DATA SCIENCE

Python Assignment

1. **What is 7 to the power of 4?**

```
7**4
=2401
```

2. **Split this string: s="Hi there Sam!" into a list.**

```
s='Hi there Sam!'
s.split()
['Hi', 'there', 'Sam!']
```

3. **Given the variables: planet = "Earth" diameter = 12742 Use .format() to print the following string:**

```
planet="Earth"
diameter=12742
print("The diameter of {} is {} kilometers.".format(planet,diameter))
The diameter of Earth is 12742 kilometers.
```

4. **Given this nested list, use indexing to grab the word "hello"**

```
lst=[1,2,[3,4],[5,[100,200,['hello']],23,11],1,7]
lst[3][1][2][0]
'hello'
```

5. **Given this nest dictionary grab the word "hello". Be prepared, this will be annoying/tricky**

```
d = {'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello']}]}]}
d['k1'][3]['tricky'][3]['target'][3]
'hello'
```

6. **What is the main difference between a tuple and a list?**

#TUPLE IS IMMUTABLE

7. **Create a function that grabs the email website domain from a string in the form: user@domain.com.So for example, passing "user@domain.com" would return: domain.com**

```
def domainGet(email):

    return email.split('@')[-1]

print(domainGet('user@domain.com'))
```

8. **Create a basic function that returns True if the word 'dog' is contained in the input string. Don't worry about edge cases like a punctuation being attached to the word dog, but do account for capitalization.**

```
def findDog(st):
    return 'dog' in st.lower().split()
print(findDog('Is there a dog here?'))

true
```

9. **Use lambda expressions and the filter() function to filter out words from a list that don't start with the letter 's'. For example: seq = ['soup','dog','salad','cat','great'] should be filtered down to: ['soup','salad']**

```
seq = ['soup','dog','salad','cat','great']
print(list(filter(lambda word: word[0]=='s',seq)))
```

10. **You are driving a little too fast, and a police officer stops you. Write a function to return one of 3 possible results: "No ticket", "Small ticket", or "Big Ticket". If your speed is 60 or less, the result is "No Ticket". If speed is between 61 and 80 inclusive, the result is "Small Ticket". If speed is 81 or more, the result is "Big Ticket". Unless it is your birthday (encoded as a boolean value in the parameters of the function) -- on your birthday, your speed can be 5 higher in all cases.**

```
def caught_speeding(speed, is_birthday):8

    if is_birthday:
        speeding = speed - 5
    else:
        speeding = speed

    if speeding > 80:
        return 'Big Ticket'
    elif speeding > 60:
        return 'Small Ticket'
    else:
        return 'No Ticket'
```

```
print(caught_speeding(81,True))
print(caught_speeding(81,False))
```

11. **Write a program to input roll no., student name, marks of physics, Chemistry and maths out of 100. (0-100) Calculate total, percentage, calculate STATUS(pass,fail) if students scores above 40 in all the 3 subjects the STATUS should be pass otherwise fail. Calculate GRADE: is status is pass.**
If percentage is above 70, then grade must be DISTINCTION if percentage is above 60, then grade must be FIRST CLASS
If percentage is above 50, then grade must be SECOND CLASS if percentage is above 40, then grade must be PASS CLASS.

```
rollno=int(input("enter your roll no:"))
name=input("enter your name:")
physics=float(input("enter marks in physics:"))
chemistry=float(input("enter marks in chemistry:"))
maths=float(input("enter marks in maths:"))
total=physics+chemistry+maths
print("total=",total)
percentage=(total/300)*100
print("percentage=",percentage)
if ((maths>=40)and (physics>=40) and (chemistry>=40)):
    print("Grade: PASS")
    if (percentage>=70):
        print("DISCTICTION")
    elif (percentage>=60):
        print("FIRST CLASS")
    elif (percentage>=50):
        print("SECOND CLASS")
    else:
        print("PASS CLASS")
else:
    print("Grade: FAIL")
```

```
enter your roll no:22000257
enter your name:nandani
enter marks in physics:56
enter marks in chemistry:26
enter marks in maths:86
total= 168.0
percentage= 56.00000000000001
Grade: FAIL
```

Python Assignment 2

1. **Write a OOP in python to input empid, name, basic salary, no. of experience in yrs. Calculate hra (35% of basic), da (58% of basic) and pf (9.5% of basic). Also calculate bonus based on experience in years. If experience in years is >=30, bonus must be 59% of basic, If experience in years is >=23, bonus must be 51% of basic, If experience in years is >=15, bonus must 45% of basic, If experience in years is >=7, bonus must be 33% of basic, If experience in years is <7, bonus must be 16% of basic Calculate netsalary as basic+da+hra+pf+bonus. Create a class, constructor to create instance variables, getter-setter for each variable, calculative functions for operative variables. A class methods/function should not contain display specific and input specific code. Such code should be added in driver part of python program.**

```
class Employee:
    def __init__(self, empid, name, basic_salary, experience_years):
        self.empid = empid
        self.name = name
        self.basic_salary = basic_salary
        self.experience_years = experience_years
        self.hra = 0
        self.da = 0
        self.pf = 0
        self.bonus = 0
        self.net_salary = 0

    # Getter and setter methods for each variable
    def get_empid(self):
        return self.empid

    def set_empid(self, empid):
        self.empid = empid

    def get_name(self):
        return self.name

    def set_name(self, name):
        self.name = name

    def get_basic_salary(self):
        return self.basic_salary

    def set_basic_salary(self, basic_salary):
        self.basic_salary = basic_salary
    def get_experience_years(self):
        return self.experience_years

    def set_experience_years(self, experience_years):
        self.experience_years = experience_years

    def calculate_hra(self):
        self.hra = 0.35 * self.basic_salary

    def calculate_da(self):
        self.da = 0.58 * self.basic_salary

    def calculate_pf(self):
        self.pf = 0.095 * self.basic_salary

    def calculate_bonus(self):
        if self.experience_years >= 30:
            self.bonus = 0.59 * self.basic_salary
        elif self.experience_years >= 23:
            self.bonus = 0.51 * self.basic_salary
        elif self.experience_years >= 15:
            self.bonus = 0.45 * self.basic_salary
        elif self.experience_years >= 7:
            self.bonus = 0.33 * self.basic_salary
        else:
            self.bonus = 0.16 * self.basic_salary

    def calculate_net_salary(self):
        self.net_salary = (
            self.basic_salary + self.da + self.hra - self.pf + self.bonus
        )


# Driver code with user input
empid = int(input("Enter Employee ID: "))
name = input("Enter Name: ")
basic_salary = float(input("Enter Basic Salary: "))
experience_years = int(input("Enter Years of Experience: "))

# Create Employee instance
emp = Employee(empid, name, basic_salary, experience_years)

# Calculate operative variables
emp.calculate_hra()
emp.calculate_da()
emp.calculate_pf()
emp.calculate_bonus()
emp.calculate_net_salary()

# Display results
```

```
print("\nEmployee Details:")
print("Employee ID:", emp.get_empid())
print("Name:", emp.get_name())
print("Basic Salary:", emp.get_basic_salary())
print("Experience (in years):", emp.get_experience_years())

print("\nOperative Variables:")
print("HRA:", emp.hra)
print("DA:", emp.da)
print("PF:", emp.pf)
print("Bonus:", emp.bonus)

print("\nNet Salary:", emp.net_salary)
```

```
Enter Employee ID: 22000257
Enter Name: nandani
Enter Basic Salary: 10000
Enter Years of Experience: 2

Employee Details:
Employee ID: 22000257
Name: nandani
Basic Salary: 10000.0
Experience (in years): 2

Operative Variables:
HRA: 3500.0
DA: 5800.0
PF: 950.0
Bonus: 1600.0

Net Salary: 19950.0
```

2. **Write an OOP based Python program which inputs n numbers in a list from keyboard. If 8 numbers are inputted, calculate sum of Oth and 7th element and save it in another list say newlist[0], sum of Ist and 6th to newlist[1], sum of 2nd and 5th to newlist[2], sum of 3rd and 4th to newlist[3] and so on. Remove duplicates from newlist if any by converting to set. Later convert it to tuple and display.**

```
class NumberProcessor:
    def __init__(self):
        self.number_list = []
        self.new_list = []

    def input_numbers(self, n):
        print(f"Enter {n} numbers:")
        for _ in range(n):
            number = float(input())
            self.number_list.append(number)
    def calculate_and_create_new_list(self):
        length = len(self.number_list)
        for i in range(length // 2):
            sum_result = self.number_list[i] + self.number_list[length - 1 - i]
            self.new_list.append(sum_result)

    def remove_duplicates_and_display(self):
        unique_set = set(self.new_list)
        unique_tuple = tuple(unique_set)
        print("Resulting Tuple after removing duplicates:", unique_tuple)


# Driver code
num_processor = NumberProcessor()
num_processor.input_numbers(8)
num_processor.calculate_and_create_new_list()
num_processor.remove_duplicates_and_display()
```

```
Enter 8 numbers:
1
2
1
2
3
1
2
2
Resulting Tuple after removing duplicates: (2.0, 3.0, 4.0, 5.0)
```

3. **Write an OOP program to perform addition, base and power, concatenation, max, min of two numbers stored in two different objects created from same class.**
n1=MyNumber(2) n2=MyNumber() n2.setNum(5) n3=n1.add(n2)
print("Addition is ",n3.getNum()) #7 n3=n1.raisedTo(n2)
print(n1.getNum()," raised to ",n2.getNum()," is ",n3.getNum()) #32 n3=n1.concat(n2)
print("Concat answer is ",n3.getNum()) #25 n3=n1.max(n2)
print("Max is ",n3.getNum())

```
class MyNumber:
    def __init__(self, value=0):
        self.value = value

    def setNum(self, value):
        self.value = value

    def getNum(self):
        return self.value

    def add(self, other):
        result = self.value + other.value
        return MyNumber(result)

    def raisedTo(self, other):
        result = self.value ** other.value
        return MyNumber(result)

    def concat(self, other):
        result = int(str(self.value) + str(other.value))
        return MyNumber(result)

    def max(self, other):
        result = max(self.value, other.value)
        return MyNumber(result)

    def min(self, other):
        result = min(self.value, other.value)
        return MyNumber(result)


# Example usage:
n1 = MyNumber(2)
n2 = MyNumber()
n2.setNum(5)

n3 = n1.add(n2)
print("Addition is", n3.getNum())  # 7

n3 = n1.raisedTo(n2)
print(n1.getNum(), "raised to", n2.getNum(), "is", n3.getNum())  # 32

n3 = n1.concat(n2)
print("Concat answer is", n3.getNum())  # 25

n3 = n1.max(n2)
print("Max is", n3.getNum())
```

```
Addition is 7
2 raised to 5 is 32
Concat answer is 25
Max is 5
```

Python Assignment 3

1. **Write a python program to access following data structure**

```
data=[(12345,"Diksha","Diksha
r@nuv.ac.in"),(45667,"Jiva","Jiva@nuv.ac.in"),(16789,"Ronit","Ronitd@nuv.ac.in"),
(68433,"Heena","Heena@nuv.ac.in")]
def searchbyid(uid):
    for tup in data:
        if tup[0]==uid:
            sid=tup[0]
            sname=tup[1]
            semail=tup[2]
            print(f'student is {sid} name is {sname} email is {semail}')


def searchbyname(uname):
    for tup in data:
        if tup[1]==uname:
            sid=tup[0]
            sname=tup[1]
            semail=tup[2]
            print(f'student is {sid} name is {sname} email is {semail}')

uid=int(input("Enter id to search:"))
searchbyid(uid)
uname=input("Enter name to search:")
searchbyname(uname)
```

2. **Write a python program to access the following dictionary: Mydict= {"Enrollment id:":[12345,45667,16789,68433], "Student Name":["Diksha","Jiva","Ronit","Heena"], "Email Id" : ["Diksha r@nuv.ac.in","Jiva@nuv.ac.in","Ronitd@nuv.ac.in","Heena@nuv.ac.in"] } print(f "{enrolid:7}{sname:12}{emailid:25}")**

```
Mydict={"Enrollment id:":[12345,45667,16789,68433],
"Student Name":["Diksha","Jiva","Ronit","Heena"],
"Email Id" : ["Diksha
r@nuv.ac.in","Jiva@nuv.ac.in","Ronitd@nuv.ac.in","Heena
@nuv.ac.in"]}
```

```
#Mydict['Enrollment id:'][0],Mydict['Student
Name'][0],Mydict['Email Id'][0]


def search(value):
    for dc in Mydict:
        #print(Mydict[dc])
        if value in Mydict[dc]:
            i=Mydict[dc].index(value)
            enrolid=Mydict["Enrollment id:"][i]
            ename=Mydict["Student Name"][i]
            eemailid=Mydict["Email Id"][i]
            print(f"{enrolid:<15}{ename:<15}{eemailid:<25}")



value=int(input("Enter the id:"))
search(value)

value=input("Enter the name:")
search(value)
```

3. **Write a python program to convert data in nested list structure to dictionary, e.g., data=[(12345,"Diksha","Diksha r@nuv.ac.in"),(45667,"Jiva","Jiva@nuv.ac.in"),(16789,"Ronit","Ronitd@nuv.ac.in"), (68433,"Heena","Heena@nuv.ac.in")] Provide key such as Enrollment Id, Student Name, and Email Id Resultant Data Mydict={"Enrollment id:":[12345,45667,16789,68433], "Student Name":["Diksha","Jiva","Ronit","Heena"], "Email Id" : ["Diksha r@nuv.ac.in","Jiva@nuv.ac.in","Ronitd@nuv.ac.in","Heena@nuv.ac.in"]}
   Display data in a tabular format**

```
data = [
    (12345, "Diksha", "Diksha r@nuv.ac.in"),
    (45667, "Jiva", "Jiva@nuv.ac.in"),
    (16789, "Ronit", "Ronit@nuv.ac.in"),
    (69433, "Heena", "Heena@nuv.ac.in")
]

# Extracting values from the nested list
enrollment_ids = [item[0] for item in data]
student_names = [item[1] for item in data]
email_ids = [item[2] for item in data]

# Creating a dictionary
mydict = {
    "Enrollment Id": enrollment_ids,
    "Student Name": student_names,
    "Email Id": email_ids
}

# Displaying data in tabular format
print(f"{'Enrollment Id':<15}{'Student Name':<15}{'Email
Id':<30}")

for i in range(len(enrollment_ids)):

print(f"{enrollment_ids[i]:<15}{student_names[i]:<15}{email
_ids[i]:<30}")

print("\nResultant Data:")
print(mydict)
```

4. **Write a menu driven Object Oriented program to store Students details like Enrollment No,Student Name and Contact Number permanently using dictionary data structure.**

Add functions to add, update, delete and display data. Save data permanently in a JSON file.

HINT:

To import json library

import json

To write any object to python

fw=open("students.json","w")

jsndata = json.dumps(dictionary of students details)#dumps converts dictionary to json

fw.write(jsndata)

To read any json object from file

fr=open("students.json","r")

allStudents=json.load(fr) #reads whole file and returns the data.

```
import json

class Student:
    def __init__(self, enrollment_no, name, contact_no):
        self.enrollment_no = enrollment_no
        self.name = name
        self.contact_no = contact_no

class StudentManager:
    def __init__(self):
        self.students = {}

    def add_student(self, enrollment_no, name,
contact_no):
        if enrollment_no not in self.students:
            self.students[enrollment_no] =
Student(enrollment_no, name, contact_no)
            print("Student added successfully.")
        else:
            print("Student with this Enrollment No already
exists.")

    def update_student(self, enrollment_no, name,
contact_no):
        if enrollment_no in self.students:
            self.students[enrollment_no].name = name
            self.students[enrollment_no].contact_no =
contact_no
            print("Student updated successfully.")
        else:
            print("Student with this Enrollment No does not
exist.")

    def delete_student(self, enrollment_no):
        if enrollment_no in self.students:
            del self.students[enrollment_no]
            print("Student deleted successfully.")
        else:
            print("Student with this Enrollment No does not
exist.")

    def display_students(self):
        for student in self.students.values():
            print(f"Enrollment No: {student.enrollment_no},
Name: {student.name}, Contact No:
{student.contact_no}")

    def save_students(self, filename):
        with open(filename, 'w') as fw:
            json_data = json.dumps({k: v.__dict__ for k, v in
self.students.items()}, indent=4)
            fw.write(json_data)
        print("Students saved successfully.")

    def load_students(self, filename):
        with open(filename, 'r') as fr:
            self.students = {int(k): Student(v['enrollment_no'],
v['name'], v['contact_no']) for k, v in json.load(fr).items()}
        print("Students loaded successfully.")

    def menu(self):
        while True:
            print("\nMenu:")
            print("1. Add Student")
            print("2. Update Student")
            print("3. Delete Student")
            print("4. Display Students")
            print("5. Save Students")
        print("6. Load Students")
        print("7. Exit")
        choice = int(input("Enter your choice: "))
        if choice == 1:
            enrollment_no = int(input("Enter Enrollment No: 
"))
            name = input("Enter Name: ")
            contact_no = input("Enter Contact No: ")
            self.add_student(enrollment_no, name, 
contact_no)
        elif choice == 2:
            enrollment_no = int(input("Enter Enrollment No: 
"))
            name = input("Enter new Name: ")
            contact_no = input("Enter new Contact No: ")
            self.update_student(enrollment_no, name, 
contact_no)
        elif choice == 3:
            enrollment_no = int(input("Enter Enrollment No: 
"))
            self.delete_student(enrollment_no)
        elif choice == 4:
            self.display_students()
        elif choice == 5:
            filename = input("Enter filename to save: ")
            self.save_students(filename)
        elif choice == 6:
            filename = input("Enter filename to load: ")
            self.load_students(filename)
        elif choice == 7:
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")

# Create an instance of StudentManager and call the 
menu function
manager = StudentManager()
manager.menu()
```

5. **A data.csv file has following data(stateid, name, population in crores, No. of Universities)**

data.csv
12001,Gujarat,10,29,12002,Maharashtra,19,36,12003,R
ajasthan,13,31,12004,Madhya
Pradesh,14,21,12005,Punjab,12,13,12006,Karnataka,23,
31,12007,Tamilnadu,25,29,1208,Keral
a,21,15
Read data.csv file. Transfer data to data structure like following
States={
12001:
{"name":"Gujarat","population":10,"no_of_uni":20},
12002:
{"name":"Maharashtra","population":19,"no_of_uni":36}
}
Using this data, construct dictionary in following manner.

```
import csv

def read_csv(file_path):
    states = {}

    with open(file_path, 'r') as file:
        csv_reader = csv.reader(file)

        for row in csv_reader:
            state_id = int(row[0])
            state_name = row[1]
            population = int(row[2])
            universities = int(row[3])

            states[state_id] = {
                "name": state_name,
                "population": population,
                "no_of_uni": universities
            }

    return states
```

```
# Assuming your data.csv file is in the current working
directory
file_path = 'data.csv'
states_data = read_csv(file_path)

# Print the constructed dictionary
for state_id, state_info in states_data.items():
    print(f"{state_id}: {state_info}")
```