CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assesment Test - II

Sub: Programming Using Python Code: 18MCA32

Date: 16.11.2019 Duration: 90 mins Max Marks: 50 Sem: III Branch: MCA

Answer Any One FULL Question from each part.

			Marks	OBE CO	OBE RBT
		Part - I			
1	(a)	Explain any 5 string functions with examples.	[10]	CO2	L2
		OR			
2	(a)	What is list in python? Describe its memory model and show the state of memory when a list element is modified. Explain in brief any three methods for lists	[10]	CO3	L2
		Part – II			
3	(a)	Consider the following list:li = [1,7,9,12,16]. Give outputs of the following commands: i) li[0:3] ii) li[0:-1] (iii) li[::-1] iv) li[-1:-4] v) li[:] vi) li[:4]	[6]	CO3	L3
	(b)	Write a Python program to check if a given string is a palindrome.	[4]	CO1	L3
		OR			
4	(a)	Describe the different ways of opening a file explain with clear example.	[6]	CO4	L2
	(b)	What are the different operations that can be done on files.	[4]	CO4	L2
		ART - III			
5	(a)	Write a python program to check whether a given number is prime or not, using for-else statement.	[5]	CO2	L3
	(b)	Write a Python program to read a string with punctuations and print the same without punctuations.	[5]	CO3	L3
		OR			
6	(a)	Consider the list qrty = {5,4,7,3,6,2,1} and white the Python code to perform the following operation without using built in methods: i) Insert an element 9 at the beginning of the list ii) Insert an element 8 at the end of the list iii) Insert an element 8 at the index position 3 of the list iv) Delete an element at the beginning of the list v) Delete an element at the end of the list vi) Delete an element at index position 3 vii) Print all the elements in reverse order viii) Delete all elements of the list	[10]	CO3	L4
		Part – IV			
7	(a)	Explain the different looping constructs in Python using examples	[10]	CO2	L2
		OR			
8	(a)	Discuss about the four techniques for reading files in detail.	[10]	CO4	L3
		Part – V			
9	(a)	What are the two ways of importing a module? Which one is more beneficial? Explain.	[4]	CO2	L2
	(b)	Explain how code in Python is tested semi-automatically.	[6]	CO5	L3
		OR			
10	(a)	What is a list of lists? Give one example along with memory model.	[5]	CO3	L3
	(b)	Give differences between lists and strings.	[5]	CO3	L2

Q 1 (a) Explain any 5 string functions with examples.
Ans: swapcase(...)
| S.swapcase() -> string
|
| Return a copy of the string S with uppercase characters
| converted to lowercase and vice versa

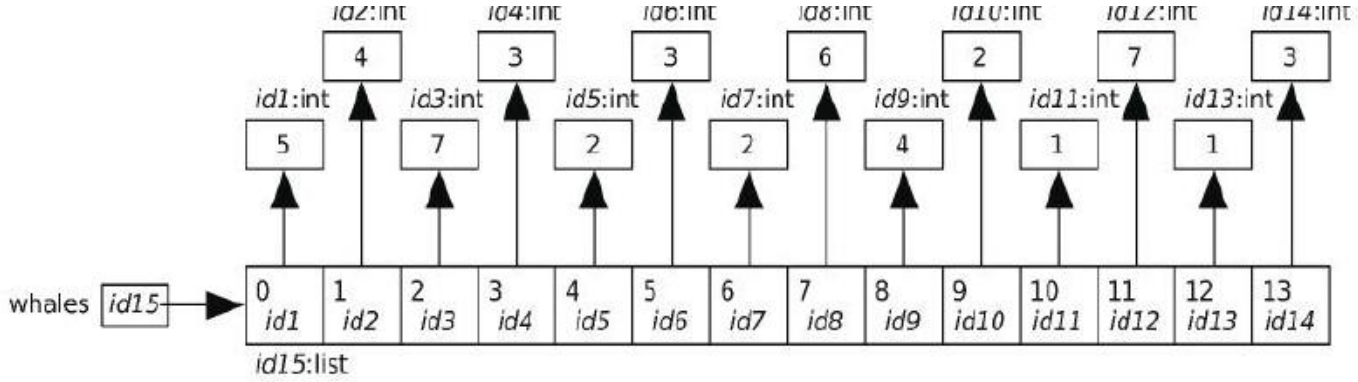
strip(...)
| S.strip([chars]) -> string or unicode
|
| Return a copy of the string S with leading and trailing
| whitespace removed.
| If chars is given and not None, remove characters in chars instead.

| startswith(...)
| S.startswith(prefix[, start[, end]]) -> bool
|
| Return True if S starts with the specified prefix, False otherwise.
| With optional start, test S beginning at that position.
| With optional end, stop comparing S at that position.
| prefix can also be a tuple of strings to try.

| split(...)
| S.split([sep [,maxsplit]]) -> list of strings
|
| Return a list of the words in the string S, using sep as the
| delimiter string. If maxsplit is given, at most maxsplit
| splits are done. If sep is not specified or is None, any
| whitespace string is a separator and empty strings are removed
| from the result.

format(...)
| S.format(*args, **kwargs) -> string
|
| Return a formatted version of S, using substitutions from args and kwargs.
| The substitutions are identified by braces ('{' and '}').

Q 2 (a) What is list in python? Describe its memory model and show the state of memory.
Ans:
A list is class representing a collection of heterogeneous elements stored in contiguous memory locations. The list elements are enclosed within [] and separated from each other by comma. E.g. li=[1,2,3] is a list of 3 elements. An empty list is represented by []. A list `[5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]` is represented in the memory as



In the above list, variable whales refers to a list with 14 elements. The list itself is an object but it also contains the memory address of the 14 elements. Note the addresses of consecutive elements in the list are stored contiguously but the values themselves might not be consecutive.

The general form of a list expression is as follows:

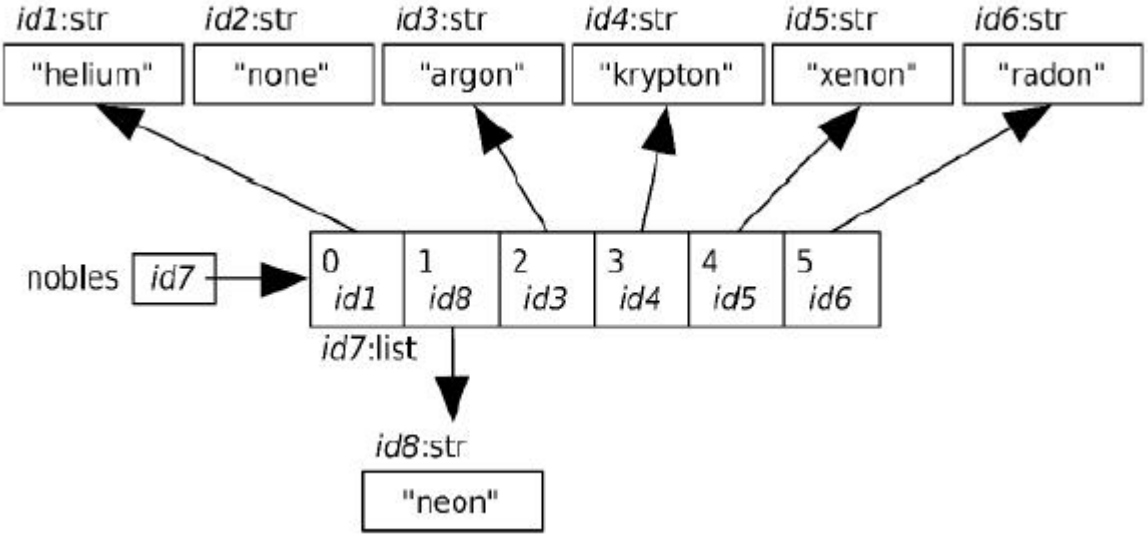
```
[«expression1», «expression2», ... , «expressionN»]
```

The items in the list are ordered and each item has an index indicating its position in the list. The index starts from 0 for the first item, 1 for the second and so on. To refer to an item we put the index in brackets after a reference to the list like whales[3] - refers to the fourth element in the list which is =3.
We can use only those indices that are in the range from zero up to one less than the length of the list. Hence for list whales the max index is 13. To acess the list from the end negative rangesmay be given. whales[-1]=whales[13]=3

Modifying list

```
>>> nobles = ['helium', 'none', 'argon', 'krypton', 'xenon', 'radon']
```

Changing the second element nobles[1]='neon' will result in memory modification as shown



(b) when a list element is modified. Explain in brief any three methods for lists.
Ans:

Three list methods:
a) extend - attaches another list to the end of current list.
e.g.
l=[1,2,3]
l1=['a','b','c']
l.extend(l1)

Output:
l=[1,2,3,'a','b','c']

b) append - attaches a single element to the end of the list
l=[1,2,3]
l.append('a')

Output:
l=[1,2,3,'a']

c) remove - remove the element specified
l=[1,2,3]
l.remove(2)

Output
l=[1,3]

Q 3 (a) Consider the following list:li = [1,7,9,12,16]. Give outputs of the following commands:
i) li[0:3] ii) li[0:-1] (iii) li[::-1] iv) li[-1:-4] v) li[:] vi) li[:4]
Ans: i) [1,7,9] ii) [1,7,9,12] iii) [16,12,9,7,1] iv) [] v) [1,7,9,12,16] vi)) [1,7,9,12] vii) li=[]

(b) Write a Python program to check if a given string is a palindrome.
Ans: s=input('Please enter the string to be checked')
rev=s[::-1]
if s==rev:
    print 'palindrome'
else:
    print 'not palindrome'

Q 4 (a) Describe the different ways of opening a file explain with clear example.
Ans: Opening a file

Method 1: using the open function - the function takes two arguments - the file name and the mode to open the file.
Example fp=open('abc.txt','r')

The above statement opens a file called abc.txt in read mode and returns . Files can be opened in three modes
'r' - read mode - for reading from the file. The file cursor is positioned at the beginning of the file. The file is expected to exist. An error is given if the file does not exist
'w' -write mode - for writing into file. If the file does not exist then a new file with the name is created. If the file exists then the contents are erased and cursor is placed at the beginning of the file.
'a' -append mode - for appending into file. The cursor is positioned at the end of file if the file exists and a write operation writes to the end of file. If the file does not exist then a new file with the name is created.

After the file operations the file needs to be closed by using fp.close()

Method 2: Using with .. as ..:

using open(<<filename>>,<<mode>>) as <<file_handle>>:
    <<body>>

Example:
using open('file1.txt','r') as fp:
    for line in fp:
        print line

The advantage of this method is that on exit of the with block the file automatically is closed and the programmer need not close it explicitly.

(b) What are the different operations that can be done on files.
Ans: i) Opening a file using function open(Format explained in Q6(a)
ii) closing a file - Format <<file handle>>.close() - Must be done when the use of file is over
iii) read ,readline,readlines - Detailed in Q6(a)
iv) seek - used to move the file cursor to different locations
Format - seek(<<number of steps>>,<<relative to>>)
seek(...)
  seek(offset[, whence]) -> None. Move to new file position.

  Argument offset is a byte count. Optional argument whence defaults to
  0 (offset from start of file, offset should be >= 0); other values are 1
  (move relative to current position, positive or negative), and 2 (move
  relative to end of file, usually negative, although many platforms allow
  seeking beyond the end of a file). If the file is opened in text mode,
  only offsets returned by tell() are legal. Use of other offsets causes
  undefined behavior.
  Note that not all file objects are seekable.
v) tell:
tell(...)
  tell() -> current file position, an integer (may be a long integer).

Q 5 (a) Write a python program to check whether a given number is prime or not, using for-else statement.
Ans:
```
num=input()

for i in range(2,int(num**0.5)):
    if num%i==0:
        print str(num) +' is composite'
        break
else:
    print str(num)+' is prime'
```

(b) Write a Python program to read a string with punctuations and print the same without punctuations.
Ans:
```
punc = "!?.',"
for i in punc:
    s=s.replace(i,'')
print 'After removing punctuation : '+s
```

Q 6 (a) Consider the list qrty = {5,4,7,3,6,2,1} and white the Python code to perform the following operation without using built in methods:

i) Insert an element 9 at the beginning of the list
ii) Insert an element 8 at the end of the list
iii) Insert an element 8 at the index position 3 of the list
iv) Delete an element at the beginning of the list
v) Delete an element at the end of the list
vi) Delete an element at index position 3
vii) Print all the elements in reverse order
viii) Delete all elements of the list

Ans:
```
qrty=[5,4,7,3,6,2,1]
qrty=[9]+qrty
print 'After inserting 9 '+str(qrty)
qrty=qrty+[8]
print 'After inserting 8 at end '+str(qrty)
qrty.insert(3,8)
print 'After inserting 8 at index 3 '+str(qrty)
qrty=qrty[1:]
print 'After deleting from beginning '+str(qrty)
qrty.pop()
print 'After deleting from end '+str(qrty)
qrty.pop(3)
print 'After deleting at index 3 '+str(qrty)
qrty.reverse()
print 'Elements in reverse order '+str(qrty)
del qrty[:]
print 'After deleting all elements '+str(qrty)
```

Q 7 (a) Explain the different looping constructs in Python using examples.

Ans: Loops allow repetition of tasks several times. There are two types of loops in Python.
i) for loops, ii) while loops

For Loop

For loop can be used to do things with each element of a sequence in turn. The format is

```
for «variable» in «list»:
    «block»
```

A for loop is exected as follows:

- The loop variable is assigned the first item in the list, and the loop block - the body of the for loop - is executed
- The loop variable is then assigned the second item in the list and the loop body is executed again.
- Finally the loop variable is assigned the last item of the list and the loop body is executed one last time

A block is a sequence of one or more statements. Each pass through a block is called an iteration. At the start of an iteration the next item in the list is assigned to loop variable.

For example:

```
lst=[1,2,3,4]
for i in lst:
    print 2*i
```

Output:

```
2
4
```

6
8

The above code assigns to i the values in the list, one value in each iteration . The body of the loop prints the double of the loop variable.
If in turn we need to double the values in the list itself then we can iterate on the index of the for loop using range function. The range function generates a sequence of values between a given range. The values include the start value and exclude the stop value. By default, rangeg generates numbers that successively increase by one - i.e. step size is one. A different step size can be specified with an optional third parameter.

The code is as follows:

```
lst=[1,2,3,4]
for i in range(len(lst)):
        lst[i] *= 2
print lst
```

Output:

[2,4,6,8]

In the above code the loop variable i takes on values from 0 to 3(one less than length of list). In the loop body the list values are modified and finally the values are printed.

It is possible to iterate over a list, string, set, tuple or dictionary in a for loop.

<u>while loop</u>

For loops are useful only if the number of iterations are known. In situations when this information is not known in advance a while loop can be used. The general form of while loop is :

```
while <<expr>>:
    <<block>>
```

The while loop expression is called the loop condition. When Python executes while loop, it evaluates the expression . If the expression evaluates to False then the loop execution stops .If the expression evaluates to True then the loop body is executed once and then again the expression is evaluated. This is repeated - expression, body, expression , body and so on until the expression evaluates to False.
An example program :

```
>>> rabbits = 3
>>> while rabbits > 0:
...     print(rabbits)
...     rabbits = rabbits - 1
...
3
2
1
```

In the above code first the expression rabbits>0 is evaluated initially. Since it is true the value of rabbits is printed and its value is decremented. The expression is then again checked. This continues till the value of variable rabbits becomes 0 in which case the expression evaluates to False and the control is transferred out of the loop.

else with loop

An else clause can be added to both for and while loop. The body of else is executed whenever the loop condition becomes False and the loop is exited. The else body is not executed if the control comes out of the loop because of break or exception. Example

```
for i in range(2):
        print(i)
else:
        print("Finished")
```

The output:

```
0
1
Finished
```

Whereas for the following code the else body is not executed since it exits due to break

```
for i in range(10):
        print(i)
        if i%7 == 0:
                break
else:
        print("Finished")
```

Output:

```
0
1
2
3
4
5
```

6
7

Nested Loop

The block of statements in a loop can contain another loop. In the code below the inner loop is executed for each item in the outer loop.

```
outer = ['Li', 'Na', 'K']
inner = ['F', 'Cl', 'Br']
for metal in outer:
    for halogen in inner:
        print(metal + halogen)
```

Output

LiF
LiCl
LiBr
NaF
NaCl
NaBr
KF
KCl
KBr

This is useful in dealing with nested list.

Q 8 (a) Discuss about the four techniques for reading files in detail.
Ans: Methods of Reading from file

a) read - format - read(<<number of characters>>)
Example -
fp = open('file1.txt','r')
fp.read(10)

This command specifies that 10 characters from the file - file1.txt needs to be read

b) readline - for reading one line from the file i.e. till the next newline

Example -
fp = open('file1.txt','r')
line=fp.readline()
print line

It returns the line as a string

c) readlines() - Reads all the lines of the file. Returns a list of strings consisting of individual lines in the file
Example -
fp = open('file1.txt','r')
lines = fp.readlines()

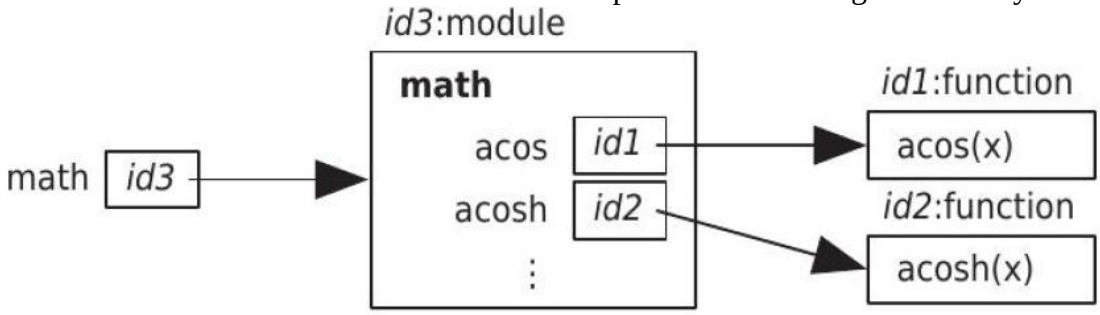
```
for l in lines:
        print l
fp.close()
```

d) For line in file - A straightforward method of processing lines in a file is using loop
Example

```
fp = open('file1.txt','r')
for line in fp:
        print l
fp.close()
```

Q 9 (a) What are the two ways of importing a module? Which one is more beneficial? Explain.
Ans: The two ways to import a module is to import the entire module using the statement:
import <module_name>
e.g. import math

In this case all the functions in the module are imported and are assigned memory.



A module variable math is created which points to module structure containg addresses of all the functions in the module. as shown in the above diagram. To invoke a function we use the format
<module name>.<function name>(args)
.E.g. math.sqrt(5).

Another way to import are specific functions and variables from a module using the format:
from <module name> import fn1,fn2
E.g. from math import math, pi

In such a case only the functions sqrt and variable pi are imported from math. The other functions are not stored . A function call be invoked by just calling the function without prepending the module name
e.g. sqrt(9)

Usually the second method is preferred since it only imports the functions and variables required. The first method causes all the methods to occupy memory irrespective of whether they are used or not.

(b) Explain how code in Python is tested semi-automatically.
Ans: Python has a module called doctest that allows to run tests that are included in the docstring all at once. The function that enables us to print such a report is testmod. It reports on whether the function calls return what we expect. It gives messages on how many tests succeeded and how many failed. The test cases can be specified in the docstring for a function as shown in the example below:

```
def large(a,b,c):
        '''
                (number, number, number) --> number

                Finds the largest of the three numbers given as input
```

```
        >>> large(5,1,3)
        5
        >>> large(4,10,7)
        10
    '''
    if a>b:
        if a>c:
            return a
        else:
            return c
    else:
        if  b > c:
            return b
        else:
            return c
```

In the above function to find maximum of 3 numbers the docstring specifies two tests for the numbers 5,1,3 and 4,10,7. The line next to the function call in docstring specifies the output expected. The testmod function parses the docstring runs the test specified and compares the result to the expected result to give the output. For instance importing the module containing the function above and typing the following commands in the python

```
>>>import doctest
>>> doctest.testmod()
```

tests all the functions in the current enviroment. In this case it will give the following output:

```
>>> doctest.testmod()
TestResults(failed=0, attempted=2)
```

This means two tests were run and none of them failed.

Q 10(a) What is a list of lists? Give one example along with memory model.
Ans: List within list[nested list]
 A list are heterogeneous, thus it contain elements of different data types.
 A nested list is a list appears as an element in another list.
Ex1. >>>list=['hello', 2.0,5,[10,20]]
In this list, the three-eth element is nested list. To extract an element from nested list here follows:
1)>>>elt=list[3]
>>>elt[0]
Output: 10
2)>>>list[3][1]
Output: 20
Ex2:>>>life=[['canada', 76.5],['united states', 75,5],['mexico', 72.0]]
>>>life[1]
Output: ['united states', 75.5]
>>>life[1][0]
Output: ['united states']
We can also assign sublist tp variables.
Assigning a sublists to a variable creates an alias for the sublist

```
>>> life = [['Canada', 76.5], ['United States', 75.5], ['Mexico', 72.0]]
>>> canada = life[0]
>>> canada
['Canada', 76.5]
>>> canada[0]
'Canada'
>>> canada[1]
76.5
```

MEMORY MODEL

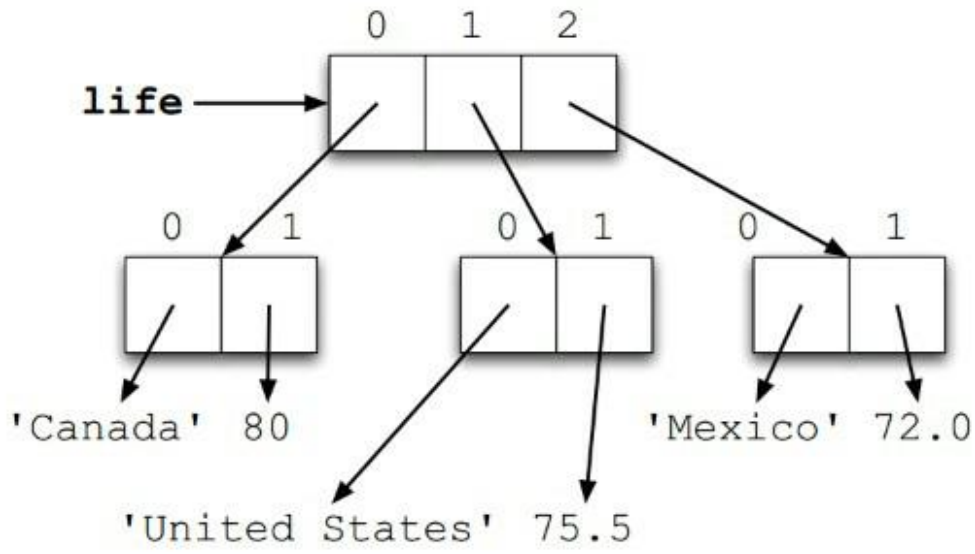


Figure 21—Nested lists

(b) Give differences between lists and strings.

Ans: One simple difference between strings and lists is that lists can any type of data i.e. integers, characters, strings etc, while strings can only hold a set of characters.

We can change the value of a list after we have created it, but we cannot in case of strings.
List is mutable whereas strings are immutable.

Q4(c) What is the result of the following python code fragment:
A=([1,4],2,3)
A[0][1]=5
A[2]=[4,5]

Sol: After the first statement A==([1,4],2,3)
After the second statement A[0][1]=5 -- A=([1,5],2,3)
The third statement A[2]=[4,5] will give an error since elements of a tuple cannot be modified since its immutable