**1.**

**Question:**

You are given an HTML form, which contains the following entries:

<input type="checkbox" class="form" name="checkbox\_2"/>

<input type="checkbox" class="form" name="checkbox\_1"/>

<input type="checkbox" class="form" name="checkbox\_3"/>

...

<input type="checkbox" class="form" name="checkbox\_10"/>

The above form has been submitted using the "POST" method.  
  
Write code that will determine which checkboxes have been checked and print, in a space separated list, the checkbox numbers that were checked.  
  
For example, if check-boxes 3, 5, and 10 are checked, you would print

3 5 10

**Answer:**

*from bottle import route, run, request*

*@route('/')*

*def hello():*

*return '''<!doctype html><html><title> Select Check box </title>*

*<body> <hi>Select Check box</h1>*

*<form action="/get\_check\_box" method="post">*

*<input type="checkbox" class="form" name="checkbox\_1"/>*

*1</br>*

*<input type="checkbox" class="form" name="checkbox\_2"/>*

*2</br>*

*<input type="checkbox" class="form" name="checkbox\_3"/>*

*3</br>*

*<input type="checkbox" class="form" name="checkbox\_4"/>*

*4</br>*

*<input type="checkbox" class="form" name="checkbox\_5"/>*

*5</br>*

*<input type="checkbox" class="form" name="checkbox\_6"/>*

*6</br>*

*<input type="checkbox" class="form" name="checkbox\_7"/>*

*7</br>*

*<input type="checkbox" class="form" name="checkbox\_8"/>*

*8</br>*

*<input type="checkbox" class="form" name="checkbox\_9"/>*

*9</br>*

*<input type="checkbox" class="form" name="checkbox\_10"/>*

*10</br>*

*<input value="select & submit" type="submit" />*

*</form></body></html>'''*

*@route('/get\_check\_box', method='POST')*

*def do\_get\_check\_box():*

*x = [int(i.replace('checkbox\_','')) for i in list(request.forms)]*

*x = [str(i) for i in sorted(x)]*

*x = ' '.join(x)*

*return '<h2> Output:</h2> <br> <br> <p> %s </p>' %x*

*run(host='localhost', port=8080, debug=True)*

**2.**

**Question:**

You are given an array in Python, which contains positive integers and/or recursively nested arrays of positive integers. It may, for example, be initialized as:

*arr = [[141,151,161],2,3,[101,202,[303,404]]]*

Write a function *def MaxArray(arr)* which returns the maximum value contained in *arr* or some array nested within *arr*. In the example, the returned value should be 404.

**Answer:**

*def maxArray(l=None):*

*if not l or l == None:*

*raise Exception("Invalid entry of array!")*

*else:*

*lis = nested\_list\_to\_list(l)*

*return max(lis)*

*def nested\_list\_to\_list(l=None):*

*result = []*

*if l != None:*

*for elem in l:*

*if type(elem) == list:*

*if len(elem) >= 2:*

*for e in nested\_list\_to\_list(elem):*

*result.append(e)*

*else:*

*result.append(elem[0])*

*else:*

*result.append(elem)*

*return result*

*if \_\_name\_\_ == '\_\_main\_\_':*

*x=maxArray([[141,151,161],2,3,[101,202,[303,404]]])*

*#x=maxArray([141,151,161,2,3,101,202,303,40])*

*print x*

**3.**

**Question:**

Write a function declared as *def SplitEmailAddress(address)*, whose argument will contain string data consisting of a valid e-mail address. This function will take the email address as the argument and return a dictionary with two keys: *user* for the username part and *domain* for the domain part of the address. For example, after calling:

*dict = SplitEmailAddress('myuser\_1@mailserver.example.com')*

*dict['user']* should contain the string *myuser\_1*, and *dict['domain']* should contain the string *mailserver.example.com*

**Answer:**

*def splitEmailAddress(address):*

*if type(address) == str and address.find('@') != -1:*

*username, domain = address.split('@')*

*return dict([('user',username),('domain',domain)])*

*else:*

*raise Exception(" Invalid input!")*

*if \_\_name\_\_ == '\_\_main\_\_':*

*dict = splitEmailAddress('myuser\_1@mailserver.example.com')*

*print dict['user']*

*print dict['domain']*

**4.**

**Question:**

Write a function *GetLongestString*, whose arguments are character strings. It should return an integer representing the length of the longest string passed as its argument; for example:

GetLongestString("a", "aaa", "aa") should return 3,

GetLongestString("a", "bcd", "efgh", "ij", "") should return 4.

**Answer:**

*def getLongestStr(a=None):*

*if a != None and hasattr(a, '\_\_iter\_\_') == True and all(isinstance(x,str) for x in a) == True:*

*return max(map(len, a))*

*else:*

*raise Exception('Invalid input!')*

*if \_\_name\_\_ == '\_\_main\_\_':*

*x = getLongestStr(("a", "bcd", "efgh", "ij", ""))*

*#x = getLongestStr((1,4,6,7))*

*print x*

**5.**

**Question:**

Write a function *GetUniqueOnes*, which accepts a single argument. The argument is an array of integers, and the function should return the unique integers separated by commas.  
  
For Example : GetUniqueOnes(arr)   
  
arr = [34,54,67,68,141,151,161,141,54,151,54]  
  
should return   
34,54,67,68,141,151,161

**Answer:**

*def getUniqueOnes(a=None):*

*if a != None:*

*return ','.join(map(str,sorted(list(set(a)))))*

*else:*

*raise Exception('Invalid input!')*

*if \_\_name\_\_ == '\_\_main\_\_':*

*arr = [34,54,67,68,141,151,161,141,54,151,54]*

*x = getUniqueOnes(arr)*

*print x*

**6.**

**Question:**

Write a function *ReadXml(xmlstr)* which accepts an XML string as its only argument. Your function should write the names of the nodes and their values.   
  
For Example :   
  
ReadXml(xmlstr) where   
xmlstr= '<Address><to>James</to><from>Jani</from><heading>Reminder</heading><body>Please check your mail.</body></Address>'   
  
should return   
Address  
to: James  
from: Jani  
heading: Reminder  
body: Please check your mail.  
  
**Note:** As the question requires the output to be displayed in the new/ next line, use "\n" instead of "<br>"

**Answer:**

*from xml.etree.ElementTree import fromstring*

*def readXML(xmlstr):*

*tree = fromstring(xmlstr)*

*display\_content(tree)*

*def display\_content(tree):*

*if tree.tag == 'br':*

*print "\n"*

*elif tree.text == None:*

*print tree.tag*

*else:*

*print "%s:%s"%(tree.tag, tree.text)*

*for child\_node in tree:*

*display\_content(child\_node)*

*if \_\_name\_\_ == '\_\_main\_\_':*

*xmlstr= '<Address><br></br><to>James</to><from>Jani</from><heading>Reminder</heading><body>Please check your mail.</body></Address>'*

*readXML(xmlstr)*

**7.**

**Question:**

Consider a database with one table called "user" having two fields:  
  
"id" (type: INTEGER, PRIMARY KEY)   
"name" (type: VARCHAR(32))  
  
Write a standard SQL query which retrieves the second highest value of "id" from the "user" table. The value returned should be represented using the column name "id".

**Answer:**

*select max(id) as id from users where id < (select max(id) from users);*

**8.**

**Question:**

Write a function *GeneratePassword* which accepts two arguments, an integer and a character string consisting of letters (a-z) and digits (0-9).  
  
When GeneratePassword(5,'abc0123') is called, it should return a random string of 5 characters taken from 'abc0123'.   
  
For Example : GeneratePassword(7,'abczxc012394') could return any of the following outputs :   
2c00acb   
2c23z93   
030b2a4

**Answer:**

*import random*

*def generatePassword(size = 0, char = None):*

*if size != 0 and char != None:*

*passwd = []*

*for i in range(size):*

*passwd.append(random.choice(char))*

*return ''.join(passwd)*

*else:*

*raise Exception('Invalid Input!')*

*if \_\_name\_\_ == '\_\_main\_\_':*

*x = generatePassword(7'abczxc012394')*

*print x*

**9.**

**Question:**

Write a program that outputs the numbers that are divisible by 8 and are between 200 and 600 (inclusive), separated by commas (without spaces or line breaks).

**Answer:**

*def generateNumbers():*

*x = [i for i in range(200,601) if i%8 == 0]*

*return ','.join(map(str,sorted(x)))*

*if \_\_name\_\_ == '\_\_main\_\_':*

*x = generateNumbers()*

*print x*

**10.**

**Question:**

Write a function declared as function ReformatPhoneNumber(number), whose argument will contain string data representing some phone number data (entered by the user). A valid phone number may consist of between 7 and 12 digits (0..9). Assume that in between some adjacent digits there may optionally appear either a single space, or a single hyphen (-). Any other phone number should be considered invalid.

If the phone number is valid, the return value of your function should contain a string containing between 7 and 12 digits, representing the same phone number after removing all hyphens and spaces. If the phone number is invalid, throw a custom Python Exception with the message

"**Invalid phone number**".

The first and the last character of the string should be a number.

For example, after calling ReformatPhoneNumber('012-345 69') the return value should be '01234569'. Calling the function with any of these values: '012345', '-012345 678', '01203- 34566', '123456678875432', '1234x567' should result in an exception.

**Answer:**

*allowed\_string = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '-', ' ']*

*def reformatPhoneNumber(number):*

*#if len(number) in range(7,13)*

*ph\_number = []*

*for i in number:*

*if not i in allowed\_string:*

*raise Exception("Phone Number contains unwanted charactors!")*

*else:*

*if not i in [' ', '-']:*

*ph\_number.append(i)*

*if len(number) in range(7,13):*

*return ''.join(ph\_number)*

*else:*

*raise Exception("Invalid Phone Number!")*

*if \_\_name\_\_ == '\_\_main\_\_':*

*x = reformatPhoneNumber('012-345 69')*

*#x = reformatPhoneNumber('012345')*

*#x = reformatPhoneNumber('-012345 678')*

*#x = reformatPhoneNumber('-012345 6781234567')*

*#x = reformatPhoneNumber('01203- 34566')*

*#x = reformatPhoneNumber('123456678875432')*

*#x = reformatPhoneNumber('1234x567')*

*print x*