CMR INSTITUTE OF TECHNOLOGY

Solution/Model Answer of IAT-1 (March 2018)
Python Application Programming-15CS664 (OE)
VI Sem – CSE
Dr. P. N. Singh, Professor(CSE)

1.

a. **What is the function of the secondary memory in a computer? Where in the computer is a variable such as "x" stored after the following Python line finishes?**

x = 123 **05**

Ans:

Examples of secondary memory are disk drives, CDs or flash memory (Pen-drives) used to store information permanently. Secondary memory is much slower than the main memory. The advantage of the secondary memory is that it can store information even when there is no power to the computer.

When value of variable is initialized by the statement:
x=123
Memory management of operating system stores this value occupying the memory cells according to size of data type & that particular region is named as x.
In python decision of data type and their size is dynamic according to initialized data. In above case as the Python finishes, internally data type is declared as of "integer" class. We can check the type.

```
>>> x=123
>>> type(x)
<class 'int'>
```

We can check the memory address also by in-built function id( )

```
>>> id(x)
1550147760
```

b. Write a Python Program to prompt the user for hours and rate per hour to compute gross pay using try and except so that your program handles invalid type of input by printing a message. The following shows two executions of the program:
Enter Hours: 20
Enter Rate: nine
Error, please enter numeric input **05**

Ans:

```
#Computing pay/wages
rate=input("Enter rate : ")
hours=input("Enter wages : ")
try:
    rate=float(rate)
    hours=float(hours)
    gross_pay=rate*hours
    print('Gross Pay = Rs.',gross_pay)
except:
```

```
    print('Please enter numeric input')
```

Expected Output:

```
Enter rate : 82
Enter wages : nine
Please enter numeric input
```

Again:

```
Enter rate : 82
Enter wages : 9
Gross Pay = Rs. 738.0
```

2.

a. Write a Python program which repeatedly reads numbers until the user enters "done". Once "done" is entered, print out the sum, minimum, maximum and average of the numbers. If the user enters anything other than a number, print an error message and continue the current iteration again. 05

Ans:

```
#largest and smallest in number
largest=smallest=None
tot=0
kount=1
while True:
    num = input("Enter a number (done to end) : ")
    if num == 'done':
        break
    try:
        n = int(num)
    except:
        print ("Invalid input")
        continue
    if largest == None or largest < n:
        largest = n
    if smallest == None or smallest > n:
        smallest = n
    tot=tot+n
    kount=kount+1

avg=tot/kount
print ("sum = ",tot,"Maximum = ", largest)
print("Minimum = ", smallest, "average = ",avg)
```

Expected output:

```
Enter a number (done to end) : 10
Enter a number (done to end) : 20
Enter a number (done to end) : 30
Enter a number (done to end) : abc
Invalid input
Enter a number (done to end) : 40
Enter a number (done to end) : 50
Enter a number (done to end) : done
sum =  150 Maximum =  50
Minimum =  10 average =  25.0
```

b. Take the following Python code that stores a string:
‘ str = ’X-DSPAM-Confidence:0.8475’
Use find and string slicing to extract the portion of the string after the colon character and then use the float function to convert the extracted string into a floating point number. 05

Ans:

```
str = 'X-DSPAM-Confidence:0.8475'
pos=str.find(':')
num=str[pos+1:]
num=float(num)
print(num)
0.8475
```

3.

a. Write a Python program to prompt for a file name, and then read through the file and print all the line starting with string “6thSem” striping the white space from left. 06

Ans:

```
fname=input(‘Enter file name : ‘)
file1=open(fname)
for line in file1:
       line=line.lstrip()
       if line.startswith('6thSem'):
     print(line,end=' ')
```

Expected Output:

```
Enter file name : file1.txt
6thSem CSE section a
6thSem ISE section b
```

b. Write a Python program to find sum and average of a list of numbers using loops. 04

Ans:

```
list1=[10,20,30,40,50]
tot=0
x=0
for n in list1:
     tot+=n
     x+=1

avg=tot/x
print('Sum = ', tot, 'Average = ', avg)
```

Expected output:

```
Sum =  150 Average =  30.0
```

4.

a. Write a Python program to prompt for a score between 0.0 and 1.0. If the score is out of range, print an error message. If the score is between 0.0 and 1.0, print a grade using the following table:

Score	Grade
>= 0.9	A
>= 0.8	B
>= 0.7	C
>= 0.6	D
< 0.6	F

06

Ans:

```
marks = input('Enter score (0 to 1.0): ')
try:
    score = float(marks)
    if score > 1.0 or score < 0:
       print ('Out of range')
       grade=None
    elif score >= .9:
       grade='A'
    elif score >= .8:
       grade='B'
    elif score >= .7:
       grade='C'
    elif score >= .6:
       grade='D'
    else:
       grade='F'
    print("Grade is",grade)
except:
    print ('Bad score\n')
```

Expected Output:

```
Enter score (0 to 1.0): 90
Out of range
Grade is None
```

Again

```
Enter score (0 to 1.0): .8
Grade is B
```

b. Write a Python Program to find maximum in 3 numbers using nested if statement. 04

Ans:

```
#nested if - to find max in 3 numbers
n1 = input("Enter number 1: ")
n2 = input("Enter number 2: ")
n3 = input("Enter number 3: ")
n1,n2,n3=int(n1),int(n2),int(n3)
if n1 > n2:
    if n1 > n3:
        max=n1
    else:
        max=n3
else:
    if n2 > n3:
```

```
        max=n2
    else:
        max=n3

print('Maximum is %d' % max) #%d for integer of decimal number system
print('Largest is', max)     #just printing value of variable max
```

5.

a. "Strings are immutable", Explain the statement with example. 05

Ans:

"Strings are immutable" means value of a string variable cannot be modified. Aliasing the string object aliasing is not as much of a problem but we can't change/modify characters of string.

```
>>> name='Paras'
>>> name[1]='o'

TypeError: 'str' object does not support item assignment
```

However ever we copy/add some contents of a string to another string.

```
>>>chat='I love you'
>>> chat2 = chat[:7]+'her'
>>> # all characters of chat before 7 and adding 'her'
>>> print(chat2)
I love her
```

b. Write a Python program to check where a string is palindrome. 05

Ans:

```
# Check whether a string is palindrome
def isPalindrome(str):
    i = 0
    j = len(str) - 1
    while(i < len(str)/2):   #comparing up to middle of string
        if(str[i] != str[j]):
            return False
        i += 1
        j -= 1
    return True


s = input("Enter a string : ")
if(isPalindrome(s)):
    print('Yes, "'+s+'" is a palindrome.')
else:
    print('No, "'+s+'" is not a palindrome.')
```

Expected Output:

```
Enter a string : malayalam
Yes, "malayalam" is a palindrome.
Again
```

```
Enter a string : able was i ere i saw elba
Yes, "able was i ere i saw elba" is a palindrome.
```

6.

a. Define a function *reverse_sum* that takes a number as an input parameter ,finds the sum of all the digits in the number and returns the sum. 05

Ans:

```
def reverse_sum(num):
tot=0
while num != 0:
      rem=num%10
      tot+=rem
      num//=10 #making it integer only
return tot
```

Expected output:

```
 print('Sum of digits of',num,' = ', reverse_sum(num))
Sum of digits of 12345  =  15
```

b. Explain any two functions from the math module and two functions from random module with suitable examples. 05

Ans:

```
import math
```

This statement creates a module object named math

log10() - return logarithm base 10

```
>>> import math
>>> math.log10(1000)
3.0
```

sqrt() – returns square root of number

```
>>> math.sqrt(2)
1.414  #more than 10 digits after decimal
```

random module: The random module provides functions that generate pseudorandom numbers

```
>>> import random
>>> random.random()
0.12317805330561149
>>>
```

To choose an element from a sequence at random, you can use choice:

```
>>> names=['Paras','Ratan','Manohar','Manoj']
>>> random.choice(names)
'Manohar'
>>> n=[3,5,7,9]
>>> random.choice(n)
5
```

7.

a. Explain arithmetic, relational and logical operators with examples. 06

Ans:

Arithmetic operators:

The operators +, -, *, /, and ** perform addition, subtraction, multiplication, division, and exponentiation, as in the following examples:

20+32 hour-1 hour*60+minute minute/60 5**2 (5+9)*(15-7)

In Python 3.x result of division will be a floating point number. To get floored integer in division result we use //

Relational Operator/Boolean Experession

Boolean Expression (Returns True or false). Their type is bool.

== > < >= <= != is is not

```
>>> x=20
>>> y=30
>>> x is y
False
>>> x is not y
True
```

There are three logical operators: and, or, and not. The semantics (meaning) of these operators is similar to their meaning in English.

and : both of the condition must be True

or: either of the condition should be True

not : negation of true is false and vice-versa

```
>>> x=20
>>> x > 0 and x < 10
False
>>> x > 0 or x < 10
True
>>> not x < 10
True
>>>
```

b. Write a Python Program to check whether a number is prime or not and print appropriate messages. 04

Ans:

```
#to check whether a number is prime
n=int(input("Enter number "))
prime=True
for d in range(2,int(n**(1/2))+1): #up to square root of n
      if n%d == 0:
              prime=False
              break
if prime:
```

```
    print(n, ' is prime')
else:
    print(n,' is not prime')
```

Expected output:

```
Enter number 47
47  is prime
Again:
Enter number 48
48  is not prime
```

8.

a. Define a function to display Fibonacci series up to range given as argument. 05

Ans:

```
def fib(n):
     a, b = 0, 1
     while a < n:
         print a,
         a, b = b, a+b

 # Now call the function we just defined:
>>> fib(50)
0 1 1 2 3 5 8 13 21 34
```

b. Explain the inbuilt functions max(), min(), len(), int(), float(), str() with examples. 05

Ans:

max( ) – returns largest (number or character) in given arguments
min() – return smallest (number or character) in given arguments
len() – return length of argument in list or string
Examples:

```
>>> max(2,34,56,7)
56
>>> min(2,34,56,7)
2
>>> max('how are you')
'y'
>>> len('how are you')
11
>>> len(2,34,56,7)
TypeError: len() takes exactly one argument (4 given)
>>> a=[2,3,4,5]
>>> len(a)
4
```

Type conversion functions:
int() – converts integer written as string to integer
float() – converts number written as string to floating point value
str() – converts arguments as string

```
>>> int('32')
```

```
32
>>> int('Hello')
ValueError: invalid literal for int() with base 10: 'Hello'
```

int can convert floating-point values to integers, but it doesn't round off; it chops off the fraction part:

```
>>> int(3.99999)
3
 >>> int(-2.3)
-2
```

float converts integers and strings to floating-point numbers:

```
>>> float(32) 32.0
>>> float('3.14159')
3.14159
```

Finally, str converts its argument to a string:

```
>>> str(32)
'32'
 >>> str(3.14159)
'3.14159'
```