**Darien Cortez**

**1: Definitions:**

Using python comments, label all lines that an OOP definition could be applied to.

class Employee:

  empCount = 0

  def \_\_init\_\_(self, name, salary):

     self.name = name

     self.salary = salary

     Employee.empCount += 1

  def displayCount(self):

    print "Total Employee %d" % Employee.empCount

  def displayEmployee(self):

     print "Name : ", self.name,  ", Salary: ", self.salary

emp1 = Employee("Zara", 2000)

emp2 = Employee("Manni", 5000)

emp1.displayEmployee()

emp2.displayEmployee()

print "Total Employee %d" % Employee.empCount

**Answer 1**

# “Employee” is a class

# “empCount = 0”, “self.name = name”, and “self.salary = salary” are Attributes

# “empCount” is a data member (class variable)

# “def\_\_init\_\_(…)”: is a constructor

# “self.name = name” is an Attribute

# “empCount” is mutable

# “displayCount” and “displayEmployee” are methods

# “emp1” and “emp2” are objects of the “Employee” class

# “self” is an instance method

**Answer 2**

# A) States.sort()

# B) def addInOrder(L, x):

#    L.append(x)

#    L.sort()

#    return (L)

#    addInOrder(L, ‘Oklahoma’)

**Answer 3**

# def addPrevious(L):

#    for i in range(len(L)):

#        L[i] = L[i] / (i + 1)

#    return (L)

**Answer 4**

# months = abbr\_months

# for k, m in abbr\_months.keys():

#    m = str(:4)

#    print(k, m)

**Answer 5**

# def miniStats(L):

#    L = L[:]

#    min = L[0]

#    max = L[0]

#    avg = 0

#    for i in range(len(L)):

#        if i < min:

#            min = i

#    for i in range(len(L)):

#        if i > max:

#            max= i

#    for i in range(len(L)):

#        avg = avg + L[i]

#    avg / len(L)

#    tuple = (min, max, avg)

#    return (tuple)

**Answer 6**

# class myPrimes(object):

#    def \_\_checkPrime(self, x):

#        self.x = x

#        if x > 1:

#            for i in range(2, x):

#                if (x % i) != 0:

#                    return x

#                else:

#                    break

#    def addPrime(self, x):

#        self.append(x)

#        return self

#    def removePrime(self, x):

#        self.remove(x)

#        return self

#    def \_\_str\_\_(self):

#        return “Prime Numbers: %d” % (self)

**2: List Functions**

Given the list below:

States = ['Alabama','Illinois','Wyoming','New York', 'Vermont', 'New Hampshire', 'Maine', 'Texas']

**A)** Sort the list

**B)** Add 'Oklahoma' to the list in alphabetical order without sorting the list again. Actually, write a function that would add an item to the list in alphabetical order. Example:

def addInOrder(L):

   #add to the list L in the proper order

   #return your ordered list

   return L

**3: Looping over Lists**

(10 Points)

Using the following list as an example: L = [10,20,30,40,50,60,70,80,90,100] write a function that would divide each value by its index location + 1. Our example list would turn into: L = [10,10,10,10,10,10,10,10,10,10]. Remember NOT to get caught up on these values. Your function should work on any list.

Usage:

L =  [10,20,30,40,50,60,70,80,90,100]

NList = addPrevious(L)

print(NList)

# prints: [10,10,10,10,10,10,10,10,10,10]

Your answer should consist of just the function definition and none of the usage I provided above.

**4: Looping over Dictionaries**

(10 Points)

Given the following dictionary:

months = { 1 : "January",

       2 : "February",

       3 : "March",

       4 : "April",

       5 : "May",

       6 : "June",

       7 : "July",

       8 : "August",

       9 : "September",

       10 : "October",

       11 : "November",

       12 : "December" }

Iterate over this dictionary, and create a new one that only uses the first three letters of the month. Also make the new months all lowercase. Your new dictionary should look like:

abbr\_months = {1:"jan",

       2 :"feb",

       3 :"mar",

       4 : "apr",

       5 : "may",

       6 : "jun",

       7 : "jul",

       8 : "aug",

       9 : "sep",

       10 : "oct",

       11 : "nov",

       12 : "dec" }

To help you look up string slicing and lower.

Your answer should include just the code that loops and creates the new dictionary.

**5: Min and Max**

(10 Points)

* Assume that pythons built in min , max , and sort functions are broken. Write a function that receives a list then traverses the list and returns the min , max, and average values in a tuple.

def miniStats(L):

"""

@Description: Finds the min,max,and average values in a list

@Params: L (list)

@Returns: tuple (int,int,double)

"""

   # Start with a copy of the list so we don’t modify the original.

   L = L[:]

When writing your answer, include the entire function definition (without the comment block).

**6: Prime Class**

Write a class called myPrimes that represents a collection of your prime numbers.

* addPrime :
  + receives a prime number and adds it to your collection of primes
  + it must be checked to make sure it's prime! (should be a private method that does this).
* removePrime:
  + a method will remove a prime from your list
* printPrimes:
  + this method will print your prime numbers out

)