

# CompSci 101

## Introduction to Computer Science



Oct 20, 2016

Prof. Rodger

Lecture by Barrett Ames

## Announcements

- Reading and RQ due next time
- Assignment 5 due in one week
- APT 5 due Tuesday
- Today:
  - Debugging
  - APT SandwichBar

## Process Exam Scores

[bit.ly/101f16-1018-5](http://bit.ly/101f16-1018-5)

- Calculate
  - total number of scores
  - Average score
  - Median score
- Print a visualization of the grades
- Get snarf file

## Problem 1: Set Examples

[bit.ly/101f16-1020-1](http://bit.ly/101f16-1020-1)

```
s = set(lista)    lista = ['apple', 'pear', 'fig', 'orange', 'strawberry']
t = set(listb)    listb = ['pear', 'lemon', 'grapefruit', 'orange']

problem1 = (s-t) | (t-s)
print problem1

problem2 = (s|t) - (s&t)
print problem2

problem3 = (s|t|(s&t))
print problem3
```

## Debugging Problems

- Today the focus is on debugging.
- There are several problems. Trace by hand to see if you can figure out if they are correct or not, or what to do to correct them.
- Enter your answers on the google form

## Debug 1 – Does it work?

[bit.ly/101f16-1020-2](http://bit.ly/101f16-1020-2)

- The function *sizes* has a parameter named *words* that is a list of strings. This function returns a list of the sizes of each string. For example, `sizes(['This', 'is', 'a', 'test'])` should return the list `[4, 2, 1, 4]`

```
def sizes(words):  
    nums = [ ]  
    for w in words:  
        nums = len(w)  
    return nums
```

## Debug 2 – Does it work?

[Bit.ly/101f16-1020-3](http://Bit.ly/101f16-1020-3)

- The function *buildword* has a parameter *words* that is a list of strings. This function returns a string that is made up of the first character from each word in the list. For example, `buildword(['This', 'is', 'a', 'test'])` returns 'Tiat'

```
def buildword(words):  
    answer = ''  
    for w in words:  
        answer += w[:1]  
    return answer
```

## Debug 3 – Does it work?

[Bit.ly/101f16-1020-4](http://Bit.ly/101f16-1020-4)

- The function *middle* has a parameter *names* that is a list of strings, which each string is in the format "firstname:middlename:lastname". This function returns a list of strings of the middlenames.

```
def middle(names):  
    middlelist = []  
    for name in names:  
        name.split(":")  
        middlelist.append(name[1])  
    return middlelist
```

## Debug 4 – Does it work?

Bit.ly/101f16-1020-5

- The function *removeOs* has one string parameter named *names*. This function returns a string equal to *names* but with all the lowercase o's removed.

```
def removeOs(word):  
    position = word.find("o")  
    while position != -1:  
        word = word[:position] +  
            word[position+1:]  
    return word
```

## Problem 5 – Does it work?

Bit.ly/101f16-1020-6

- The function *uniqueDigits* has one int parameter *number*. This function returns the number of unique digits in *number*. If the number is 456655, then it returns 3.

```
def uniqueDigits(number)  
    digits = [ ]  
    while number > 0:  
        digits.append(number % 10)  
        number = number / 10  
    return len(digits)
```

## APT SandwichBar

### 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.

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.

### 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
```

## APT SandwichBar

Example:

```
[ "cheese", "cheese", "cheese", "tomato" ]  
[ "ham ham ham", "water", "pork", "bread", "cheese tomato cheese", "beef" ]
```

Returns: 4