Introduction to:
Computers & Programming:
Post-Midterm 1 Review

Adam Meyers
New York University
Summary

• Some Procedural Matters
• Review of Variables, Functions and Loops
• Review of Part 1
• Review of Part 2
• Lab Tomorrow
# Grading Curve

<table>
<thead>
<tr>
<th>Letter Grade</th>
<th>Raw Score 009</th>
<th>Raw Score 004</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>93 and Above</td>
<td>TBA</td>
</tr>
<tr>
<td>A-</td>
<td>88-92</td>
<td>TBA</td>
</tr>
<tr>
<td>B+</td>
<td>84-87</td>
<td>TBA</td>
</tr>
<tr>
<td>B</td>
<td>79-83</td>
<td>TBA</td>
</tr>
<tr>
<td>B-</td>
<td>75-78</td>
<td>TBA</td>
</tr>
<tr>
<td>C+</td>
<td>71-74</td>
<td>TBA</td>
</tr>
<tr>
<td>C</td>
<td>66-70</td>
<td>TBA</td>
</tr>
<tr>
<td>D</td>
<td>55-65</td>
<td>TBA</td>
</tr>
<tr>
<td>F</td>
<td>54 and Below</td>
<td>TBA</td>
</tr>
</tbody>
</table>
Raw Score vs Letter Score

• Only the Letter Score counts for purposes of the final grade, e.g., there is no difference between a 71 and a 73 if both are part of the same range.

• It is only worth haggling over your grade if:
  – the change in score will effect your letter grade, e.g.,
    • raw score is 91 (an A-) and discrepancy is worth 2 points
    • discrepancy is worth 5-10 points
    • Etc.

• Of course, understanding everything you got wrong is important regardless of the grade.
Grading Considerations

• If Midterm 2 is significantly better than Midterm 1, Midterm 2 will count more towards the final grade than Midterm 1.
  – If it is a real outlier, I will not count it at all, e.g.,
    • If (1\textsuperscript{st} Midterm == D) and (2\textsuperscript{nd} Midterm == A) and (Final == A) and (Homework Grade == A):
      Final Grade = A

• In other words improvement over the course of the class can have a big influence on final grade. I have not figured out exactly how to quantify this, but it will be a major factor.

• **Main purpose of final grade**: An indicator of state of knowledge at the end of the class.

• **Administrative purpose**: Only allow qualified students to progress to higher level CS classes. C is the minimum grade in this class if you want to take the JAVA programming class.
Variable Assignment & Input Statements

- Input statement is a function
  - It takes one parameter, a message to a user
  - It returns one value, a string entered by a user
  - You can assign this value to a variable

- Correct: `Variable1 = input('Give me a Number')`
  - ## Variable1 is set to whatever the user inputs

- Incorrect: `input('Give me a Number')`
  - ## The input function is used, but the output returned by the statement is essentially thrown away
How Loops Work with Accumulating Variables

• def intersperse (string1,string2):
    
    big_string = " ## initialize big_string as empty
    if len(string1)==len(string2):
        for index in range(len(string1)):
            big_string=big_string+string1[index]+string2[index]
        print(big_string)
    return(big_string)

• def triangle_number(number):
    triangle_output = 0
    for num in range(1,number+1):
        triangle_output = triangle_output+num
    print(num,triangle_output)
    return(triangle_output)
def ninety_nine_bottles:
    more = True
    num = 99
    while more:
        print(num, 'bottles of beer on the wall')
        print(num, 'bottles of beer...if one of those bottles should happen to fall')
        num = num-1
        if num == 0:
            more = False
        else:
            answer = input('More?')
            answer = answer.lower()
            if answer == 'no':
                more = False
Using Non-Booleans as Booleans

• Most non-booleans can be equivalent to True
• The following exceptions which can be equivalent to False
  – The empty string ("")
  – The number zero (0)
  – range(0)
  – None (what a void function like print returns)
  – And a small number of others objects, all of which play a similar role in their object-type
Boolean Expressions containing Non-Boolean

• Non-Boolean behave with connectives and control structures as per previous slide
• False or 'red' → evaluates as 'red'
• if False or 'red':
  print('hi')  ## 'hi' will print
• False and '' → evaluates as ''
• if False and '':
  print('hi')  ## hi will not print
How This Relates To Test Examples

• Answer1 == 'red' or 'blue'
  – Always evaluates as True
  – Equivalent to (Answer1 == 'red') or 'blue'
    • (Answer1 == 'red) evaluates as True or False
    • 'blue' is treated as True no matter what
    • The statement is always True
  – Not Equivalent to: (Answer1 == 'red') or (Answer1 == 'blue')

• Answer1 == 'red' and 'blue'
  – Equivalent to (Answer1 == 'red') and 'blue'
    • True whenever (Answer1 == 'red') is True
    • Not Equivalent to (Answer1 == 'red') and (Answer1 == 'blue')
The Tests in PDF form
Answers as PY files

• 004 test:
  – http://cs.nyu.edu/courses/spring16/CSCI-UA.0002-004/midterm1_version1.py

• 009 test:
  – http://cs.nyu.edu/courses/spring16/CSCI-UA.0002-004/midterm1_version2.pdf
  – http://cs.nyu.edu/courses/spring16/CSCI-UA.0002-004/midterm1_version2.py