APPLICATIONS OF PYTHON PROGRAMMING

Skill Oriented Course-II

B. Tech IV semester (AR20)

Course Code: 201SC4L17

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

ADITYA ENGINEERING COLLEGE(A)

(Approved by AICTE, Permanently Affiliated to JNTUK & Accredited by NBA, NAAC with 'A' Grade. Recognized by UGC under the sections 2(f) and 12(B) of the UGC act 1956)

Aditya Nagar, ADB Road, Surampalem – 533 437

Experiment 1: Basic numpy Programs (Part-1)

Date:

Aim: Write a program for performing following operation.

- Python program to demonstrate the creation 0-D, 1-D, 2-D and 3-D arrays and their array characteristics.
- Python program to demonstrate array creation techniques.
- Python program to demonstrate indexing in numpy

Description:

Program:

A. Python program to demonstrate the creation 0-D, 1-D, 2-D and 3-D arrays and their array characteristics.

```
import numpy as np
arr=np.array(37)  # 0-Dimension array
print("Array is of type: ", type(arr))    # Printing type of arr object
print("No. of dimensions: ", arr.ndim) # Printing array dimensions
print("Shape of array: ", arr.shape)      # Printing shape of array
print("Size of array: ", arr.size) # size (total No. of elements) of array
print("Array stores elements of type: ", arr.dtype) # type of elements in
array

arr2 = np.array( [[ 1, 2, 3], [ 4, 2, 5]] )  # Creating 2-D array object
print("Array is of type: ", type(arr2))    # Printing type of arr object
print("No. of dimensions: ", arr2.ndim)    # Printing array dimensions
print("Shape of array: ", arr2.shape)         # Printing shape of array
print("Size of array: ", arr2.size) # size (total No. of elements) of array
print("Array stores elements of type: ", arr2.dtype) # Printing type of
elements in array

arr3=np.array( [[1, 2, 3], [4, 5, 6]],[[7, 8, 9],[6, 7, 5]] ) # 3-D array
print("Array is of type: ", type(arr3))    # Printing type of arr object
print("No. of dimensions: ", arr3.ndim)    # Printing array dimensions
print("Shape of array: ", arr3.shape)         # Printing shape of array
print("Size of array: ", arr3.size) # Printing size (total number of
elements) of array
print("Array stores elements of type: ", arr3.dtype)   # Printing type of
elements in array
```

output:

B. Python program to demonstrate array creation techniques

```
# Creating a 3X4 array with all zeros
c = np.zeros((3, 4))
print ("\nAn array initialized with all zeros:\n", c)
d=np.ones((3,4))
print ("\nAn array initialized with all ones:\n", d)
# Create an array with random values
e = np.random.random((2, 2))
print ("\nA random array:\n", e)
# Create a sequence of integers from 0 to 30 with steps of 5
 f = np.arange(0, 30, 5)
print ("\nA sequential array with steps of 5:\n", f)
# Create a sequence of 10 values in range 0 to 5
g = np.linspace(0, 5, 10)
print ("\nA sequential array with 10 values between 0 and 5:\n", g)
# create a matrix of with the same elements
h=np.full((3,3),8)
print ("\nA array with all valuesbeing 8:\n", h)
```

output:

C. *Python program to demonstrate indexing in numpy*

```
import numpy as np
# An example array
arr = np.array([[-1, 2, 0, 4], [4, -0.5, 6, 0], [2.6, 0, 7, 8], [3, -7, 4,
2.0]])
# Slicing array
temp = arr[:2, ::2]
print ("Array with first 2 rows and alternate columns(0 and 2):\n", temp)
# Integer array indexing example
temp = arr[[0, 1, 2, 3], [3, 2, 1, 0]]
print ("\nElements at indices (0, 3), (1, 2), (2, 1),(3, 0):\n", temp)
# boolean array indexing example
cond = arr > 0 # cond is a boolean array
temp = arr[cond]
print ("\nElements greater than 0:\n", temp)
```

Output:

Experiment 2 : Basic NumPy Programs (Part-2)

Date:

Aim: Write a program for performing following operation.

- Python program to demonstrate arithmetic operations on arrays
- Python program to demonstrate universal functions in numpy

Description:

Program:

A. Python program to demonstrate arithmetic operations on arrays

```
import numpy as np
# operations on single array
a = np.array([1, 2, 5, 3])
print ("Adding 1 to every element:", a+1)     # add 1 to every element
print ("Subtracting 3 from each element:", a-3) # subtract 3 from each
element
print ("Multiplying each element by 10:", a*10) # multiply each element by 10
print ("Squaring each element:", a**2) # square each element
a *= 2  # modify existing array
print ("Doubled each element of original array:", a)
# Arithmetic operations
arr1 = np.array([[1., 2., 3.], [4., 5., 6.]])
arr2=np.array([[4., 5., 7.], [3., 6., 9.]])
# Addition of arrays
s_arr= arr1+arr2
s_arr1=np.add(arr1,arr2)
print ("Addition of two arrays:", s_arr)
print ("Addition of two arrays using add function:", s_arr1)
# subtraction of arrays
su_arr=arr1-arr2
su_arr1=np.subtract(arr1,arr2)
print ("Subtraction of two arrays:", su_arr)
print ("Subtraction of two arrays using subtract function:", su_arr)
# Element-wise multiplication
m_arr=arr1*arr2
print ("Element-wise multiplication of two arrays:", m_arr)
# Matrix multiplication
```

```
m_arr1=np.dot(arr1, arr2)
m_arr2=np.matmul(arr1, arr2)
m_arr3=arr1@arr2
print ("Matrix multiplication using dot:", m_arr1)
print ("Matrix multiplication using matmul:", m_arr2)
print ("Matrix multiplication using @ symbol:", m_arr3)
# Element-wise Division
d_arr=arr1/arr2
d_arr1=np.divide(arr1,arr2)
print ("Division of arrays:", d_arr)
print ("Division of arrays using divide function:", d_arr)
```

Output:

B. Python program to demonstrate universal functions in numpy

```python
# Python code to demonstrate trigonometric function
import numpy as np

# create an array of angles
angles = np.array([0, 30, 45, 60, 90, 180])

# conversion of degree into radians
# using deg2rad function
radians = np.deg2rad(angles)

# sine of angles
print('Sine of angles in the array:')
sine_value = np.sin(radians)
print(np.sin(radians))

# inverse sine of sine values
print('Inverse Sine of sine values:')
print(np.rad2deg(np.arcsin(sine_value)))

# hyperbolic sine of angles
print('Sine hyperbolic of angles in the array:')
sineh_value = np.sinh(radians)
print(np.sinh(radians))

# inverse sine hyperbolic
print('Inverse Sine hyperbolic:')
print(np.arcsinh(sineh_value))

# hypot function demonstration
base = 4
height = 3
print('hypotenuse of right triangle is:')
print(np.hypot(base, height))
```

Output:

```
# other mathematical functions
import numpy as np
# natural logarithm
a=np.array([[1, np.e, np.e**2, 0]])
print("Element-wise logarithm of the array:", np.log(a))
# log base 10
b=np.array([100, 1000, 50, 1e-15])
print("Element-wise logarithm of the array:", np.log10(a))
# log base 2
c=np.array([0, 1, 2, 4, 8, 16,32])
print("Element-wise logarithm of the array:", np.log2(a))
# square root
d=np.array([1, 4, 9, 36, 49, 64, 81, 100])
print("square roots of elements of the array", np.sqrt(d))

# minimum and maximum of the elements in the array
arr = np.array([[1, 5, 6], [4, 7, 2], [3, 1, 9]])
print ("Largest element is:", arr.max())     # maximum element of array
print ("Row-wise maximum elements:",  arr.max(axis = 1))
print ("Column-wise minimum elements:",   arr.min(axis = 0)) # minimum
element of array

# sum of elements in the array
```

```
e=np.array([[7, 8, 9, 12, 13, 15, 17],[18, 19, 20, 11, 13, 19, 21]])
print(e)
print(e.sum())
print(e.sum(axis=0))   # summation along row (axis=0)
print(e.sum(axis=1))   # summation along column (axis=1)
```

Output

```
# Python code demonstrate statistical functions
import numpy as np

# construct a weight array
weight = np.array([50.7, 52.5, 50, 58, 55.63, 73.25, 49.5, 45])

# minimum and maximum
print('Minimum and maximum weight of the students: ')
print(np.amin(weight), np.amax(weight))

# range of weight i.e. max weight-min weight
print('Range of the weight of the students: ')
print(np.ptp(weight))

# mean
print('Mean weight of the students: ')
print(np.mean(weight))

# median
print('Median weight of the students: ')
print(np.median(weight))

# standard deviation
print('Standard deviation of weight of the students: ')
print(np.std(weight))

# variance
print('Variance of weight of the students: ')
print(np.var(weight))

# average
print('Average weight of the students: ')
print(np.average(weight))
```

Output

Experiment 3 : NumPy Array Manipulation

Date:

Aim: Write a program for performing following operation.

- Python program to demonstrate reshaping of the array
- Python program to demonstrate addition and removal of elements in the array
- Python program to demonstrate concatenation of arrays

Description:

Program:

A. Python program to demonstrate reshaping of the array

```
a=np.array([1,2,3,4,5,6])
# reshape an 1-D array into 2-D array
print("reshaping the array into an array of size (3,2)\n",a.reshape(3,2))
print("reshaping the array into an array of size (2,3)\n",a.reshape(2,3))
print("reshaping the array into an array of size (6,1)\n",a.reshape(6,1))
# reshape an array into 3 dimensional matrix
a1=np.arange(0,9)
print("reshaping the array into an array of size (3,2)",a1.reshape(3, 1, 3))
# reshaping the array into a larger with repeated copies of the original
array
a2=np.array([[0,1],[2,3]])
Print(np.resize(a2,(2,3)))
Print(np.resize(a2,(1,4)))
Print(np.resize(a2,(2,4)))
```

output:

B. Python program to demonstrate addition and removal of elements in the array

```
# appending a value at the end of the array using append function
import numpy as np
a = np.array([1, 2, 3])
b=np.append(a,4)
print(“appending a value at the end of the array”, b)
a1 = np.array([[1, 2, 3], [7, 8, 9]])
c=np.append(a1,4)
print(“appending a value to the 2-D array”, c)
# appending two arrays with axis=None
a = np.array([[1, 2, 3], [7, 8, 9]])
b = np.array([[11, 21, 31], [42, 52, 62]])
print("The first array:\n",a)
print("The second array:\n",b)
c = np.append(a,b)
print("The resultant array after appending a & b:", c)
# With axis=0
c1=np.append(a,b,axis=0)
print("The resultant array after appending a & b along row:", c1)
# With axis=1
c2=np.append(a,b,axis=1)
print("The resultant array after appending a & b along column:", c2)
```

output:

```
# inserting a value at particular index in the array
a = np.array([[1, 2], [3, 4], [5, 6]])

print("Axis parameter not passed. The input array is flattened before
insertion.")
print(np.insert(a,3,[11,12]))

print("Axis parameter passed. The values array is broadcast to match input
array.")
print('Broadcast along axis 0:')
print(np.insert(a,1,[11],axis = 0))

print('Broadcast along axis 1:')
print(np.insert(a,1,[11],axis = 1))

# inserting a value at different positions
arr = np.arange(12).reshape(3, 4)
print('\n',arr)
a = np.insert(arr, (2, 4), 9)
print('\n',a)

# inserting and broadcasting at different positions in the array
b = np.insert(arr, (0, 3), 66, axis = 1)
print('\n',b)
```

output:

```
# Deleting the values at particular positions in the array
#Working on 1D
arr = np.array([5,6,7,8,1,2])
print("arr : \n", arr)
print("Shape : ", arr.shape)

# deletion from 1D array
object = 2
a = np.delete(arr, object)
print("\ndeleteing {} from array : \n {}".format(object,a))
print("Shape : ", a.shape)

object = [1, 3]
b = np.delete(arr, object)
#print("\ndeleteing {} from array : \n {}".format(object,a))
#print("Shape : ", a.shape)
print("\n",b)
print("\n",b.shape)

arr = np.arange(12).reshape(3, 4)
print("arr : \n", arr)
print("Shape : ", arr.shape)

# deletion from 2D array
a = np.delete(arr, 1, 0)
print("\ndeleteing arr 2 times : \n", a)
print("Shape : ", a.shape)

# deletion from 2D array
a = np.delete(arr, 1, 1)
print("\ndeleteing arr 2 times : \n", a)
print("Shape : ", a.shape)
```

output:

C. Python program to demonstrate concatenation of arrays

```
# np.concatenate
import numpy as np
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6]])
print(“Concatenate two arrays along the row”, np.concatenate((a, b), axis=0))
print(“Concatenate two arrays along the column”, np.concatenate((a, b.T),
axis=1))
print(“Concatenate the values of two arrays with no axis”,np.concatenate((a,
b), axis=None)

# vstack
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
print(“Stacking arrays in sequence vertically”, np.vstack((a,b)))

#hstack
print(“Stacking arrays in sequence horizontally”, np.hstack((a,b)))
```

output:

Experiment 4 : Application of Python Programming: Linear Algebra on Numpy arrays

Date:

Aim: Write a program for performing following operation.

- Compute the Eigen values of a matrix
- Compute the multiplicative inverse of a matrix
- Solve a linear matrix equation such as $3x^0 + x^1 = 9,\ x^0 + 2x^1 = 8$
- Find the roots of the polynomial expression $3.2x^2 + 2x + 1$
- Compute the rank of a matrix
- Compute the determinant of an array

Description:

Program:

A. Compute the Eigen values of a matrix

```
#eigen values of a matrix
import numpy as np
a=np.array([[1,2],[3,4]])
eigvalues,eigvectors=np.linalg.eig(a)
print("eigen value:",eigvalues,"eigen vector:",eigvectors)
```

B. Compute the multiplicative inverse of a matrix

```
#multiplicative inverse
import numpy as np
a=np.array([[3,1],[1,2]])
a_inv=np.linalg.inv(a)
print("a inverse:",a_inv)
```

C. Solve a linear matrix equation such as $3x_0 + x_1 = 9,\ x_0 + 2x_1 = 8$

```
#linear matric equation
Import numpy as np
a=np.array([[3,1],[1,2]])
b=np.array([[9],[8]])
a_inv=np.linalg.inv(a)
e=np.matmul(a_inv,b)
print(“linear equation:”,e)
```

```
# method 2
e=np.linalg.solve(a,b)
print(“solution”, e)
```

Task: The admission fee at a small fair is $1.50 for children and $4.00 for adults. On a certain day, 2200 people enter the fair and $5050 is collected. How many children and how many adults attended?

```
# roots of the polynomial expression
p = [3.2, 2, 1]
g = np.roots(p)
print (“roots of the polynomial expression 3.2*x**2 +2*x+1 are”, g)
```

D. Compute the rank of a matrix

```
#matrix rank
a=np.array([[3,1],[1,2]])
b=np.linalg.matrix_rank(a)
print(“rank:”,b)
```

E. Compute the determinant of an array

```
a=np.array([[3,1],[1,2]])
b=np.linalg.det(a)
print(“determinant:”,b)
```

Experiment 5 : Generation of Basic Signals

Date:

Aim: Write a program for performing following operation.

- Python program to generate basic signals and plot all the signals using matplotlib module

Description:

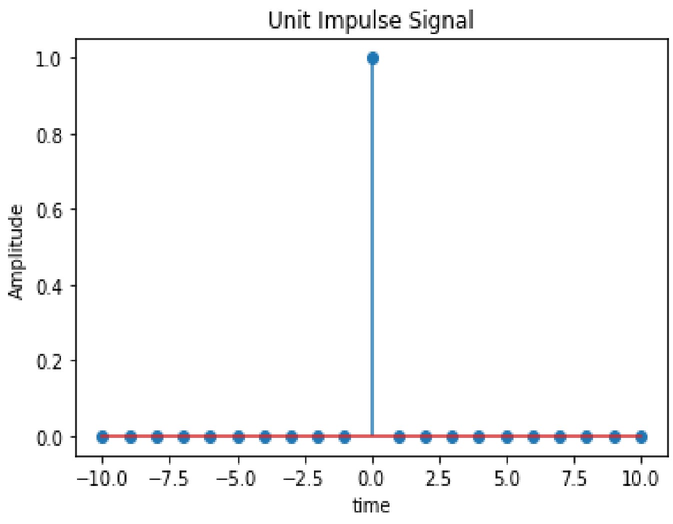
Program:

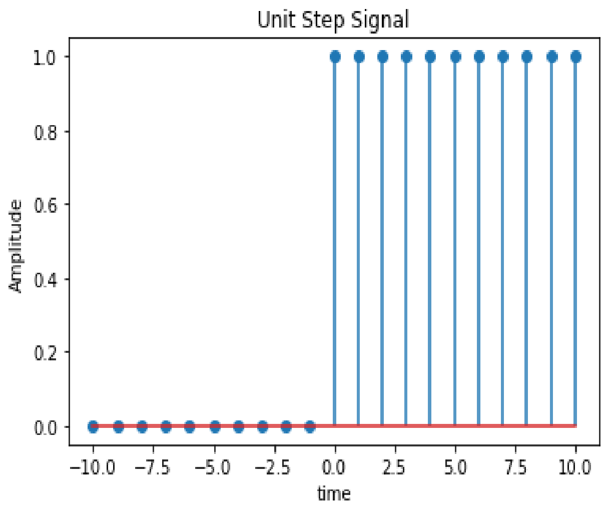
A. Python program to generate basic signals and plot all the signals using matplotlib module

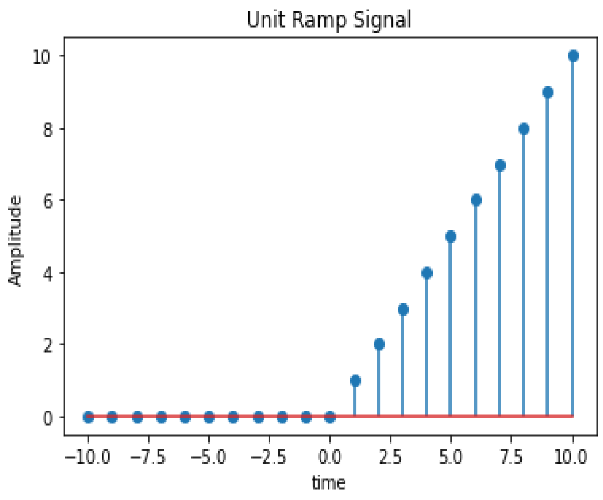
```
import numpy as np
import matplotlib.pyplot as plt

# Impulse Signal
t=np.arange(-10,11)
plt.close('all')
x=np.zeros(len(t))
x[t==0]=1
plt.stem(t,x)
plt.xlabel('time')
plt.ylabel('Amplitude')
plt.title('Unit Impulse Signal')

# Step Signal
t=np.arange(-10,11)
plt.close('all')
x=np.zeros(len(t))
x[t>=0]=1
plt.stem(t,x)
plt.xlabel('time')
plt.ylabel('Amplitude')
plt.title('Unit Step Signal')

#Ramp Signal
t=np.arange(-10,11)
plt.close('all')
x=np.zeros(len(t))
```

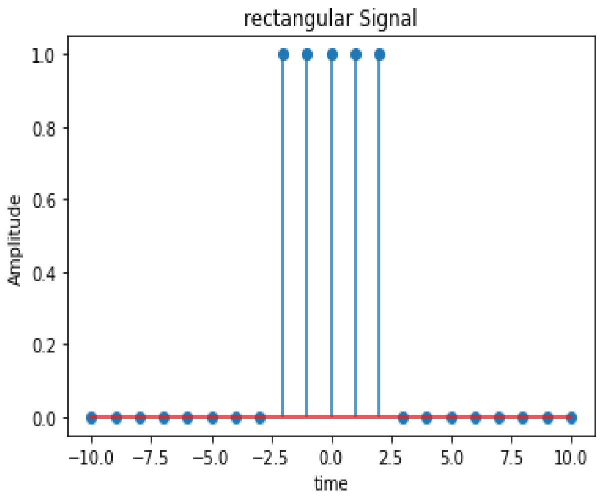
```
for i in t:
  if t[i]>=0:
    x[i]=t[i]
plt.stem(t,x)
plt.xlabel('time')
plt.ylabel('Amplitude')
plt.title('Unit Ramp Signal')

# Rectangular signal
t=np.arange(-10,11)
plt.close('all')
x=np.zeros(len(t))
for i in t:
  if abs(t[i])<=2:
    x[i]=1
plt.stem(t,x)
plt.xlabel('time')
plt.ylabel('Amplitude')
plt.title('rectangular Signal')

# Sinusoidal Signal
t = np.linspace(0, 1, 1000)
A = 2
f0 = 5
x = A*np.cos(2*np.pi*f0*t)
plt.plot(t, x, '-b')
plt.title('$x(t) = 2cos(2\pi*5t)$')
plt.xlabel('t (in s)')
plt.grid()
```

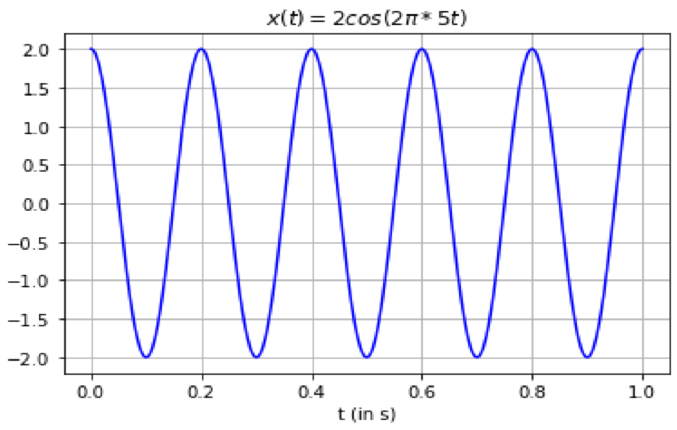
output:


Unit Impulse Signal
Amplitude
time


Unit Step Signal
Amplitude
time


Unit Ramp Signal
Amplitude
time


rectangular Signal
Amplitude
time


$x(t) = 2cos(2\pi * 5t)$
t (in s)

Experiment 6: Shifting and Time Reversal Transformations on Discrete time Signals

Date:

Aim: Write a program for performing shifting and time reversal transformations on discrete time signals. Use NumPy and Matplotlib package in python to plot following signals

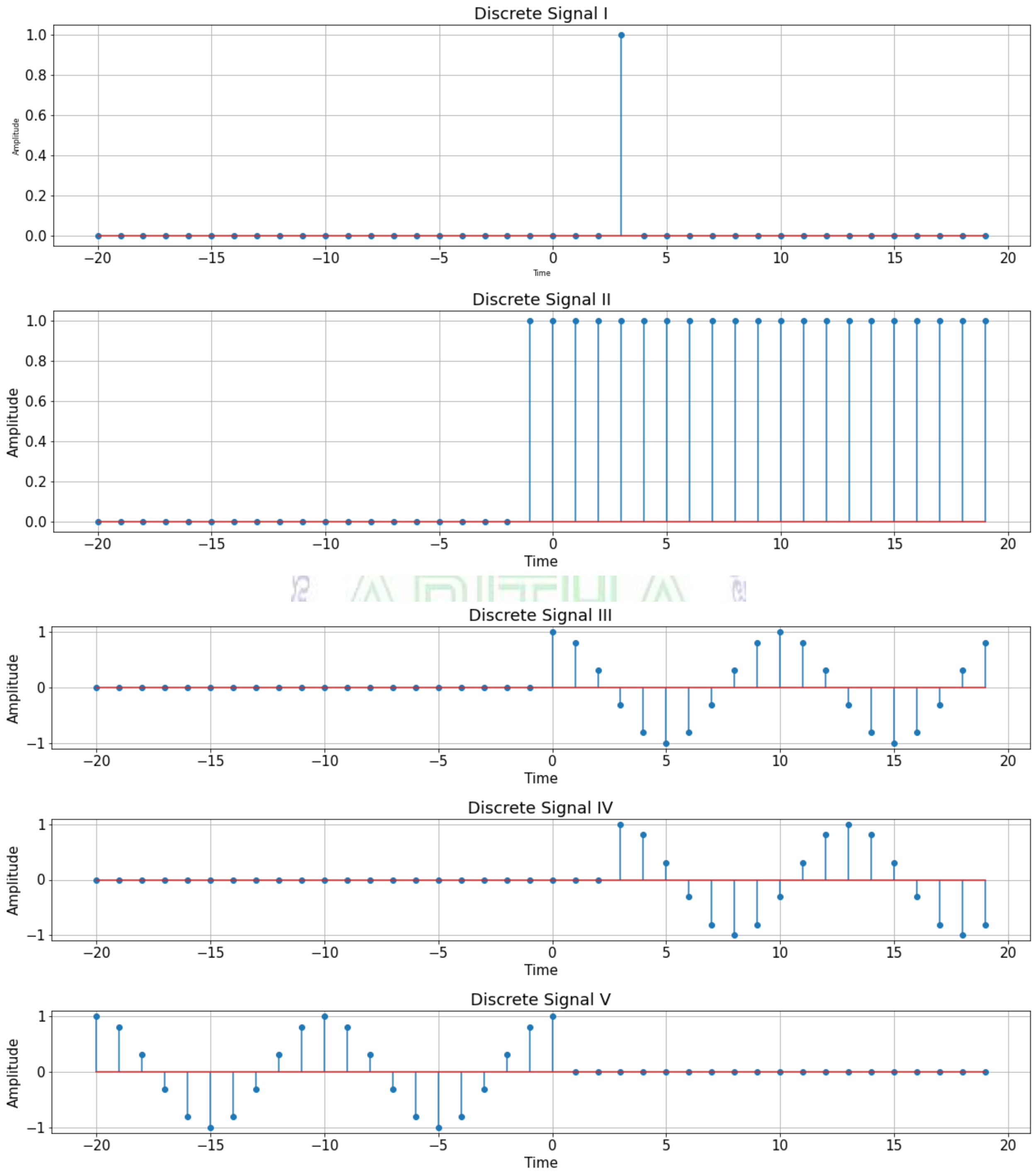
I. $\delta[n-3]$
II. $u[n+1]$
III. $x[n] = \cos\left(\frac{\pi n}{5}\right) u[n]$
IV. $x_1[n] = x[n-3]$
V. $x_2[n] = x[-n]$

Description:

Program:

```
import numpy as np
import matplotlib.pyplot as plt
U=lambda T:1.0*(T>=0)
I=lambda T:1.0*(T==0)

sig1=lambda T:I(T-3)
sig2=lambda T:U(T+1)
sig3=lambda T:np.cos((np.pi/5)*T)*U(T)
sig4=lambda T:sig3(T-3)
sig5=lambda T:sig3(-T)
T=np.arange(-20, 20, 1, dtype=int)

plt.figure(1)
plt.subplot(2, 1, 1)
plt.stem(T, sig1(T))
plt.xlabel('Time', fontsize = 8)
plt.ylabel('Amplitude', fontsize = 8)
plt.title ('Discrete Signal I')
plt.grid(True)
plt.subplot(2, 1, 2)
plt.stem(T, sig2(T))
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title ('Discrete Signal II')
plt.grid(True)
```

```
fig = plt.gcf()
fig.set_size_inches(16, 9)
plt.tight_layout()
plt.show()

plt.figure(2)
plt.subplot(3, 1, 1)
plt.stem(T, sig3(T))
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title ('Discrete Signal III')
plt.grid(True)

plt.subplot(3, 1, 2)
plt.stem(T, sig4(T))
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title ('Discrete Signal IV')
plt.grid(True)

plt.subplot(3, 1, 3)
plt.stem(T, sig5(T))
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title ('Discrete Signal V')
plt.grid(True)

fig = plt.gcf()
fig.set_size_inches(16, 9)
plt.tight_layout()
plt.show()
```


Discrete Signal I
Amplitude
Time
Discrete Signal II
Amplitude
Time
Discrete Signal III
Amplitude
Time
Discrete Signal IV
Amplitude
Time
Discrete Signal V
Amplitude
Time
−20
−15
−10
−5
0
5
10
15
20
0.0
0.2
0.4
0.6
0.8
1.0
−1
0
1

Experiment 7 : Application of Python in Communication Engineering

Date:

Aim: Write a program for performing following operation.

- Generate message signal
- Generate Carrier signal
- Perform Amplitude modulation
- Display the results in time and Frequency domain

Description:

#Carrier wave c(t)=A_c*cos(2*pi*f_c*t)

#Modulating wave m(t)=A_m*cos(2*pi*f_m*t)

#Modulated wave s(t)=A_c[1+mu*cos(2*pi*f_m*t)]cos(2*pi*f_c*t)

Program:

```
import numpy as np
import matplotlib.pyplot as plt
Variable1 = 5
Variable2 = 7
A_c = 2 + .25* Variable1
f_c = 50 + .5* Variable1
A_m = 2 + .25* Variable2
f_m = 10 + .25* Variable2
modulation_index = .5

# Number of sample points
N = 600

# Sample spacing
T = 1.0 / 800.0
t = np.linspace(0.0, N*T, N)

carrier = A_c*np.cos(2*np.pi*f_c*t)        # Carrier Signal
message = A_m*np.cos(2*np.pi*f_m*t)  # Message Signal

# Performing Modulation
modulated =
A_c*(1+modulation_index*np.cos(2*np.pi*f_m*t))*np.cos(2*np.pi*f_c*t)
```

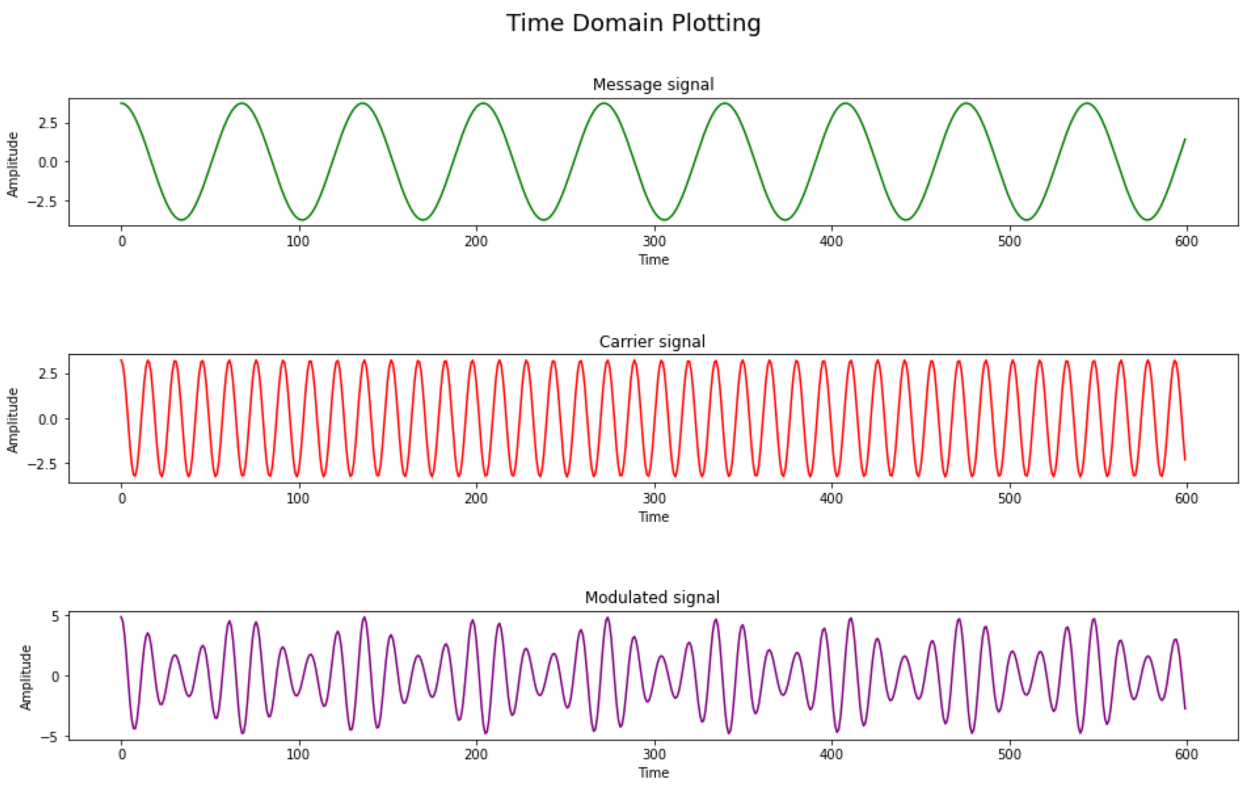
```
# Plotting the result in time domain
plt.figure()
plt.subplot(3,1,1)
plt.plot(message,'g')
plt.ylabel('Amplitude')
plt.xlabel('Time')
plt.title('Message signal')


plt.subplot(3,1,2)
plt.plot(carrier, 'r')
plt.ylabel('Amplitude')
plt.xlabel('Time')
plt.title('Carrier signal')


plt.subplot(3,1,3)
plt.plot(modulated, color="purple")
plt.ylabel('Amplitude')
plt.xlabel('Time')
plt.title('Modulated signal')


plt.subplots_adjust(hspace=1)
plt.rc('font', size=15)
fig = plt.gcf()
fig.set_size_inches(16, 9)

plt.suptitle('Time Domain Plotting')

fig.savefig('Time_RollNo.png', dpi=100)

carrier_Freq = np.fft.fft(carrier)
message_Freq  = np.fft.fft(message)
modulated_Freq  = np.fft.fft(modulated)
freq  = np.linspace(0.0, 1.0/(2.0*T), int(N/2))

plt.figure(2)
plt.subplot(3,1,1)
plt.title('Message signal')
plt.plot(freq, 2.0/N * np.abs(message_Freq[0:int(N/2)]))
plt.xlim([0,100])
plt.grid()
plt.ylabel('Amplitude')
plt.xlabel('Frequency')


plt.subplot(3,1,2)
plt.plot(freq, 2.0/N * np.abs(carrier_Freq[0:int(N/2)]))
plt.xlim([0,100])
plt.grid()
plt.title('Carrier signal')
plt.ylabel('Amplitude')
plt.xlabel('Frequency')

plt.subplot(3,1,3)
plt.plot(freq, 2.0/N * np.abs(modulated_Freq[0:int(N/2)]))
plt.xlim([0,100])
plt.grid()
plt.title('Modulated signal')
plt.ylabel('Amplitude')
plt.xlabel('Frequency')

plt.suptitle('Frequency Domain Plotting')

plt.subplots_adjust(hspace=1)
plt.rc('font', size=15)
fig = plt.gcf()
fig.set_size_inches(16, 9)
plt.show()

fig.savefig('Spectrum_Rollnumber.png', dpi=100)
```

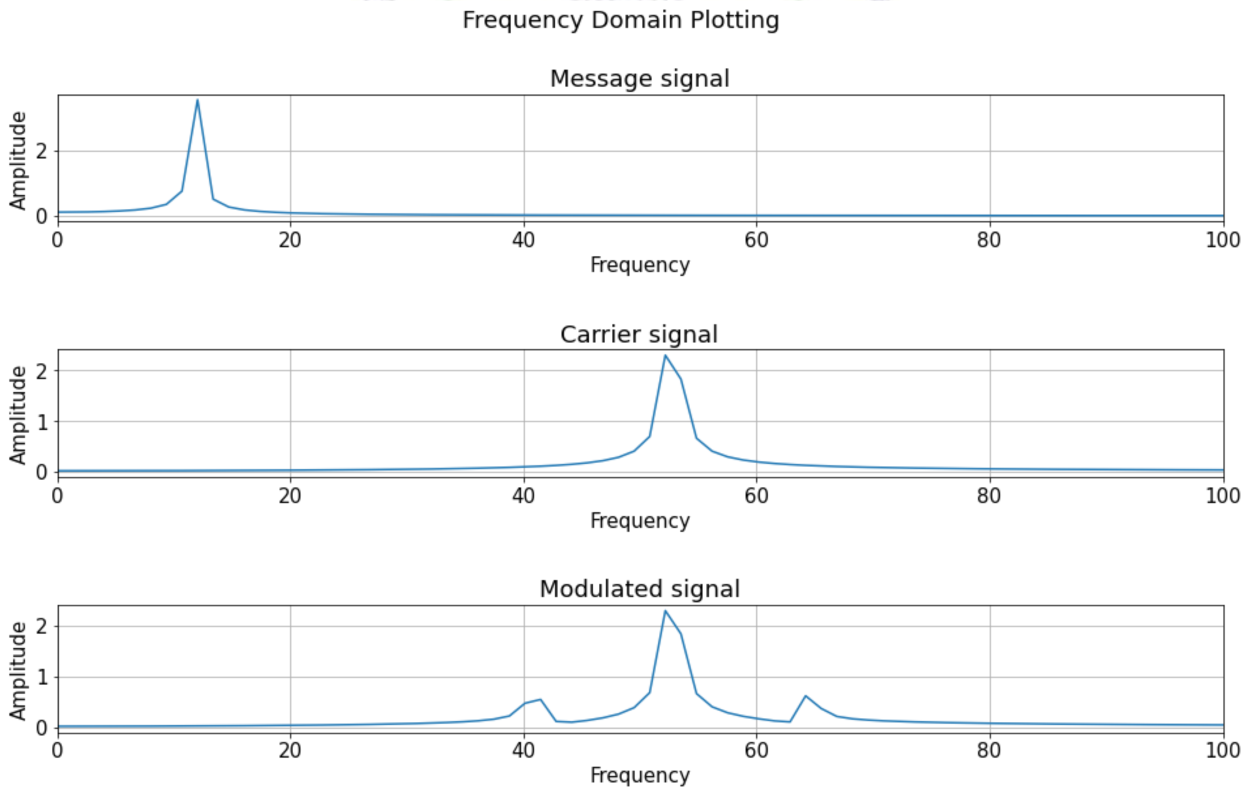
Variable - 1

Last Digit 2nd-Last Digit/Letter	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0 (Zero)	8	9	9	4	2	2	2	3	9	4
1 (One)	9	6	1	6	3	6	8	2	9	8
2 (Two)	2	1	4	3	6	5	5	2	5	1
A	9	8	4	7	5	1	9	8	5	1
B	6	9	7	3	4	4	1	6	4	2
C	1	7	8	5	8	2	4	5	9	6
D	3	7	2	7	6	8	1	2	4	7
E	5	7	5	9	5	3	9	8	2	6
F	9	4	5	9	9	5	1	6	8	5
G	9	6	6	5	3	2	7	4	4	5
H	2	2	7	2	7	6	8	5	3	3
I	9	7	7	2	7	3	8	4	4	7
J	9	1	3	3	4	6	1	1	1	2
K	5	3	7	8	6	7	4	3	2	7
L	8	1	6	3	1	7	3	2	9	2
M	2	1	2	8	1	5	8	2	9	4
N	4	8	2	3	5	1	4	3	6	6
O	9	7	5	9	8	3	9	4	1	8
P	8	3	9	4	9	9	2	1	3	1

Variable - 2

Last Digit 2nd-Last digit/Letter	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0 (Zero)	9	6	3	6	4	6	5	8	1	6
1 (One)	7	2	4	3	7	6	1	1	7	8
2 (Two)	5	3	6	5	2	8	8	4	3	7
A	4	5	3	9	7	8	8	5	4	2
B	5	3	6	5	1	6	7	4	5	4
C	3	8	7	5	6	2	2	6	9	5
D	5	2	2	3	5	3	6	6	4	9
E	5	3	2	5	8	8	5	3	9	2
F	8	2	3	6	7	1	9	4	3	8
G	8	3	3	7	9	5	6	1	7	6
H	6	4	4	4	9	2	8	9	6	4
I	4	3	5	4	4	9	5	2	5	2
J	8	9	1	9	7	7	4	1	7	4
K	5	4	3	1	2	5	8	4	6	5
L	4	2	8	8	1	5	1	2	2	2
M	9	9	1	9	7	1	2	5	2	6
N	8	9	9	8	5	7	2	4	9	3
O	5	4	7	1	5	1	4	9	2	4
P	6	2	5	3	9	1	8	9	1	6


Time Domain Plotting
Message signal
Carrier signal
Modulated signal
Amplitude
Time


Frequency Domain Plotting
Message signal
Carrier signal
Modulated signal
Amplitude
Frequency

Experiment 8 : Pandas Dataframes: Understanding Data

Date:

Aim: Write a program for performing following operation.

- Loading data from CSV, excel and text files
- Compute the basic statistics of given data – index, no. of columns, size, shape, memory usage, no. of dimensions.
- Perform indexing and Selecting data from a pandas dataframe.
- Check the datatype of the data and convert variable datatype into necessary format.
- Visualize the data using scatterplot, histogram and barplot

Dataset: ‘Iris_data_sample.csv’, ‘Iris_data_sample.xlsx’, ‘Iris_data_sample.txt’, ‘Toyota.csv’

Description:

Program:

A. Loading data from CSV, excel and text files

```
# importing necessary libraries
import os
import pandas as pd
# changing the current working directory
os.chdir(r"D:\aditya\evensem 22-23\APP\pandas") # specify the path where data
files are stored

# importing data from csv files
data_csv=pd.read_csv('Iris_data_sample.csv')
# blank cells are read as ‘nan’
# Removing the extra id column by passing index_col=0
data_csv=pd.read_csv('Iris_data_sample.csv',index_col=0)
# Replacing ‘??’ and ‘# # #’ as missing values
data_csv=pd.read_csv('Iris_data_sample.csv',index_col=0,
na_values=["??","###"])

# importing data from excel files
data_xlsx= pd.read_excel('Iris_data_sample.xlsx', sheet_name='Iris_data')
# blank cells are read as ‘nan’
# Removing the extra id column by passing index_col=0
data_xlsx=pd.read_excel('Iris_data_sample.xlsx',sheet_name='Iris_data',index_
col=0)
```

```
# Replacing ‘??’ and ‘# # #’ as missing values
data_csv=pd.read_excel('Iris_data_sample.xlsx',sheet_name='Iris_data',index_c
ol=0, na_values=["??","###"])

# importing data from text files
data_txt=pd.read_table(“Iris_data_sample.txt”)
# All columns read and stored in a single column of dataframe
# In order to avoid this, provide a delimiter to the parameters ‘sep’ or
‘delimiter’
data_txt=pd.read_table(“Iris_data_sample.txt”, sep= ‘\t’)
data_txt1=pd.read_table(“Iris_data_sample.txt”, delimiter= ‘\t’)
# Tab delimiter might not always work
#  Other commonly used delimiters are commas and blanks
data_txt1=pd.read_table(“Iris_data_sample.txt”, delimiter= “ ”)
# blank cells are read as ‘nan’
# Removing the extra id column by passing index_col=0
data_txt1=pd.read_table(“Iris_data_sample.txt”, delimiter= “ ”, index_col=0)
# Replacing ‘??’ and ‘# # #’ as missing values
data_txt1=pd.read_table(“Iris_data_sample.txt”, delimiter= “ ”, index_col=0,
na_values=["??","###"])
```

Output:

B. Compute the basic statistics of given data – index, no. of columns, size, shape, memory usage, no. of dimensions.

```
# importing necessary libraries
import os
import pandas as pd
# changing the current working directory
os.chdir(r"D:\aditya\evensem 22-23\APP\pandas") # specify the path where data
files are stored
importing data from csv file
cars_data=pd.read_csv('Toyota.csv',index_col=0)

# copying the data into new dataframe
cars_data1=cars_data.copy(deep=True)

# To get the index (row labels) of the dataframe
```

```
print(“row labels of the dataframe are”, cars_data1.index)

# To get the column labels of the dataframe
print(“row labels of the dataframe are”, cars_data1.columns)

#no of columns
cols=len(cars_data1.axes[1])
print('no of columns:', cols)

# To get the total number of elements from the dataframe
print(' total number of elements from the dataframe:', cars_data1.size)

# The memory usage of each column in bytes in the dataframe
print(' memory usage of each column in bytes:', cars_data1.memory_usage())

# The number of axes / array dimensions in the dataframe
print(' number of array dimensions:', cars_data1.ndim)
```

Output:

C. Perform indexing and Selecting data from a pandas dataframe.

```
# importing necessary libraries
import os
import pandas as pd
# changing the current working directory
os.chdir(r"D:\aditya\evensem 22-23\APP\pandas") # specify the path where data
files are stored
importing data from csv file
cars_data=pd.read_csv('Toyota.csv',index_col=0)

# The first n rows from the dataframe
print(' first 6 rows from the dataframe:', cars_data1.head(6))

# The last n rows from the dataframe
print(' first 6 rows from the dataframe:', cars_data1.tail(5))

# To access a scalar value, the fastest way is to use the at and iat methods
# at provides label based scalar lookups
```

```
print('access 4th row from FuelType column:', cars_data1.at[4, ‘FuelType’])

 # iat provides integer based lookups
print('access 5th row from 6th column:', cars_data1.iat[5, 6])

# To access a group of rows and columns by label(s) .loc[] can be used
print('access all elements from FuelType column:', cars_data1.loc[:,
‘FuelType’])
```

Output:

D. Check the datatype of the data and convert variable datatype into necessary format.

```
# importing necessary libraries
import os
import pandas as pd
# changing the current working directory
os.chdir(r"D:\aditya\evensem 22-23\APP\pandas") # specify the path where data
files are stored
importing data from csv file
cars_data=pd.read_csv('Toyota.csv',index_col=0)

# dtypes returns a series with the data type of each column
print('data type of each column:', cars_data1.dtypes)

# selecting data based on data types
print('selecting the data that excludes object data type:',
cars_data1.select_dtypes(exclude=[object]))

# Concise summary of a dataframe
print(' returns a concise summary of a dataframe', cars_data1.info())

# converting datatypes from one to another
print(‘Datatype of Metcolor variable’, cars_data1[‘Metcolor’].dtypes)
print(‘Datatype of Automatic variable’, cars_data1[‘Automatic’].dtypes)
Cars_data1[‘MetColor’]= Cars_data1[‘MetColor’].astype(‘object’)
Cars_data1[‘Automatic’]= Cars_data1[‘Automatic’].astype(‘object’)
print(' datatype of MetColor variable after conversion',
cars_data1[‘MetColor’].dtypes)
```

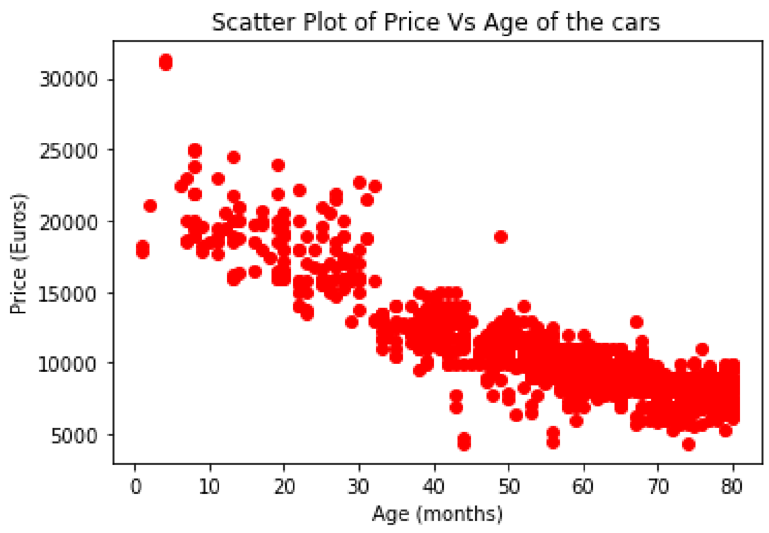
```
print(' datatype of Automatic variable after conversion ',
cars_data1[‘Automatic’].dtypes)
```

Output:

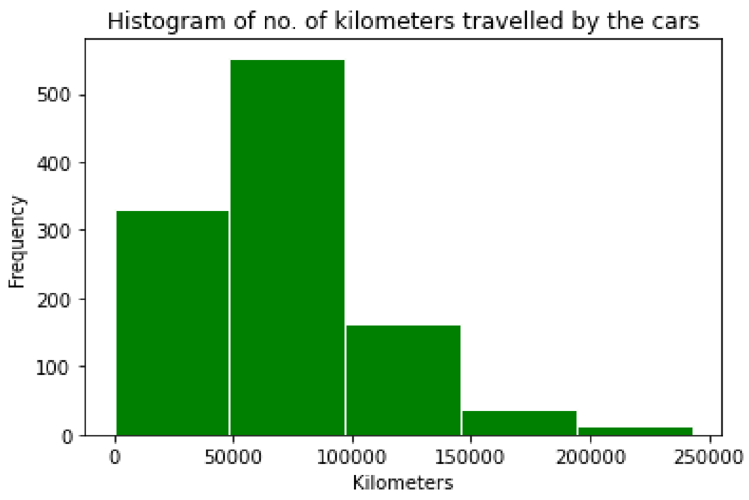
E. Visualize the data using scatter plot, histogram and bar plot

```
# importing necessary libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# changing the current working directory
os.chdir(r"D:\aditya\even sem 22-23\APP\pandas") # specify the path where
data files are stored
#importing data from csv file
cars_data=pd.read_csv('Toyota.csv',index_col=0, na_values=["??","????"])

# Removing missing values from the dataframe
cars_data.dropna(axis=0, inplace=True)

# Scatter plot
plt.scatter(cars_data['Age'], cars_data['Price'], c='red')
plt.title('Scatter Plot of Price Vs Age of the cars')
plt.xlabel('Age (months)')
plt.ylabel('Price (Euros)')
plt.show()

# Histogram
f1=plt.figure()
f2=plt.figure()
```

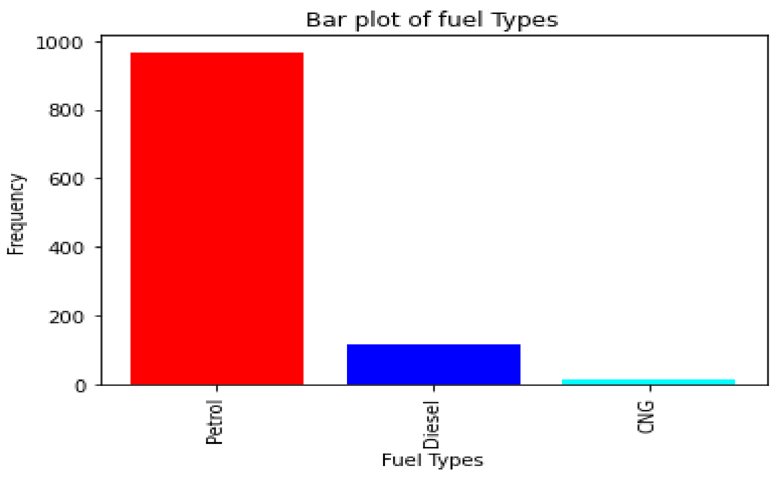
```
ax1 = f1.add_subplot(111)
ax2 = f2.add_subplot(111)
ax1.hist(cars_data['KM'])
plt.title('Histogram of no. of kilometers travelled by the cars')
plt.xlabel('Kilometers')
plt.ylabel('Frequency')


ax2.hist(cars_data['KM'], color='green', edgecolor='white',bins=5)
plt.title('Histogram of no. of kilometers travelled by the cars')
plt.xlabel('Kilometers')
plt.ylabel('Frequency')
plt.show()

# barplots
counts=np.array([cars_data['FuelType'].value_counts()])
counts=counts.flatten()
fuelType=('Petrol','Diesel', 'CNG')
index=np.arange(len(fuelType))

plt.bar(index, counts, color=['red','blue','cyan'])
plt.title('Bar plot of fuel Types')
plt.xlabel('Fuel Types')
plt.ylabel('Frequency')
plt.xticks(index,fuelType,rotation=90)
plt.show()
```



Analysis: The price of the car decreases as age of the car increases



Analysis: Frequency distribution of kilometre of the cars shows that most of the cars have travelled between 50000 100000 km and there are only few cars with more distance travelled.



Analysis: Bar plots of Fuel types of the cars shows that most of the cars are having fuel type as Petrol and very few number of cars have fuel type as CNG.

Experiment 8 : Pandas Dataframes: Data Cleaning and Handling Missing Data

Date:

Aim: Write a program for performing following operation.

- Perform data cleaning in the dataframe
- Remove missing rows/columns
- Fill the missing data in the dataframe using mean, median and mode approaches.

Dataset: ‘Toyota.csv’

Description:

Program:

```
# importing necessary libraries
import os
import pandas as pd
import numpy as np
# changing the current working directory
os.chdir(r"D:\aditya\even sem 22-23\APP\pandas") # specify the path where
data files are stored
#importing data from csv file
cars_data1=pd.read_csv('Toyota.csv',index_col=0,na_values=["??","###","????"]
)

# cleaning Doors column

# checking the unique values of variable Doors
print(np.unique(cars_data1['Doors']))
# ['2' '3' '4' '5' 'five' 'four' 'three'] there are strings along with
integer values

# use replace() to replace a value with desired value
cars_data1['Doors'].replace('three', 3, inplace=True)
cars_data1['Doors'].replace('four', 4, inplace=True)
cars_data1['Doors'].replace('five', 5, inplace=True)
```

```
# converting Doors to int64
cars_data1['Doors']= cars_data1['Doors'].astype('int64')

# rechecking the Doors column for datatype and unique values
print(np.unique(cars_data1['Doors']))
print(cars_data1.info())

# removing row/columns with nan values

#removing all rows with missing values
cars1=cars_data1.dropna(axis=0, how='any')
print(cars1.info())
#removing rows with all nan values
cars1=cars_data1.dropna(axis=0, how='all')
print(cars1.info())
#removing all columns with missing values
cars2=cars_data1.dropna(axis=1, how='any')
print(cars2.info())
#removing columns with all nan values
cars2=cars_data1.dropna(axis=1, how='all')
print(cars2.info())



#Handling the missing data/ nan values
cars_data2=cars_data1.copy()
cars_data3=cars_data1.copy()

#To check the count of missing values present in each column
Dataframe.isnull.sum () is used
print('count of missing values in each column:\n', cars_data2.isnull().sum())
print('count of missing values in each column:\n', cars_data2.isna().sum())
# Subsetting the rows that have one or more missing values
missing=cars_data2[cars_data2.isnull().any(axis=1)]

# imputing the missing values
# Two ways of approach to fill the missing data
'''
1. Fill the missing values by mean / median ,
in case of numerical variable
2. Fill the missing values with the class which has maximum count ,
in case of categorical variable
'''
'''
Using DataFrame.describe() Generate descriptive statistics  like
count, mean, std, min, 25%, 50%, 75%, and max that summarize the
central tendency, dispersion and shape of a dataset’s
distribution, excluding NaN values'''
pd.set_option('max_columns', 50)

c_d=cars_data2.describe()
print(c_d)

# imputing the missing values with mean/median for numerical datatype
''' The mean and median of Age variable is almost same.
hence replace the missing values with mean'''

Age_mean=cars_data2['Age'].mean()

cars_data2['Age'].fillna(cars_data2['Age'].mean(), inplace=True)

print('count of missing values in Age column:\n',
cars_data2['Age'].isnull().sum())

'''
The mean and median of KM variable are very far from each other.
This is because of high extreme values present.
Hence use median value to replace the missing values'''

KM_median=cars_data2['KM'].median()

cars_data2['KM'].fillna(cars_data2['KM'].median(), inplace=True)

print('count of missing values in KM column:\n',
cars_data2['KM'].isnull().sum())

''' The mean and median of HP variable is almost same.
```

```
hence replace the missing values with mean'''

HP_mean=cars_data2['HP'].mean()

cars_data2['HP'].fillna(cars_data2['HP'].mean(), inplace=True)

print('count of missing values in HP column:\n',
cars_data2['HP'].isnull().sum())

# Imputing missing values of 'FuelType'

'''
Series.value_counts()
• Returns a Series containing counts of unique values

• The values will be in descending order so that the
first element is the most frequently occurring
element
• Excludes NA values by default'''

print('count of unique values from FuelType column:\n',
cars_data2['FuelType'].value_counts())

''' As frequency of petrol category is more when compared to others,
replace the missing values with Petrol category'''

cars_data2['FuelType'].fillna(cars_data2['FuelType'].value_counts().index[0],
inplace=True)

print('count of missing values in Fueltype column:\n',
cars_data2['FuelType'].isnull().sum())

# Imputing missing values of ‘MetColor' using mode
Met_mode=cars_data2['MetColor'].mode()
cars_data2['MetColor'].fillna(cars_data2['MetColor'].mode()[0], inplace=True)

print('count of missing values in MetColor column:\n',
cars_data2['MetColor'].isnull().sum())
```

```
# rechecking the null values in the data frame after filling the data

print('count of missing values in each column:\n', cars_data2.isnull().sum())

# Imputing missing values using lambda function
cars_data3=cars_data3.apply(lambda x:x.fillna(x.mean()) if x.dtype=='float'
else x.fillna(x.value_counts().index[0]))
print('count of missing values in each column:\n', cars_data3.isnull().sum())
```

Output:

Experiment 10 : Pandas Dataframes: Exploratory Data Analysis

Date:

Aim: Write a program to perform Exploratory data analysis using.

- Frequency tables
- Two-way tables
- Two-way tables – joint probability
- Two-way tables – marginal probability
- Two-way tables – conditional probability
- Correlation matrix

Dataset: ‘Iris_data_sample.csv’, ‘Iris_data_sample.xlsx’, ‘Iris_data_sample.txt’, ‘Toyota.csv’

Description:

Program:

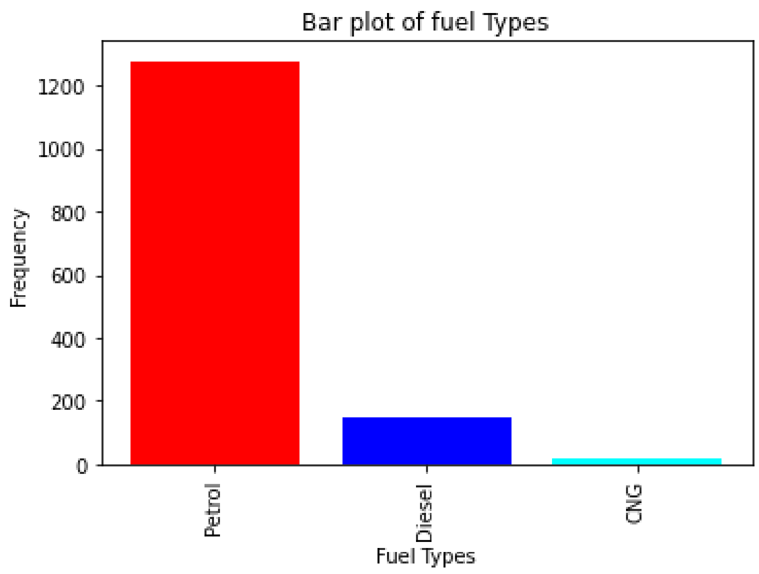
```
# importing necessary libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# changing the current working directory
os.chdir(r"D:\aditya\even sem 22-23\APP\pandas") # specify the path where
data files are stored
#importing data from csv file
cars_data=pd.read_csv('cars_cleaned.csv',index_col=0)

#checking for missing values in the dataframe
print('count of missing values in each column:\n', cars_data.isnull().sum())

#creating copy of original data
cars_data2=cars_data.copy()

# Frequency Tables
'''
pandas.crosstab()
--->To compute a simple cross tabulation of one, two (or more)
```

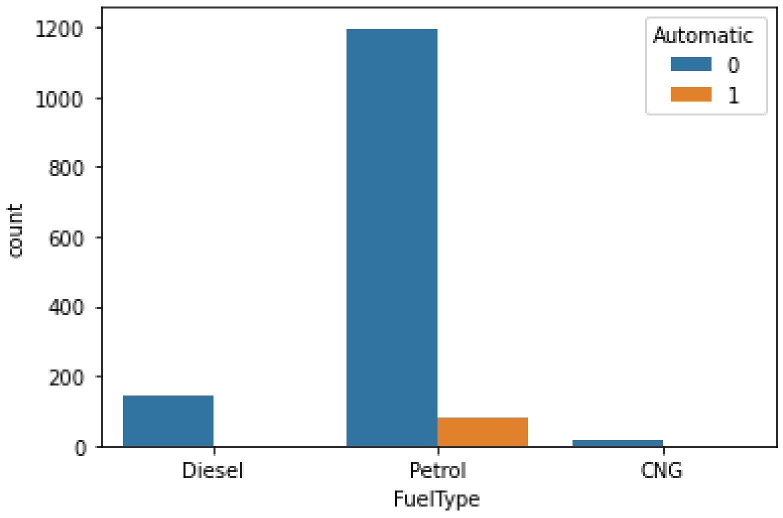
```
factors

--->By default computes a frequency table of the factors'''

cross=pd.crosstab(index=cars_data2['FuelType'], columns='count')
print("Frequency of categorial data in FuelType column is:", cross)

# It is observed most of the cars have petrol as fuel type

# This is similar to barplot
counts=np.array([cars_data2['FuelType'].value_counts()])
counts=counts.flatten()
fuelType=('Petrol','Diesel', 'CNG')
index=np.arange(len(fuelType))

plt.bar(index, counts, color=['red','blue','cyan'])
plt.title('Bar plot of fuel Types')
plt.xlabel('Fuel Types')
plt.ylabel('Frequency')
plt.xticks(index,fuelType,rotation=90)
plt.show()

#Two-way tables
'''
--->To look at the frequency distribution of gearbox types
with respect to different fuel types of the cars
'''
cross2=pd.crosstab(index=cars_data2['Automatic'],
             columns=cars_data2['FuelType'])
print("Frequency relationship between FuelType and Automatic variable: \n",
cross2)

# It is observed that cars with Automatic gearbox system are of Petrol fuel
type

# This can be visualized by countplot function in seaborn
sns.countplot(x='FuelType', data=cars_data2, hue='Automatic')
```

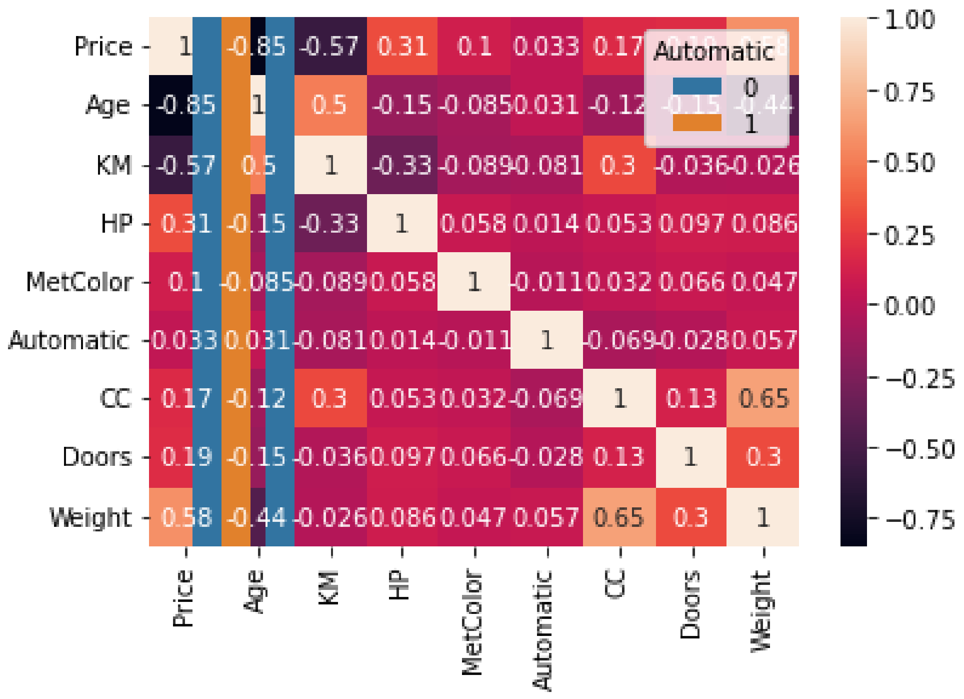
```
# Two-way table: joint probability
'''
Joint probability is the likelihood of two independent events
happening at the same time'''

cross3=pd.crosstab(index=cars_data2['Automatic'],
             columns=cars_data2['FuelType'],
             normalize=True)
print("joint probability between FuelType and Automatic variable: \n",
cross3)

# Two-way table: marginal probability
cross4=pd.crosstab(index=cars_data2['Automatic'],
             columns=cars_data2['FuelType'],
             margins=True,
             normalize=True)
print("marginal probability of FuelType and Automatic variables within joint
probability: \n", cross4)

# probability of cars having manual gear box when the fuel type are CNG or
Diesel or Petrol is 0.95

# Two-way table: conditional probability
'''
---> Conditional probability is the probability of an event ( A ),
given that another event ( B ) has already occurred
---> Given the type of gear box, probability of different fuel type'''
cross5=pd.crosstab(index=cars_data2['Automatic'],
             columns=cars_data2['FuelType'],
             margins=True,
             normalize='index')
print("marginal probability of FuelType when Automatic variable has already
occured: \n", cross5)

cross6=pd.crosstab(index=cars_data2['Automatic'],
             columns=cars_data2['FuelType'],
             margins=True,
             normalize='columns')
```

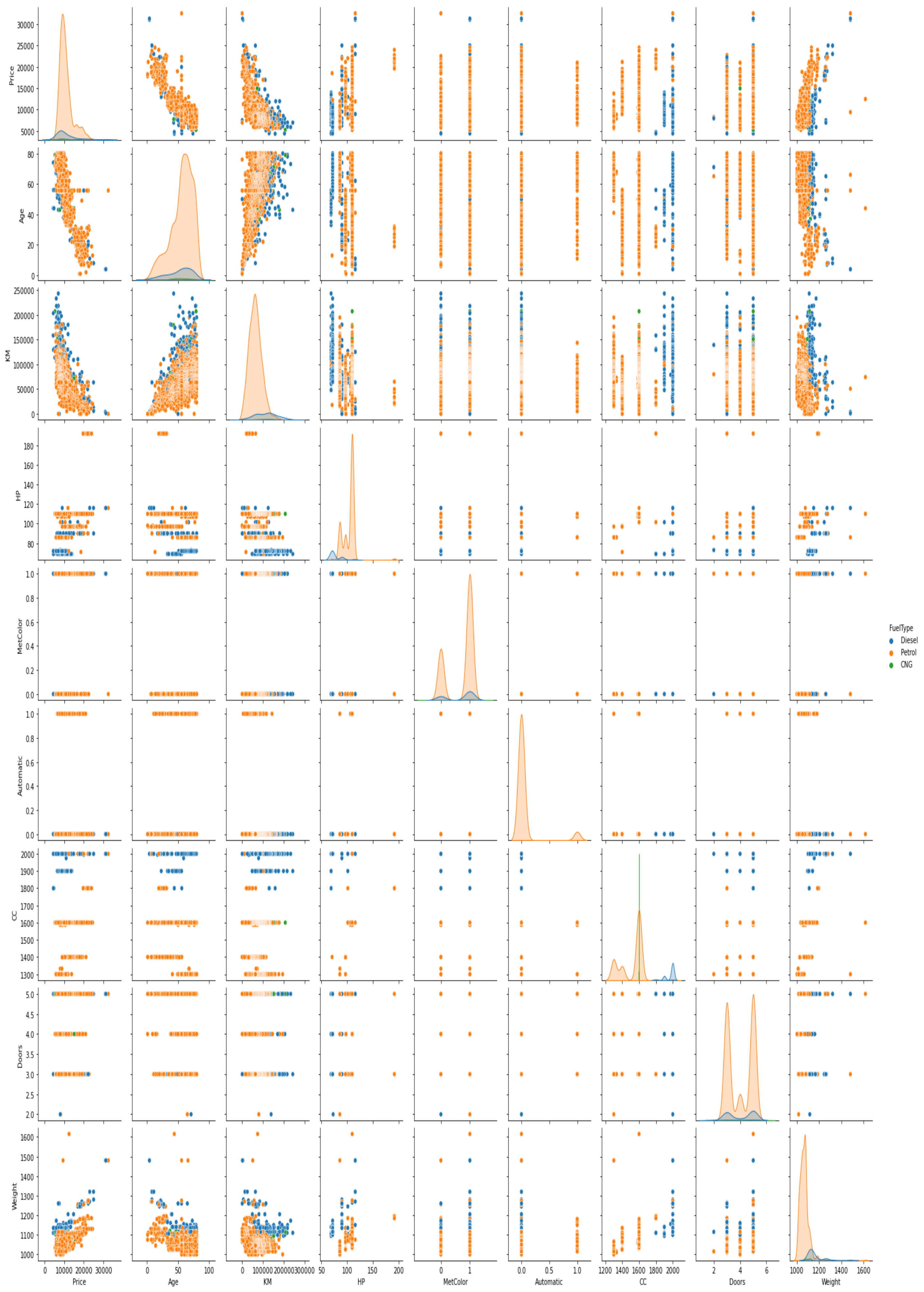
```
print("marginal probability of Automatic when FuelType variable has already
occured: \n", cross6)

# correlation
'''
DataFrame.corr(self, 'pearson')
--->Correlation between numerical variables
--->To compute pairwise correlation of columns excluding NA/null
values
--->Excluding the categorical variables to find the Pearson’s
correlation
'''
numerical_data=cars_data2.select_dtypes(exclude=[object])
print(numerical_data.shape)
corr_matrix=numerical_data.corr()
print('correlation between numerical variables is', corr_matrix)

# heat maps for visualizing correlation
sns.heatmap(corr_matrix, annot=True)

# pairwise plots

'''
--->It is used to plot pairwise relationships in a dataset
--->Creates scatterplots for joint relationships and histograms for
univariate distributions'''
# pairwise plots with hue as fueltype
sns.pairplot(cars_data2, kind="scatter", hue="FuelType")
plt.show()
```

output:


Bar plot of fuel Types
Frequency
0
200
400
600
800
1000
1200
Petrol
Diesel
CNG
Fuel Types


count
0
200
400
600
800
1000
1200
Automatic
0
1
Diesel
Petrol
CNG
FuelType


Price
Age
KM
HP
MetColor
Automatic
CC
Doors
Weight
Automatic
0
1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


Price
Age
KM
HP
MetColor
Automatic
CC
Doors
Weight
FuelType
Diesel
Petrol
CNG