**THIRD INTERNAL ASSESSMENT TEST KEY**

**Subject code and Name: GE8151, PROBLEM SOLVING AND PYTHON PROGRAMMING:**

**DATE:05.12.2018 MARKS:100**

**TIME:03.00-04.30PM DURATION:01:30HRS**

**PART-A:**

**ANSWER ALL THE QUESTIONS: 5x2=10**

**1) Difference between text file and binary file**

Ans:

In the text mode, we get strings when reading from the file.

On the other hand, binary mode returns bytes and this is the mode to be used when dealing

with non-text files like image or exe files.

**2)Exception in python?**

Ans:

Python has many built-in exceptions which forces your program to output an error when

something in it goes wrong.

When these exceptions occur, it causes the current process to stop and passes it to the calling

process until it is handled. If not handled, our program will crash.

**3)Comment line with example?**

Ans:

Comments indicate Information in a program that is meant for other programmers (or

anyone reading the source code) and has no effect on the execution of the program. In Python,

we use the hash (#) symbol to start writing a comment.

Eg:

#This is a comment

#print out Hello

print('Hello')

**4)Tupule**

Ans:

Tuple is an ordered sequence of items same as list.The only difference is that tuples are

immutable. Tuples once created cannot be modified.

Tuples are used to write-protect data and are usually faster than list as it cannot change

dynamically.

It is defined within parentheses () where items are separated by commas.

Eg:

>>> t = (5,'program', 1+3j)

>>> a=(5,7.9,10)

**5)Module**

Ans:

A module is a file containing Python definitions and statements. The file name is the

module name with the suffix .py appended. A module can contain executable statements as well

as function definitions. Each module has its own private symbol table, which is used as the

global symbol table by all functions defined in the module. Modules can import other modules.

**PART-B: 4x10=40**

**ANSWER ANY FOUR QUESTIONS:**

**1)Sorting:**

Ans:

The sort() method sorts the elements of a given list in a specific order - Ascending or Descending. The syntax of sort() method is: list.sort(key=..., reverse=...) Alternatively, you can also use Python's in-built function sorted() for the same purpose.

**(i) Program for Selection sorting:**

def selectionSort(alist):

for fillslot in range(len(alist)-1,0,-1):

positionOfMax=0

for location in range(1,fillslot+1):

if alist[location]>alist[positionOfMax]:

positionOfMax = location

temp = alist[fillslot]

alist[fillslot] = alist[positionOfMax]

alist[positionOfMax] = temp

alist = [54,26,93,17,77,31,44,55,20]

selectionSort(alist)

print(alist)

**Output:**

[17, 20, 26, 31, 44, 54, 55, 77, 93]

**(ii) Program for Insertion sorting:**

def insertionSort(alist):

for index in range(1,len(alist)):

currentvalue = alist[index]

position = index

while position>0 and alist[position-1]>currentvalue:

alist[position]=alist[position-1]

position = position-1

alist[position]=currentvalue

alist = [54,26,93,17,77,31,44,55,20]

insertionSort(alist)

print(alist)

**Output:**

[17, 20, 26, 31, 44, 54, 55, 77, 93]

**2)List with Built-in examples:**

Ans:

A list is a sequence of values. In a string, the values are characters; in a list, they can

be any type. The values in a list are called elements or sometimes items.

There are several ways to create a new list; the simplest is to enclose the elements in

squarebrackets ([ and ]):

[10, 20, 30, 40]

['crunchy frog', 'ram bladder', 'lark vomit']

The first example is a list of four integers. The second is a list of three strings. The

elements of a list don’t have to be the same type. The following list contains a string, a float,

an integer, and (lo!) another list:

['spam', 2.0, 5, [10, 20]]

A list within another list is nested. A list that contains no elements is called an empty

list; you can create one with empty brackets, [].

**Built in functions:**

1.Python float() returns floating point number from number, string

2.Python len() Returns Length of an Object

3.Python max() returns largest element

4.Python min() returns smallest element

5.Python map() Applies Function and Returns a List

6.Python next() Retrieves Next Element from Iterator

**3)Exception handling:**

Ans:

Python has many built-in exceptions which forces your program to output an error when

something in it goes wrong.

When these exceptions occur, it causes the current process to stop and passes it to the calling

process until it is handled. If not handled, our program will crash.

For example, if function A calls function B which in turn calls function C and an exception

occurs in function C. If it is not handled in C, the exception passes to B and then to A.

If never handled, an error message is spit out and our program come to a sudden, unexpected halt.

**3 programs:**

**(i) try....except statement:**

try:

a= int(input("First Number: "))

b= int(input(" Second Number: "))

result=a/b

print(" Result= ", result)

except zeroDivisionError:

print(" Division by zero")

else:

print("Seccessful Division")

**Output:**

First Number:10

Second Number:0

Division by Zero

**(ii) except clause with no exceptions:**

try:

a= int(input("First Number:"))

b=int(input("Second Number:"))

result = a/b

Print("Result=", result)

except:

print ("Error occured")

else:

print("Successful Division")

**Output:**

First Number:2

Second Number:0

Error Occured

**(iii)except clause with multiple exception:**

try:

a= int(input("First Number:"))

b=int(input("Second Number:"))

result = a/b

Print("Result=", result)

except(ZeroDivisionError, TypeError):

print("Error Occured")

else:

print("Successful Division")

**Output:**

First Number:10

Second Number:0

Error Occured.

**4)File:**

File is a named location on disk to store related information. It is used to permanently store

data in a non-volatile memory (e.g. hard disk).

Since, random access memory (RAM) is volatile which loses its data when computer is

turned off, we use files for future use of the data.

When we want to read from or write to a file we need to open it first. When we are done, it

needs to be closed, so that resources that are tied with the file are freed.

Hence, in Python, a file operation takes place in the following order.

1. Open a file

2. Read or write (perform operation)

3. Close the file

**(i)program for reading a file:**

**Syntax:**

fileObject.write(str);

**Program:**

file=open ("new2.txt, "w")

file.write(" United Global Publishers\n")

file.write("Chennai\n\n")

file.write("Tamilnadu\n")

**Output:**

United Global Publishers

Chennai

Tamilnadu

**(ii)To read a file:**

**Syntax:**

fileobject.readline([size])

**Program:**

>>>file.readline()

'United Global Publishers\n'

>>>file.readline()

'Chennai\n'

>>>file.readline()

'Tamilnadu\n'

>>>file.readline() #Empty String

' '

**5)Merge Sorting:**

**(i)Concept:**

Merge Sort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. The merge() function is used for merging two halves. The merge(arr, l, m, r) is key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted sub-arrays into one.

**(ii)program:**

def mergeSort(nlist):

print("Splitting ",nlist)

if len(nlist)>1:

mid = len(nlist)//2

lefthalf = nlist[:mid]

righthalf = nlist[mid:]

mergeSort(lefthalf)

mergeSort(righthalf)

i=j=k=0

while i < len(lefthalf) and j < len(righthalf):

if lefthalf[i] < righthalf[j]:

nlist[k]=lefthalf[i]

i=i+1

else:

nlist[k]=righthalf[j]

j=j+1

k=k+1

while i < len(lefthalf):

nlist[k]=lefthalf[i]

i=i+1

k=k+1

while j < len(righthalf):

nlist[k]=righthalf[j]

j=j+1

k=k+1

print("Merging ",nlist)

nlist = [14,46,43,27,57,41,45,21,70]

mergeSort(nlist)

print(nlist)

Splitting [14, 46, 43, 27, 57, 41, 45, 21, 70]

Splitting [14, 46, 43, 27]

Splitting [14, 46]

Splitting [14]

Merging [14]

Splitting [46]

-------

Merging [14, 21, 27, 41, 43, 45, 46, 57, 70]

[14, 21, 27, 41, 43, 45, 46, 57

**Output:**

[14, 21, 27, 41, 43, 45, 46, 57, 70].

×End×