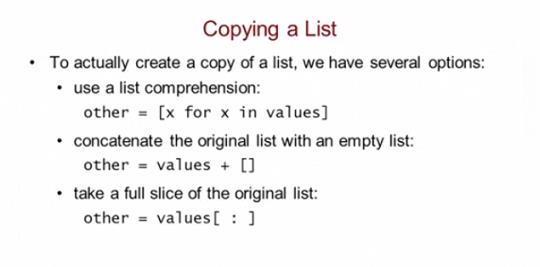
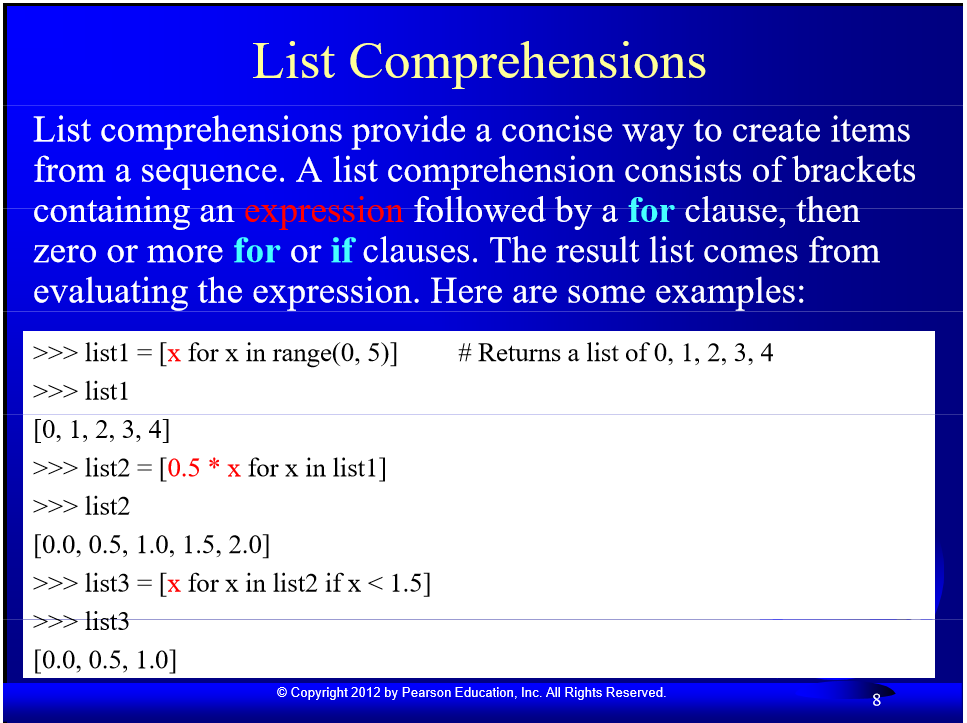
**Last minute Study Tools**

Slicing Strings & Lists:

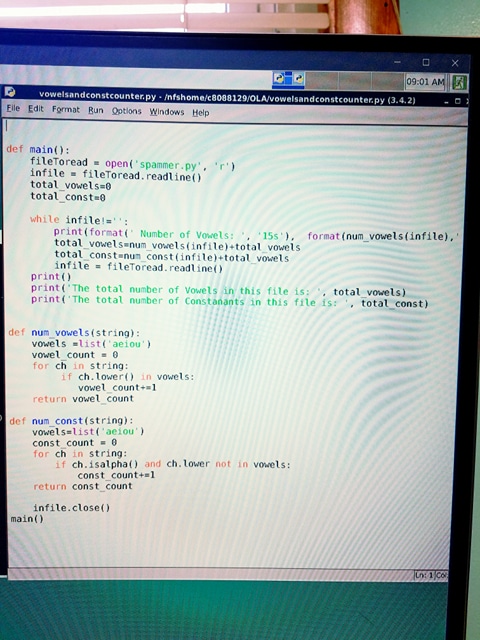
[**http://codingbat.com/doc/python-strings.html**](http://codingbat.com/doc/python-strings.html)

[**http://codingbat.com/doc/python-lists.html**](http://codingbat.com/doc/python-lists.html)

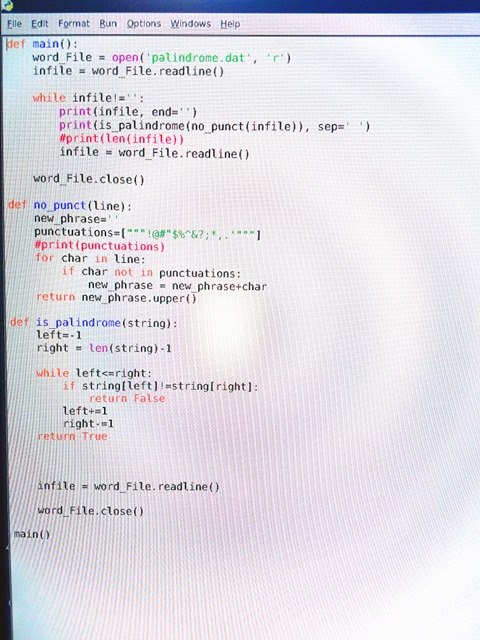
****

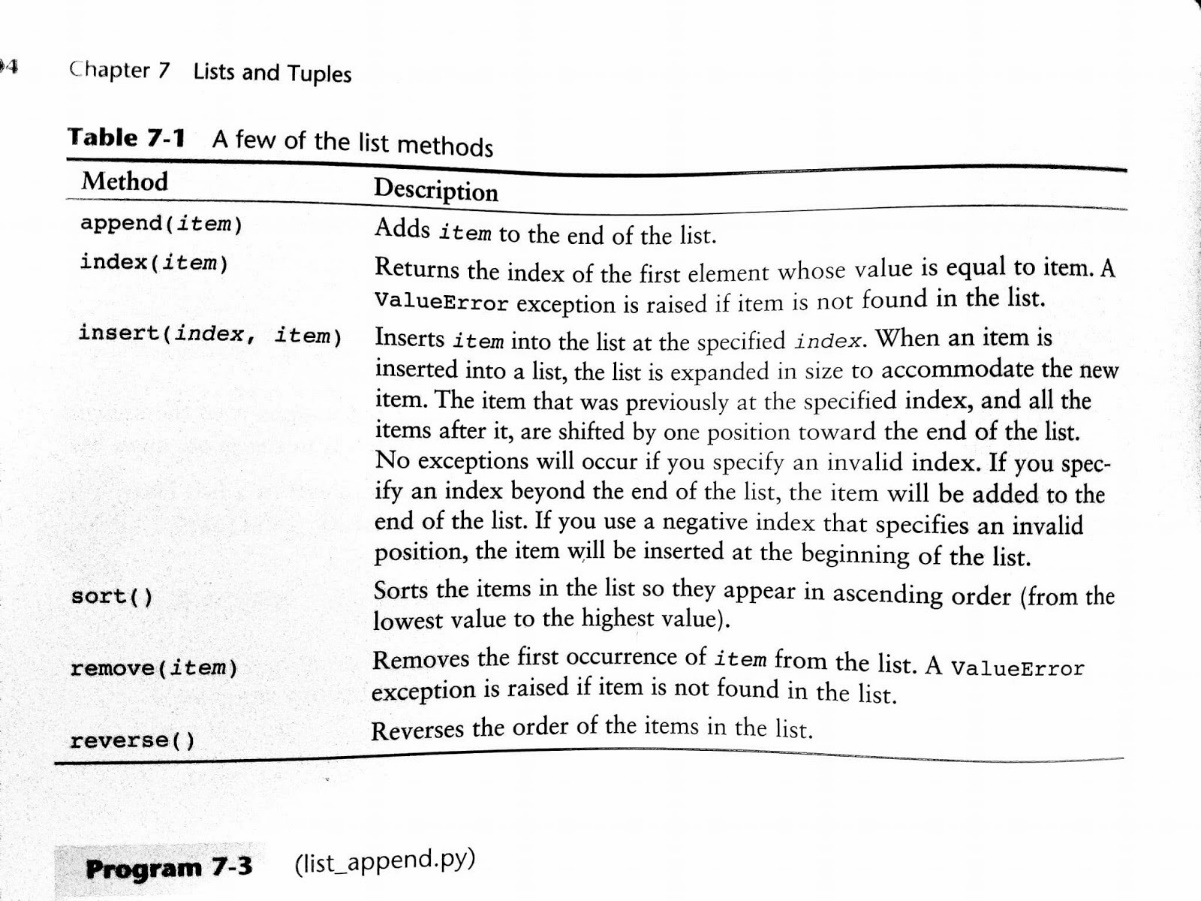
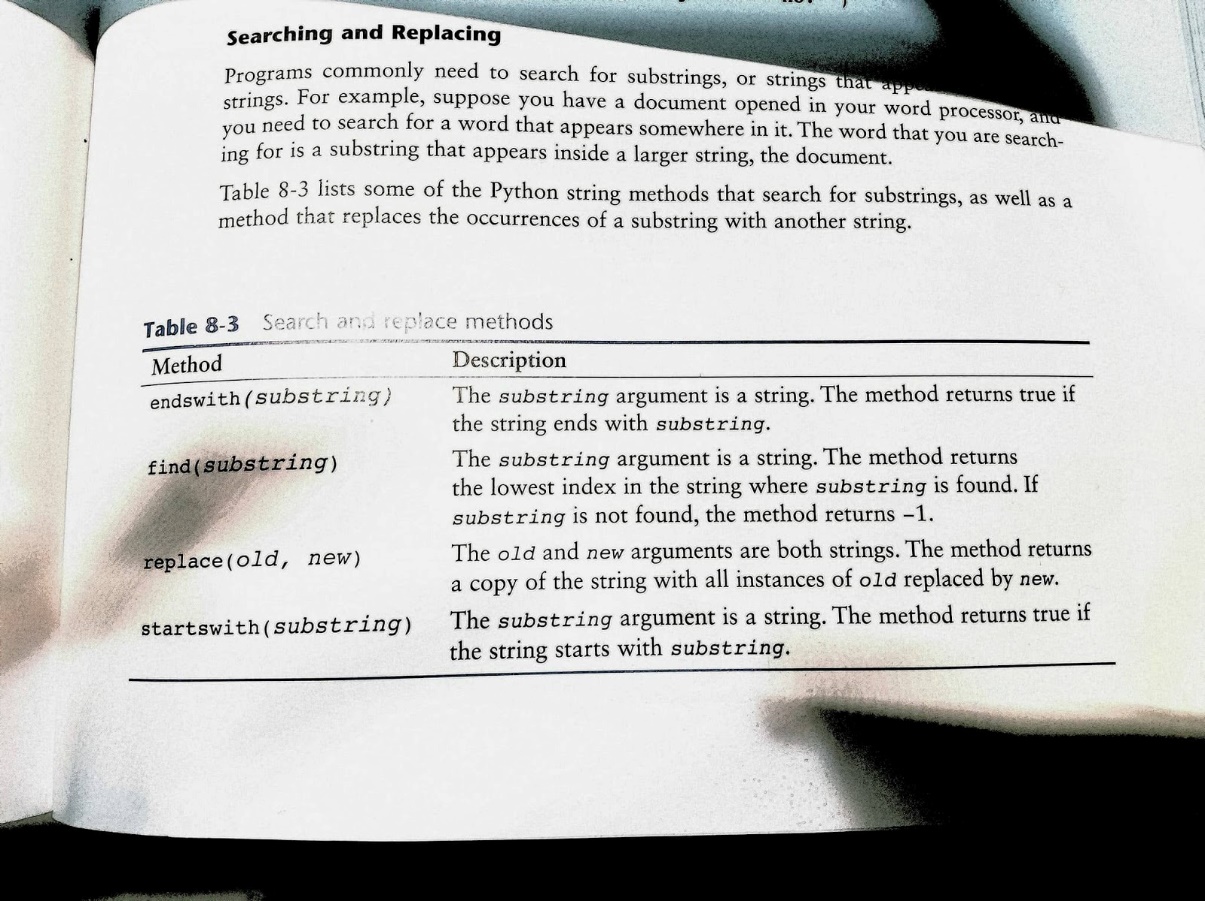
****

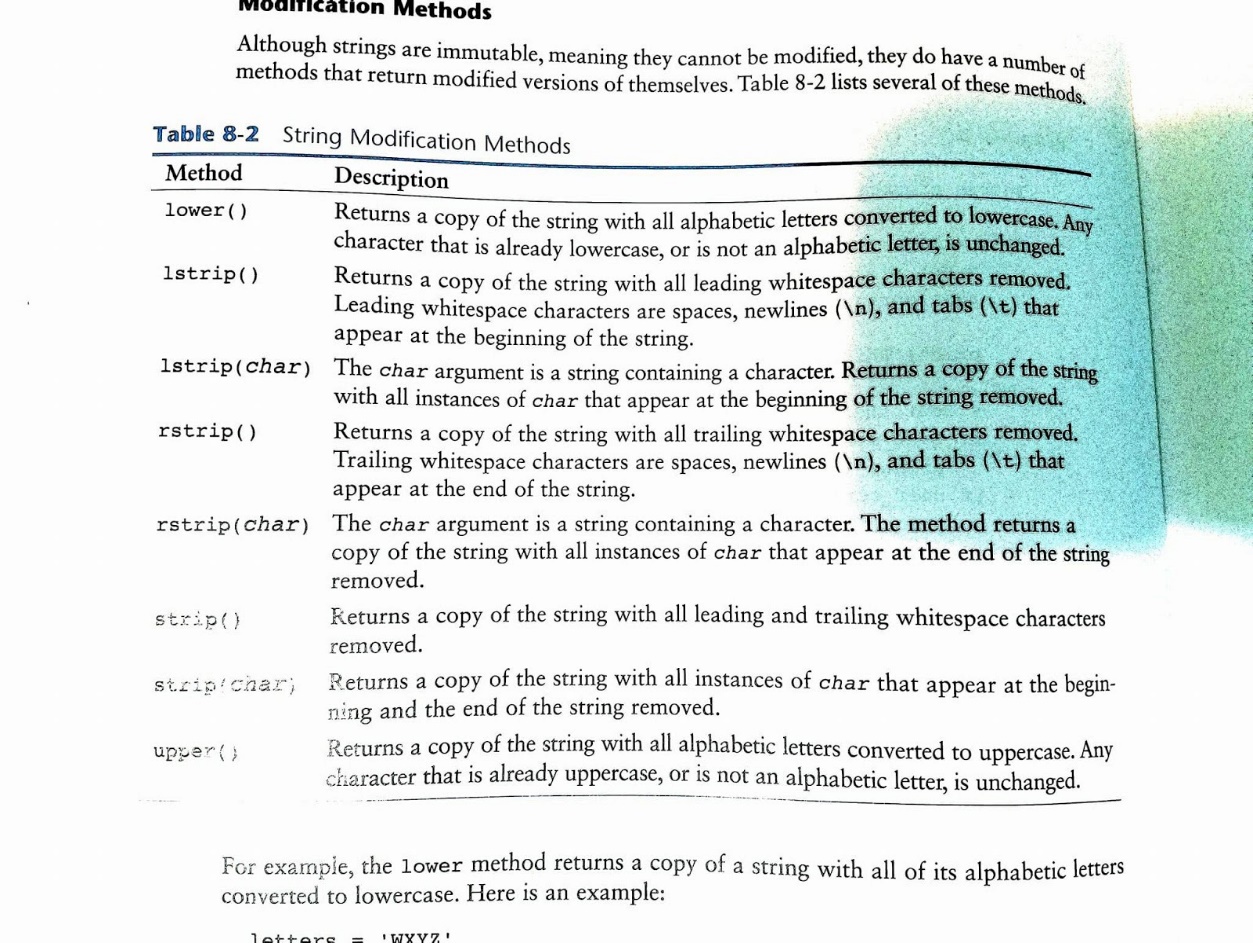
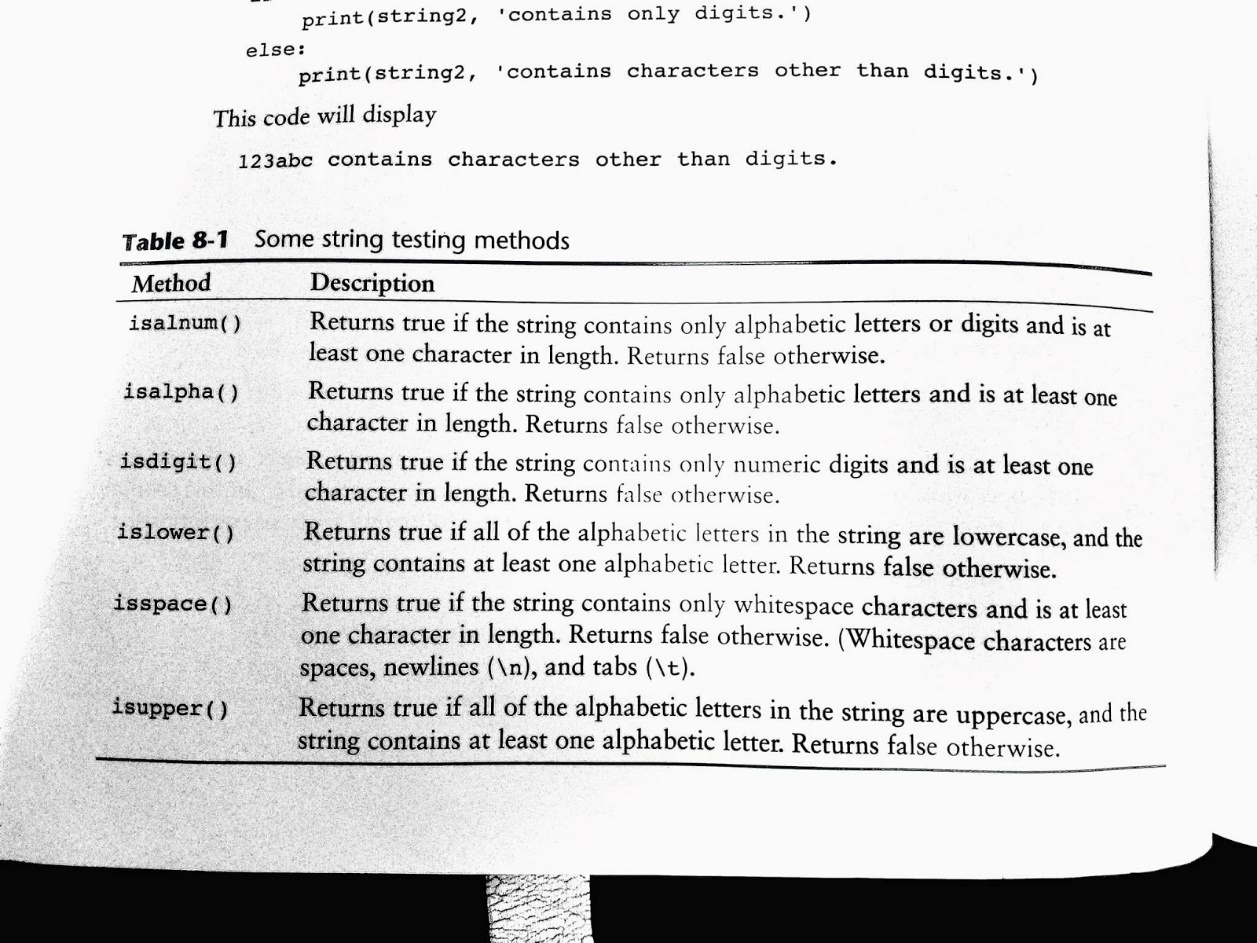
**To find the number of Vowels or Consonants.**

****

**Palindrome function/Way to remove punctuation without .isalpha()**

****

****

****

**CSCI 1170-002 (Untch)               Exam 3 Review Sheet**

**Monday**

|  |  |  |
| --- | --- | --- |
| **Nov 20** | |  | | --- | | [**Exam 3**](https://www.cs.mtsu.edu/~untch/1170/private/exam3results.txt) (Bring a [short-form Scantron](https://www.cs.mtsu.edu/~untch/share/ScantronShortForm19641.jpg) form and #2 pencil)     [[Test Protocol](https://www.cs.mtsu.edu/~untch/1170/public/TestTakingProtocol3.html)] | |

   The exam will be drawn from the material covered **from the beginning of  
the course up to the present, excepting 2-D lists and tuples.**  The focus  
topics of the exam include:

 \* Looping constructs (for,while)

Notes:

*Ex:*

 \* Multi-way IFs (if-elif-else)

Notes:

*Ex:*

grade = (#)

if grade >= 90:

print(“A”)

elif grade >= 80:

print(“B”)

elif grade >= 70:

print(“C”)

elif grade >= 60:

print(“D”)

else:

print(“F”)

 \* Python Data Types (int,float,str,bool)

Notes:

***Int*:** integer. A solid number.

***Float:*** a partial number- fraction or decimal.

***Str:*** String. A word/phrase written with letters. If any numbers are involved, the computer will not see them as numbers until you *tell* the computer to. (using int/float).

***Bool:*** Boolean expression- and/or. They will be registered as either True or False. Use DeMorgan’s Law (deprecate) to assess whether they’re true/false.

 \* Boolean Logic, Boolean Expression Evaluation, and DeMorgan's Laws

Notes:

**Boolean Expression:** Any expression that evaluates to either True or False.

**DeMorgan’s Law:** not(a and b) ←→ not a or not b

                               not(a or b) ←→ not a and not b

**\* Binary numbers**

***Example:***

Base 2:

16  8  4  2  1

    1   0  0  1  0 = 16 + 2 = 18

 \* Functions (void and non-void)

Notes:

**Void Function:** A function that does not return a value, there is no parameter.

***Example:***move(); turn\_left(); turn\_right():

**Non-Void Function:** A function that returns a value(value-returning function)

***Example:***

def sum(num1, num2):

    result  =  num1 + num2

    Return  result

 \* Argument/Parameter conventions

Notes:

*Ex:*

 \* Files

Notes:

**Opening the file** - Opening the file creates a connection between the file and the program. Opening an output file usually creates the file on the disk and allows the program to write the data to it.

***Ex:***

file\_variable = (**filename, mode**)     #*Modes are ‘r’ for read, ‘w’ for  write, and ‘a’ for  append.*

**Process the file** - In this step data is either written to the file (if it is an output file) or read from the file (if it is an input file).

**Close the file** - When the program is finished using the file, the file must be closed. Closing a file disconnects the file from the program.

 \* Strings

Notes:

*Ex:*

 \* Lists (but not two-dimensional lists)

Notes:

**lower()**Return a copy of the string converted to lowercase.

**lstrip([chars])**, **rstrip([chars])**Same as the strip except left(leading) or right(trailing) only.

**strip([chars])**Return a copy of the string with the leading and trailing characters removed. The chars argument is a string specifying the set of characters to be removed. If omitted or None, the chars argument defaults to removing whitespace.

**upper()**Return a copy of the string converted to uppercase.

**replace(old, new[, count])**

Return a copy of the string with all occurrences of substring old replaced by new. If the optional argument count is given, only the first count occurrences are replaced.

**split([sep [,maxsplit]])**Return a list of the words in the string, using (optional) sep as the delimiter string. If maxsplit is given, at most maxsplit splits are done.

*Ex:*

 \* plus ... everything we've covered so far (except 2-D lists and tuples).  
    This corresponds to textbook chapters 1 through 8.  
 *I* do not *intend to ask any UNIX command questions on exam 3.  
  
    I* do not *intend to ask any questions on Hierarchy Charts*  
  
   \* You may be asked to write a function or an entire program.  You will  
be asked \* How to open, read, write, and close files.

Notes:

*Ex:*

 \* You will be expected to demonstrate your mastery of \* String and list manipulation; be sure you understand \* Slicing in strings and lists.  
Notes:

**See links in \*Last minute Study Tools ^**

*Ex:*

**CSCI 1170 (Untch)                  Exam 2 Review Sheet**

**\* Looping constructs (for,while) Designing Loops… Know your loop!**

* How does it terminate? When the sentinel value is encountered
* Do (local) variables need to be updated within the loop? Yes *Ex: count += 1*
* What is the initialization? A necessary preceding command of a function
* What is being repeated? (What is the body doing?)
* What must be accomplished at the end of the loop?

**\* Multi-way IFs (if-elif-else)**

***Example:***

grade = (#)

if grade >= 90:

print(“A”)

elif grade >= 80:

print(“B”)

elif grade >= 70:

print(“C”)

elif grade >= 60:

print(“D”)

else:

print(“F”)

**\* Python Data Types (int,float,str,bool)**

**Notes:**

***Int*:** integer. A solid number.

***Float:*** a partial number- fraction or decimal.

***Str:*** String. A word/phrase written with letters. If any numbers are involved, the computer will not see them as numbers until you *tell* the computer to. (using int/float).

***Bool:*** Boolean expression- and/or. They will be registered as either True or False. Use DeMorgan’s Law (deprecate) to assess whether they’re true/false.

**\* Boolean Logic, Boolean Expression Evaluation, and DeMorgan's Laws**

**Notes:**

**Boolean Expression:** Any expression that evaluates to either True or False.

**DeMorgan’s Law:** not(a and b) ←→ not a or not b

                               not(a or b) ←→ not a and not b

**\* Binary numbers**

***Example:***

Base 2:

16  8  4  2  1

    1   0  0  1  0 = 16 + 2 = 18

**\* Functions (void and non-void)  and  Argument/Parameter conventions**

**Notes:**

**Void Function:** A function that does not return a value, there is no parameter.

***Example:***move(); turn\_left(); turn\_right():

**Non-Void Function:** A function that returns a value(value-returning function)

***Example:***

def sum(num1, num2):

    result  =  num1 + num2

    Return  result

**\* Files:**

**Opening the file** - Opening the file creates a connection between the file and the program. Opening an output file usually creates the file on the disk and allows the program to write the data to it.

***Ex:***

file\_variable = (**filename, mode**)     #*Modes are ‘r’ for read, ‘w’ for  write, and ‘a’ for  append.*

**Process the file** - In this step data is either written to the file (if it is an output file) or read from the file (if it is an input file).

**Close the file** - When the program is finished using the file, the file must be closed. Closing a file disconnects the file from the program.

**Plus ... everything we've covered so far. This corresponds to textbook chapters 1 through 6.  I'd review for this exam by studying as follows:**

1. Review notes over the lecture material and view all assigned VideoNotes.

2. Review Textbook: Chapters 2 and 3.  *Be sure* you understand section 3.3

  on string comparison and section 3.4 on the if-elif-else construct.

*3. Review Textbook:  Chapter 4.  Although "while" loops should be well understood by now, be sure to study section 4.3 and watch the assigned VideoNotes on "for loops" to master this construct.  Be sure you understand the concept of sentinels and are able to write code that uses a sentinel to mark the end-of-data.*

4. Skim Textbook:  **Sections 5.1, 5.2, and 5.3.** ("Skim" because you've seen

  this material repeatedly.)

5. Study: Rest of **Chapter 5.4-5.9  (Exam will not cover 5.10)**

6  Study: **Chapter 1.3.**  Review: Handout 914 (Binary Numbers)

7. Study: **Chapter 6.1-6.3 on Files.  Exam coverage of Chapter 6.4, on Exceptions, will be very light; if you remember the lecture you'll be in good shape.**

8. Review **Closed Labs** done since Exam 1.

9. Look over all **handouts** (whether hardcopy or off the web).

We have covered a lot of terminology and definitions.  Be sure you know this terminology.  For example: **"What are the three types of program maintenance?**  **What is precedence?  What is an operand? What are DeMorgan's Laws?  What is a file object?  What are arguments and parameters?  What are global versus local variables?"**

We have studied Python syntax and semantics.  We have become familiar with Reeborg's programming environment.  We have seen how to refactor programs. We have seen how to develop programs using **Stepwise Refinement**. We have seen how to evaluate expressions.  We have studied input and formatted output in Python.  We have learned about the **boolean operators** and how to combine them.  You will be asked to demonstrate your mastery of these items.

                For example, you will be **asked to write a function and/or an entire program**. You may be asked to simplify code.  You may be asked what certain code does.

                You may be asked to evaluate expressions.  You will be asked how to open, read, write, and close files.  You will be expected to demonstrate your command of the technical terminology that we've introduced; for ***Example:***

DEPRECATED: Doesn’t apply anymore, no longer applicable, possibly outdated.

SHORT-CIRCUIT: When using an “and” or “or” expression, the evaluation is stopped at the first evaluated “True” or “False”

***Ex:***some\_condition **or** die;    *# Abort execution if some\_condition is true.*

Reason: Because if either of the conditions are true, the whole “or” statement is true.  
some\_condition **and** die;   *# Abort execution if some\_condition is false.*

Because if either of the conditions are false, the whole “and” statement is false.

ARGUMENT:An argument is any piece of data that is passed into a function when the function is called.

BLOCK:A block is simply a set of statements that belong together as a group.

OPERAND:The values on the **RIGHT AND LEFT** of the operator.

***Ex:*** *2+3*

*# The operands are 2 & 3.*

PARAMETER: A Parameter is a variable that receives an argument that is passed into a function.

***Ex:*** *Def greeting(name):*

*print(‘Hello’, name)         # (name) is the parameter*

ASSOCIATIVITY: Left-to-Right, Right-to-Left; direction python reads code

I do not intend to ask any UNIX command questions on exam 2.

(Not even any questions on file redirection, which is a UNIX feature.)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Working Code Examples**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*clearpapers.py / OLA 106     Due: Midnight, October 9, 2017

Link to problem:

<https://www.cs.mtsu.edu/~untch/1170/private/clearpapers.html>

Working Code:

def main():

   exit\_house()

   clean\_perimeter()

   deposit\_beepers()

   go\_to\_bed()

   turn\_off()

#Reeborg finds vestibule and exits

def exit\_house():

   face\_north()

   if left\_is\_clear():

       turn\_left()

       move()

   else:

       while not left\_is\_clear():

           if not front\_is\_clear():

               turn\_right()

           move()

       turn\_left()

       move()

def clean\_perimeter():

   if not left\_is\_clear():

       move()

   turn\_left()

   while on\_beeper():

       pick\_beeper()

   move()

   if left\_is\_clear():

       turn\_left()

       while on\_beeper():

           pick\_beeper()

       move()

   if not front\_is\_clear():

       turn\_right()

       if on\_beeper():

           while on\_beeper():

               pick\_beeper()

       while on\_beeper():

           pick\_beeper()

       move()

   while not left\_is\_clear():

       if on\_beeper():

           while on\_beeper():

               pick\_beeper()

       move()

   turn\_left()

   while on\_beeper():

       pick\_beeper()

   move()

   while not left\_is\_clear():

       if on\_beeper():

           while on\_beeper():

               pick\_beeper()

       move()

   turn\_left()

   while on\_beeper():

       pick\_beeper()

   move()

   while not left\_is\_clear():

       if on\_beeper():

           while on\_beeper():

               pick\_beeper()

       move()

   turn\_left()

   while on\_beeper():

       pick\_beeper()

   move()

   while not left\_is\_clear():

       if on\_beeper():

           while on\_beeper():

               pick\_beeper()

       if not front\_is\_clear():

           turn\_right()

       move()

   turn\_left()

   while on\_beeper():

       pick\_beeper()

   move()

   while not left\_is\_clear():

       if on\_beeper():

           while on\_beeper():

               pick\_beeper()

       move()

       if not front\_is\_clear():

           turn\_right()

   turn\_left()

   move()

# Reeborg enters vestibule, drops all but one beeper

def deposit\_beepers():

   if left\_is\_clear():

       turn\_left()

       move()

   while carrying\_beepers():

       put\_beeper()

   pick\_beeper()

# Face west in southeast corner.

def go\_to\_bed():

   if facing\_west():

       move\_to\_wall()

       face\_south()

       move\_to\_wall()

       turn\_right()

   if facing\_south():

       move\_to\_wall()

       turn\_right()

   if facing\_north():

       if not right\_is\_clear():

           move()

       if left\_is\_clear():

           turn\_left()

           move\_to\_wall()

       else:

           turn\_left()

   if facing\_east():

       if not left\_is\_clear():

           move()

           turn\_left()

           turn\_left()

       else:

           turn\_right()

           move\_to\_wall()

           turn\_right()

# Go to room's northwest street corner

def go\_to\_northwest\_corner():

   face\_north()

   move\_to\_wall()

   turn\_right()

   move\_to\_wall()

# Advance forward to a wall

def move\_to\_wall():

   while front\_is\_clear():

       move()

# Move along outer wall of house

def move\_along\_wall():

   if front\_is\_clear():

       move()

   else:

       move\_one\_avenue\_north()

# Move east to adjacent avenue corner

def move\_one\_avenue\_north():

   if facing\_west():

       turn\_right()

       move()

       turn\_right()

   else:

       turn\_left()

       move()

       turn\_left()

# Face Reeborg north

def face\_north():

   while not facing\_north():

       turn\_left()

# Face Reeborg south

def face\_south():

   while not facing\_south():

       turn\_left()

#Face Reeborg west

def face\_west():

   while not facing\_west():

       turn\_left()

# Pivot 90 degrees to the right

def turn\_right():

   for x in range(3):

       turn\_left()

# Call the main function.

main()

\*\*\*ulam.py / OLA109         Due: Midnight, October 23, 2017

Link to problem:

<https://www.cs.mtsu.edu/~untch/1170/private/ulam.html>

Working Code:

import sys

def main():

   file\_Z = open('ulam.dat', 'r')

   number = file\_Z.readline()

   while number != '':

       number = int(number)

       x = ulam\_sequence\_length(number)

       print((format(number, '>10')),format(x, '>4'))

       number = file\_Z.readline()

def ulam\_sequence\_length(number):

   x = 1

   while number != 1:

       if number%2 == 0:

           number//=2

       else:

           number = number\*3+1

       x+=1

   return x

main()

\*\*\*exercise12p8.py                       Lab

def main():

   filename = input('Enter name of the file to read')

   try:

       file\_A = open(filename, 'r')

       lowest = 100

       playersName = 0

       name = file\_thing.readLine()

       while  name !='':

           score = int(file\_thing.readLine() )

           name = name.rstrip('\n')

           print(name.score)

           playersName += 1

           if lowest > score:

               lowest = score

               winner = name

           name = file\_A.readLine()

               print("There are", playersName, "players in file", filename)

               print("The winner of this tournament is", winner, "with a score of", lowest)

               file\_A.close()

               except:

                   print('There was an error opening the file.')

main()

\*\*\*check\_digit Program:                    Lab

Def main():

print(“CSCI OLAxyz by Author”)

print(“ Basic Number    Check-Digit”)

#Assign a number

Basic\_num = int(input())

While basic\_num > 0:

Suffix = check\_digit(basic\_num)

Print(format(basic\_num, “12d”), format(suffix, “8d”))

Basic\_num = int(input())

Def check\_digit(basic\_num):

Cross\_sum = 0

Remnant = basic\_num

While remnant > 0:

Last\_digit = remnant%10

Product = last\_digit\*2

Cross\_sum = (product//10)+(product%10)

Remnant //=10

Last\_digit = remnant%10

Cross\_sum += last\_digit

Remnant //=10

#Cycles through until basic\_num is completely worked through

Return cross\_sum

Suffix = (10-(cross\_sum%10))%10

main()

\*\*\*persist.py                    Due:Midnight, November 1, 2017

Link to problem:

<https://www.cs.mtsu.edu/~untch/1170/private/persistence.html>

Working Code:

def main():  
    HIGH\_LIMIT = 7  
    LOW\_LIMIT = 3  
    number = 10  
    p = persistence(number)  
  
    print("CSCI 1170 OLA110 BY Change toYourName")  
    for limit in range (LOW\_LIMIT,n HIGH\_LIMIT+1):  
        number = 10  
        while persistence(number) < limit:  
            number += 1  
          
        print("The smallest integer with a persistence of", limit, "is:", number)  
             
         
def persistence(number):  
    count = 0  
    while  number > 9:  
        product = 1  
        while number >= 1:  
            last\_digit = number%10  
            number = number//10  
            product \*= last\_digit  
        count += 1  
        number = product  
    return count  
          
main()

\*\*\*path.py / OlaX (Optional)   Due: Midnight, November 17, 2017

Link to problem:

<https://www.cs.mtsu.edu/~untch/1170/private/path.htm>

Working Code:

def main():

   return\_home()

   drop\_all\_beepers()

   face\_north()

   turn\_off()

def return\_home():

   count=0

   while count<2:

       count=0

       if front\_is\_clear():

           move()

           if not on\_beeper():

               turn\_left()

               turn\_left()

               move()

               turn\_right()

               if front\_is\_clear():

                   move()

                   if not on\_beeper():

                       turn\_left()

                       turn\_left()

                       move()

                       if front\_is\_clear():

                           move()

               else:

                   turn\_left()

                   turn\_left()

                   move()

       elif left\_is\_clear():

           turn\_left()

           move()

           if not on\_beeper():

               turn\_left()

               turn\_left()

               move()

               if front\_is\_clear():

                   move()

                   if not on\_beeper():

                       turn\_left()

                       turn\_left()

                       move()

                       if front\_is\_clear():

                           move()

       elif right\_is\_clear():

           turn\_right()

           move()

       while on\_beeper():

           pick\_beeper()

           count+=1

def drop\_all\_beepers():

   while carrying\_beepers():

       put\_beeper()

def face\_north():

   while not facing\_north():

       turn\_left()

def turn\_right():

   for x in range(3):

       turn\_left()

main()

\*\*\*is\_Digit is\_Alpha / Ch13p7.py              Lab

def main():

   try:

       input\_file\_name = input("What is the File Name?")

       infile = open(input\_file\_name, 'r')

   except:

       print("ERROR", input\_file\_name, "does not exist")

   line = infile.readline()

   numLines = 0

   alpha\_Char = 0

   digit\_Char = 0

   while line !='':

       for letter in line:

           if isAlpha(letter) == True:

               alpha\_Char+=1

           if isDigit(letter) == True:

               digit\_Char+=1

       line = infile.readline()

       numLines +=1

   print("The number of alphabetic letters is", alpha\_Char)

   print("The number of digits is", digit\_Char)

   print("The number of lines is", numLines)

def isAlpha(letter):

   if (letter>="A" and letter <= "Z") or (letter >="a" and letter <="z"):

       return True

def isDigit(letter):

   if letter >="0" and letter<="9":

       return True

main()

\*\*\*gravity.py                          Lab

def main():

   start\_second = 1                       # At the Start of the Fall.

   end\_second = 11                      # At the End of the Fall.

   print("Time\tFalling Distance\n----\t----------------" )

   for time in range(start\_second, end\_second):

       distance = falling\_distance(time)

       print(time,"\t", format(falling\_distance(time), ".2f"))

def falling\_distance(falling\_seconds):

   gravity = 9.8

   distance=(1/2)\*gravity\*(falling\_seconds\*\*2)

   return distance                          # The distance Value being returned.

main()

\*\*\*spammer.py                 Due: Midnight November 8, 2017

Link to problem:

<https://www.cs.mtsu.edu/~untch/1170/private/spammer.html>

Working Code:

def main():

   infile = open('mail.dat', 'r')         # Open the Read file.

   outfile = open("addresses.dat", "w")   # Open the Write file.

   line = infile.readline()               # Priming Read.

   while line !=' ':

       for word in line.split():

           if word.find('@')!=-1:

               outfile.write(str(word.rstrip(',')))

               outfile.write(str('\n'))

       line = infile.readline()           # Reoccuring Read.

   infile.close()                         # Close both files.

   outfile.close()

main()

\*\*\*consolidate.py / OLA112     Due: Midnight, November 15, 2017

Link to problem:

<https://www.cs.mtsu.edu/~untch/1170/private/consolidate.html>

Working Code:

import os.path

                                                                               # File name constants

FILE\_NAME1  = "store1.dat"

FILE\_NAME2  = "store2.dat"

MERGED\_FILE = "merged.dat"

REJECT\_FILE = "reject.dat"

def main():

                                                                               # Open the input files

   # if not os.path.isfile(store1.dat) or not os.path.isfile(store2.dat):      # In source file, don't fully understand.

   #    return

   file1  = open('store1.dat','r')

   file2  = open('store2.dat','r')

   merged = open('merged.dat', 'w')

   reject = open('reject.dat', 'w')

   # Conflates the files

   eof1, sku1, amt1, uom1 = obtain\_record(file1)                               # Priming read

   eof2, sku2, amt2, uom2 = obtain\_record(file2)

   while not eof1 or not eof2:                                                 # Left in for note:

     # amt1, uom1 = convert\_to\_metric(amt1, str(uom1))                         # \*Can't convert 'all at once' here, need

     # amt1, uom2 = convert\_to\_metric(amt2, str(uom2))                         #  to write rejected skus to file-unchanged\*

       if eof2 or (not eof1 and sku1 < sku2):

           if is\_valid\_sku(sku1)==True:

               amt1, uom1 = convert\_to\_metric(amt1, str(uom1))                 # Covert to Metric

               #print(format(sku1, '9d'), format(amt1, '9.1f'), uom1)          # Print Test

               adjusted\_sku1=format(sku1, '9d')

               merged.write(str(adjusted\_sku1))

               adjusted\_amt1=format(amt1, '9.1f')

               merged.write(str(adjusted\_amt1))

               adjusted\_uom1=(' '+uom1)

               merged.write(str(adjusted\_uom1))

               merged.write('\n')

               #merged.write(format(sku1, '9d'), format(amt1, '8.1f'), uom1)

               eof1, sku1, amt1, uom1 = obtain\_record(file1)                   # Reoccuring read

           else:

               #print(format(sku1, '9d'), "\t\tSKU NOT VALID!")                # Print Test

               #reject.write(format(sku1, '9d'), format(amt1, '8.1f'), uom1)   # Left in for note: To many Arguments

               adjusted\_sku1=format(sku1, '9d')                                # Solution to 'merge.write' formatting -->

               reject.write(str(adjusted\_sku1))

               adjusted\_amt1=format(amt1, '9.1f')

               reject.write(str(adjusted\_amt1))

               adjusted\_uom1=(' '+uom1)

               reject.write(str(adjusted\_uom1))

               reject.write('\n')                                              # --------------------------

               eof1, sku1, amt1, uom1 = obtain\_record(file1)                   # Reoccuring read

       elif eof1 or sku1 > sku2:

           if is\_valid\_sku(sku2)==True:

               amt2, uom2 = convert\_to\_metric(amt2, str(uom2))                 # Covert to Metric

               #print(format(sku2, '9d'), format(amt2, '9.1f'), uom2)          # Print Test

               adjusted\_sku2=format(sku2, '9d')                                # Solution to 'merge.write' formatting -->

               merged.write(str(adjusted\_sku2))

               adjusted\_amt2=format(amt2, '9.1f')

               merged.write(str(adjusted\_amt2))

               adjusted\_uom2=(' '+uom2)

               merged.write(str(adjusted\_uom2))

               merged.write('\n')                                              # --------------------------

               eof2, sku2, amt2, uom2 = obtain\_record(file2)                   # Reoccuring read

           else:

               #print(format(sku2, '9d'), "\t\tSKU NOT VALID!")                # Print Test

               #reject.write(format(sku2, '9d'), format(amt2, '8.1f'), uom2)

               adjusted\_sku2=format(sku2, '9d')                                # Solution to 'reject.write' formatting -->

               reject.write(str(adjusted\_sku2))

               adjusted\_amt2=format(amt2, '9.1f')

               reject.write(str(adjusted\_amt2))

               adjusted\_uom2=(' '+uom2)

               reject.write(str(adjusted\_uom2))

               reject.write('\n')                                              # --------------------------

               eof2, sku2, amt2, uom2 = obtain\_record(file2)                   # Reoccuring read

       else:                                                                   # Implied 'if sku1 == sku2:'

           if is\_valid\_sku(sku1)==True:

               amt1, uom1 = convert\_to\_metric(amt1, str(uom1))                 # Covert to Metric

               #print(format(sku1, '9d'), format(amt1, '9.1f'), uom1)          # Print Test

               adjusted\_sku1=format(sku1, '9d')                                # Solution to 'merge.write' formatting -->

               merged.write(str(adjusted\_sku1))

               adjusted\_amt1=format(amt1, '9.1f')

               merged.write(str(adjusted\_amt1))

               adjusted\_uom1=(' '+uom1)

               merged.write(str(adjusted\_uom1))

               merged.write('\n')                                              # --------------------------

               eof1, sku1, amt1, uom1 = obtain\_record(file1)                   # Reoccuring read

               eof1, sku2, amt2, uom2 = obtain\_record(file2)

           else:

               #print(format(sku1, '9d'), "\t\tSKU NOT VALID!")                # Print Test

               #reject.write(str(sku1), str(amt1), uom1)                       # Left in for note: To many Arguments

               adjusted\_sku1=format(sku1, '9d')                                # Solution to 'reject.write' formatting -->

               reject.write(str(adjusted\_sku1))

               adjusted\_amt1=format(amt1, '9.1f')

               reject.write(str(adjusted\_amt1))

               adjusted\_uom1=(' '+uom1)

               reject.write(str(adjusted\_uom1))

               reject.write('\n')                                              # --------------------------

               eof1, sku1, amt1, uom1 = obtain\_record(file1)

               eof1, sku2, amt2, uom2 = obtain\_record(file2)

   file1.close()                                                               # Close all files.

   file2.close()

   merged.close()

   reject.close()

# Determine validity of SKU number

def is\_valid\_sku(sku):

   #list

   inventory\_number = sku // 10

   check\_suffix     = sku %  10

   if check\_digit(inventory\_number)==check\_suffix:

       return True

   else:

       return False

# Obtain an inventory item from specified file.

def obtain\_record(file):

   record = file.readline()

   if record=='':

       eof = True

       return eof, 0, 0.0, ''

   else:

       eof = False

       sku, amt, unit\_of\_measure = record.split()

       return eof, int(sku), float(amt), unit\_of\_measure

# Calculate the Modulus-10 check-digit of the input parameter.

def check\_digit(number):

   crossfoot = 0

   while number>0:

       product = number%10 \* 2

       crossfoot +=  product//10 + product%10

       number //= 10

       crossfoot += number%10

       number //= 10

   return (10 - crossfoot%10) % 10

# If a record is in english units, convert to metric system

def convert\_to\_metric(amount, unit\_of\_measure):

   if unit\_of\_measure=='I':

       amount = amount\*6.4516      # Square Inches to Square Centimeters

       unit\_of\_measure='C'

   elif unit\_of\_measure=='P':

       amount = amount\*0.45359237

       unit\_of\_measure='K'

   elif unit\_of\_measure=='F':

       amount = amount\*0.3048

       unit\_of\_measure='M'

   return float(amount), str(unit\_of\_measure)

# Invoke main

main()

\*\*\*percentage.py / OLA113     Due: **Midnight, November 29, 2017**

Link to problem:

<https://www.cs.mtsu.edu/~untch/1170/private/percentage.html>

Working Code:

# this is what I got for my code for this program

def main():

   infile = open("exam.dat", "r")

   lst = []

   line = infile.readline()

   while line != '':

       num = int(line)

       lst.append(num) # Add each line of numbers to the list

       line = infile.readline() # Each Loop starts with the next line

   print(format("Sum Equals:",">11s"), format(sum(lst),">6")) # use print statement for sum

   for num in lst: # Print each number in the list plus the percentage

       print(format(num,">11"), format((num/sum(lst)),"7.2%")) #use formatting to get %

   infile.close()

main()

\*\*\*bakery.py / OLA114    Due: **Midnight, December 6th, 2017**

Link to problem:

<https://www.cs.mtsu.edu/~untch/1170/private/bakery.html>

Working Code:

\*\*\*        /            **Additional/Duplicate Code Examples**

Working Code:

\*\*\*        /            **Additional/Duplicate Code Examples**

Working Code:

\*\*\*        /            **Additional/Duplicate Code Examples**

Working Code:

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Final Exam Schedule!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

|  |  |
| --- | --- |
| **Dec 6** | University calendar: **Last Day of Classes**     (Reminder: Midnight is the absolute deadline for submitting via the Gus **handin** system.) |
| **Dec 7** | University calendar: **Study Day** |
| **Dec 11 (002)** | Final Exam - Section 002: 12:30pm-2:30pm    (Bring a [short-form Scantron](https://www.cs.mtsu.edu/~untch/share/ScantronShortForm19641.jpg) form and #2 pencil) |