PYTHON APPLICATION PROGRAMMING		
MODEL QUESTION PAPER-2		

1	a	Write python program to swap two numbers using functions. (write without using intermediate/temporary variables). Prompt the user for input.
		def swap(a,b):
		  print ('Before swapping a=%s b=%s' %(a,b))
		a,b=b,a
		  print ('After swapping a=%s b=%s' %(a,b))
		
		a=input("Enter a:")
		b=input("Enter b:")
		swap(a,b)

	b	Find the area and perimeter of a circle using functions. Prompt the user for input.
		defcalc(r):
		  import math
		  area=math.pi*r**2
		  perimeter=2*math.pi*r
		  return area,perimeter
		
		print("to find perimeter and area of circle")
		r=float(input("enter radius:"))
		a,b=calc(r)
		print("area:",a)
		print("perimeter:",b)

	c	Explain variable names, keywords, operators, operands and order of operations with examples. Variable Names: • A variable is a name that refers to value. • The value is assigned to variable using equal sign(=). • Variable names can be arbitrarily long. They can contain both letters and numbers, but they cannot start with a number. • Legal to use uppercase letters, but it is a good idea to begin variable names with a lowercase letter. • Underscore character(_) can appear in name • Eg: name = 'CBIT' Keywords: • Python interpreter uses keywords to recognize the structure of the program, and they cannot be used as variable names. • Python reserves 31 keywords. • Like..... and,as ,assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, if, is not, pass, print, while etc., • Eg: word='mango' for word in words: Print word Operators: • Operators are special symbols that represent computations like addition and subtraction. • The values the operator is applied to are called Operands

		• The operators +, -,*, /, and ** performs addition , subtraction, multiplication, division and exponentiation • Eg: 20+32 hour*60+minute hour-1 minute/60 5**2 (5+9)*(15-7) Order of operations: • When more than one operator appears in an expression, the order of evaluation depends on the rules of precedence • For mathematical operations, python follows mathematical convention. • Acronym PEDMAS is a useful way to remember the rules: ✓ Parantheses have the highest precedence and can be used to force an expression to evaluate in the order you want. Since expression in parantheses are evaluated first, 2*(3-1) is 4. ✓ Exponentiation has the next highest precedence, so 2**1+1 is 3 not 4. ✓ Multiplication and division have same precedence ehich is higher than addition and subtraction which also have same precedence. So 2*3-1 is 5 and 6+4/2 is 8. ✓ Operators with same precedence are evaluated from left to right. So the expression 5-3-1 is 1

2	a	Write pythonic code to solve the quadratic equation ax**2+bx+c=0 by getting input for coefficients from the user.
		Solution:
		import cmath
		# To take coefficient input from the users
		a = float(input('Enter a: '))
		b = float(input('Enter b: '))
		c = float(input('Enter c: '))
		
		# calculate the discriminant
		d = (b**2) - (4*a*c)
		
		# find two solutions
		sol1 = (-b-cmath.sqrt(d))/(2*a)
		sol2 = (-b+cmath.sqrt(d))/(2*a)
		
		print('The solution are {0} and {1}'.format(sol1,sol2))

	b	Write a function called **is_palindrome** that takes a string argument and returns True if it is a palindrome and False otherwise. Use built in function to check the length of a string. Prompt the user for input.
		Solution:
		word = raw_input('Want to see if it is a palindrome?\n')
		def is_palindrome(word):
		if len(word) <= 2 and word[0] == word[-1]:
		print 'True'
		elif word[0] == word[-1]:
		is_palindrome(word[1:-1])
		else:
		print 'False'
		is_palindrome(word)

		check the length of a string >>>Word=input('enter a string':) >>>len(word)
	c	Explain syntax errors and logic errors. Write pythonic code which prompts the user for a Celsius temperature. Convert the temperature to Fahrenheit and print out the converted temperature. Use try and except so that your program handles non-numeric input gracefully by printing a message and exit the program. Solution: **Syntax errors:** Syntax error means that violated the grammar rules of python. Python does its best to point right at the line and character where it noticed it was confused. it indicates in a syntax error may just be a starting point for investigation. **Logic errors:** A logic error is when program has good syntax but there is a mistake in the order of the statements or perhaps a mistake in how the statements relate to one another. **Celcius to Fahrenheit** inp=input('Enter celcius temperature:') try: celsius=float(inp) fahrenheit = (celsius * 1.8) + 32 print(fahrenheit) except: print('please enter a number')

3	a)	"Strings in python are immtable". Explain this statement with example. Write pythoniccode to find the factorial of any number entered through the keyboard.
		Solution:
		>>>greeting='Hello, world!'
		>>>greeting[0]='J'
		The above expression gives typeerror: object does not support item assignment
		The reason for error is immutable, which means existing strings cant be changed.
		The best can be done is create a new string that is a variation on the original:
		>>> greeting='Hello, world!'
		>>>new_greeting='J'+ greeting [1:]
		>>> print new_greeting
		Jello, world!
		This example concatenates a new first letter onto a slice of greeting. It has no effect on the original string.
		Factorial of number:
		number=int(input('enter a number to calculate the fact'))
		fact=1
		while number>0:
		fact=fact*number
		number=number-1
		print("factorial is",fact)

	b)	A number with more than one digit is input through the keyboard. Write pythonic code to reverse the digits in the number and find the sum of all the digits in the reversed number. Solution: Number=int(input('please enter number:'))

		Reverse=0 While(number>0): Remainder=number%10 Reverse=(Reverse*10)+Remainder Number=Number/10 Print("\n reverse of entered number is = %d" %Reverse)

	c)	Explain the following string methods in detail a) upper() and b) find(). Write a python program to check whether a number is prime or not using while loop and print appropriate messages.
		Solution:
		
		Upper(): It takes a string and returns a new string with all uppercase letters
		>>>word= 'banana'
		>>>new=word.upper()
		>>>print(new)
		
		Find(): searches for the position of one string within another
		>>>word='banana'
		>>>index=word.find('a')
		>>>print(index)
		
		number=int(input('enter a number:'))
		i=2
		prime=True
		while i<=int(math.sqrt(number)):
		if number%i==0:
		prime=False
		break
		i=i+1
		if number<2:
		prime=False
		if prime:
		print(number,"is a prime number")
		else:
		print(number,"Is not a prime number")

4	a	Explain break and continue statements with examples in python. Write pythonic code to multiply two matrices using nested loops and also perform transpose of the resultant matrix. Solution: • Python break statement. It terminates the current loop and resumes execution at the next statement, just like the traditional break statement in C. The most common use for break is when some external condition is triggered requiring a hasty exit from a loop. The break statement can be used in both while and for loops. • Python continue statement. It returns the control to the beginning of the while loop.. The continue statement rejects all the remaining statements in the

		current iteration of the loop and moves the control back to the top of the loop.
		The continue statement can be used in both while and for loops.
		`# Program to multiply two matrices using nested loops`
		`# take a 3x3 matrix`
		`A = [[12, 7, 3],`
		`[4, 5, 6],`
		`[7, 8, 9]]`
		
		`# take a 3x4 matrix`
		`B = [[5, 8, 1, 2],`
		`[6, 7, 3, 0],`
		`[4, 5, 9, 1]]`
		
		`result = [[0, 0, 0, 0],`
		`[0, 0, 0, 0],`
		`[0, 0, 0, 0]]`
		`# iterating by row of A`
		`for i in range(len(A)):`
		`# iterating by coloum by B`
		`for j in range(len(B[0])):`
		
		`# iterating by rows of B`
		`for k in range(len(B)):`
		`result[i][j] += A[i][k] * B[k][j]`
		`for r in result:`
		`print(r)`

	b	Write pythonic code to read a file. Count and print the lines that start with the word "**From**". Prompt the user for the file name. also use try/except to handle bad file names. Explain format operator with examples in python.
		Solution:
		`fhand=open('mbox.txt')`
		`for line in fhand:`
		`if line.startswith('From')`
		`print(line)`
		
		Format operator:
		The format operator, % allows us to construct strings, replacing parts of the strings with the data stored in variables. The first operand is the format string, which contains one or more format sequence that specify how the second operand is formatted. The result is string.
		eg: >>> camels=42
		>>>'%d' % camels
		'42'
		>>>'I have spotted %d camels.' % camels
		'I have spotted 42 camels.'

	c	Write pythonic code that iteratively prompts the user for input. It should continue until

		the user enters 'done' and return the average value.
		
		sum=0
		count=0
		average=0
		while(True):
		  try:
		  line=input('enter a number')
		  if(line=='done'):
		  break
		  value=float(line)
		  sum=value+sum
		  count=count+1
		  average=sum/count
		except:
		  print("invalid input")
		
		print('Avg',average)

5	a)	Why do you need regular expressions in python? Consider a file **"ait.txt".** Write a python program to read the file and look for lines of the form
		X-DSPAM-Confidence:0.8475
		X-DSPAM-Probability:0.458
		Extract the number from each of the lines using a regular expression. Compute the average of the numbers and print out the average.
		Solution:
		import re
		hand=open('ait.txt')
		for line in hand:
		  line=line.rstrip()
		ifre.findall('^X\S*: ([0-9.]+)',line)
		  print(line)
		average of numbers:
		sum=0
		count=0
		average=0
		while (True):
		  try:
		  line=input('enter a number')
		  if(line=="done")
		  break
		  value=float(line)
		  sum=value+sum
		  count=count+1
		  average=sum/count
		  except:
		  print("invalid input")
		print (sum,count,average)

	b)	How are dictionaries and tuples used together? Demonstrate the use of tuple assignment with dictionaries to traverse the keys and values of dictionary.

		Solution: Dictionaries have a method called items that returns a list of tuples, where each tuple is a key value pair. >>>d={'a':10, 'b':1,'c':22} >>>t=d.items( ) >>>print t [(' a',10),('c':22),('b',1)] tuple assignment with dictionaries to traverse the keys and values >>>d={'a':10, 'b':1,'c':22} >>>t=d.items( ) . . for key,val in d.items(): print val,key ouput: 10 a 22 c 1 b

6	a	Write pythonic code to create a function called most_frequent that takes a string and prints the letters in decreasing order of frequency using dictionary.
		Solution:
		defmost_frequent(string):
		d=dict()
		for key in string:
		if key not in d:
		d[key]=1
		else:
		d[key]+=1
		return d
		print (most_frequent('aabbbc'))

	b	Consider the string 'brontosaurus'. Write pythonic code that implements and returns the functionality of histogram using dictionaries for the given string. Also, write the function print_hist to print the keys and their values in alphabetical order from the values returned by the histogram function.
		Solution:
		word='brontosaurus'
		d=dict( )
		for c in word:
		if c not in d:
		d[c]=1
		else:
		d[c]+=1
		print(d)

```
>>> L = 'abracadabra'
>>> histogram(L)
{'a': 5, 'b': 2, 'c': 1, 'd': 1, 'r': 2}


def histogram(L):
    d = {}
    for x in L:
        if x in d:
            d[x] += 1
        else:
            d[x] = 1
    return d
```

7 a) write an __init__ method for the point class that takes x and y as optional parametres and assigns them to corresponding attributes. Write an add method for Points that works with either a Point object or a tuple. If the second operand is a Point, the method should return a new Point whose x coordinate is the sum of the x coordinates of the operands, and likewise for the y coordinates. If the second operand is a tuple, the method should add the first element of the tuple to the x coordinate and the second element to the y coordinate, and return a new Point with the result.

Solution:

```
def __add__(self, other):
                point_ = Point()
                if isinstance(other, Point):
            point_.x += self.x + other.x
            point_.y += self.y + other.y
                    return point_
            elif type(other) == tuple:
            point_.x += self.x + other[0]
            point_.y += self.y + other[1]
                return point_

            def __radd__(self, other):
                return self.__add__(other)

            def __str__(self):
                return "(%s, %s)" % (self.x, self.y)

              point1 = Point(1, 6)
              point2 = (5, 2)
              point3 = point1 + point2
              point4 = point2 + point1
              print point3, point4
```

b) Write a function called distance that takes two points as arguments and returns the

		distance between them.
		Solution:
		import math
		class Point(object):
		point_one = Point()
		point_two = Point()
		point_one.x, point_one.y = 6.0, 1.0
		point_two.x, point_two.y = 2.0, 6.0
		
		def distance(p1, p2):
		delta_x = p2.x - p1.x
		delta_y = p2.y - p1.y
		return math.sqrt(delta_x ** 2 + delta_y ** 2)
		
		print "The distance between point one at (%g,%g)" % (point_one.x, point_one.y)
		print "and point two at (%g,%g)" % (point_two.x, point_two.y)
		print "is %.3f" % distance(point_one, point_two)

8	a)	Write pythonic code to compute the end time of movie by specifying the start time and duration by considering all relevant conditions.
		Solution:
		import time
		start_time = time.time()
		# your script
		elapsed_time = time.time() - start_time
		time.strftime("%H:%M:%S", time.gmtime(elapsed_time))

	b)	What is operator overloading? Write a pythonic code to overload "+","-" and "*" operators by providing the methods __add__,__sub__ and __mul__. Solution: By defining other special methods, you can specify the behavior of operators on p # #inside class Time: def __add__(self, other): seconds = self.time_to_int() + other.time_to_int() return int_to_time(seconds) def__sub__(self,other): seconds = self.time_to_int() – other.time_to_int() return int_to_time(seconds) def__mul__(self,other): seconds = self.time_to_int() * other.time_to_int() return int_to_time(seconds)

		And here is how you could use it:
		>>> start = Time(9, 45)
		>>> duration = Time(1, 35)
		>>> print(start + duration)
		>>> print(start - duration)
		>>>print(start * duration)
		When you apply the + operator to Time objects, Python invokes __add__. When you print the result, Python invokes __str__.
		So there is a lot happening behind the scenes! Changing the behavior of an operator so that it works with programmer-defined types is called operator overloading.

9	a)	State a need for urlib in python. Explain why data is retrieved in blocks. Write pythonic code to read any sized binary file using urlib without using up all of the memory you have in your computer. Solution: import urlib img=urlib.urlopen('http://data.pr4e.org/cover.jpg') fhand=open('cover.jpg','w') size=0 while True: info=img.read(10000) if len(info)<1:break size=size+len(info) fhand.write(info) print(size,'characters copied' fhand.close())
	b)	Give an example to construct a simple web page using HTML. State the need for BeautifulSoup library in python. Write pythonic code to read a web page usingurlib and then BeautifulSoup library to extract the href attributes from the anchor(a) tags. Solution: import urlib from BeautifulSoup import * url = raw_input('enter') html=urlib.urlopen(url).read() soup=BeautifulSoup(html) tags=soup('a') for tag in tags: print tag.gte('href', None)
10	a)	Define XML. Construct a simple XML document and represent it with a diagram. Write pythonic code to loop through XML nodes in the document.

		Solution: XML is used for exchanging document style data Simple XML document <person> <name>chuck</name> <phone type='intl'> +12345890</phone> <email hide="yes"/> </person> loop through XML nodes in the document import xml.etree.ElementTree as ET input=' ' <stuff> <users> <user x="2"> <id>001</id> <name>chuck</name> </user> </users> </stuff>' ' stuff=ET.fromstring(input) lst=stuff.findall('users/user') print 'user count:', len(lst) for item in lst: print 'Name', item.find('name').text print 'Id', item.find('id').text print 'Attribute', item.get('x')

	b)	Define JSON. Construct a simple JSON document. Bring out the differences between XML and JSON. Write pythonic code to parse JSON document.
		Solution:
		JSON : object and array format used in the javascript language
		It maps directly to some combinations of lists and dictionaries
		Natural format for two cooperating programs exchange data
		Simple JSON document
		{
		"name": "chuck",
		"phone": {
		"type": "intl",
		"number":"+123456789"
		},
		"email" : {
		"hide": "yes"
		}
		}
		parse JSON document
		import json

```
input = ' ' '
[
  { "id" : "001"
  "x" : "2"
   "name" : "chuck"
  },
  { "id" : "009"
    "x" : "7"
    "name" : "Brent"
   }
] ' ' '
Info = json.loads(input)
Print 'user count:', len(info)

for item in info:
 print 'Name', item['name']
 print 'Id', item['id']
 print 'Attribute', item['x']
```