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

Adam Meyers
New York University
Summary

• Some Procedural Matters
• Review of Concepts from Test
• Review of Part 1
• Review of Part 2
• Lab Tomorrow
<table>
<thead>
<tr>
<th>Letter Grade</th>
<th>Raw Score Test 1 (Section 004)</th>
<th>Raw Score Test 2 (Section 009)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>93 and Above</td>
<td>90 and Above</td>
</tr>
<tr>
<td>A-</td>
<td>88-92</td>
<td>85-89</td>
</tr>
<tr>
<td>B+</td>
<td>84-87</td>
<td>80-84</td>
</tr>
<tr>
<td>B</td>
<td>79-83</td>
<td>74-79</td>
</tr>
<tr>
<td>B-</td>
<td>75-78</td>
<td>69-73</td>
</tr>
<tr>
<td>C+</td>
<td>70-74</td>
<td>60-68</td>
</tr>
<tr>
<td>C</td>
<td>61-69</td>
<td>51-59</td>
</tr>
<tr>
<td>C-</td>
<td>51-60</td>
<td>41-50</td>
</tr>
<tr>
<td>D</td>
<td>40-50</td>
<td>31-40</td>
</tr>
<tr>
<td>F</td>
<td>39 and Below</td>
<td>30 and Below</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 on the borderline between 2 grades
    • discrepancy is worth 5-10 points
    • Etc.

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

• The Midterm1 grade may not be counted if other grades are higher
  – Grade1 = (.05 * Quiz) + (.2 * HW) + (.2 * Mid1) + (.2 * Mid2) + (.35 * Final)
  – Grade2 = (.05 * Quiz) + (.2 * HW) + (.3 * Max(Mid1, Mid2)) + (.45 * Final)
  – Grade = max(grade1, grade2)

• Thus improvement over the course of the class can influence the final grade.

• **Main purpose of final grade:** Indicator of state of knowledge at end of class.

• **Administrative purposed:**
  – C is the minimum grade in this class if you want to take the JAVA programming class.
  – A or A- average is recommended if you want to be a CS major.
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

- It is usually not useful to redefine parameters with input statements, e.g.,
  - def tweeze_eyebrows(left_eyebrow,right_eyebrow):
    - left_eyebrow = input('Enter Left Eyebrow')
    - right_eyebrow = input('Enter Right Eyebrow')
    etc. …..
Print

• print statements do not return anything
  – Do not assign a variable to the value of print
    • Abc = print(input('blah blah'))
    • Abc = input('blah blah')

• print statements put spaces between arguments by default and \n at the end
  – print(var1,var2,var3,var4)

• The parameters sep and end change this
  – print(var1,var2,var3,var4,sep='',end='')
Accumulator 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)
While loops with Accumulators and Sentinel Variables

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
Other Ways to exit Loops

• return – exits function, will exit loop also
  def blah():
    while True:
      if ...
        return(157)

• Break – only returns from inner-most loop
  def blah_blah():
    for number in range(5000):
      if number >= 57:
        break
    print(number)
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-Booleans

• Non-Boolean behave with connectives and control structures as per previous slide
• False or 'No' → evaluates as 'No'
• if False or 'No':
  
  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 == 'Yes' or 'Y'
  - Always evaluates as True
  - Equivalent to (Answer1 == 'Yes') or 'Y'
    - (Answer1 == 'Yes') evaluates as True or False
    - 'Y' is treated as True no matter what
    - The statement is always True
  - Not Equivalent to: (Answer1 == 'Yes') or (Answer1 == 'Y')

- Answer1 == 'Yes' and 'Y'
  - Equivalent to (Answer1 == 'Yes') and 'Y'
    - True whenever (Answer1 == 'Yes') is True
    - Not Equivalent to (Answer1 == 'Yes') and (Answer1 == 'Y')
    - Not Equivalent to Answer1 in ['Yes','Y']
while and if are different

- Only if is followed by else or elif
  - if …:
    - XXX
  - elif …:
    - XXX
  - else:
    - XXX

- while begins a loop and is not connected to elifs or else
  - while expression_containing_variable:
    - XXX (including possible change of variable that ends loop)
break and return

• **break**
  – ends a loop, statements following break at the same level of indent do not execute
  – not covered in class because it is possible to exit a loop without *break*

• **return**
  – returns a value from a function
  – The function is exited immediately after *return*
  – Statements following *return* don't execute
Embedded loops

• A loop with in a loop gets all combinations
  – for num1 in range(1,10):
    for num2 in range(1,10):
      print(num1,num2)

• This is very general. It applies to all combinations of sequences.
Tests and Answers

• Test Version1 (Section 004)
  – Test:  http://cs.nyu.edu/courses/spring17/CSCI-UA.0002-004/midterm1_version1.pdf
  – Answers:  http://cs.nyu.edu/courses/spring17/CSCI-UA.0002-004/midterm1_version1.py

• Test Version2 (Section 009)
  – Test:  http://cs.nyu.edu/courses/spring17/CSCI-UA.0002-004/midterm1_version2.pdf
  – Answers:  http://cs.nyu.edu/courses/spring17/CSCI-UA.0002-004/midterm1_version2.py
Middle of Term Grades

• .05 * Quizzes + .45 * HW + .5 * Midterm
• Unofficial Grade – will not go on your transcript
• Purpose: Advisory Only