Practical 1: Dictionaries (Part I)

Q1 Add key/value pair to a dictionary by using subscript operator.

```
Keys = ["name", "occupation"], Values = ["Om", "Teacher"]
```

Q2 Replace value of existing key = 'occupation' = 'Manager'

```
Dict = {'name: 'Om', 'occupation': 'Teacher'}
```

Q3 Write a Python program to check if value 30 exists in the following dictionary:

```
D = {"age1": 20, "age2" : 30, "age3" : 40}
```

Q4 Remove a key from a dictionary

```
Dict = {"name": "Om", "age": 20, "salary": 8000, "city": "Mumbai"}
```

Q5 Assume that the variable data refers to the dictionary `{'b':20, 'a':35}`. Write the expressions that perform the following tasks:

a) Replace the value at the key 'b' in data with that value's negation.
b) Add the key/value pair 'c': 40 to data.
c) Remove the value at key 'b' in data, safely.
d) Print the keys in data in alphabetical order.

Q6 Give three examples of real-world objects that behave like a dictionary.

Q7 Assume that the variable `data` refers to the dictionary `{'b':20, 'a':35}`. Write the values of the following expressions:

```
a) data['a']
b) data.get('c', None)
c) len(data)
d) data.keys()
e) data.values()
f) data.pop('b')
g) data
```

Q8 Sort dictionaries by key in Python.

```
{'ravi': 10, 'rajnish': 9, 'sanjeev': 15, 'yash': 2, 'suraj': 32}
```

Signature of the instructor **Date**

Assignment Evaluation

0:Not done	2:Late Complete	4:Complete	
1:Incomplete	3:Needs improvement	5:Well Done	

Practical 1: Dictionaries (Part I)

1)

```
d = {}
d["name"] = "Om"
d["occupation"] = "Teacher"
```

2)

```
d = {}
d["name"] = "Om"
d["occupation"] = "Teacher"
d['occupation'] = 'Manager'
```

3)

```
D = {'age1': 20, 'age2' : 30, 'age3' : 40}
print(30 in D.values())
```

4)

```
d = {"name": "Om", "age": 20, "salary": 8000, "city": "Mumbai"}
del d["city"]
print(d)
```

5)

```
d={'b':20, 'a':35}
d['b']=-d['b']
d
{'b': -20, 'a': 35}
d['c'] = 40
d
```

{'b': -20, 'a': 35, 'c': 40}

del d['b']

d

{'a': 35, 'c': 40}

sorted(d)

['a', 'c']

6)

Phone Contact List

- **Key** → Person's name
- **Value** → Phone number

English Dictionary (word book)

- **Key** → Word
- **Value** → Meaning

Student Roll Register

- **Key** → Roll number
- **Value** → Student name/details

- PIN code → City
- Vehicle number → Owner
- Username → Password

7)

data={'b':20, 'a':35}

data['a']

35

```
data.get('c', None)
data
{'b': 20, 'a': 35}
len(data)
2
data.keys()
dict_keys(['b', 'a'])
data.values()
dict_values([20, 35])
data.pop('b')
20
data
{'a': 35}
```

8)

```
d={'ravi': 10, 'rajnish': 9, 'sanjeev': 15, 'yash': 2, 'suraj': 32}
sorted(d)
['rajnish', 'ravi', 'sanjeev', 'suraj', 'yash']
```

Practical 2: Dictionaries (Part II)

Q1 Merge the following dictionaries

```
d1 = {"a" : 1, "b" : 2, "c" : 3};    d2 = {"d" : 4, "e" : 5, "f" : 6}
```

Q2 Create the ordered dictionary for the data

```
keys = {0, 1, 4, 5, 6, 3, 2};     values = {-9, -7, -4, -6, -3, 3, 4}
```

Q3 Create dictionary with keys as `{1, 2, 3..., 8}` and corresponding values

```
{'one', 'two',..., 'eight'}
```

Q4 Find minimum, maximum form the given dictionary. Also, sort the dictionary

```
adict = { 'n' : 1, 's' : 2, 'r' : 2, 'u' : 2, 't' : 1}
```

Q5 Extract the 3 largest elements from the list

```
l = [5, 3, 4, 7, 8, -6, -8, -3, 0]
```

Q6 Write a Python program to concatenate the following dictionaries to create a new one.

```
Dict1 = {1:10, 2:20};    Dict2 = {3:30, 4:40};    Dict3 = {5:50,6:60}
```

```
Expected Result = {1:10, 2:20, 3:30, 4:40, 5:50, 6:60}
```

Q7 . Finding the minimum and maximum of a sequence of sequences.

```
L=[[5,3,4,7,8,-6,-8,-3,0],  [6,5,6,3],  [7,4,3,5]]
```

Q8 Write a Python program to combine two dictionary by adding values for common keys.

```
D1 = {'a':100, 'b':200, 'c':300};   D2 = {'a':300, 'b':200, 'd':400}
```

Expected Result : `{'a':400, 'b':400, 'c':300, 'd':400}`

Signature of the instructor ☐ **Date** ☐

Assignment Evaluation

0:Not done	☐	**2:Late Complete**	☐	**4:Complete**	☐
1:Incomplete	☐	**3:Needs improvement**	☐	**5:Well Done**	☐

Practical 2: Dictionaries (Part II)

1)

```
d1 = {'a' : 1, 'b' : 2, 'c' : 3}
d2 = {'d' : 4, 'e' : 5, 'f' : 6}
d={**d1,**d2}
d
{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6}
```

2)

```
keys = [0, 1, 4, 5, 6, 3, 2]
values = [-9, -7, -4, -6, -3, 3, 4]
from collections import OrderedDict
d = OrderedDict()
for i in keys:
    d[i]=values[i]
d
OrderedDict({0: -9, 1: -7, 4: -3, 5: 3, 6: 4, 3: -6, 2: -4})
```

3)

```
keys = [1,2,3,4,5,6,7,8]
values = ["one", "two", "three", "four", "five", "six", "seven", "eight"]
d = dict(zip(keys, values))
print(d)
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five', 6: 'six', 7: 'seven', 8:
'eight'}
```

4)

```
adict = {'n': 1, 's': 2, 'r': 2, 'u': 2, 't': 1}
min(adict)
'n'
max(adict)
'u'
```

5)

```
import heapq
l = [5, 3, 4, 7, 8, -6, -8, -3, 0]
heapq.nlargest(3, l)
[8, 7, 5]
```

6)

```
Dict2 = {3:30, 4:40};
Dict1 = {1:10, 2:20}
Dict2 = {3:30, 4:40}
Dict3 = {5:50,6:60}
result = {**Dict1, **Dict2, **Dict3}
print(result)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

7)

```
L = [[5,3,4,7,8,-6,-8,-3,0], [6,5,6,3], [7,4,3,5]]
min_val = min(min(sublist) for sublist in L)
max_val = max(max(sublist) for sublist in L)
print("Minimum:", min_val)
```

Minimum: -8

print("Maximum:", max_val)

Maximum: 8

8)

```
D1 = {'a':100, 'b':200, 'c':300}

D2 = {'a':300, 'b':200, 'd':400}

result = {k: D1.get(k,0) + D2.get(k,0) for k in D1.keys() | D2.keys()}

print(result)
```

{'a': 400, 'b': 400, 'c': 300, 'd': 400}

Practical 3: Simple Graphics (Part I)

Q1 Draw a letter “M” using turtle methods with width 2.
Q2 Draw a circle of radius 200 pixel.
Q3 Initialize the Turtle canvas and draw the following commands:

a) Move forward 200 pixel
b) Turn 90 degrees left.
c) Move forward 50 pixel.
d) Change width of the pencil to 5.
e) Move forward 100 pixel with red color.
f) Turn 90 degrees right.
g) Move forward 200 pixel.

Q4 Write a program to draw these regular polygons.

a) An equilateral triangle.
b) A Square
c) A octagon
d) A hexagon.

Q5 Write a program to change background color of turtle’s screen.
Q6 Draw a half circle in clockwise direction of radius 200 pixel.
Q7 Plot a rectangle of a sides 100 and 150 pixels starting with its current position.

Signature of the instructor **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**		**4:Complete**		
1:Incomplete		**3:Needs improvement**		**5:Well Done**		

Practical 3: Simple Graphics (Part I)

1)

```
from turtle import *
width(2)
speed(1)
penup()
goto(-100, -50)
pendown()
left(90)
forward(100)
right(150)
forward(60)
left(120)
forward(60)
right(150)
forward(100)
```

2)

```
from turtle import *
t=circle(200)
```

3)

```
from turtle import *
speed(1)
forward(200)
left(90)
```

```
forward(50)
width(5)
color("red")
forward(100)
right(90)
forward(200)
```

4)

```
from turtle import *
speed(1)
width(2)
penup()
goto(-300, 0)
pendown()
for i in range(3):
    forward(100)
    left(120)
penup()
goto(-100, 0)
pendown()
for i in range(4):
    forward(100)
    left(90)
penup()
goto(100, 0)
```

```
pendown()
for i in range(8):
    forward(70)
    left(45)
penup()
goto(300, 0)
pendown()
for i in range(6):
    forward(80)
    left(60)
```

5)

```
from turtle import *
bgcolor('blue')
```

6)

```
from turtle import *
circle(200,-180)
```

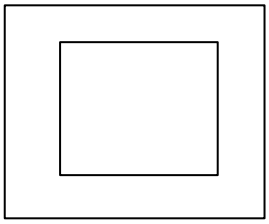
7)

```
from turtle import *
speed(1)
width(2)
forward(150)
left(90)
forward(100)
left(90)
```

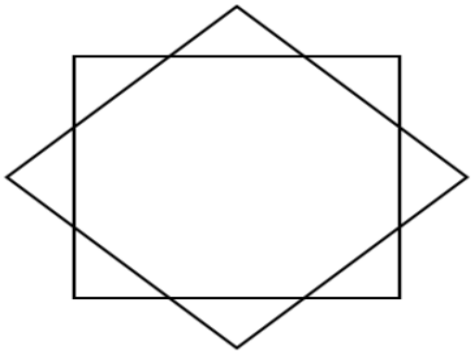
```
forward(150)
left(90)
forward(100)
```

Practical 4: Simple Graphics (Part II)

Q1 Write a Python Program to draw a shape like this:



Q2 Write a Python Program to draw a shape like this:



Q3 Draw a circle of different radius with random colours.
Q4 Draw a letter “MATH” using turtle method with width 4.
Q5 Draw a circle clockwise in of radius 200 pixel in 7 steps.

Signature of the instructor **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**	**4:Complete**
1:Incomplete		**3:Needs improvement**	**5:Well Done**

Practical 4: Simple Graphics (Part II)

1)

```
from turtle import *
speed(3)
width(2)
for i in range(2):
    forward(200)
    left(90)
    forward(150)
    left(90)
penup()
forward(40)
left(90)
forward(30)
right(90)
pendown()
for i in range(2):
    forward(120)
    left(90)
    forward(80)
    left(90)
```

2)

```
from turtle import *
speed(3)
```

```
width(2)
penup()
goto(-100, -100)
pendown()
for i in range(4):
    forward(200)
    left(90)
penup()
goto(0, -150)
setheading(45)
pendown()
for i in range(4):
    forward(212)
    left(90)
```

3)

```
from turtle import *
import random
speed(3)
width(2)
for r in range(50, 251, 40):
    color(random.random(), random.random(), random.random())
    circle(r)
```

4)

```
from turtle import *
speed(3)
width(4)
# ---- M ----
penup()
goto(-300, -50)
pendown()
left(90)
forward(100)
right(135)
forward(70)
left(90)
forward(70)
right(135)
forward(100)
# ---- A ----
penup()
goto(-150, -50)
setheading(75)
pendown()
forward(105)
right(150)
forward(105)
```

```
penup()
goto(-165, 0)
setheading(0)
pendown()
forward(70)
# ---- T ----
penup()
goto(0, 50)
setheading(0)
pendown()
forward(100)
penup()
goto(50, 50)
setheading(-90)
pendown()
forward(100)
# ---- H ----
penup()
goto(180, -50)
setheading(90)
pendown()
forward(100)
penup()
goto(260, -50)
```

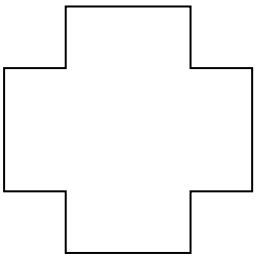
```
pendown()
forward(100)
penup()
goto(180, 0)
setheading(0)
pendown()
forward(80)
```

5)

```
from turtle import *
speed(3)
width(2)
r = 200
steps = 7
angle = -360 / steps
for i in range(steps):
    circle(r, angle)
```

Practical 5: Simple Graphics (Part III)

Q1 Write a Python program to draw a shape like this:



Q2 Draw a random walk using random function.

Q3 Draw a random walk with random color having 50 steps using random function.

Q4 Draw a random walk with 100 steps using random function.

Q5 Using random function, draw a random walk with 50 steps which travels forward between 30 and 100 and moves right between 0 and 360 randomly

Signature of the instructor **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**		**4:Complete**			
1:Incomplete		**3:Needs improvement**		**5:Well Done**			

Practical 5: Simple Graphics (Part III)

1)

```
from turtle import *
speed(3)
width(2)
penup()
goto(-50, -150)
pendown()
forward(100)
left(90)
forward(100)
right(90)
forward(100)
left(90)
forward(100)
left(90)
forward(100)
right(90)
forward(100)
left(90)
forward(100)
left(90)
forward(100)
right(90)
```

```
forward(100)
left(90)
forward(100)
left(90)
forward(100)
right(90)
forward(100)
```

2)

```
from turtle import *
import random
speed(0)
width(2)
for i in range(100):
    forward(20)
    turn = random.choice([0, 90, 180, 270])
    setheading(turn)
    color(random.random(), random.random(), random.random())
```

3)

```
from turtle import *
import random
speed(0)
width(2)
for i in range(50):
    forward(20)
```

```
    turn = random.choice([0, 90, 180, 270])
    setheading(turn)
    color(random.random(), random.random(), random.random())
```

4)

```
from turtle import *
import random
speed(0)
width(2)
for i in range(100):
    forward(20)
    turn = random.choice([0, 90, 180, 270])
    setheading(turn)
    color(random.random(), random.random(), random.random())
```

5)

```
from turtle import *
import random
speed(0)
width(2)
for i in range(50):
    fd(random.randint(30, 100))
    right(random.randint(0, 360))
```

Practical 6: Complex Numbers in Python

Q1 Consider complex number `z1= 2+3j, z2 = 4+5j`; perform following operations:

A) `z1+z2`
B) `z1 + 7`
C) `z1 - z2`
D) `z1 * z2`
E) `z1/z2`

Q2 Convert `x=2, y=3` into complex number.

Q3 Convert complex number `3+2i` into polar form.

Q4 Write a Python Program to print a complex number and its real and imaginary parts.

Q5 Write a Python Program to find conjugate of `z = 4+3j`.

Q6 Write a Python program to find argument of complex number.

Signature of the instructor **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**		**4:Complete**			
1:Incomplete		**3:Needs improvement**		**5:Well Done**			

Practical 6: Complex Numbers in Python

1)

```
z1 = 2 + 3j

z2 = 4 + 5j

print('A) z1 + z2 =', z1 + z2)

print('B) z1 + 7 =', z1 + 7)

print('C) z1 - z2 =', z1 - z2)

print('D) z1 * z2 =', z1 * z2)

print('E) z1 / z2 =', z1 / z2)
```

```
A) z1 + z2 = (6+8j)

B) z1 + 7 = (9+3j)

C) z1 - z2 = (-2-2j)

D) z1 * z2 = (-7+22j)

E) z1 / z2 = (0.5609756097560976+0.04878048780487805j)
```

2)

```
x = 2

y = 3

z = complex(x, y)

print(z)
```

```
(2+3j)
```

3)

```
from cmath import *

z = 3 + 2j

r, theta = polar(z)
```

```
print("Modulus (r) =", r)

print("Argument (theta in radians) =", theta)
```

Modulus (r) = 3.605551275463989

Argument (theta in radians) = 0.5880026035475675

4)

```
z = 3 + 4j
print("Complex number =", z)
print("Real part =", z.real)
print("Imaginary part =", z.imag)
```

Complex number = (3+4j)
Real part = 3.0
Imaginary part = 4.0

5)

```
z = 4 + 3j

zc = z.conjugate()

print("Complex number =", z)

print("Conjugate =", zc)
```

Complex number = (4+3j)

Conjugate = (4-3j)

6)

```
from cmath import *

z = 3 + 4j

theta = phase(z)

print("Complex number =", z)
print("Argument (in radians) =", theta)
```

Complex number = (3+4j)
Argument (in radians) = 0.9272952180016122

Practical 7: Complex Number in Python

Q1 Trigonometric functions for a complex number

```
a = 3 + 4j
```

Show the result for following expression.

```
A)print('Sine:', cmath,sin(a))
B)print('Cosine', cmath.cos(a))
C)print('Tangent:', cmath.tan(a))
D)print('ArcSin:', cmath.asin(a))
E)print('ArcCosine:', cmath.acos(a))
F)print('ArcTan:', cmath.atan(a))
```

Q2 Hyperbolic Functions

```
b = 2 - 5j
```

Show the result for following expression.

```
A)print('Hyperbolic Sine:', cmath.sinh(b))
B)print('Hyperbolic Cosine:', cmath.cosh(b))
C)print('Hyperbolic Tangent:', cmath.tanh(b))
D)print('Inverse Hyperbolic Sine:', cmath.asinh(b))
E)print('Inverse Hyperbolic Cosine:', cmath.acosh(b))
F)print('Inverse Hyperbolic Tangent:', cmath.atanh(b))
```

Q3 Exponential and Logarithmic Functions.

```
d = 3+7j
A)print('e^d =', cmath.exp(d))
B)print('log2(d) =', cmath.log(d,2))
C)print('log10(d) =', cmath.log10(d))
D)print('sqrt(d) =', cmath.sqrt(d))
```

Signature of the instructor **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**		**4:Complete**			
1:Incomplete		**3:Needs improvement**		**5:Well Done**			

Practical 7: Complex Number in Python

1)

```
import cmath
a = 3 + 4j
print('Sine:', cmath.sin(a))
print('Cosine', cmath.cos(a))
print('Tangent:', cmath.tan(a))
print('ArcSin:', cmath.asin(a))
print('ArcCosine:', cmath.acos(a))
print('ArcTan:', cmath.atan(a))
```

```
Sine: (3.853738037919377-27.016813258003936j)
Cosine (-27.034945603074224-3.8511533348117775j)
Tangent: (-0.00018734620462947 84+0.999355987381473j)
ArcSin: (0.6339838656391766+2.305509031243477j)
ArcCosine: (0.9368124611557198-2.305509031243477j)
ArcTan: (1.4483069952314644+0.15899719167999918j)
```

2)

```
import cmath
b = 2 - 5j
print('Hyperbolic Sine:', cmath.sinh(b))
print('Hyperbolic Cosine:', cmath.cosh(b))
print('Hyperbolic Tangent:', cmath.tanh(b))
print('Inverse Hyperbolic Sine:', cmath.asinh(b))
print('Inverse Hyperbolic Cosine:', cmath.acosh(b))
```

```
print('Inverse Hyperbolic Tangent:', cmath.atanh(b))
```

```
Hyperbolic Sine: (1.0288031496599335+3.6076607742131563j)

Hyperbolic Cosine: (1.0671926518731156+3.477884485899157j)

Hyperbolic Tangent:
(1.0310080051524912+0.0205530165682555j)

Inverse Hyperbolic Sine: (2.37054853731792-1.184231684275022j)

Inverse Hyperbolic Cosine: (2.3830308809003258-
1.1961255219693696j)

Inverse Hyperbolic Tangent: (0.06706599664866984-
1.399284356584545j)
```

3)

```
import cmath

d = 3+7j

print('e^d =', cmath.exp(d))

print('log2(d) =', cmath.log(d,2))

print('log10(d) =', cmath.log10(d))

print('sqrt(d) =', cmath.sqrt(d))
```

```
e^d = (15.142531566086864+13.195928586605717j)

log2(d) = (2.928990497563786+1.682044698743433j)

log10(d) = (0.8817139967814686+0.5063459083693581j)

sqrt(d) = (2.3038850997677716+1.5191729832155236j)
```

Practical 8: File Handling (Part I)

Q1 Create a file called “Example.txt” with some text in it. Save the file in the same folder as that of the Python file. Try to read text file using Python.
Q2 Write a Python program to read first n lines of a file.
Q3 Write a Python program to append text to a file and display the text.
Q4 Write a Python program to read a file line by line store it into a list.
Q5 Write a Python program to count the number of lines in a text file.
Q6 Write a Python program to get the size of a plain file.

Signature of the instructor **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**		**4:Complete**			
1:Incomplete		**3:Needs improvement**		**5:Well Done**			

Practical 8: File Handling (Part I)

1)

```
file = open("Example.txt", "w")
file.write("This is an example text file.\n")
file.write("Python file handling is easy.")
file.close()
file = open("Example.txt", "r")
content = file.read()
file.close()
print("Contents of the file:")
print(content)
```

```
Contents of the file:
This is an example text file.
Python file handling is easy.
```

2)

```
n = int(input('Enter number of lines to read: '))
file = open('Example.txt', 'r')
for i in range(n):
    line = file.readline()
    if not line:
        break
    print(line, end='')
file.close()
```

```
Enter number of lines to read: 25
This is an example text file.
Python file handling is easy.
```

3)

```
file = open('Example.txt', 'a')
file.write('\nThis line is appended to the file.')
file.close()
file = open('Example.txt', 'r')
content = file.read()
file.close()
print('File contents:')
print(content)
```

```
File contents:
This is an example text file.
Python file handling is easy.
This line is appended to the file.
```

4)

```
file = open('Example.txt', 'r')
lines = []
for line in file:
    lines.append(line.strip())
file.close()
print('Lines stored in list:')
print(lines)
```

```
Lines stored in list:
['This is an example text file.', 'Python file handling is easy.', 'This line is appended to the file.']
```

5)

```
file = open('Example.txt', 'r')
count = 0
for line in file:
    count += 1
file.close()
print('Number of lines in the file =', count)
```

Number of lines in the file = 3

6)

```
import os
file_name = "Example.txt"
size = os.path.getsize(file_name)
print("Size of the file (in bytes) =", size)
```

Size of the file (in bytes) = 96

Practical 9: File Handling (Part II)

Q1 Write a Python program to assess if a file is closed or not.
Q2 Create a text file and write a function in python to count and display the total number of words in a text file.
Q3 Write a Python program for read content from one file and write it into another file.
Q4 Create Python program to write a dictionary to a file in Python.
Q5 Write a Python program to find the most repeated word in a text file.

Signature of the instructor **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**		**4:Complete**			
1:Incomplete		**3:Needs improvement**		**5:Well Done**			

Practical 9: File Handling (Part II)

1)

```
f = open('Example.txt', 'r')
print('File closed?', f.closed)
f.close()
print('File closed after closing?', f.closed)
```

File closed? False

File closed after closing? True

2)

Python is easy to learn

Python is very powerful

```
def count_words(filename):
    f = open(filename, "r")
    text = f.read()
    words = text.split()
    f.close()
    print("Total number of words =", len(words))
count_words("sample.txt")
```

Total number of words = 9

3)

```
f1 = open("source.txt", "r")
f2 = open("destination.txt", "w")
data = f1.read()
f2.write(data)
```

```
f1.close()
f2.close()
print("Content copied successfully")
```

Content copied successfully

4)

```
d = {'a': 10, 'b': 20, 'c': 30}
f = open("dict.txt", "w")
f.write(str(d))
f.close()
print("Dictionary written to file")
```

Dictionary written to file

5)

```
def most_repeated_word(filename):
    f = open(filename, "r")
    text = f.read().lower()
    f.close()
    words = text.split()
    freq = {}
    for w in words:
        freq[w] = freq.get(w, 0) + 1
    max_word = max(freq, key=freq.get)
    print("Most repeated word:", max_word)
    print("Number of times:", freq[max_word])
most_repeated_word("sample.txt")
```

Most repeated word: a

Number of times: 10

Practical 10: NumPy (Part I)

Q1 Write a NumPy program to get the NumPy version and show the NumPy build configuration.
Q2 Write a NumPy program to get help with the add function.
Q3 Write a NumPy program to test if any of the elements of a given array are non-zero.

```
Array = [1 0 0 0]
```

Q4 Write a NumPy program to create an array with the values 1, 7, 13, 105 and determine the size of the memory occupied by the array.
Q5 Write a NumPy program to create an array of 10 zeros, 10 ones, and 10 fives.
Q6 Write a NumPy program to create an array of all even integers from 30 to 70.
Q7 Write a NumPy program to create a 3x3 identity matrix.
Q8 Write a NumPy program to create a vector of length 10 with values evenly distributed between 5 and 50.
Q9 Write a NumPy program to multiply the values of two given vectors.
Q10 Write a NumPy program to create a 10x10 matrix, in which the elements on the borders will be equal to 1, and inside 0.

Signature of the instructor **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**		**4:Complete**	
1:Incomplete		**3:Needs improvement**		**5:Well Done**	

Practical 10: NumPy (Part I)

1)

```
from numpy import *
print("NumPy Version:")
print(__version__)
print("\nNumPy Build Configuration:")
show_config()
```

```
NumPy Version:
2.3.2
NumPy Build Configuration:
```

2)

```
from numpy import *
help(add)
```

3)

```
from numpy import *

a = array([1, 0, 0, 0])

result = any(a)

print('Array:', a)

print('Any non-zero element:', result)
```

```
Array: [1 0 0 0]

Any non-zero element: True
```

4)

```
from numpy import *
a = array([1, 7, 13, 105])
print("Array:", a)
print("Memory size in bytes:", a.nbytes)
```

```
Array: [  1   7  13 105]
Memory size in bytes: 32
```

5)

```
from numpy import *
a = zeros(10)
b = ones(10)
c = full(10, 5)
result = concatenate((a, b, c))
print(result)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 5. 5. 5. 5.
 5. 5. 5. 5. 5. 5.]
```

6)

```
from numpy import *

a = arange(30, 71, 2)

print(a)
```

```
[30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70]
```

7)

```
from numpy import *
I = identity(3)
print(I)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

8)

```
from numpy import *
v = linspace(5, 50, 10)
print(v)
```

```
[ 5. 10. 15. 20. 25. 30. 35. 40. 45. 50.]
```

9)

```
from numpy import *
a = array([1, 2, 3, 4])
b = array([5, 6, 7, 8])
c = a * b
print('Result:', c)
```

```
Result: [ 5 12 21 32]
```

10)

```
from numpy import *
a = zeros((10, 10), dtype=int)
a[0, :] = 1
a[-1, :] = 1
a[:, 0] = 1
a[:, -1] = 1
print(a)
```

```
[[1 1 1 1 1 1 1 1 1 1]
 [1 0 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0 0 1]
 [1 1 1 1 1 1 1 1 1 1]]
```

Practical 11: NumPy (Part II)

Q1 Write a NumPy program to convert a list of numeric values into a one-dimensional NumPy array.

```
List = [12.23,13,100,36.5,39]
```

Q2 Write a NumPy program to convert an array to a floating type.

```
Array = [1 2 3 4]
```

Q3 Write a NumPy program to create a 2D array with 1 on the border and 0 inside.

Q4 Write a NumPy program to append values to the end of an array.

```
Array = [10 20 30]
```

Q5 Write a NumPy program to find common values between two arrays.

```
Array1 = [0 10 20 40 60]
Array2 = [10 30 40]
```

Q6 Write a NumPy program to sort along the first and last axes of an array.

```
Array = [[2, 5],[4, 6]]
```

Q7 Write a NumPy program to save a NumPy array to a text file.

Q8 Write a NumPy program to find the memory size of a NumPy array.

Signature of the instructor **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**		**4:Complete**			
1:Incomplete		**3:Needs improvement**		**5:Well Done**			

Practical 11: NumPy (Part II)

1)

```
from numpy import *
L = [12.23, 13, 100, 36.5, 39]
a = array(L)
print(a)
```

```
[ 12.23  13.   100.    36.5   39.  ]
```

2)

```
from numpy import *
a = array([1, 2, 3, 4])
b = a.astype(float)
print(b)
```

```
[1. 2. 3. 4.]
```

3)

```
from numpy import *
a = zeros((5, 5), dtype=int)
a[0, :] = 1
a[-1, :] = 1
a[:, 0] = 1
a[:, -1] = 1
print(a)
```

```
[[1 1 1 1 1]
 [1 0 0 0 1]
 [1 0 0 0 1]
 [1 0 0 0 1]
 [1 1 1 1 1]]
```

4)

```
from numpy import *
a = array([10, 20, 30])
a = append(a, [40, 50])
print(a)
```

```
[10 20 30 40 50]
```

5)

```
from numpy import *
a = array([0, 10, 20, 40, 60])
b = array([10, 30, 40])
c = intersect1d(a, b)
print("Common values:", c)
```

```
Common values: [10 40]
```

6)

```
from numpy import *
a = array([[2, 5],[4, 6]])
print('Sort along first axis (axis=0):')
print(sort(a, axis=0))
print('\nSort along last axis (axis=1):')
print(sort(a, axis=1))
```

```
Sort along first axis (axis=0):
[[2 5]
 [4 6]]

Sort along last axis (axis=1):
[[2 5]
 [4 6]]
```

7)

```
from numpy import *
a = array([10, 20, 30, 40, 50])
savetxt("array.txt", a)
print("Array saved to file successfully")
```

Array saved to file successfully

8)

```
from numpy import *
a = array([1, 2, 3, 4, 5])
print("Memory size in bytes:", a.nbytes)
```

Memory size in bytes: 40

Practical 12: NumPy (Part III)

Q1 Write a NumPy program to add, subtract, multiply, divide arguments element-wise.

```
Elements = 1.2, 4.6
```

Q2 Write a NumPy program to get the powers of an array values element-wise.

Note: First array elements raised to powers from second array

Q3 Write a NumPy program to calculate the absolute value element-wise.

```
Array = [-10.2 122 -2]
```

Q4 Write a NumPy program to get the floor, ceiling and truncated values of the elements of a numpy array.

```
Array = [-1.6 -1.5 -0.3 0.1 1.4 1.8]
```

Q5 Write a NumPy program to sort a given array of shape 2 along the first axis, last axis and on flattened array.

Original array: `[[10 40][30 20]]`

Q6 Write a Python program to find the maximum and minimum value of a given flattened array.

Flattened array: `[[0 1][2 3]]`

Q7 Write a NumPy program to compute the mean, standard deviation, and variance of a given array along the second axis.

```
Array = [0 1 2 3 4 5]
```

Signature of the instructor | **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**		**4:Complete**	
1:Incomplete		**3:Needs improvement**		**5:Well Done**	

Practical 12: NumPy (Part III)

1)

```
from numpy import *
a = 1.2
b = 4.6
A=add(a, b)
S=subtract(a, b)
M=multiply(a, b)
D=divide(a, b)
print('Addition=',A)
print('Subtraction=',S)
print('Multiplication=',M)
print('Division=',D)
```

```
Addition= 5.8
Subtraction= -3.3999999999999995
Multiplication= 5.52
Division= 0.2608695652173913
```

2)

```
from numpy import *
a = array([2, 3, 4, 5])
b = array([1, 2, 3, 2])
result = power(a, b)
print("First array:", a)
print("Second array:", b)
```

```
print("Result:", result)
```

```
First array: [2 3 4 5]
Second array: [1 2 3 2]
Result: [ 2  9 64 25]
```

3)

```
from numpy import *
a = array([-10.2, 122, -2])
result = absolute(a)
print("Original array:", a)
print("Absolute values:", result)
```

```
Original array: [-10.2 122.   -2. ]
Absolute values: [ 10.2 122.    2. ]
```

4)

```
from numpy import *
a = array([-1.6, -1.5, -0.3, 0.1, 1.4, 1.8])
print("Original array:", a)
print("Floor values:", floor(a))
print("Ceiling values:", ceil(a))
print("Truncated values:", trunc(a))
```

```
Original array: [-1.6 -1.5 -0.3  0.1  1.4  1.8]
Floor values: [-2. -2. -1.  0.  1.  1.]
Ceiling values: [-1. -1. -0.  1.  2.  2.]
Truncated values: [-1. -1. -0.  0.  1.  1.]
```

5)

```
from numpy import *
a = array([[10, 40],[30, 20]])
print("Original array:")
print(a)
print("\nSort along first axis:")
print(sort(a, axis=0))
print("\nSort along last axis:")
print(sort(a, axis=1))
print("\nSort flattened array:")
print(sort(a, axis=None))
```

```
Original array:
[[10 40]
 [30 20]]
Sort along first axis:
[[10 20]
 [30 40]]
Sort along last axis:
[[10 40]
 [20 30]]
Sort flattened array:
[10 20 30 40]
```

6)

```
from numpy import *
a = array([[0, 1],[2, 3]])
f = a.flatten()
print("Flattened array:", f)
print("Minimum value:", f.min())
print("Maximum value:", f.max())
```

```
Flattened array: [0 1 2 3]
Minimum value: 0
Maximum value: 3
```

7)

```
from numpy import *
a = array([0, 1, 2, 3, 4, 5])
a = a.reshape(2, 3)
print("Array:")
print(a)
print("\nMean along second axis:", mean(a, axis=1))
print("Standard deviation along second axis:", std(a, axis=1))
print("Variance along second axis:", var(a, axis=1))
```

```
Array:
[[0 1 2]
 [3 4 5]]
Mean along second axis: [1. 4.]
Standard deviation along second axis: [0.81649658 0.81649658]
Variance along second axis: [0.66666667 0.66666667]
```

Practical 13: Matrix Algebra in Python (Part I)

Q1 Create a row vector `u = [1,2,3,4]`

Q2 Create a column vector `v =` $\begin{bmatrix}2\\3\end{bmatrix}$

Q3 Construct the following matrices.

A)Identity matrix of order 4

B)Zero matrix of order 3 × 2

C)Diagonal matrix of order 3 × 3

Q4 For the following matrices

$$A = \begin{pmatrix}2 & 4 & 2\\1 & 1 & -1\\1 & 2 & 4\end{pmatrix} \qquad B = \begin{pmatrix}1 & 5 & 3\\3 & -5 & 7\\1 & 3 & -1\end{pmatrix}$$

a) `A+B`

b) `A-B`

c) `A*B`

d) Find inverse of A if it's exist.

e) Find B transpose.

Q5 For Matrix

$$C = \begin{pmatrix}10 & 2 & 3\\12 & 7 & 15\\-15 & 10 & -11\end{pmatrix}$$

a) Find its transpose.

b) Find its Determinant

c) Find its inverse if exists.

Signature of the instructor **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**		**4:Complete**	
1:Incomplete		**3:Needs improvement**		**5:Well Done**	

Practical 13: Matrix Algebra in Python (Part I)

1)

```
from numpy import *
u = array([1, 2, 3, 4])
print(u)
```

```
[1 2 3 4]
```

2)

```
from numpy import *
v = array([[2],[3]])
print(v)
```

```
[[2]
 [3]]
```

3)

```
from numpy import *
A = eye(4)
print("Identity Matrix (4x4):")
print(A)
B = zeros((3, 2))
print("\nZero Matrix (3x2):")
print(B)
C = diag([1, 2, 3])
print("\nDiagonal Matrix (3x3):")
print(C)
```

Identity Matrix (4x4):

[[1. 0. 0. 0.]

 [0. 1. 0. 0.]

 [0. 0. 1. 0.]

 [0. 0. 0. 1.]]

Zero Matrix (3x2):

[[0. 0.]

 [0. 0.]

 [0. 0.]]

Diagonal Matrix (3x3):

[[1 0 0]

 [0 2 0]

 [0 0 3]]

4)

```
from numpy import *
A = array([[2, 4, 2],[1, 1, -1],[1, 2, 4]])
B = array([[1, 5, 3],[3, -5, 7],[1, 3, -1]])
print("A + B =")
print(A + B)
print("\nA - B =")
print(A - B)
print("\nA * B =")
print(dot(A, B))
print("\nInverse of A =")
```

```
print(linalg.inv(A))
print("\nB Transpose =")
print(B.T)
A + B =
[[ 3  9  5]
 [ 4 -4  6]
 [ 2  5  3]]
A - B =
[[ 1 -1 -1]
 [-2  6 -8]
 [ 0 -1  5]]
A * B =
[[16 -4 32]
 [ 3 -3 11]
 [11  7 13]]
Inverse of A =
[[-1.         2.         1.        ]
 [ 0.83333333 -1.        -0.66666667]
 [-0.16666667  0.         0.33333333]]
B Transpose =
[[ 1  3  1]
 [ 5 -5  3]
 [ 3  7 -1]]
```

5)

```
from numpy import *
C = array([[10, 2, 3],[12, 7, 15],[-15, 10, -11]])
print("Transpose of C =")
print(C.T)
print("\nDeterminant of C =")
print(linalg.det(C))
print("\nInverse of C =")
print(linalg.inv(C))
```

```
Transpose of C =
[[ 10  12 -15]
 [  2   7  10]
 [  3  15 -11]]
Determinant of C =
-1780.9999999999995
Inverse of C =
[[ 0.12745649 -0.02919708 -0.00505334]
 [ 0.05221786  0.03649635  0.06400898]
 [-0.12633352  0.0729927  -0.02582819]]
```

Practical 14: Matrix Algebra in Python (Part II)

Q1 Assuming that $B = \begin{bmatrix} 1 & 2 & -3 \\ 3 & 4 & -1 \end{bmatrix}$, $A = \begin{bmatrix} 2 & -5 & 1 \\ 1 & 4 & 5 \\ 2 & -1 & 6 \end{bmatrix}$, $y = \begin{bmatrix} 2 \\ -4 \\ 1 \end{bmatrix}$, $z = \begin{bmatrix} -15 \\ -8 \\ -22 \end{bmatrix}$

Calculate each of the following:

A) BA

B) AB^T

C) Ay

D) $y^T z$

E) Yz^T

Q2 Given that $A = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$, $C = \begin{bmatrix} -2 & 0 \\ 0 & 2 \end{bmatrix}$ and $b = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$

A)Find A^{-1}.

B) Ab

C) AC

D) Cb

E) A + C

F) C^{-1}

Signature of the instructor **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**		**4:Complete**	
1:Incomplete		**3:Needs improvement**		**5:Well Done**	

Practical 14: Matrix Algebra in Python (Part II)

1)

```
from numpy import *
B = array([[1, 2, -3],[3, 4, -1]])
A = array([[2, -5, 1],[1, 4, 5],[2, -1, 6]])
y = array([[2],[-4],[1]])
z = array([[-15],[-8],[-22]])
print("BA =")
print(dot(B, A))
print("\nAB^T =")
print(dot(A, B.T))
print("\nAy =")
print(dot(A, y))
print("\ny^T z =")
print(dot(y.T, z))
print("\ny z^T =")
print(dot(y, z.T))
```

```
BA =
[[-2  6 -7]
 [ 8  2 17]]
AB^T =
[[-11 -15]
 [ -6  14]
 [-18  -4]]
```

```
Ay =
[[25]
 [-9]
 [14]]
y^T z =
[[-20]]
y z^T =
[[-30 -16 -44]
 [ 60  32  88]
 [-15  -8 -22]]
```

2)

```
from numpy import *
A = array([[1,2],[3,0]])
C = array([[-2,0],[0,2]])
b = array([[4],[6]])
print("A inverse =")
print(linalg.inv(A))
print("\nAb =")
print(dot(A,b))
print("\nAC =")
print(dot(A,C))
print("\nCb =")
print(dot(C,b))
```

```
print("\nA + C =")
print(A + C)
print("\nC inverse =")
print(linalg.inv(C))
```

```
A inverse =
[[ 0.         0.33333333]
 [ 0.5       -0.16666667]]
Ab =
[[16]
 [12]]
AC =
[[-2  4]
 [-6  0]]
Cb =
[[-8]
 [12]]
A + C =
[[-1  2]
 [ 3  2]]
C inverse =
[[-0.5 -0. ]
 [ 0.   0.5]]
```

Practical 15: NumPy and Matrix

Q1 Write a NumPy program to compute the multiplication of two given matrices.

```
A = [[1, 0], [0, 1]]      B = [[1, 2], [3, 4]]
```

Q2 Write a NumPy program to compute the determinant of an array.

```
A = [[1, 2], [3,4]]
```

Q3 Write a NumPy program to compute the inverse of a given matrix.

```
A = [[1, 2], [3,4]]
```

Q4 Write a NumPy program to compute the sum of the diagonal element of a given array.

```
B = [[1, 2], [3,4]]
```

Q5 Write a NumPy program to compute the cross product of two given vectors.

```
A = [[1, 0], [0, 1]]    B = [[1, 2], [3, 4]]
```

Q6 Write a NumPy program to find a matrix or vector norm.

Signature of the instructor **Date**

Assignment Evaluation

0:Not done		**2:Late Complete**		**4:Complete**			
1:Incomplete		**3:Needs improvement**		**5:Well Done**			

Practical 15: NumPy and Matrix

1)

```
from numpy import *
A = array([[1, 0],[0, 1]])
B = array([[1, 2],[3, 4]])
result = dot(A, B)
print("Multiplication of A and B:")
print(result)
```

```
Multiplication of A and B:
[[1 2]
 [3 4]]
```

2)

```
from numpy import *
A = array([[1, 2],[3, 4]])
det_A = linalg.det(A)
print("Determinant of A:", det_A)
```

```
Determinant of A: -2.0000000000000004
```

3)

```
from numpy import *
A = array([[1, 2],[3, 4]])
A_inv = linalg.inv(A)
print("Inverse of A:")
print(A_inv)
```

```
Inverse of A:
```

[[-2. 1.]

 [1.5 -0.5]]

4)

```
from numpy import *
B = array([[1, 2],[3, 4]])
diag_sum = trace(B)
print("Sum of diagonal elements:", diag_sum)
```

Sum of diagonal elements: 5

5)

```
from numpy import *
A = array([[1, 0],[0, 1]])
B = array([[1, 2],[3, 4]])
cross_prod = cross(A, B)
print("Cross Product:")
print(cross_prod)
```

Cross Product:

[2 -3]

6)

```
from numpy import *
v = array([3, 4])
norm_v = linalg.norm(v)
print("Vector Norm:", norm_v)
```

Vector Norm: 5.0