Python Activity -3

Dictionary and Multidimensional data structure

Q1. Given DBI={'db1':'sql','db2':'sqlite','db3':'mysql'}

Write a python program to

a. Display 'db2' value

b. Replace mysql with pl/sql

c. Display list of keys from DBI

--

Q2.Write a python program

a. Create an empty dictionary

b. Display the size of dictionary

c. Read the following details from STDIN - employee name,ID,dept,place and

initialize them to dictionary.

d. Using keys() function, display dictionary details(key,value)

Q3. Write a python program

Given list structure

----

emp=["e123,ram,sales,pune,1000","e132,kumar,prod,bglore,3423",
"e456,arun,prod,chennai,2456","e544,vijay,hr,mumbai,6500"]

a. create an empty dictionary and name it as EMP

b. convert the above given list into dict format.

Note:- employee id as a key, emp name as value

c. display list of key,value pairs from EMP dict

Q4. Write a python program

hosts={"alias1":"host1.example.com",

"alias2":"host2.example.com","alias3":"host3.example.com"}

a. display key,value pairs to monitor.

b. read a alias name from STDIN(Keyboard)

   test input key (alias) name is existing or not.

c. if key exists,update the lo (LOCAL HOST) address (127.0.0.1) to input key.

d. display key,value pairs to monitor.

Q5. Write a python program and convert below declaration into dictionary format as specified.

```
sub1="python"
sub2="ruby"
sub3="perl"
sub4="java"
sub5="oracle"

os1="OL5"
os2="OL6"
os3="OL7"
```

a. create a dict name as "Course"

b. create two keys named as "Subject" and "OS"

c. initialize list of subject names to "Subject" key and list of os names to "OS" key

d. display list of item pair to screen.

Q6. Given dictionary

conf={"f1":"/etc/passwd","f2":"/etc/group","f3": "/etc/sysconfig","f4":None}

a. Determine the size of conf dictionary
b. Add new configuration file (/etc/pam.d)
c. Using keys() and get() display key,value details

Q7. Create an empty dictionary student.

using setdefault() function, add the following student details(student name,ID,dept,DOB to student dictionary.

a. Using keys() and get(), display the student details

b. using items() display student details

# understand the difference between return value of keys() & items()

Q8. Given dict structure

```
Proc={'pid':12,'fs':'/proc','user':'root','sh':'/bin/bash'}
using pop() - delete 'fs' and 'sh' key entries
using del() - delete 'pid' and 'user' entries
# what's the difference between pop() and del()
using popitem() - delete Proc structure
```

Q9. Identify the errors

a)

```
 car =   {
"brand": "Ford"
"model": "Mustang"
"year": 1964
}
```

b) fs={"ftype":"ext4","proto":"tcp/ip","port":80}

```
>>>fs["ext4"]
```

Q10. Identify the errors

a) `import sys;sys.modules=[os]`

b)
```
d1={"k1":"v1"}
d1.pop()
```

c)
```
d2={}
d2.setdefault("K1","V1","K2","V2")
```

Multidimensional Data structures

Q1.

```
action_model = {
        'request': {
            'operation': 'DeleteTags',
            'params': [{
                'target': 'Resources[0]',
                'source': 'identifier',
                'name': 'Id'
            }]
        }
    }
```

A. Determine the given structure type

B. How to print 'name' value

Q2.

```
Cloudwatch={
    AlarmName:"Web_Server_CPU_Utilization",
    ComparisonOperator:'GreaterThanThreshold',
    EvaluationPeriods:1,
    MetricName:'CPUUtilization',
    Namespace:'AWS/EC2',
    Period:60,
    Statistic:'Average',
    Threshold:70.0,
    ActionsEnabled:False,
    AlarmDescription:'Alarm when server CPU exceeds 70%',
    Dimensions:[
        {
          'Name': 'InstanceId',
          'Value': 'INSTANCE_ID'
        },
    ],
    Unit:'Seconds'
}
```

a. print structure type

b. How to display 'Namespace' and 'Threshold' values

c. How to add the following as a 'Dimensions' value{'Name1':'InstanceID1','Value1':'InstanceID2'}

d. modify 'Unit' value to 'Minits'

Q3. Given Structure

```
S={
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "cloudwatch:Describe*",
            "ec2:Describe*",
            "ec2:RebootInstances",
            "ec2:StopInstances*",
            "ec2:TerminateInstances"
         ],
         "Resource": [
            "*"
         ] }
         ]}
```

a. How to display the list of instances from 'Action' key ?

Q4. Given structure

```
namedtuple=(
    'ServiceContext',
    ['service_name', 'service_model', 'service_waiter_model',
     'resource_json_definitions']
)
```

Display the tuple data members

Q5.

```
# Create a relationship definition and attach it
# to the model, such that all identifiers must be
# supplied by the user. It will look something like:
#
         # {
          #   'resource': {
          #     'type': 'ResourceName',
          #     'identifiers': [
          #       {'target': 'Name1', 'source': 'input'},
          #       {'target': 'Name2', 'source': 'input'},
          #       ...
          #     ]
          #   }
          # }
          #
          fake_has = {
              'resource': {
                  'type': name,
                  'identifiers': [] } }
```

a. Display the 'identifiers' value.

b. Add new entries ({'target': 'Name2', 'source': 'input'}) to 'identifiers'

Q6.

```
my_managed_policy = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "RESOURCE_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:Scan",
                "dynamodb:UpdateItem"
            ],
            "Resource": "RESOURCE_ARN"
        }
    ]
}
```

Understand the above structure.

a. display each structure elements

b. insert following data members to Statement key

```
{
"Effect":"Allow",
"Action":"logs:CreateLogMembers",
"Resource":["RESOURCE_ARN","RESOURCE_SAB","RESO
URCE_AB"]}
```

Q7. Write a python program

Given dictionary structure

```
ResponseMetadata={
'RequestId': 'nnnnn-e323-nn-a9a3-254nnnn2c3b6',
'RetryAttempts': 0,
'HTTPHeaders': None,
'transfer-encoding': 'chunked',
'content-type': 'text/xml',
'vary': 'Accept-Encoding',
'server': 'AmazonEC2',
'HTTPStatusCode': 200
}
```

a. Display key,value details to screen.

b. Assign a value to 'HTTPHeaders'

c. Modify the value 'text/xml' into 'text/html'

Note : before modifying 'text/xml' value, test input key 'content-type' exists or NOT

(use : get() function)

d. Display key,value details to screen (compare 'a' statement)

Q8. Given structure

```
stry = {
  ad_   : { class : 'DBD::AnyData',  },
  ad2_   : { class : 'DBD::AnyData2', },
  ado_  : { class : 'DBD::ADO',  },
  amzn_  : { class : 'DBD::Amazon',  },
  best_  : { class : 'DBD::BestWins', },
  csv_   : { class : 'DBD::CSV', },
  dbi_   : { class : 'DBI', },
  dbm_  : { class : 'DBD::DBM',},
  df_  : { class : 'DBD::DF',},
  examplep_    : { class : 'DBD::ExampleP', },
}
```

a. how to add new DBD into existing dictionary
   (ex: db2 : DBD::DB2)
b. display list of keys from **stry** dictionary

Q9. Given structure

```
EXPORT_TAGS = {
   sql_types :[
        SQL_GUID
        SQL_WLONGVARCHAR
        SQL_WVARCHAR
        SQL_WCHAR
        SQL_BIGINT
        SQL_BIT
        SQL_TINYINT
        SQL_LONGVARBINARY
        SQL_VARBINARY
        SQL_BINARY ]
}
```

Display each **key,value** details to screen.

Q10. Convert Given structure to dictionary of dictionary format.

```
EXPORT_TAGS={"html2":["h1":"header1","h2":"header2"],
             "html3":["cgi":"param"],
             "html5":["https":"urllib2","requests":"bs4"]
}
```

Q11.write a python program

Given list structure

```
emp=["e123,ram,sales,pune,1000",
          "e132,kumar,prod,bglore,3433",
    "e456,arun,prod,chennai,2456",
    "e544,vijay,hr,mumbai,6500"
]
```

a. create a empty dictionary name as EMP

b. convert the above given list into dict format

c. employee id as a key, rest of the details(name,dept,place,cost) as values

d. display list of **key,value** pairs from EMP dict

Expected structure is

```
EMP={"e123":["ram","sales","pune",1000],...}
```

Q12. Convert the given list to dictionary of list

OS=["OL5","RHL5","OL6","OL7","RHL7","DEB5",

"OL6","OL7","RHL5"]

# Note - ignore duplicate values

# Expected Result is

# os={"os"=>["OL5","RHL5","OL6","OL7","RHL7","DEB5"]}