SMRITI PRIYA
101765006
ENC-7

Cloud Computing Lab-1

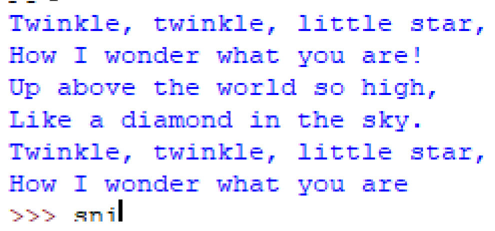
Part A

1. Write a Python program to print the following string in a specific format (see the output).
Twinkle, twinkle, little star,
How I wonder what you are!
Up above the world so high,
Like a diamond in the sky.
Twinkle, twinkle, little star,
How I wonder what you are

code:

```
print(‘Twinkle, twinkle, little star,’)
print(‘How I wonder what you are!’)
print(‘Up above the world so high,’)
print(‘Like a diamond in the sky.’)
print(‘Twinkle, twinkle, little star,)
print(‘How I wonder what you are’)
```

output:



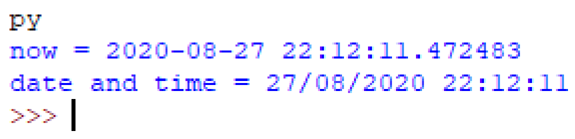
```
Twinkle, twinkle, little star,
How I wonder what you are!
Up above the world so high,
Like a diamond in the sky.
Twinkle, twinkle, little star,
How I wonder what you are
>>> sni
```

2. Write a Python program to display the current date and time.

code:

```
from datetime import datetime
now = datetime.now()
print("now =", now)
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("date and time =", dt_string)
```

output:



```
py
now = 2020-08-27 22:12:11.472483
date and time = 27/08/2020 22:12:11
>>>
```

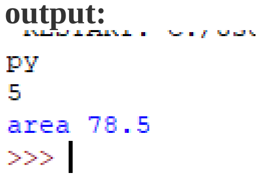
SMRITI PRIYA
101765006
ENC-7

3. Write a Python program which accepts the radius of a circle from the user and compute the area.

code:

```
n=int(input())
print('area',str(3.14*n**2))
```

output:



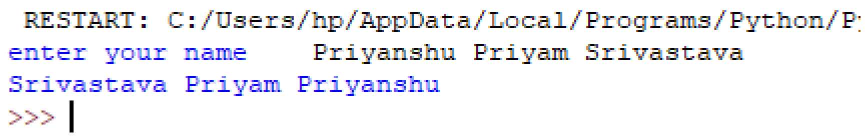
```
 RESTART: C:/Us
py
5
area 78.5
>>> 
```

4. Write a Python program which accepts the user's first and last name and print them in reverse order with a space between them.

code:

```
s=input('enter your name    ')
l=[i for i in s.split()]
for i in l[::-1]:
    print(i,end=' ')
print()
```

output:



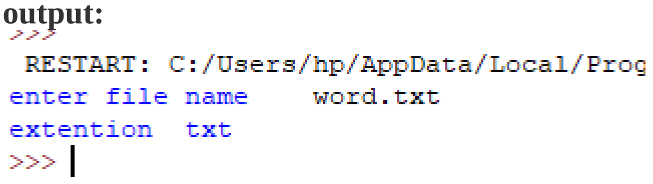
```
 RESTART: C:/Users/hp/AppData/Local/Programs/Python/P
enter your name    Priyanshu Priyam Srivastava
Srivastava Priyam Priyanshu
>>> 
```

5. Write a Python program to accept a filename from the user and print the extension of that.

code:

```
s=input('enter file name    ')
l=[i for i in s.split('.')]
print('extention ',l[-1])
```

output:



```
>>>
 RESTART: C:/Users/hp/AppData/Local/Prog
enter file name    word.txt
extention  txt
>>> 
```

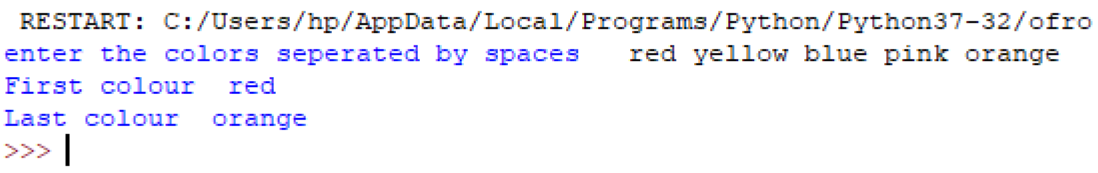
SMRITI PRIYA
101765006
ENC-7

6. Write a Python program to display the first and last colours from a list.

code:

```
s=input('enter the colors seperated by spaces   ')
l=[i for i in s.split()]
print('First colour ',l[0])
print('Last colour ',l[-1])
```

output:



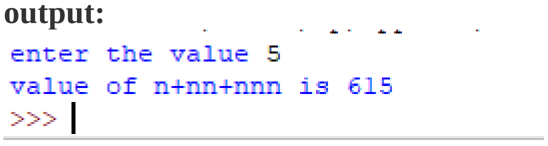
```
 RESTART: C:/Users/hp/AppData/Local/Programs/Python/Python37-32/ofro
enter the colors seperated by spaces   red yellow blue pink orange
First colour  red
Last colour  orange
>>> 
```

7. Write a Python program that accepts an integer (n) and computes the value of n+nn+nnn.

code:

```
n=input('enter the value ')
print('value of n+nn+nnn is ',end='')
l=int(n*2)
m=int(n*3)
n=int(n)
print(l+m+n)
```

output:



```
enter the value 5
value of n+nn+nnn is 615
>>> 
```

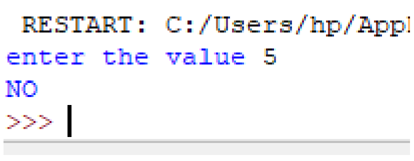
8. Write a Python program to test whether a number is within 100 of 1000 or 2000.

code:

```
n=input('enter the value ')
s='NO'
for i in range(100 ,10001):
    if int(n)==i:
        s='YES'
        break
print(s)
```

SMRITI PRIYA
101765006
ENC-7

output:



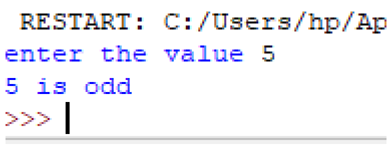
```
 RESTART: C:/Users/hp/Appl
enter the value 5
NO
>>> 
```

9. Write a Python program to find whether a given number (accept from the user) is even or odd, print out an appropriate message to the user.

code:

```
n=int(input('enter the value '))
if n%2==0:
   print(str(n)+' is even')
else:
   print(str(n)+' is odd')
```

output:



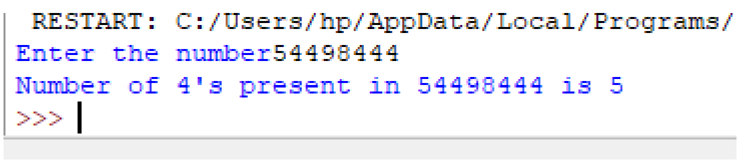
```
 RESTART: C:/Users/hp/Ap
enter the value 5
5 is odd
>>> 
```

10. Write a Python program to count the number 4 in a given list.

code:

```
n=input('Enter the number')
print("Number of 4's present in "+str(n)+" is "+str(n.count('4')))
```

output:



```
 RESTART: C:/Users/hp/AppData/Local/Programs/
Enter the number54498444
Number of 4's present in 54498444 is 5
>>> 
```

Part B

To Understand AWS Storage service (S3 and EBS) and summarize the learning in one page.

Amazon S3

SMRITI PRIYA
101765006
ENC-7

Amazon Simple Storage Service is storage for the Internet. It is designed to make web-scale computing easier for developers. Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

Following are some advantages:

1. Creating buckets: Create and name a bucket that stores data. Buckets are the fundamental containers in Amazon S3 for data storage

2. Storing data – Store an infinite amount of data in a bucket. Upload as many objects as you like into an Amazon S3 bucket. Each object can contain up to 5 TB of data. Each object is stored and retrieved using a unique developer-assigned key.

3. Downloading data – Download your data or enable others to do so. Download your data anytime you like, or allow others to do the same.

4. Permissions – Grant or deny access to others who want to upload or download data into your Amazon S3 bucket. Grant upload and download permissions to three types of users. Authentication mechanisms can help keep data secure from unauthorized access.

5. Standard interfaces – Use standards-based REST and SOAP interfaces designed to work with any internet-development toolkit.

Amazon EBS

Amazon EBS allows you to create storage volumes and attach them to Amazon EC2 instances. Once attached, you can create a file system on top of these volumes, run a database, or use them in any other way you would use block storage. Amazon EBS volumes are placed in a specific Availability Zone where they are automatically replicated to protect you from the failure of a single component. All EBS volume types offer durable snapshot capabilities and are designed for 99.999% availability.

Amazon EBS provides a range of options that allow you to optimize storage performance and cost for your workload. These options are divided into two major categories: SSD-backed storage for transactional workloads, such as databases and boot volumes (performance depends primarily on IOPS), and HDD-backed storage for throughput intensive workloads, such as MapReduce and log processing (performance depends primarily on MB/s).

SSD-backed volumes include the highest performance Provisioned IOPS SSD (io2 and io1) for latency-sensitive transactional workloads and General Purpose SSD (gp2) that balance price and performance for a wide variety of transactional data. HDD-backed volumes include Throughput Optimized HDD (st1) for frequently accessed, throughput intensive workloads and the lowest cost Cold HDD (sc1) for less frequently accessed data.

SMRITI PRIYA
101765006
ENC-7

Elastic Volumes is a feature of Amazon EBS that allows you to dynamically increase capacity, tune performance, and change the type of live volumes with no downtime or performance impact. This allows you to easily right-size your deployment and adapt to performance changes.

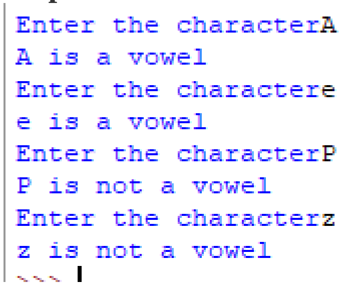
Cloud Computing Lab-2

1. Write a Python program to test whether a passed letter is a vowel or not.

code:

```
n=input('Enter the character')
s='aeiouAEIOU'
if n in s:
    print(str(n)+' is a vowel')
else:
    print(str(n)+' is not a vowel')
```

output:



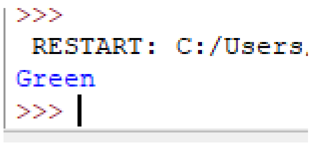
```
Enter the characterA
A is a vowel
Enter the charactere
e is a vowel
Enter the characterP
P is not a vowel
Enter the characterz
z is not a vowel
>>>
```

2. Write a Python program to print out a set containing all the colours from color_list_1 which are not present in color_list_2.
color_list_1 = set(["White", "Black", "Red"])
color_list_2 = set(["Red", "Green"])

code:

```
color_list_1 = set(["White", "Black", "Red"])
color_list_2 = set(["Red", "Green"])
for i in color_list_2:
    if i not in color_list_1:
        print(i)
```

output:



```
>>>
 RESTART: C:/Users,
Green
>>>
```

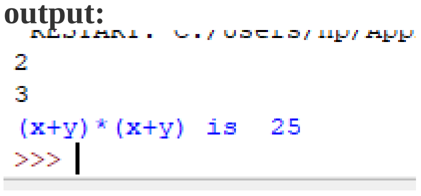
SMRITI PRIYA
101765006
ENC-7

3. Write a Python program to solve (x + y) * (x + y).

code:

```
x=int(input())
y=int(input())
print('(x+y)*(x+y) is ', end=' ')
print((x+y)**2)
```

output:



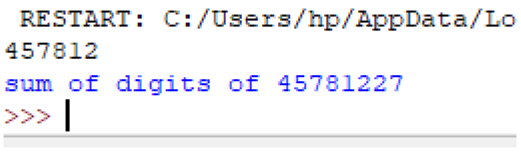
```
 RESTART: C:/Users/hp/App
2
3
(x+y)*(x+y) is  25
>>> 
```

4. Write a Python program to calculate the sum of the digits in an integer.

code:

```
n=input()
l=[int(i) for i in n]
print('sum of digits of '+ str(n)+str(sum(l)))
```

output:



```
 RESTART: C:/Users/hp/AppData/Lo
457812
sum of digits of 45781227
>>> 
```

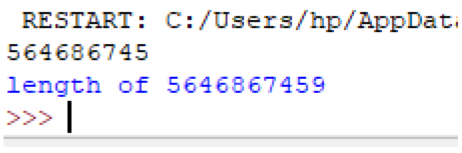
5. Write a Python program to calculate the length of a string.

code:

```
n=input()

print('length of '+ str(n)+str(len(n)))
```

output:



```
 RESTART: C:/Users/hp/AppDat
564686745
length of 5646867459
>>> 
```

6. Write a Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string.
Sample String : 'abc', 'xyz'
Expected Result : 'xyc abz'

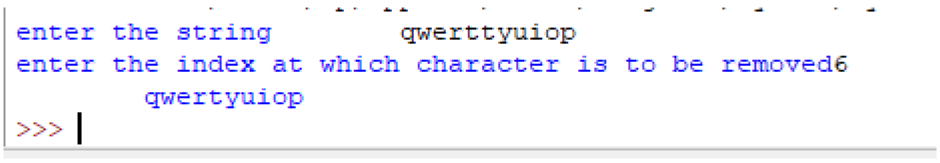
SMRITI PRIYA
101765006
ENC-7

7. Write a Python program to remove the nth index character from a nonempty string.

Code:

```
s=input('enter the string  ')
n=int(input('enter the index at which character is to be removed'))
ns=''
ns=s[:n]
ns+=s[n+1:]
print(ns)
```

Output:



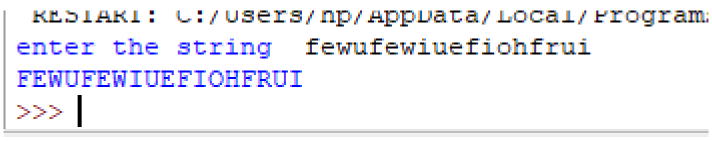
```
enter the string        qwerttyuiop
enter the index at which character is to be removed6
        qwertyuiop
>>> 
```

8. Write a Python script that takes input from the user and displays that input back in upper and lower cases

code:

```
s=input('enter the string  ')
print(s.upper())
```

output.



```
 RESTART: C:/Users/hp/AppData/Local/Program
enter the string  fewufewiuefiohfrui
FEWUFEWIUEFIOHFRUI
>>> 
```

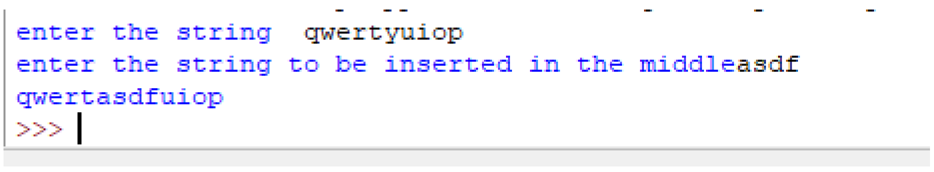
9. Write a Python function to insert a string in the middle of a string.

code:

```
s=input('enter the string  ')
i=input('enter the string to be inserted in the middle')
mid=(len(s)+1)//2
ns=''
ns=s[:mid]
ns+=i
ns+=s[mid+1:]
print(ns)
```

SMRITI PRIYA
101765006
ENC-7

output:



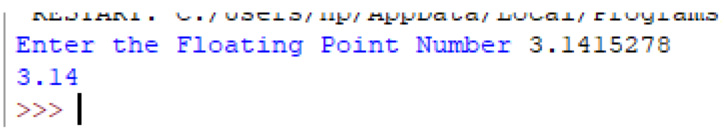
```
enter the string  qwertyuiop
enter the string to be inserted in the middleasdf
qwertasdfuiop
>>> 
```

10. Write a Python program to print the floating numbers upto 2 decimal places with a sign.

code:

```
a_float = input('Enter the Floating Point Number ')
a_float=float(a_float)
formatted_float = "{:.2f}".format(a_float)
print(formatted_float)
```

output:



```
Enter the Floating Point Number 3.1415278
3.14
>>> 
```

Part B

To Understand AWS compute service (EC2 & lightsail) and summarize the learning in one page.

EC2

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

Amazon EC2 provides the following features:

1. Virtual computing environments, known as instances

2. Preconfigured templates for your instances, known as Amazon Machine Images (AMIs), that package the bits you need for your server (including the operating system and additional software)

3. Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types

4. Secure login information for your instances using key pairs (AWS stores the public key, and you store the private key in a secure place)

5. Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as instance store volumes

6. Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes

7. Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as Regions and Availability Zones

8. A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups

9. Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses

10.Metadata, known as tags, that you can create and assign to your Amazon EC2 resources

11.Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs)

Lightsail

AWS Lightsail is an entry-level AWS service, offering app developers access to a configurable virtual private server (VPS) and a suite of easy to use tools. Amazon Lightsail provides an easy, lightweight way for new cloud users to take advantage of AWS' cloud computing services.

Advantages:

1. Simplified UI: It runs through pre-installed software. There will be less confusion for users who are using it for the first time.

2. Familiarity: Developers and companies who use Amazon Lightsail for web hosting and application development will find it more alluring.

3. Affordability: The subscription is free for the first month and is $5/month for the cheapest tier.

4. Reliability: The need for replacement is rare as the services run through Amazon infrastructure and data centers.

AWS Lightsail is mostly used by first-time cloud users and developers looking to test out app prototypes, or by startups and small scale production studios for whom the limitations of the Lightsail environment are sufficient.

SMRITI PRIYA
101765006
ENC-7

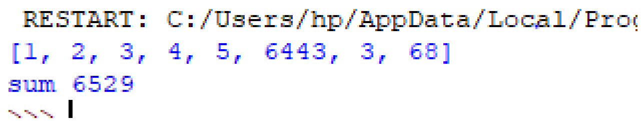
Cloud Computing Lab-3

Part A

1. Write a Python program to sum all the items in a list
code:

```
l=[1,2,3,4,5,6443,3,68]
print(l)
print('sum',sum(l))
```

output:

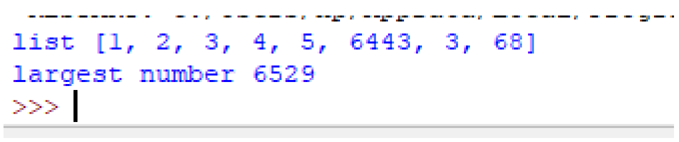


```
 RESTART: C:/Users/hp/AppData/Local/Pro
[1, 2, 3, 4, 5, 6443, 3, 68]
sum 6529
>>> 
```

2. Write a Python program to get the largest number from a list.
code:

```
l=[1,2,3,4,5,6443,3,68]
print('list',l)
print('largest number',sum(l))
```

output:

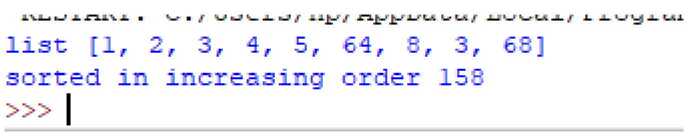


```
list [1, 2, 3, 4, 5, 6443, 3, 68]
largest number 6529
>>> 
```

3. Write a Python program to get a list, sorted in increasing order by the last element in each tuple from a given list of non-empty tuples
code:

```
l=[1,2,3,4,5,64,8,3,68]
print('list',l)
print('sorted in increasing order',sum(l))
```

output:



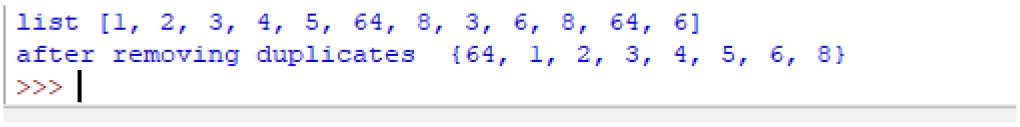
```
list [1, 2, 3, 4, 5, 64, 8, 3, 68]
sorted in increasing order 158
>>> 
```

4. Write a Python program to remove duplicates from a list
code:

```
l=[1,2,3,4,5,64,8,3,6,8,64,6]
print('list',l)
print('after removing duplicates ',set(l))
```

SMRITI PRIYA
101765006
ENC-7

output:

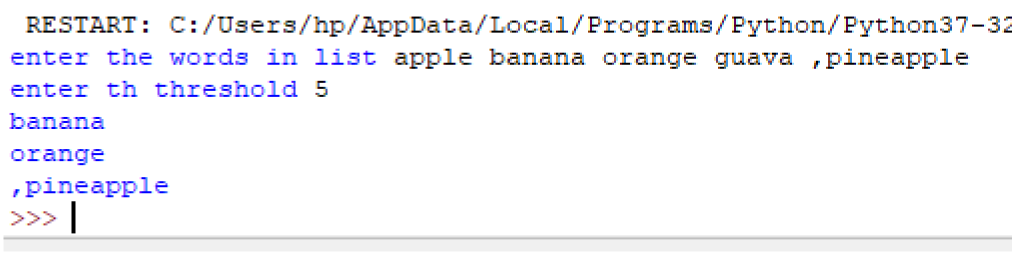


```
list [1, 2, 3, 4, 5, 64, 8, 3, 6, 8, 64, 6]
after removing duplicates  {64, 1, 2, 3, 4, 5, 6, 8}
>>> 
```

5. Write a Python program to find the list of words that are longer than n from a given list of words
code:

```
s=input('enter the words in list ')
l=[i for i in s.split()]
n=int(input('enter th threshold '))
for i in l:
    if len(i)>n:
        print(i)
```

output:

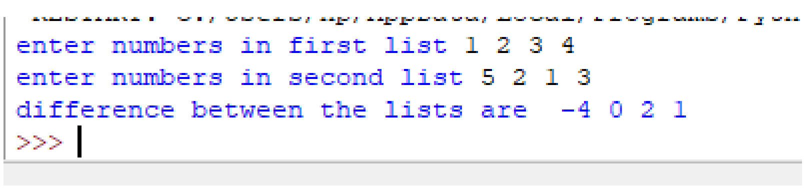


```
 RESTART: C:/Users/hp/AppData/Local/Programs/Python/Python37-32
enter the words in list apple banana orange guava ,pineapple
enter th threshold 5
banana
orange
,pineapple
>>> 
```

6. Write a Python program to get the difference between the two lists
code:

```
s=input('enter numbers in first list ')
t=input('enter numbers in second list ')
s=[int(i) for i in s.split()]
t=[int(i) for i in t.split()]
l=[s[i]-t[i] for i in range(len(s))]
print('difference between the lists are ',*l)
```

output:



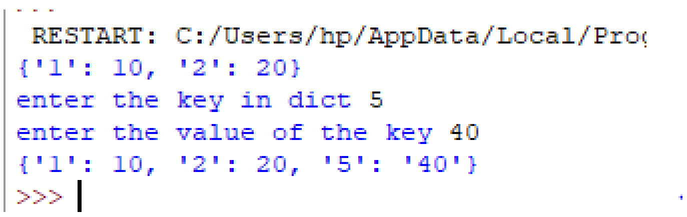
```
 RESTART: C:/Users/hp/AppData/Local/Programs/Pyth
enter numbers in first list 1 2 3 4
enter numbers in second list 5 2 1 3
difference between the lists are  -4 0 2 1
>>> 
```

SMRITI PRIYA
101765006
ENC-7

7. Write a Python script to add a key to a dictionary
code:

```
dic1={'1':10, '2':20}
print(dic1)
i=input('enter the key in dict ')
v=input('enter the value of the key ')
dic1[i]=v
print(dic1)
```

output:

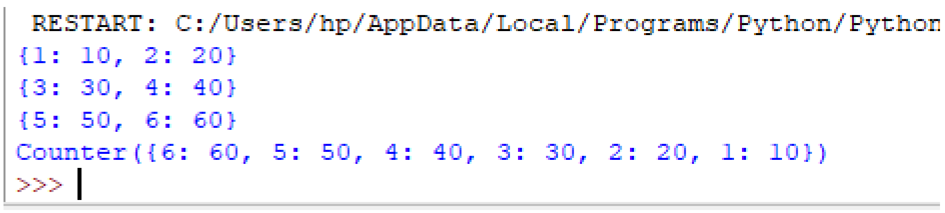


```
 RESTART: C:/Users/hp/AppData/Local/Pro
{'1': 10, '2': 20}
enter the key in dict 5
enter the value of the key 40
{'1': 10, '2': 20, '5': '40'}
>>>
```

8. Write a Python script to concatenate following dictionaries to create a new one.
Sample Dictionary :
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
code:

```
from collections import Counter
dic1={1:10, 2:20}
print(dic1)
dic2={3:30, 4:40}
dic3={5:50,6:60}
print(dic2)
print(dic3)
dic1=Counter(dic1)
dic2=Counter(dic2)
dic3=Counter(dic3)
dic4=Counter()
dic4=dic1+dic2+dic3
print(dic4)
```

output:



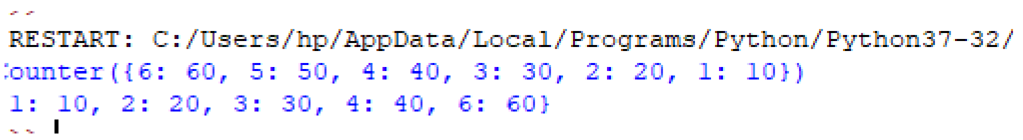
```
 RESTART: C:/Users/hp/AppData/Local/Programs/Python/Python
{1: 10, 2: 20}
{3: 30, 4: 40}
{5: 50, 6: 60}
Counter({6: 60, 5: 50, 4: 40, 3: 30, 2: 20, 1: 10})
>>>
```

SMRITI PRIYA
101765006
ENC-7

9. Write a Python program to remove a key from a dictionary
code:

```
from collections import Counter
dic1={1:10, 2:20}
#print(dic1)
dic2={3:30, 4:40}
dic3={5:50,6:60}
#print(dic2)
#print(dic3)
dic1=Counter(dic1)
dic2=Counter(dic2)
dic3=Counter(dic3)
dic4=Counter()
dic4=dic1+dic2+dic3
print(dic4)
dic4=dict(dic4)
del(dic4[5])
print(dic4)
```

output:



```
RESTART: C:/Users/hp/AppData/Local/Programs/Python/Python37-32/
ounter({6: 60, 5: 50, 4: 40, 3: 30, 2: 20, 1: 10})
1: 10, 2: 20, 3: 30, 4: 40, 6: 60}
```

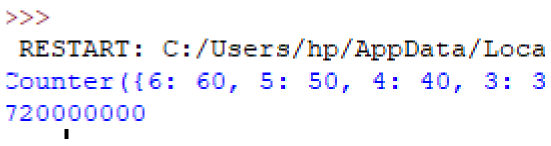
10. Write a Python program to multiply all the items in a dictionary
code:

```
from collections import Counter
dic1={1:10, 2:20}
#print(dic1)
dic2={3:30, 4:40}
dic3={5:50,6:60}
#print(dic2)
#print(dic3)
dic1=Counter(dic1)
dic2=Counter(dic2)
dic3=Counter(dic3)
dic4=Counter()
dic4=dic1+dic2+dic3
print(dic4)
dic4=dict(dic4)
mul=1
```

SMRITI PRIYA
101765006
ENC-7

```
for i in dic4:
    mul*=dic4[i]
print(mul)
```

output:



```
>>>
 RESTART: C:/Users/hp/AppData/Loca
Counter({6: 60, 5: 50, 4: 40, 3: 3
720000000
```

Part B

To Understand AWS database service (Aurora, Redshift & DynamoDB) and summarize the learning in one page.

Aurora

Amazon Aurora (Aurora) is a fully managed relational database engine that's compatible with MySQL and PostgreSQL. Aurora includes a high-performance storage subsystem. Its MySQL- and PostgreSQL-compatible database engines are customized to take advantage of that fast distributed storage. The underlying storage grows automatically as needed, up to 64 tebibytes (TiB). Aurora also automates and standardizes database clustering and replication, which are typically among the most challenging aspects of database configuration and administration. Aurora is part of the managed database service Amazon Relational Database Service (Amazon RDS). Amazon RDS is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. If you are not already familiar with Amazon RDS

Redshift

Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. It is also used to perform large scale database migrations.

Redshift's column-oriented database is designed to connect to SQL-based clients and business intelligence tools, making data available to users in real time. Based on PostgreSQL 8, Redshift delivers fast performance and efficient querying that help teams make sound business analyses and decisions.

Redshift is Amazon's analytics database, and is designed to crunch large amounts of data as a data warehouse. Those interested in Redshift should know that it consists of clusters of databases with dense storage nodes, and allows you to even run traditional relational databases in the cloud.

Amazon Redshift automatically handles many of the time-consuming tasks associated with managing your own data warehouse, including:

Setup: With Amazon Redshift, you simply create a data warehouse cluster, define your schema, and begin loading and querying your data. You don't have to manage provisioning, configuration or patching.

Data Durability: Amazon Redshift replicates your data within your data warehouse cluster and continuously backs up your data to Amazon S3, which is designed for eleven nines of durability. Amazon Redshift mirrors each drive's data to other nodes within your cluster. If a drive fails, your queries will continue with a slight latency increase while Redshift rebuilds your drive from replicas. In case of node failure(s), Amazon Redshift automatically provisions new node(s) and begins restoring data from other drives within the cluster or from Amazon S3. It prioritizes restoring your most frequently queried data so your most frequently executed queries will become performant quickly.

Scaling: You can add or remove nodes from your Amazon Redshift data warehouse cluster with a single API call or via a few clicks in the AWS Management Console as your capacity and performance needs change. You can also schedule your scaling and resize operations by using the scheduler capability in Redshift.

Automatic Updates and Patching: Amazon Redshift automatically applies upgrades and patches your data warehouse so you can focus on your application and not on its administration.

Exabyte Scale Query Capability: Redshift Spectrum enables you to run queries against exabytes of data in Amazon S3. There is no loading or ETL required. Even if you don’t store any of your data in Amazon Redshift, you can still use Redshift Spectrum to query datasets as large as an exabyte in Amazon S3

DynamoDb

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. DynamoDB also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data.

DynamoDB provides on-demand backup capability. It allows you to create full backups of your tables for long-term retention and archival for regulatory compliance needs.DynamoDB allows you to delete expired items from tables automatically to help you reduce storage usage and the cost of storing data that is no longer relevant.

DynamoDB is a particularly good fit for the following use cases:

1. Applications with large amounts of data and strict latency requirements. As your amount of data scales, JOINs and advanced SQL operations can slow down your queries. With DynamoDB, your queries have predictable latency up to any size, including over 100 TBs!

2. Serverless applications using AWS Lambda. AWS Lambda provides auto-scaling, stateless, ephemeral compute in response to event triggers. DynamoDB is accessible via an HTTP API and performs authentication & authorization via IAM roles, making it a perfect fit for building Serverless applications.

3. Data sets with simple, known access patterns. If you're generating recommendations and serving them to users, DynamoDB's simple key-value access patterns make it a fast, reliable choice.

SMRITI PRIYA
101765006
ENC-7

Cloud Computing Lab-4

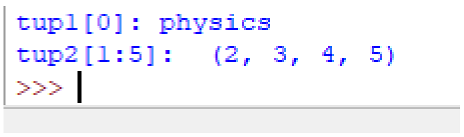
Part A

1. Write a Python program to create a tuple with different data types.

code:

```
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5, 6, 7 )
print("tup1[0]:" , tup1[0])
print("tup2[1:5]: ", tup2[1:5])
```

output:



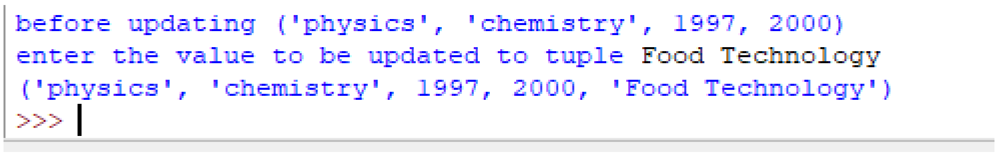
```
tup1[0]: physics
tup2[1:5]:  (2, 3, 4, 5)
>>> 
```

2. Write a Python program to add an item in a tuple

code:

```
tup1 = ('physics', 'chemistry', 1997, 2000)
print('before updating',tup1)
#tup2 = (1, 2, 3, 4, 5, 6, 7 )
ad=input('enter the value to be updated to tuple ')
tup2=(ad,)
print(tup1+tup2)
```

output:



```
before updating ('physics', 'chemistry', 1997, 2000)
enter the value to be updated to tuple Food Technology
('physics', 'chemistry', 1997, 2000, 'Food Technology')
>>> 
```

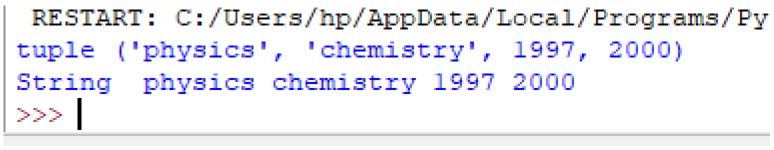
3. Write a Python program to convert a tuple to a string

code:

```
tup1 = ('physics', 'chemistry', 1997, 2000)
print('tuple',tup1)
#tup2 = (1, 2, 3, 4, 5, 6, 7 )
#ad=input('enter the value to be updated to tuple ')
#tup2=(ad,)
s=''
for i in tup1:
    s+=str(i)+' '
print('String ',s)
```

SMRITI PRIYA
101765006
ENC-7

output:

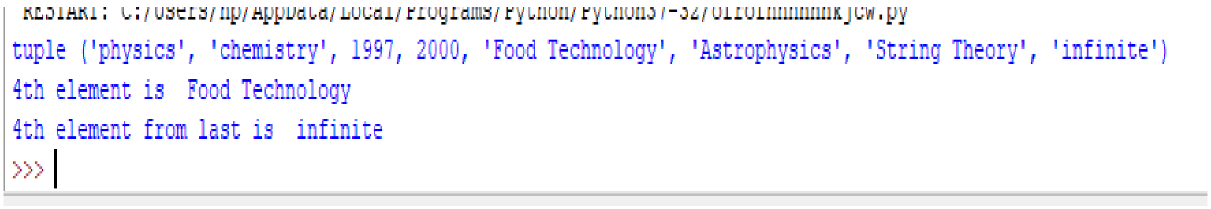


```
 RESTART: C:/Users/hp/AppData/Local/Programs/Py
tuple ('physics', 'chemistry', 1997, 2000)
String  physics chemistry 1997 2000
>>> 
```

4. Write a Python program to get the 4th element and 4th element from last of a tuple.
code:

```
tup1 = ('physics', 'chemistry', 1997, 2000,'Food Technology','Astrophysics','String Theory','infinite')
print('tuple',tup1)
#tup2 = (1, 2, 3, 4, 5, 6, 7 )
#ad=input('enter the value to be updated to tuple ')
#tup2=(ad,)
print('4th element is ',tup1[4])
print('4th element from last is ',tup1[-1])
```

output:

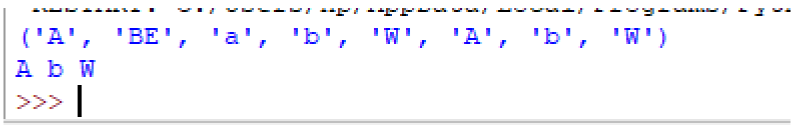


```
 RESTART: C:/Users/hp/AppData/Local/Programs/Python/Python37-32/[illegible].py
tuple ('physics', 'chemistry', 1997, 2000, 'Food Technology', 'Astrophysics', 'String Theory', 'infinite')
4th element is  Food Technology
4th element from last is  infinite
>>> 
```

5. Write a Python program to find the repeated items of a tuple
code:

```
t=('A','BE','a','b','W','A','b','W')
print(t)
l=[]
for i in t:
    if t.count(i)>1 and i not in l:
        l.append(i)
print(*l)
```

output:



```
 RESTART: C:/Users/hp/AppData/Local/Programs/Pyt
('A', 'BE', 'a', 'b', 'W', 'A', 'b', 'W')
A b W
>>> 
```

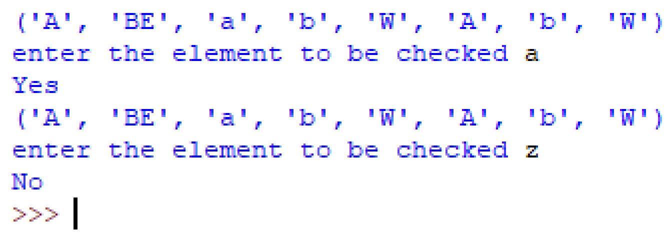
6. Write a Python program to check whether an element exists within a tuple
code:

SMRITI PRIYA
101765006
ENC-7

```
t=('A','BE','a','b','W','A','b','W')
print(t)

i=input('enter the element to be checked ')
if i in t:
   s='Yes'
else:
   s='No'
print(s)
```

output:

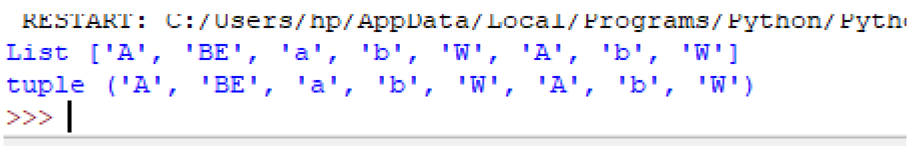


```
('A', 'BE', 'a', 'b', 'W', 'A', 'b', 'W')
enter the element to be checked a
Yes
('A', 'BE', 'a', 'b', 'W', 'A', 'b', 'W')
enter the element to be checked z
No
>>> 
```

7. Write a Python program to convert a list to a tuple
code:

```
t=['A','BE','a','b','W','A','b','W']
print('List',t)
ts=tuple(t)
print('tuple',ts)
```

output:



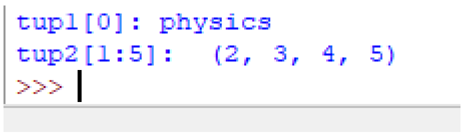
```
 RESTART: C:/Users/hp/AppData/Local/Programs/Python/Pyth
List ['A', 'BE', 'a', 'b', 'W', 'A', 'b', 'W']
tuple ('A', 'BE', 'a', 'b', 'W', 'A', 'b', 'W')
>>> 
```

8. Write a Python program to slice a tuple
code:

```
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5, 6, 7 )
print("tup1[0]:" , tup1[0])
print("tup2[1:5]: ", tup2[1:5])
```

output:

SMRITI PRIYA
101765006
ENC-7

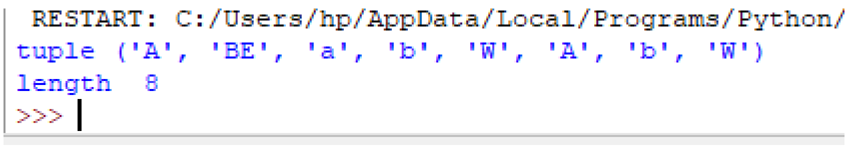


```
tupl[0]: physics
tup2[1:5]:  (2, 3, 4, 5)
>>> 
```

9. Write a Python program to find the length of a tuple
code:

```
t=['A','BE','a','b','W','A','b','W']
#print('List',t)
ts=tuple(t)
print('tuple',ts)
print('length ',len(ts))
```

output:

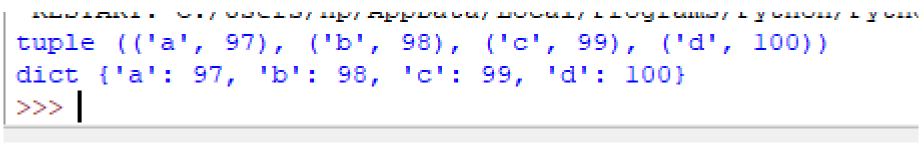


```
 RESTART: C:/Users/hp/AppData/Local/Programs/Python/
tuple ('A', 'BE', 'a', 'b', 'W', 'A', 'b', 'W')
length  8
>>> 
```

10. Write a Python program to convert a list of tuples into a dictionary
code:

```
t=(('a',97),('b',98),('c',99),('d',100))
print('tuple',t)
d={}
for pair in t:
    d[pair[0]]=pair[1]
print('dict',d)
```

output:



```
tuple (('a', 97), ('b', 98), ('c', 99), ('d', 100))
dict {'a': 97, 'b': 98, 'c': 99, 'd': 100}
>>> 
```

Part B

To Understand Map-Reduce Paradigm and summarize the learning in one page.

MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment.

MapReduce consists of two distinct tasks — Map and Reduce.

As the name MapReduce suggests, reducer phase takes place after the mapper phase has been completed.

So, the first is the map job, where a block of data is read and processed to produce key-value pairs as intermediate outputs.

The output of a Mapper or map job (key-value pairs) is input to the Reducer.

The reducer receives the key-value pair from multiple map jobs.

Then, the reducer aggregates those intermediate data tuples (intermediate key-value pair) into a smaller set of tuples or key-value pairs which is the final output.

Advantages of MapReduce:

1. Parallel Processing

In MapReduce, we are dividing the job among multiple nodes and each node works with a part of the job simultaneously. So, MapReduce is based on Divide and Conquer paradigm which helps us to process the data using different machines. As the data is processed by multiple machines instead of a single machine in parallel, the time taken to process the data gets reduced by a tremendous amount

2. Data Locality

Instead of moving data to the processing unit, we are moving the processing unit to the data in the MapReduce Framework. In the traditional system, we used to bring data to the processing unit and process it. But, as the data grew and became very huge, bringing this huge amount of data to the processing unit posed the following issues:

a. Moving huge data to processing is costly and deteriorates the network performance.

b. Processing takes time as the data is processed by a single unit which becomes the bottleneck.

c. Master node can get over-burdened and may fail.

As data is distributed among multiple nodes where each node processes the part of the data residing on it, it has the following advantages

1. It is very cost effective to move the processing unit to the data.

2. The processing time is reduced as all the nodes are working with their part of the data in parallel.

3. Every node gets a part of the data to process and therefore, there is no chance of a node getting overburdened.

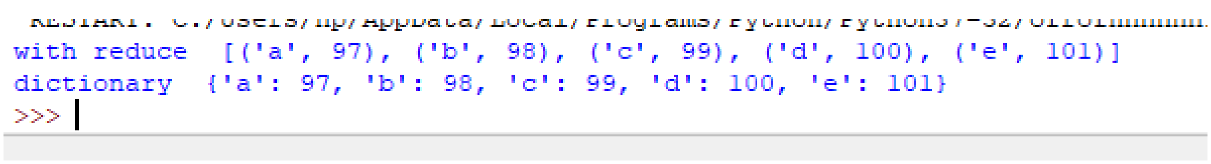
SMRITI PRIYA
101765006
ENC-7

Cloud Computing Lab-5

1. Create a text file named “INPUT” that comprises of some text. Using Mapper and Reducer program generate a KEY and COUNT matrix for the file.
code:

```
from functools import reduce
a=reduce(lambda output, current: output + [(current, ord(current))], 'abcde', [])
print('with reduce ',a)
print('dictionary ',dict(a))
```

output:



```
RESTART: C:/Users/hp/AppData/Local/Programs/Python/Python37-32/[illegible]
with reduce  [('a', 97), ('b', 98), ('c', 99), ('d', 100), ('e', 101)]
dictionary  {'a': 97, 'b': 98, 'c': 99, 'd': 100, 'e': 101}
>>> 
```

2. XYZ.com is an online music website where users listen to various tracks, the data gets collected like shown below. Write a map reduce program to get following stats

1. Number of unique listeners
2. Number of times the track was shared with others
3. Number of times the track was listened to on the radio
4. Number of times the track was listened to in total
5. Number of times the track was skipped on the radio

The data is coming in log files and looks like as shown below.
UserId|TrackId|Shared|Radio|Skip
111115|222|0|1|0
111113|225|1|0|0
111117|223|0|1|1
111115|225|1|0|0

code:

output:

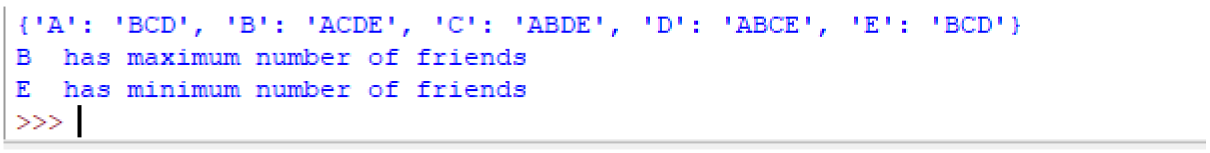
3. Five persons A, B, C, D, E have a friendship relation as shown below.
A -> B C D
B -> A C D E
C -> A B D E
D -> A B C E
E -> B C D
Identify who is having maximum friends, minimum friends and no friends.

code:

SMRITI PRIYA
101765006
ENC-7

```
d={'A':'BCD','B':'ACDE','C':'ABDE','D':'ABCE','E':'BCD'}
print(d)
maxx='A'
minn='A'
for i in d:
   if len(d[i])>len(d[maxx]):
      maxx=i
   if len(d[i])<len(d[maxx]):
      minn=i
print(maxx,' has maximum number of friends')
print(minn,' has minimum number of friends ')
```

output:



```
{'A': 'BCD', 'B': 'ACDE', 'C': 'ABDE', 'D': 'ABCE', 'E': 'BCD'}
B  has maximum number of friends
E  has minimum number of friends
>>> 
```