In [ ]:
```
#-----------------------------------------------#
# PYTHON EXERCISES
# September 12, 2016 #
# EECS 445: Machine Learning #
# Author: Valliappa Chockalingam (valli@umich.edu)
# -----------------------------------------------#
```

Question 1: Numbers and Data Structures

- Example: If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.
- ***Question: Find the sum of all the multiples of 3 or 5 below 1000 in 3 ways.***
- *Hint 1:* Loop through all the possible numbers, i.e., $x \in \mathbb{N} \wedge x < 1000$ and simply add up the multiples as you go through them. Use two loops first and then try making your code more concise.
- *Bonus:* Use the inclusion-exclusion principle. Write a function that calculates the sum of the first n numbers in an arithmetic series. Hint: $S_n = \frac{n}{2}(u_1 + u_n) = \frac{n}{2}(2u_1 + (n-1) \cdot d)$

In [ ]:
```
# Open Ended, No Skeleton Code for this question.
```

Question 2: Strings and Data Structures

Note: *You are free to use any functions and libraries that come with Python.* The aim is not to necessarily implement code from scratch, but to get familiar with Python (specifically the syntax and data structures) and, to some extent, write concise readable working code.

- ***(a): Given a non-empty string like "Code" return a string like "CCoCodCode".***
- ***(b): Given strings a, b and c, write a function that will replace all occurences of a in b with c.***
- ***(c): Given a string s that has been Caeser enciphered with a numeric shift n. Return the deciphered string.*** For example, s = "vjku ku c eqfg" and n = -2, returns "this is a code" Note: You can assume the string is all in lowercase without any special characters.
- ***(d): Given a string s, return whether s is a palindrome, i.e, it is spelt the same when read from either direction.*** Note: Ignore case, special characters and spacing.
- ***(e): Using the higher order function filter(), define a function filter_long_words() that takes a list of words and an integer n and returns the list of words that are longer than n (in the same order).***
- ***(f): Given a string s and a number n, return a list of size n consisting of tuples with two elements, the first being the correct string in the look-and-say-sequence and the second being a dictionary of character: count pairs.*** For example, if s = "1" and n = 5, the look-and-say-sequence is as follows:

```
1 ("1", {"1" : 1})
11 ("11", {"1" : 2})
21 ("21", {"1" : 1, "2" : 1})
1211 ("1211", {"1" : 3, "2" : 1})
111221 ("111221", {"1" : 4, "2" : 2})
correct output: [("1", {"1" : 1}), ("11", {"1" : 2}), ("21", {"1" :
1, "2" : 1}),
("1211", {"1" : 3, "2" : 1}), ("111221", {"1" : 4, "2" : 2})]
```

In [12]:
```
# Implement part (a) below
def string_repeater(s):
    pass

# Implement part (b) below
def string_occurence_remover(a, b, c):
    pass

# Implement part (c) below
def caeser_decipher(s, n):
    pass

def is_palindrome(s):
    pass

def filter_long_words(l):
    pass

def look_and_say(s):
    pass
```

In [ ]:
```
# Simple Tests for part (a)
assert(string_repeater("Code") == "CCoCodCode")
assert(string_repeater("EECS445") == "EEEEECEECSEECS4EECS44EECS445")


# Simple Tests for part (b)
assert(string_occurence_remover("Boring", "PythonIsBoring", "Fun") == "P
ythonIsFun")
assert(string_occurence_remover("12", "Today is September 12 and 12 is m
y favorite number.", "9") == \
       "Today is September 9 and 9 is my favorite number.")

# Simple Tests for part (c)
assert(caeser_decipher("vjku ku eqfg", -2) == "this is code")
assert(caeser_decipher("h khjd bnlotsdqr", 1) == "i like computers")

# Simple Tests for part (d)
assert(is_palindrome("Rats live on no evil star."))
assert(is_palindrome("On a clover, if alive, erupts a vast pure evil; a
 fire volcano"))
assert(not is_palindrome("Hello, this is Jupyter Notebook speaking."))
assert(not is_palindrome("I am currently in a hands-on lecture."))

# Simple Tests for part (e)
assert(filter_long_words(['a', '', '0', 'a0', 'a0b02030', 'ee', 'cs', 'e
ecs', 'eeccss'], 2) == ['a0b02030', 'eecs', 'eeccss'])
assert(filter_long_words(['1', '2'], 2) == ['1', '2'])
```

Question 3: Classes and Interactive I/O

(a) Define a class which has at least two methods, getString: to get a string from console input and printString: to print the string in upper case. Also write a simple test to check the functionality class methods.

In [ ]:

(b) Write a program able to play the "Guess the number"-game, where the number to be guessed is randomly chosen between 1 and 20.

(Source: http://inventwithpython.com (http://inventwithpython.com)) This is how it should work when run in a terminal:

```
Hello! What is your name?
Valli
Well, Valli, I am thinking of a number between 1 and 20.
Take a guess.
10
Your guess is too low.
Take a guess.
15
Your guess is too low.
Take a guess.
18
Good job, Valli! You guessed my number in 3 guesses!
```

In [2]:
```
import random
class GuessTheNumber():
    # Create a constructor here (__init__ function) that takes two numbe
rs, a minimum and a maximum for
    # the range that guesses can take. Save these in variable min_guess
 and max_guess. Create and
    # initialize a Boolean called incorrect to be true. (Note: Python bo
oleans use capitalization, >T<rue or >F<alse)

    def play():
        print("Hello! What is your name?")
        # Write code to get input from the user and save it into a strin
g variable name.
        print("Well, " + name + ", I am thinking of a number between 1 a
nd 20.")
        answer = random.randrange(self.min_guess, self.max_guess + 1)
        # Write the main loop to collect guesses and check whether it's
 write. Also remember to save the count!
        # Additionally, if the guess is out of range or input is unexpec
ted (like type mismatch or non-numeric input),
        # simply print an error message and break from the loop.
```

In [ ]:
```
# Test out your game!
g = GuessTheNumber(1, 20)
g.play()
```

Question 4: Pandas and Data Exploration

In [100]:
```
# Generate some Data for analysis
from sklearn.datasets import make_classification
X, y = make_classification(1000, n_features=5, n_informative=2,
                           n_redundant=2, n_classes=2, random_state=0)
```

In [94]:
```
# (a) Get a glimpse of the data by making a Pandas DataFrame from the da
ta and then printing the first few and last
# few rows.
```

In [95]:
```
# (b) Plot a boxplot of each column to visualize the distribution of the
 data column values.
```

In [96]:
```
# (c) Try using the description() function of the DataFrame.
```

In [97]:
```
# (d) Install Seaborn if it is not already installed and import it belo
w. Then, perform a pairwise plot using the data.
```

In [98]:
```
# (e) Now try Seaborn's correlation plot (Heatmap).
```

In [102]:
```
# (Optional) Using the first 70% of the data as a training set and the l
ast 30% as a test set, construct a classifier
# and see how well it performs. You will be certainly able to do this at
 the end of the course!
```

Question 6: Numpy Exercises

In [13]:
```
# (a) Write a function that takes in a tuple and a string that can eithe
r
# be 'zero', 'one' or 'gaussian' and correspondingly return a NumPy arra
y that contains those elements. For 'gaussian',
# assume sampling with mean = 0, std = 1.
```

In [14]:
```
# (b) Write a function that returns a n x n identity matrix with n as a
 parameter.
```

In [15]:
```
# (c) Write a function that normalizes a matrix to [0, 1] and returns th
e normalized matrix.
```

In [16]:
```
# (d) Write code that creates a NumPy array and makes it immutable.
```

In [17]:
```
# (e) Write a function that finds the closest value to a given scalar s.
```

In [19]:
```
# (f) Write a function that subtracts the mean of each row from a matrix
 and returns it.
```

In [20]:
```
# (g) Write a function that sorts an array by the nth column and returns
 the sorted array.
```

Question 7: Numpy + First ML algorithm!

In [3]:
```
# (a) Write a function that implements Ordinary Least Squares given an i
nput matrix X and a vector of targets y.
# We will go over the method in the forthcoming lecture, but the equatio
n is given in
# https://en.wikipedia.org/wiki/Linear_regression#Estimation_methods
# Note: Use NumPy here, but do NOT make use of library functions that do
 this for you.
```

In [ ]:
```

```

In [5]:
```
from sklearn.datasets import make_regression
X, y = make_regression(1000, n_features=1, random_state=0)
```

In [8]:
```
# (b) Run your function on the above data and plot the data as well as t
he decision boundary (trendline)
# generated by your classifer using matplotlib.
```

In [ ]:
```

```