**Practical 1: Working and Implementation of Infrastructure as a Service (IaaS)**

```
# Simple demonstration of IaaS - Creating Virtual Machines on demand

class VirtualMachine:
    def __init__(self, cpu, ram, storage):
        self.cpu = cpu
        self.ram = ram
        self.storage = storage
        self.status = "Stopped"

    def start(self):
        self.status = "Running"

    def stop(self):
        self.status = "Stopped"

    def details(self):
        return f"VM -> CPU: {self.cpu}, RAM: {self.ram}GB, Storage: {self.storage}GB, Status: 
{self.status}"



# Cloud provider giving infrastructure resources
class IaaSProvider:
    def __init__(self):
        self.vms = []

    def create_vm(self, cpu, ram, storage):
        vm = VirtualMachine(cpu, ram, storage)
        self.vms.append(vm)

        print("Virtual Machine Created!")

        return vm



# ----------- Main Program --------------

provider = IaaSProvider()


# User creates virtual machines (like AWS EC2)

vm1 = provider.create_vm(cpu="2 vCPU", ram=4, storage=100)

vm1.start()


vm2 = provider.create_vm(cpu="4 vCPU", ram=8, storage=200)

vm2.start()


# Show details

print(vm1.details())

print(vm2.details())
```

**Practical 2: Working and Implementation of Software as a Service (SaaS)**

```
# SaaS Example: A simple online text editor service

class TextEditorSaaS:
    def __init__(self):
        self.documents = {}

    def create_document(self, user, content):
        self.documents[user] = content
        print(f"Document created for {user}")

    def update_document(self, user, new_content):
        if user in self.documents:
            self.documents[user] = new_content
            print(f"Document updated for {user}")
        else:
            print("No document found for this user.")

    def view_document(self, user):
        return self.documents.get(user, "No document found.")
```

```
# ----------- Main Program --------------
app = TextEditorSaaS()

# User uses the software online (like Google Docs)
app.create_document("Gaurav", "This is my first online document.")
app.update_document("Gaurav", "Updated content in the cloud document."
print(app.view_document("Gaurav"))
```

**Practical 3: Working and Implementation of Platform as a Service (PaaS)**

```
# PaaS Example: A platform that allows users to deploy and run applications

class PaaSPlatform:
    def __init__(self):
        self.apps = []

    def deploy_app(self, app_name, code):
        app = {"name": app_name, "code": code, "status": "Deployed"}
        self.apps.append(app)
        print(f"App '{app_name}' deployed successfully!")

    def run_app(self, app_name):
        for app in self.apps:
            if app["name"] == app_name:
                print(f"Running app '{app_name}':")
                exec(app["code"])   # Simulates running user code
                return
        print("App not found!")

    def list_apps(self):
        print("Deployed Applications:")
        for app in self.apps:
            print(f"- {app['name']} ({app['status']})")


# ----------- Main Program --------------
platform = PaaSPlatform()
```

```
# User deploys an application on PaaS (like Heroku / Google App Engine)
user_code = """
print('Hello from my cloud application!')
"""

platform.deploy_app("MyCloudApp", user_code)

# Run the app
platform.run_app("MyCloudApp")

# List all deployed apps
platform.list_apps()
```

**Practical 4: Practical Implementation of Storage as a Service (STaaS)**

```
# Storage as a Service Example: Uploading, downloading, and listing files in cloud storage

class CloudStorage:
    def __init__(self):
        self.storage = {}

    def upload_file(self, filename, content):
        self.storage[filename] = content
        print(f"File '{filename}' uploaded successfully!")

    def download_file(self, filename):
        if filename in self.storage:
            print(f"Downloading '{filename}'...")
            return self.storage[filename]
        else:
            return "File not found."

    def list_files(self):
        print("Files in Cloud Storage:")
        for file in self.storage:
            print("-", file)


# ----------- Main Program --------------
cloud = CloudStorage()

# Uploading files (like Google Drive / Dropbox)
```

```
cloud.upload_file("notes.txt", "This is my cloud note.")

cloud.upload_file("photo.png", "<binary image data>")


# Download a file

print(cloud.download_file("notes.txt"))


# List all files stored

cloud.list_files()
```

**Practical 5: Working of Google Drive to Make Spreadsheet and Notes**

```
# Simulation of Google Drive – Creating Notes & Spreadsheets

class GoogleDrive:
  def __init__(self):
    self.notes = {}
    self.spreadsheets = {}

  # Create a note
  def create_note(self, title, content):
    self.notes[title] = content
    print(f"Note '{title}' created!")

  # Create spreadsheet (rows & columns)
  def create_spreadsheet(self, name, rows, cols):
    sheet = [[0 for _ in range(cols)] for _ in range(rows)]
    self.spreadsheets[name] = sheet
    print(f"Spreadsheet '{name}' created with {rows} rows & {cols} columns!")

  # Update a cell in spreadsheet
  def update_cell(self, name, row, col, value):
    if name in self.spreadsheets:
      self.spreadsheets[name][row][col] = value
      print(f"Updated cell ({row},{col}) with value: {value}")
    else:
      print("Spreadsheet not found!")

  # View spreadsheet
```

```
    def view_spreadsheet(self, name):
        if name in self.spreadsheets:
            for row in self.spreadsheets[name]:
                print(row)
        else:
            print("Spreadsheet not found!")

    # View note
    def read_note(self, title):
        return self.notes.get(title, "Note not found.")



# ---------------- Main Program ----------------
drive = GoogleDrive()

# Create a note (like Google Keep)
drive.create_note("My Note", "This is my note stored in cloud.")
print(drive.read_note("My Note"))

# Create spreadsheet (like Google Sheets)
drive.create_spreadsheet("BudgetSheet", 3, 3)

# Update spreadsheet cells
drive.update_cell("BudgetSheet", 0, 0, "Month")
drive.update_cell("BudgetSheet", 0, 1, "Income")
drive.update_cell("BudgetSheet", 0, 2, "Expense")

# View the spreadsheet
drive.view_spreadsheet("BudgetSheet")
```

**Practical 6: Web Feed (RSS Feed Reader Example in Python)**

```
import xml.etree.ElementTree as ET

rss_data = """
<rss>
  <channel>
    <title>Tech News</title>
    <item>
      <title>Cloud Computing Updates</title>
      <link>http://example.com/cloud</link>
    </item>
    <item>
      <title>AI Breakthrough</title>
      <link>http://example.com/ai</link>
    </item>
  </channel>
</rss>
"""

# Parse RSS feed
root = ET.fromstring(rss_data)

print("RSS Feed Items:")
for item in root.findall("./channel/item"):
    title = item.find("title").text
    link = item.find("link").text
    print(f"- {title}: {link}")
```

**Practical 7: Virtualization in Cloud Computing – Basics & Benefits (Simulation)**

```
class VirtualMachine:
    def __init__(self, os_name):
        self.os = os_name
        self.state = "Stopped"

    def start(self):
        self.state = "Running"
        print(f"{self.os} VM started.")

    def stop(self):
        self.state = "Stopped"
        print(f"{self.os} VM stopped.")

    def info(self):
        return f"OS: {self.os}, State: {self.state}"


class KVM_Hypervisor:
    def __init__(self):
        self.vms = []

    def create_vm(self, os_name):
        vm = VirtualMachine(os_name)
        self.vms.append(vm)
        print(f"{os_name} VM created on KVM.")
        return vm
```

```
# Main Program

hypervisor = KVM_Hypervisor()

vm1 = hypervisor.create_vm("Ubuntu")

vm1.start()

print(vm1.info())
```

**Practical 8: Demonstration of Cloud with Single Sign-On (SSO)**

```
class AuthServer:
  def __init__(self):
    self.tokens = {}

  def login(self, username):
    token = username + "_TOKEN"
    self.tokens[username] = token
    print("Login Successful! Token generated.")
    return token


class App:
  def access(self, token):
    if "_TOKEN" in token:
      print("Access granted using SSO token.")
    else:
      print("Access denied.")


# Main Program
auth = AuthServer()
app = App()

# User logs in once
token = auth.login("Gaurav")
# Same token works for multiple apps
app.access(token)
```

**Practical 9: Cloud Hadoop Installation & Query Demo (Simulation)**

```
class HadoopCluster:
    def __init__(self):
        self.data = []

    def upload_data(self, item):
        self.data.append(item)
        print("Data uploaded to Hadoop Cluster.")

    def query(self, keyword):
        print(f"Querying Hadoop for: {keyword}")
        return [d for d in self.data if keyword.lower() in d.lower()]


# Main Program
cluster = HadoopCluster()
cluster.upload_data("Cloud Computing Notes")
cluster.upload_data("Big Data with Hadoop")
cluster.upload_data("Hadoop MapReduce Program")

result = cluster.query("hadoop")
print("Query Result:", result)
```

**Practical 10: Installing and Developing Application Using Google App Engine (Simulation)**

```
class GoogleAppEngine:
    def __init__(self):
        self.apps = {}

    def deploy(self, name, code):
        self.apps[name] = code
        print(f"App '{name}' deployed on Google App Engine!")

    def run(self, name):
        if name in self.apps:
            print("Running App:")
            exec(self.apps[name])
        else:
            print("App not found.")


# Main Program
gae = GoogleAppEngine()

app_code = """
print("Hello from Google App Engine!")
"""

gae.deploy("MyGAEApp", app_code)
gae.run("MyGAEApp")
```

**Practical 11: Working and Implementation of Identity Management**

```
class IdentityManager:
    def __init__(self):
        self.users = {}

    def register(self, username, role):
        self.users[username] = role
        print(f"User '{username}' registered with role '{role}'.")

    def authenticate(self, username):
        if username in self.users:
            print("Authentication Successful.")
            return True
        else:
            print("Authentication Failed.")
            return False

    def authorize(self, username, required_role):
        if self.users.get(username) == required_role:
            print("Authorization Successful. Access Granted.")
        else:
            print("Authorization Failed. Access Denied.")



# Main Program
idm = IdentityManager()
idm.register("Gaurav", "Admin")
```

```
if idm.authenticate("Gaurav"):
    idm.authorize("Gaurav", "Admin")
```