18CSC207J Advanced Programming Practice

CT 1 Total marks : 25

PART A 05 X 01 = 05

1. What will be the output of the following Python code?

```
print(0xA + 0xB + 0xC)
```

a) 0xA0xB0xC
b) Error
c) 0x22
d) 33

2. What will be the output of the following Python code?

```
>>>t=(1,2,4,3)
>>>t[1:-1]
```

a) (1, 2)
b) (1, 2, 4)
c) (2, 4)
d) (2, 4, 3)

2. What will be the output of the following Python code?

```
List = [10, 20, 30, 40, 50, 60, 70, 80]
print(aList[2:5])
print(aList[3:])
```

a. [30, 40, 50]
[40, 50, 60, 70, 80]

b. [40, 50, 60]
[40, 50, 60, 70, 80]

c. [30, 40, 50]
[30, 40, 50, 60, 70]

d. [30, 40, 50]
[40, 50, 60, 70]

3. Which of the following is correct with respect to OOP concept in Python?

A. Objects are real world entities while classes are not real.
B. Classes are real world entities while objects are not real.
C. Both objects and classes are real world entities.
D. Both object and classes are not real

4.Which one of the following is correct?

A. In python, a dictionary can have two same keys with different values.
B. In python, a dictionary can have two same values with different keys
C. In python, a dictionary can have two same keys or same values but cannot have two same key-value pair
D. In python, a dictionary can neither have two same keys nor two same values.

Answer Key

B2-SET A

PART B

6.

```
list = [1, 2, 3, 4, 5]
for x in range(1, len(list), 2):
    print(list[x])
```

or

```
list = [1, 2, 3, 4, 5, 6]
for i in range(0, len(list)):
    if i % 2 != 0:
        print(list[i])
```

7.

```
def func(name, age, reg_no, dept):
    print("Name: ", name)
    print("Age: ", age)
    print("Registration No.: ", reg_no)
    print("Department: ", dept)

func("XYZ", 20, 831, "CSE")
```

Output:

```
Name:  XYZ
Age:  20
Registration No.:  831
Department:  CSE
```

8.

Public Access Modifier

The members of a class that are declared public are easily accessible from any part of the program. All data members and member functions of a class are public by default.

e.g.

```
class xyz:
    def __init__(self):
        self.x = 0
        self.y = 0
```

Private Access Modifier

The members of a class that are declared private are accessible within the class only, private access modifier is the most secure access modifier. Data members of a class are declared private by adding a double underscore '__' symbol before the data member of that class.

e.g.

```
class xyz:
    __x = 0
    def __func(self):
        print(self.__x)
```

9.

```
for i in range(100, 401):
    if int(str(i)[0]) % 2 == 0 and int(str(i)[1]) % 2 == 0 and
int(str(i)[2]) % 2 == 0:
        print(i, end=",")
```

10.

```
word = input("Enter the word: ")
print(word[::-1])
```

Sample Input and Output:

```
Enter the word: Hello
olleH
```

PART C

11.

```
import random
val = str(random.randint(100, 999))
num = ""
while(len(num) != 3):
    num = input("Enter a three digit number: ")
    if(len(num) != 3):
        print("Enter a valid number.")
awardMoney = 0
if(val[0] == num[0] and val[1] == num[1] and val[2] == num[2]):
    awardMoney = 10000
else:
    count = 0
    for x in num:
        for y in val:
            if x == y:
                count += 1
    if(count == 3):
        awardMoney = 3000
    elif(count == 1):
        awardMoney = 1000

print(f"Your Award Money is: ${awardMoney}")
```

12.

```python
def repeat_times(n):
  s = 0
  n_str = str(n)
  while (n > 0):
    n -= sum([int(i) for i in list(n_str)])
    print(n)
    n_str = list(str(n))
repeat_times(25)
```

Output:

```
18
9
0
```

13.

```python
class Student:
    def __init__(self, reg, name, dept, m1, m2, m3, m4, m5):
        self.reg_no, self.name, self.dept = reg, name, dept
        self.m1, self.m2, self.m3, self.m4, self.m5 = m1, m2, m3, m4, m5
    def total(self):
        self.total = self.m1 + self.m2 + self.m3 + self.m4 + self.m5
        print(self.total)
    def percent(self):
        # Let the full marks = 50
        print(((self.m1 + self.m2 + self.m3 + self.m4 + self.m5)/50) * 100)
    def grade(self):
        if(self.total >= 40):
            print("Distinction")
        elif(self.total >= 30 and self.total < 40):
            print("Merit")
        elif(self.total >= 25 and self.total < 30):
            print("Pass")
        else:
            print("Fail")

obj = Student("831", "XYZ", "CSE", 10, 10, 10, 10, 10)
obj.total()
obj.percent()
obj.grade()
```

Output:

```
50
100.0
Distinction
```

14.

```
class SRMIST:
    specialization = ""
    val = True
    def __init__(self, *args):
        if(len(args) == 3):
            self.school, self.dept3, self.dept4 = args[0], args[1], args[2]
            self.val = False
        else:
            self.school, self.dept1, self.dept2, self.dept3, self.dept4 =
args[0], args[1], args[2], args[3], args[4]
    def display(self):
        if self.val:
            print("School : " + self.school, "\nDept1 : ", self.dept1,
"\nDept2 : ", self.dept2, "\nDept3 : ", self.dept3, "\nDept4 : ",
self.dept4)
        else:
            print("School : " + self.school, "\nDept3 : ",
                  self.dept3, "\nDept4 : ", self.dept4, "\nSpecialization :
", self.specialization)

obj_855 = SRMIST("School of Computing", "CSE", "ECE", "EEE", "CSE w/s AI")
obj_855.display()

# Setting Specialization
SRMIST.specialization = "Cloud Computing"

# After deleting Dept1 and Dept2
obj_855 = SRMIST("School of Computing", "EEE", "CSE w/s AI")
print("\n")
obj_855.display()
```

Sample Input and Output:

```
School : School of Computing
Dept1 :  CSE
Dept2 :  ECE
Dept3 :  EEE
Dept4 :  CSE w/s AI


School : School of Computing
Dept3 :  EEE
Dept4 :  CSE w/s AI
Specialization :  Cloud Computing
```

18CSC207J Advanced Programming Practice

CT 1 Total marks : 25

PART A 05 X 01 = 05

1. What will be the output of above Python code?

```
d1={"abc":5,"def":6,"ghi":7}
print(d1[0])
```

A. abc
B. 5
C. {"abc":5}
D. Error

2. What will be the output of below Python code?

```
tuple1=(2,4,3)
tuple3=tuple1*2
print(tuple3)
```

A. (4,8,6)
B. (2,4,3,2,4,3)
C. (2,2,4,4,3,3)
D. Error

3. Choose the correct option with respect to Python.

A. Both tuples and lists are immutable.
B. Tuples are immutable while lists are mutable.
C. Both tuples and lists are mutable.
D. Tuples are mutable while lists are immutable.

4. How many objects and reference variables are there for the given Python code?

```
class A:
        print("Inside class")
A()
A()
obj=A()
```

A. 2 and 1
B. 3 and 3
C. 3 and 1
D. 3 and 2

5.Which of the following is correct with respect to OOP concept in Python?

A. Objects are real world entities while classes are not real.
B. Classes are real world entities while objects are not real.
C. Both objects and classes are real world entities.
D. Both object and classes are not real.

Answer Key

B2-SET B

PART B

6. 15

7.
```
list = [1, 2, 3, 4, 5]
for x in range(1, len(list), 2):
    print(list[x])
```

or

```
list = [1, 2, 3, 4, 5, 6]
for i in range(0, len(list)):
    if i % 2 != 0:
        print(list[i])
```

8.
Public Access Modifier
The members of a class that are declared public are easily accessible from any part of the program. All data members and member functions of a class are public by default.
e.g.
```
class xyz:
    def __init__(self):
        self.x = 0
        self.y = 0
```

Private Access Modifier
The members of a class that are declared private are accessible within the class only, private access modifier is the most secure access modifier. Data members of a class are declared private by adding a double underscore '__' symbol before the data member of that class.
e.g.
```
class xyz:
    __x = 0
    def __func(self):
        print(self.__x)
```

9.
An immutable object is one that, once created, will not change in its lifetime. In Python, strings are made immutable so that programmers cannot alter the contents of the object (even by mistake). This avoids unnecessary bugs. Some other immutable objects are integer, float, tuple, and bool.

10.
No Error
Because in line number 03 def _init_(self, name, age): can be used as a function

```
class Student:
    schoolName = 'XYZ School'
    def _init_(self, name, age):
        self.name = name
        self.age = age
        print(self.name, self.age)

obj = Student()
obj._init_("XYZ", 20)

# Output:
# XYZ 20
```

10.
If we consider _init_ as the initializer for constructor then the sytax is wrong, we should use __init__ , the double underscore at both, start and end of the keyword 'init'

PART C

11.

```
class Student:
    avg = 0
    def __init__(self, stNo, m1, m2, m3, m4, m5):
        self.studentNum = stNo
        self.mark1, self.mark2, self.mark3, self.mark4, self.mark5 = m1,
m2, m3, m4, m5
    def avgMark(self):
        self.avg = (self.mark1 + self.mark2 + self.mark3 + self.mark4 +
self.mark5) / 5
    def grade(self):
        if(self.avg >= 80):
            print("Distinction")
        elif(self.avg >= 65 and self.avg < 80):
            print("Merit")
        elif(self.avg >= 50 and self.avg < 65):
            print("Pass")
        else:
            print("Fail")

obj = Student("20202", 10, 20, 30, 40, 80)
print(obj.studentNum)
obj.grade()
```

Output:

```
20202
Fail
```

12.

```
# Variable to track number of times character is occured
times = 0
def occurence(_a, n, str):
    #To access 'times' variable which is outside this function
    global times
    # Base Case
    if n < 0:
        return times
    if str[n] == _a:
        times += 1
        # Recursion
    return occurence(_a, n-1, str)

character = 'd'
str = "ddlj"
slen = len(str) - 1
print(occurence(character, slen, str))
```

Output:
2

13.

```
# Create time class and initialize it with hours and minutes
class Time:
    def __init__(self, hours, minutes):
        self.hours = hours
        self.minutes = minutes

    def addTime(self, hr, mt):
        self.hours += hr
        self.minutes += mt
        if self.minutes >= 60:
            self.hours += self.minutes // 60
            self.minutes = self.minutes % 60
        if self.hours >= 24:
            self.hours = self.hours % 24
        return self.hours, self.minutes
    # display time in minute
    def displayTime(self):
        print(self.hours, ":", self.minutes)

    def displayMinutes(self):
        print("Total minutes: ", self.hours * 60 + self.minutes)

obj = Time(2, 50)
obj.addTime(1, 20)
obj.displayTime()
obj.displayMinutes()
```

Output:
4 : 10

Total minutes: 250

14.

```python
def isPrime():
    # Prime number entered by user
    n = int(input("Enter a number : "))
    # Variable to count the number of times n is divisible
    count = 0
    for i in range(2, n+1):
        if n % i == 0:
            count += 1
        if count > 1:
            return False
     # If above condition is not false it means 'n' is prime so return True
    return True
print("Given Number is a Prime Number." if isPrime() else "Given number is 
not a Prime Number.")
```

Sample Input and Output:

```
Enter a number : 5
Given Number is a Prime Number.
```