
```

C)
```python
def bagelCount(orders):
    return orders[0] + orders[0]//12 + bagelCount(orders[1:])
```

D)
```python
def bagelCount(orders):
    if len(orders)>1:
        return orders[1] + orders[1]//12 + bagelCount(orders[2:])
    else:
        return bagelCount(orders[0])
```
How is Python like all other programming languages, how is it different?
Find all unique/different words in a file, in sorted order
def main():
    f = open('/data/melville.txt', 'r')
    words = f.read().strip().split()
    allWords = set(words)

    for word in sorted(allWords):
        print(word)

if __name__ == "__main__":
    main()
import java.util.*;
import java.io.*;
public class Unique {
    public static void main(String[] args)
        throws IOException{
        Scanner scan =
            new Scanner(new 
                File("/data/melville.txt").
                TreeSet<String> set = new TreeSet<String>();
            while (scan.hasNext()){
                String str = scan.next();
                set.add(str);
            }
            for(String s : set){
                System.out.println(s);
            }
        }
    }
Unique words in C++

```cpp
#include <iostream>
#include <fstream>
#include <set>
using namespace std;

int main(){
    ifstream input("/data/melville.txt");
    set<string> unique;
    string word;
    while (input >> word){
        unique.insert(word);
    }
    set<string>::iterator it = unique.begin();
    for(; it != unique.end(); it++){
        cout << *it << endl;
    }
    return 0;
}
```
<?php

$wholething = file_get_contents("file:///data/melville.txt");
$wholething = trim($wholething);

$array = preg_split("/\s+/",$wholething);
$uni = array_unique($array);
sort($uni);
foreach ($uni as $word){
    echo $word."<br>";
}

?>
What is next?

• CompSci 201
  • Java, efficiency, other ways to organize data
• CompSci 230 – can take concurrently with 201
  • Discrete Mathematics

• CompSci 260 Computational Biology
• CompSci 216 Everything Data
• CompSci 240 Race, Gender, Class and Computing
Reminders

• Work smarter, not harder
• Design first
• Get smaller parts working, then build on it
• Try to identify where you are stuck
  • Identify resources to help solve problem
• Leverage your design and PythonTutor to understand program flow of control
  • [http://pythontutor.com](http://pythontutor.com)