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

INTERNAL ASSESSMENT TEST – I

Sub:	Python Programming							Code:	21EC643
Date:	06 / 06 / 2024	Duration:	90 mins	Max Marks:	50	Sem:	VI	Branch:	ECE

Answer any 5 full questions

		Marks	CO	RBT
1a	Explain math operators in Python with their precedence. ➢ Following are the math operators in Python from highest to lowest preference. **Operator** / **Operation** / **Example** / **Evaluates To** ** / Exponent / 2**3 / 8 % / Modulus/Remainder / 5%2 / 1 // / Integer Division / 7//2 / 3 / / Division / 5/2 / 2.5 * / Multiplication / 5*2 / 10 - / Subtraction / 5-2 / 3 + / Addition / 5+2 / 7 ➢ We can use parenthesis to override the usual precedence. ➢ For example 2+3*4 evaluates to 14 as * has higher precedence than + ➢ But (2+3)*4 evaluates to 20 as 2+3 is within parenthesis.	5	CO1	L2

1b	Write a Python function "fun_fact()" to find the factorial of a given number.
	
	Ideal solution
	`n=int(input())`
	`def fun_fact(n):`
	`  fact=1`
	`  if n==0:`
	`  return 0`
	`  for i in range(1,n+1):`
	`  fact=fact*i`
	`  return fact`
	`print(fun_fact(n))`
		5	CO1	L3

2a	Define and explain variables. What are the rules governing the naming of variables? Explain with examples.	5	CO1	L2

- A variable is like a box in the computer's memory where we can store a single value.
- We can store values in variables with an assignment statement.
- An assignment statement consists of a variable name, an equal sign (called the assignment operator), and the value to be stored.
- For example x=3 is an assignment statement.
- This statement will store the value 3 in the variable x.
- A variable is initialized (or created) the first time a value is stored in it.
- After that, we can use it in expressions with other variables and values.
- When a variable is assigned a new value, the old value is forgotten. This is called **overwriting** the variable.
- Following are the rules governing the naming of variable.
- **Rule 1 :** It can be only one word.
- **Rule 2 :** It can use only letters, numbers, and the underscore character.
- **Rule 3 :** It can't begin with a number.

- Following table illustrates some of the valid and invalid variable names.

Valid Variable Names	Invalid Variable Names
Name	First Name (Spaces are not allowed)
Name1	1Name (Can't begin with a number)
_Name	First-Name (Hyphens are not allowed)
First_Name	First$Name (Special characters not allowed)

2b

Write a Python function "Fun_Fib()" that takes an integer as input and generates a Fibonacci series up to that integer.

5 CO1 L3

```
n=int(input())
def Fun_Fib(n):
    list1=[]
    if n==1:
        list1=[0]
    elif n==2:
        list1=[0,1]
    else:
        list1=[0,1]
        i=2
        while(i<n-1):
            next=list1[i-1]+list1[i-2]
            list1.append(next)
            i=i+1
    print(*list1,sep=",")

Fun_Fib(n)
```

		Marks	CO	RBT
3a	Explain the following functions in Python with examples. a) print() b) input() c) len() d) str() e) float() a) print() **a) print()** ➢ The print() function displays the string value inside the parentheses on the screen. ➢ For example, print('Hello') displays Hello on the screen ➢ The quotes are not displayed on the screen. ➢ They just mark where the string begins and ends; they are not part of the string value. ➢ When Python execute the instruction print('Hello'), we can say that Python is calling the print() function and the string value is being passed to the function. ➢ A value that is passed to a function called as an argument. ➢ We can also use print() function to put a blank line on the screen; just call print() with nothing in between the parentheses. **b) input()** ➢ The input() function waits for the user to type some text on the keyboard and press enter. ➢ For example, name=input() This instruction takes a user input of string type and assigns it to the variable 'name'. **c) len()** ➢ len() function takes a string value and returns the number of characters in that string. ➢ For example, len('Python') returns 6 which is equal to the number of characters in the string 'Python'. **d) str()** ➢ str() function is used to convert integer or float value to string value. ➢ For example, str(-3.14) converts float value -3.14 into string value '-3.14' In the resulting value, -, 3, ., 1, 4 all are considered as string values. ➢ The str() function is useful when we have an integer or float which we want to concatenate to a string.	5	CO1	L2

	f) float() ➢ float() function can convert int or string values into float values. ➢ For example, float('2.4') converts string '2.4' into float 2.4 float(4) converts integer 4 into float 4.0 ➢ float() function is useful when we have a number as a string value which we want t use in some mathematical operation.			

3b	You are given a list of strings. Write a Python program to remove vowels from every string and print the new list.
	`n=int(input())`
	
	`list1=[]`
	`for i in range(n):`
	`  list1.append(input())`
	
	`list2=[]`
	`for i in list1:`
	`  str1=""`
	`  for j in i:`
	`  if j not in "aeiouAEIOU":`
	`  str1=str1+j`
	`  list2.append(str1)`
	
	`print(*list2,sep="\n")`
		5	CO1	L3

4a	Explain exception handling in Python.
	➢ We don't want our program to crash when an error is encountered.
	➢ Instead, we want our program to detect errors, handle them, and continue to run.
	➢ For example, consider the following code.
	`x=int(input('Enter an integer'))`
	`answer=10/x`
	`print(answer)`
	➢ If we enter 0 as input, this program results in ZeroDivisionError
		5	CO1	L2

	➢ This can be avoided by using try-except method. ➢ Consider the following code. x=int(input('Enter an integer')) try: answer=10/x print(answer) except: print('Error : x cannot be 0') ➢ When code in a try clause causes an error, the program execution immediately moves to the code in the except clause. ➢ After running that code, the execution continues as normal. ➢ In the above program, if the user enters 0 for x, except clause is entered and 'Error : x cannot be 0' will be printed. ➢ Once the execution jumps to the code in the except clause, it does not return to the try clause. ➢ Instead, it just continues moving down as normal.			

4b	You are given a list of strings. Write a Python program to print only the wonderful strings. A wonderful string is a string which is made up of exactly three different letters. Print "None" if there are no wonderful strings in the given list.
	Sample input
	abca pqrs tuvt xyz
	Sample output
	abca tuvt xyz
	Ideal solution
	`list1=input()`
	`list1=list1.split()`
	`list2=[]`
	`for i in list1:`
	`if len(set(i))==3:`
	`list2.append(i)`
	`if len(list2)==0:`
	`print("None")`
	`else:`
	`print(*list2)`
		5	CO1	L3

5	Explain dictionary data type with examples. Explain the following methods with respect to dictionary. (a) keys() (b) values() (c) items() (d) get() (e) set() (f) setdefault()	10	CO2	L2

- Like a list, dictionary is also a collection of many values.
- But in a list, indexes are always integer values.
- In a dictionary, index can be any immutable type of data such as integer and string.
- The indexes for dictionaries are called keys.
- Another difference between list and dictionary is that the list is ordered, but the dictionary is unordered.
- For example,
  x={'Name':'Alice','Age':10,'Gender':'Female'}
- x['Name'] returns 'Alice'
- x['Age'] returns 10
- We can append a new value to the dictionary as follows.
  x['Fav_Color']='Blue'

- Now the updated dictionary will be

  x={'Name':'Alice','Age':10,'Gender':'Female','Fav_Color':'Blue'}

a) keys()

- The keys() method returns list-like values of the dictionary's keys.
- The values returned by keys() are not true lists.
- They cannot be modified and they do not have append method.
- But these data types can be used in for loops.
- For example,

```
x={'Name':'Alice','Age':10,'Gender':'Female','Fav_Color':'Blue'}
for i in x.keys():
    print(i)
#The output of this code will be
'Name'
'Age'
'Gender'
'Fav_Color'
```

b) values()

- The values() method returns list-like values of the dictionary's values.
- The values returned by values() are not true lists.
- They cannot be modified and they do not have append method.
- But these data types can be used in for loops.
- For example,

```
x={'Name':'Alice','Age':10,'Gender':'Female','Fav_Color':'Blue'}
for i in x.values():
    print(i)
#The output of this code will be
'Alice'
10
Female
'Blue'
```

c) items()

- The items() method returns tuples containing key-value pair.
- For example,

```
x={'Name':'Alice','Age':10,'Gender':'Female','Fav_Color':'Blue'}
for i in x.items():
    print(i)
#The output of this code will be
('Name','Alice')
('Age',10)
('Gender','Female')
('Fav_Color','Blue')
```

d) get()

- The get() method is used to obtain the value corresponding to a key.
- The get() method takes two arguments : the key of the value to retrieve and a fallback value if that key does not exist.
- The fallback value helps to avoid the error if the key does not exist.
- For example,

```
x={'Name':'Alice','Age':10,'Gender':'Female','Fav_Color':'Blue'}
print(x.get('Name',-1)) #prints 'Alice'
print(x.get('Country',-1)) #prints -1 because 'Country' is not a key in x
```

	e) setdefault()
	➢ The setdefault() method assigns a value to a key only if that key does not already have a value.
	➢ If that key already has a value, setdefault() method returns the existing value.
	➢ For example,
	`x={'Name':'Alice','Age':10,'Gender':'Female','Fav_Color':'Blue'}`
	`x.setdefault('Name','Bob') #Returns 'Alice'`
	`x.setdefault('Country','India') #Returns 'India'`
	`#And the updated dictionary will be`
	`#x={'Name':'Alice','Age':10,'Gender':'Female','Fav_Color':'Blue','Country':'India'}`			

6a	Explain string indexing and slicing. ➢ Just as an index can get a single value from a string, a slice can get several values from a string, in the form of a new string. ➢ A slice is typed between square brackets, like an index, but it has two integers separated by a colon. ➢ In a slice, the first integer is the index where the slice starts. ➢ The second integer is the index where the slice ends, but it will not include the value at the second index. ➢ A slice evaluates to a new string. ➢ For example, str1='electronics' ➢ str1[1:4] returns 'lec' ➢ We can leave out one or both of the indexes on either side of the colon in the slice. ➢ Leaving out the first index is the same as using 0, or the beginning of the string. ➢ Leaving out the second index is the same as using the length of the string, which will slice to the end of the string. ➢ For example, str1[:4] returns 'elec' ➢ str1[7:] returns 'nics' ➢ str1[:] returns 'electronics' ➢ We can use negative integers for the index. ➢ The integer value -1 refers to the last index in a string, the value -2 refers to the second-to-last index in a string, and so on. For example, str1[-1] returns 's', str1[-2] returns 'c'	5	CO2	L2
6b	You are given a string. Write a Python program to print the number of uppercase and lowercase letters. **Sample input** `Cmrit Ece 2000`	5	CO2	L3

Sample output

```
2,6
```

Ideal solution

```
str1=input()

upper_count=0
lower_count=0

for i in str1:
    if i.isalpha():
        if i.isupper():
            upper_count=upper_count+1
        elif i.islower():
            lower_count=lower_count+1

print("{},{}".format(upper_count,lower_count))
```