USN										

Internal Assessment Test 1 –Nov 2022 Solutions

Sub:	Python Application Programming					Sub Code:	18CS752	Branch :	ECE,EEE, MECH
Date:	22.10.2022	Duration:	90 min's	Max Marks:	50	Sem/Sec :	7 A, B, C		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1 (a)	What is a program? Explain the building blocks of programs. Solve the problem by analyzing Coordinate the use of resources (primary/secondary memory/networked connections /IO Devices) "Talking to the CPU" Stored Instructions : Program • The definition of a *program* at its most basic is a sequence of Python statements that have been crafted to do something. • *hello.py* script is a program. It is a one-line program and is not particularly useful, but in the strictest definition, it is a Python program. The act of writing the instructions and ensuring it is correct : Programming **Input** : Get data from the "outside world". **Output** : Display the results of the program on a screen or store them in a file, speaker,etc. **Sequential Execution** : Perform statements one after another in the order they are encountered in the script. **Conditional Execution** - Check for certain conditions and then execute or skip a sequence of statements. **Repeated Execution -** Perform some set of statements repeatedly, usually with some variation. **Reuse** -Write a set of instructions once and give them a name and then reuse those instructions as needed throughout your program.	[05]	CO1	L2
(b)	Explain the logic and write program using function for finding maximum of 3 numbers. (code below)	[05]	CO1	L3

```
def max_num(a,b,c):
    if a>b:
        if a>c:
            m=a
        else:
            m=c
    else:
        if b>c:
            m=b
        else:
            m=c

    return m
```

```
m_num = max_num(5,8,9)
print('max number is', m_num)
print('using inbuilt function', max(5,8,9))
max_func.py =====
max number is 9
using inbuilt function 9
```

2 (a) What is the role of a programmer? List and elaborate on two skills required for a programmer. [04] CO1 L2

- Solve the problem by analyzing
- Coordinate the use of resources (primary/secondary memory/networked
- connections /IO Devices)
- “Talking to the CPU”
- Stored Instructions : Program
- The act of writing the instructions and ensuring it is correct : Programming

Skills required

Master the language (Python) – the vocabulary and the grammar.

Become familiar with the syntax, the various building blocks of Python, input, output, sequential execution, repeated execution, reuse.

- Solve the problem – by combining words to form sentences and essentially craft a story

-use logic to combine the various building blocks for a particular purpose

(b) Write a program print whether a given year is a leap year. [06] CO1 L3

To be a leap year, the year number must be divisible by four – except for end-of-century years, which must be divisible by 400. This means that the year 2000 was a leap year, although 1900 was not. 2020, 2024 and 2028 are all leap years.

```
if (year%4==0 and year%100 !=0) or year%400==0:
    print('leap year')
```

[OR]

```
year=int(input('Enter year:'))
leap = False
if year%4==0:
    if year %100==0:
        if year%400 ==0:
            leap=True

    else:
        leap = True

if leap:
    print('Leap year')
else:
    print('Not a leap year')
Output:
Enter year:2000
Leap year
Enter year:1900
Not a leap year
```

3 (a) Compare and contrast syntax error, logic error and semantic error with examples. [04] CO1 L2

Syntax Errors : grammatical mistakes, easy to fix

Eg. if x%2   == 0

        print('number is even')

In the above code Python expects a : following the if statement. If it is missing, it shows a syntax error.

Logic Errors : good syntax, but mistake in the order or the relation of statements to one another

Eg. Check if number is even

Code :

```
x=5
if x%2 != 0:
   print('Even')
```

This is logically wrong.

The correct code is

```
If x%2 ==0:
   Print('Even')
```

Semantic Errors : syntactically perfect and logically correct, but the program just does not do what it is meant to do (most difficult to identify and rectify)

Eg. x=1,000,000,000

Print(x)

(1,0,0,0)

If the programmer wanted to split big numbers, then they should have used _ (underscore). In python a comma treats it as a tuple. If the programmer expected 1000000000 to be printed, and did not know that comma cannot be used to split large numbers, they will not be able to correct this error.

(b) [06] CO1 L3

```
def is_prime(i):
   j=2
   isprime=True
   while j<= i//2: # for j in range(j, i//2+1)
      #print(i,'%',j)
      if i%j==0:
         isprime=False
         break
      j+=1 #remove if using for
   return isprime

m=2
n=50
while m<=n:
   if is_prime(m):
      print(m, end=' ')

   m+=1
```

Output

```
======= RESTART: D:/ prime.py =======
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

4 (a) Differentiate compiler and interpreter. [02]+[04] CO1 L2

- An *interpreter* reads the source code of the program as written by the programmer, parses the source code, and interprets the instructions on the fly. Python is an interpreter and when we are running Python interactively we can type a line of Python (a sentence) and Python processes it immediately and is ready for us to type another line of Python.

- A compiler converts the source code in high-level language to low-level language such as object code that can be used to create an executable program.

Explain type conversion, math functions using inbuilt functions with code snippets

- When you put an integer and floating point in an expression, the integer is implicitly converted to a float
- You can control this with the built-in functions int() and float()

```
>>> print(float(99) / 100)
0.99
>>> i = 42
>>> type(i)
<class 'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class 'float'>
>>> print(1 + 2 * float(3) / 4 – 5)
-2.5
>>>
```

Math runctions

- math module has to be imported to use this.
- Creates a module object named math
- Module object contains the functions and variables defined in the module
- To access, specify the name of the module and name of the function separated by a dot.

```
>>> import math
>>> decibels = 10*math.log10(5/4)
>>> decibels
0.9691001300805642
>>>
>>> radians =0.3
>>> x=math.sin(radians)
>>> x
0.29552020666133955
>>>
```

(b)	Write a python program using Exceptions, so that your program handles non-numeric input gracefully by printing an error message “Error, Please enter numeric input” and exiting the program	[04]	CO1	L3

Example Ouput:

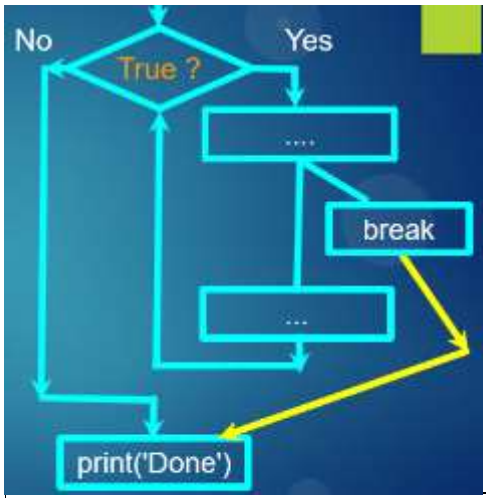
```
Enter Hours: 20
Enter Rate: nine
Error, Please enter numeric input
```

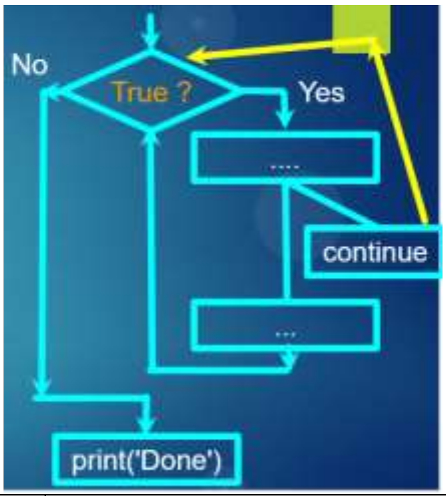
```
try:
   hours = int(input('Enter hours:'))
   rate = float(input('Enter rate:'))
except:
    print('Please enter number input')
```

```
except_q.py ==
Enter hours:20
Enter rate:nine
Please enter number input
```

5 (a) Differentiate break and continue statements with the help of flowchart and code. [04] CO1 L2

- The break statement ends the current loop and jumps to the statement immediately following the loop
- It is like a loop test that can happen anywhere in the body of the loop
- The continue statement ends the current iteration and jumps to the top of the loop and starts the next iteration





```
while True:
    line = input('> ')
    if line == 'done' :
        break
    print(line)
print('Done!')
```

```
while True:
    line = raw_input('> ')
    if line[0] == '#' :
        continue
    if line == 'done' :
        break
    print(line)
print('Done!')
```

(b) Write a python program using the list *items* to demonstrate counting, summing and average of elements using loops. Write appropriate comments and output. Do not use in-built functions. [06] CO1 L2

items =[2,6,8,9,4,9]

```
items =[2,6,8,9,4,9]
#demonstrate counting, summing and average of elements using loops

count = 0
total = 0
average =0

for i in items:
    count=count+1 #increment count by 1
    total+=i  # add each item to total

average = total/count #calculate average
```

```
print('Count:', count)
print('Total:', total)
print('Average:', average)
demo_loop.py =====
Count: 6
Total: 38
Average: 6.333333333333333
```

6(a) Evaluate the following expressions: [03] CO1 L2

(i) 3/2*4+3+(10/4)**3-2

3/2*4+3+**(10/4)****3-2
= 3/2*4+3+**(2.5)**3**-2
= **3/2***4+3+15.625-2
=**1.5*4**+3+15.625-2
= **6.0+3+**15.625-2
= **9+15.625**-2
= 24.625-2 =22.625

(ii) -17%3

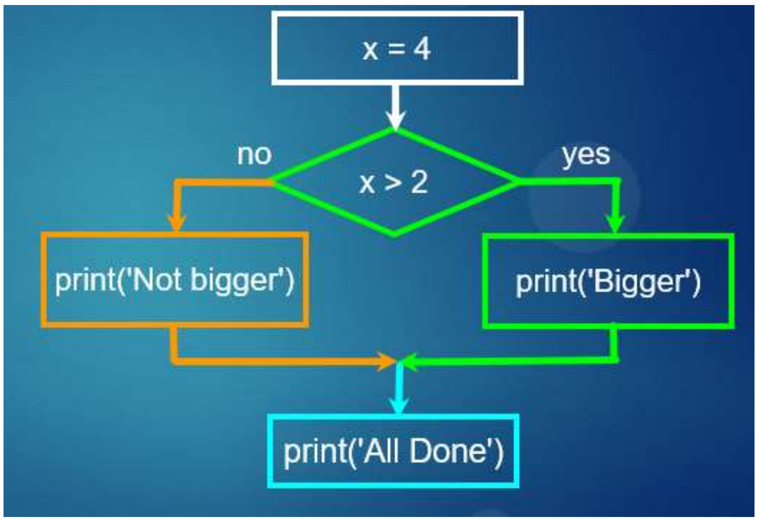
-17 – math.floor(-17/3) * 3 = -17 – (math.floor(-5.67)*3) = -17 – (-6*3) = -17+18=1

(iii) 8%2 + 9//3

0+3 = 3

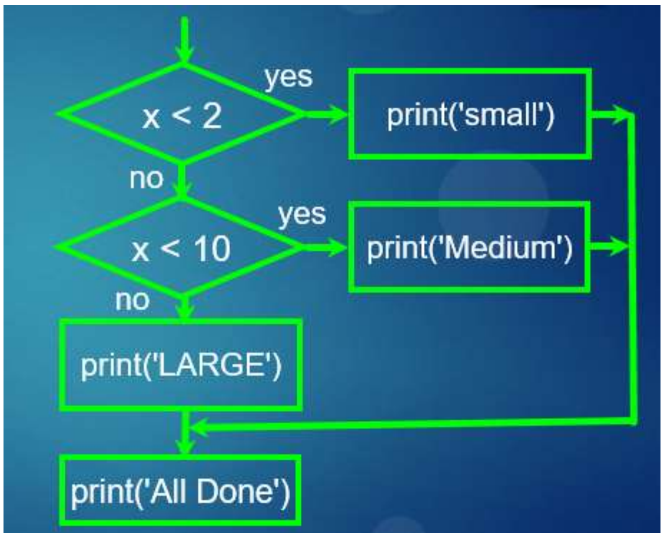
(b) Explain concept of conditional alternate execution and chained conditionals using diagrams and code snippets. [07] CO1 L2

- Sometimes we want to do one thing if a logical expression is true and something else if the expression is false
- It is like a fork in the road - we must choose one or the other path but not both
- Alternatives are called branches



```
x = 4
if x > 2 :
    print('Bigger')
else :
    print('Smaller')
print('All done')
```

Chained Conditionals – when there are multiple branches.



```
if x < 2 :
    print('small')
elif x < 10 :
    print('Medium')
else :
    print('LARGE')
print('All done')
```