Lab 5: NumPy and Basic Plotting

Name: _____ University Number: _____

Exercise 1: Chebyshev Polynomial of the First Kind

AIM:

Write a Python program that uses the NumPy `Polynomial` class to print a table of the first ten Chebyshev polynomials of the first kind. Here is the table generated by this program:

```
T_0(x) = 1
T_1(x) = x
T_2(x) = 2*x^2 - 1
⋮
```

ALGORITHM:

1. Start
2. Import the `numpy` module and the `Polynomial` class from numpy
3. Generate the first ten Chebyshev polynomials of the first kind
4. Print a table of these polynomials
5. End

PROGRAM:

```
# Exercise 1: Chebyshev Polynomial of the First Kind
# Written by F K Chow, HKU
# Last update: 2022/2/28

# Import the numpy module and the Polynomial class from numpy
import numpy as np
from numpy.polynomial import Polynomial

# Generate the first ten Chebyshev polynomials of the first kind
Chebyshev = np.polynomial.Chebyshev
n = 10
T = []
```

```
for i in range(n):
    T.append(Chebyshev.basis(i))

# Print a table of these polynomials
for i, Ti in enumerate(T):
    Tipoly = Polynomial.cast(Ti)
    print("T_{:d}(x) =".format(i), end=' ')
    d = len(Tipoly.coef) - 1   # Degree of the polynomial
    Tistr = ""
    for j, c in enumerate(Tipoly.coef[::-1]):
        if c != 0:
            Tistr += " + {:.0f}*x^{:d}".format(c, d-j)
    # Adjust the layout
    Tistr = Tistr.replace("+ -", "- ")
    Tistr = Tistr.replace("*x^0", "")
    Tistr = Tistr.replace(" 1*", " ")
    Tistr = Tistr.replace("x^1", "x")
    if Tistr[:3] == " + ":    # Remove initial +
        Tistr = Tistr[3:]
    if Tistr[:3] == " - ":    # Fix spaces for initial -
        Tistr = "-" + Tistr[3:]
    print(Tistr)
```

OUTPUT:

```
T_0(x) = 1
T_1(x) = x
T_2(x) = 2*x^2 - 1
T_3(x) = 4*x^3 - 3*x
T_4(x) = 8*x^4 - 8*x^2 + 1
T_5(x) = 16*x^5 - 20*x^3 + 5*x
T_6(x) = 32*x^6 - 48*x^4 + 18*x^2 - 1
T_7(x) = 64*x^7 - 112*x^5 + 56*x^3 - 7*x
T_8(x) = 128*x^8 - 256*x^6 + 160*x^4 - 32*x^2 + 1
T_9(x) = 256*x^9 - 576*x^7 + 432*x^5 - 120*x^3 + 9*x
```

Exercise 2: Sound Intensity from a Point Source

AIM:

In an experiment, Mary measured the variation of the intensity I of the sound produced by a point source with the distance r from the source. Here is her measurement result:

r (m)	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4
I (10^{-5} W/m^2)	0.987	0.662	0.525	0.373	0.308	0.262	0.191	0.184

On the other hand, physical theories tell us that for a point source of sound of power P, the sound intensity I and the distance r from the source are related by:

$$I = \frac{P}{4\pi r^2}$$

Write a Python program that uses the `np.linalg` method `lstsq` to find the best least-square fit of

$$\ln I = m \ln r + k$$

for the given data and then display the fitting result together with the theoretical prediction. Your program should output a table of the values of the fitting parameters m and k found from the fitting and their theoretical values as well as the root-mean-square of the residual of the fitting. Assume that the point source emits sound with a power of $P = 4\pi\times10^{-5}$ W.

ALGORITHM:

1. Start
2. Import the `numpy` module
3. Construct the lists of the measured values of sound intensity I and distance r from the source
4. Find the best least-square fit of the data
5. Compute the theoretical values of the fitting parameters
6. Display the fitting result together with the theoretical prediction
7. End

PROGRAM:

```
# Exercise 2: Sound Intensity from a Point Source
# Written by F K Chow, HKU
# Last update: 2022/4/4

# Import the numpy module
```

```
import numpy as np

# Construct the lists of the measured values of sound intensity I (in
# W/m^2) and distance r (in m^2) from the source
I = np.array([0.987, 0.662, 0.525, 0.373, 0.308, 0.262, 0.191,
              0.184])*10**(-5)
r = np.linspace(1.0, 2.4, 8)

# Find the best least-square fit of the data
n = len(r)
A = np.vstack((np.log(r), np.ones(n))).transpose()
b = np.log(I)
x, resid = np.linalg.lstsq(A, b, rcond=None)[0:2]
mf, kf = x

# Compute the theoretical values of the fitting parameters
mt = -2.0
kt = np.log(10**(-5))    # k = np.log(P/4pi)

# Display the fitting result together with the theoretical prediction
print('{:>19s} {:>20s}'.format('Fitting results',
                               'Theoretical values'))
print('-'*40)
print('{:>1s} {:>17.5f} {:>20.5f}'.format('m', mf, mt))
print('{:>1s} {:>17.5f} {:>20.5f}'.format('k', kf, kt))
print('Rms residual of the fitting = {:.8f}'.format(np.sqrt(resid[0])))
```

OUTPUT:

```
    Fitting results   Theoretical values
----------------------------------------
m         -1.95977             -2.00000
k        -11.53947            -11.51293
Rms residual of the fitting = 0.12549249
```

Exercise 3: Forced Vibration with Damping

AIM:

A small block of mass m suspended vertically by a spring with spring constant k is driven by an external force $F(t) = F_0 \cos(\omega t)$. The block is moving in a viscous medium with a damping force of the form $-bv$ where $b > 0$ is the damping constant and v is its instantaneous velocity. Taking downward as the positive direction, the vibration of the block is modeled by the differential equation:

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = F_0 \cos(\omega t)$$

where $x(t)$ is the displacement of the block from its equilibrium position at time t. It can be shown that the steady state solution (i. e. $x(t)$ when time $t \to \infty$) is

$$x_s(t) = (MF_0/k)\cos(\omega t - \phi)$$

In this formula, M is the magnification ratio and ϕ is the phase lag defined by

$$M = \frac{1}{\sqrt{(1-\omega^2/\omega_0{}^2)^2 + 4\zeta^2(\omega/\omega_0)^2}}, \quad \phi = \tan^{-1}\left[\frac{2\zeta(\omega/\omega_0)}{1-(\omega/\omega_0)^2}\right],$$

where $\omega_0 = \sqrt{k/m}$ is the natural frequency and $\zeta = b/(2\sqrt{km})$ is the damping ratio. Write a Python program that uses the Matplotlib `Axes` class method `plot` to plot the magnification ratio M over the interval of frequency ratio ω/ω_0 from 0 to 2.0 for damping ratio $\zeta = 0.1$, 0.2, 0.4, 0.6, and 0.8, respectively, on the same graph. You should label your graph with proper axis labels, title, and legends. From your graph, you can observe how the peak value of M depends on ζ, i. e. the effect of damping on the resonance frequency of the block.

ALGORITHM:

1. Start
2. Import the `numpy` module and the `matplotlib.pyplot` module
3. Set up the data and parameters for the plot
4. Create the plot of the magnification ratio M versus the frequency ratio ω/ω_0 for different values of damping ratio ζ
5. End

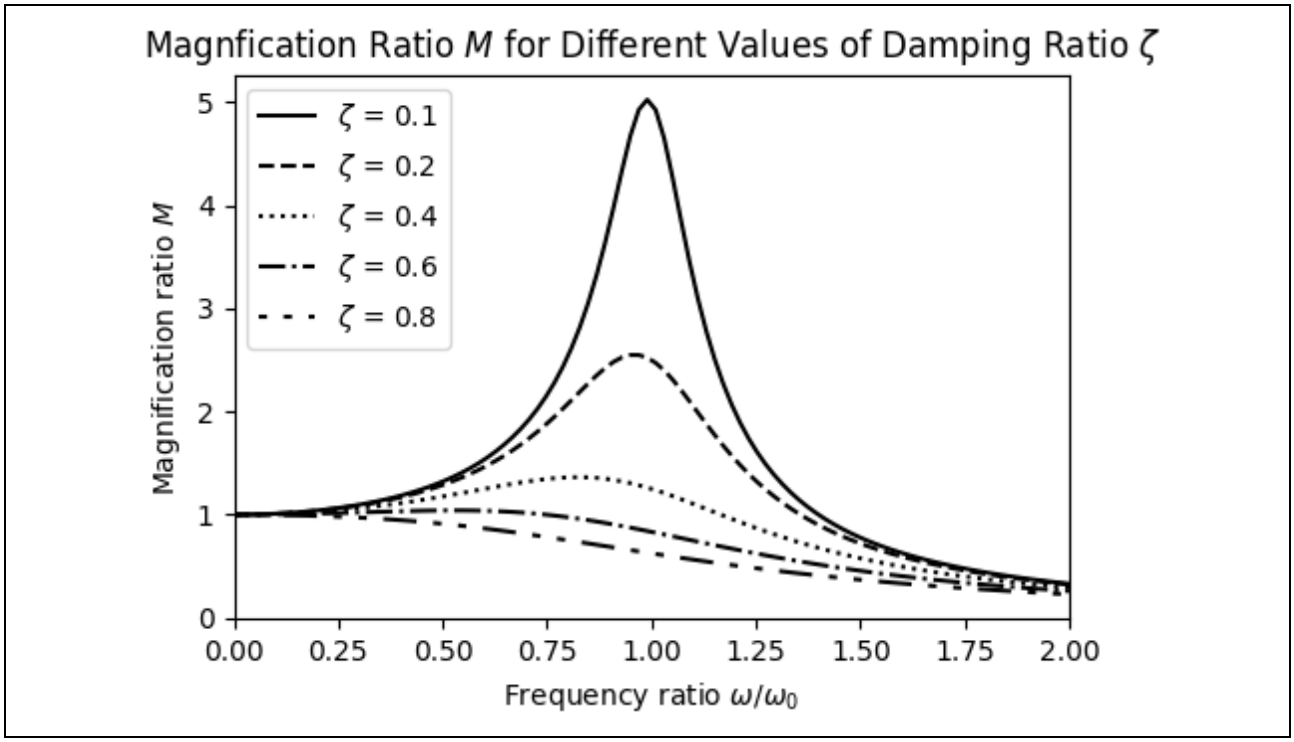
PROGRAM:

```
# Exercise 3: Forced Vibration with Damping
# Written by F K Chow, HKU
```

```
# Last update: 2022/3/2

# Import the numpy module and the matplotlib.pyplot module
import matplotlib.pyplot as plt
import numpy as np

# Set up the data and style for the plot
dpratio = np.array([0.1, 0.2, 0.4, 0.6, 0.8])
fratio = np.linspace(0, 2.0, 100)
linestyles = [{"ls": "-"}, {"ls": "--"}, {"ls": ":"}, {"ls": "-."},
              {"dashes": [2, 4, 2, 4, 8, 4]}]

# Create the plot of the magnification ratio vs the frequency ratio
# for different values of damping ratio
fig = plt.figure()
ax = fig.add_subplot(111)
for i, dpr in enumerate(dpratio):
    lstr = r'$\zeta$ = {:.1f}'.format(dpr)
    ax.plot(fratio, 1/np.sqrt((1-fratio**2)**2 + 4*dpr**2*fratio**2),
            label=lstr, c='k', **linestyles[i])
ax.legend(loc="upper left")
ax.set_xlim(0, 2.0)
ax.set_xlabel(r'Frequency ratio $\omega/\omega_0$')
ax.set_ylabel(r'Magnification ratio $M$')
ax.set_title('Magnfication Ratio $M$ for Different Values of Damping \
Ratio $\zeta$')
plt.show()
```

(The output is shown on the next page.)

OUTPUT:


Magnfication Ratio M for Different Values of Damping Ratio ζ
Magnification ratio M
Frequency ratio ω/ω0
ζ = 0.1
ζ = 0.2
ζ = 0.4
ζ = 0.6
ζ = 0.8
0
1
2
3
4
5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Exercise 4: Employees in Hong Kong's Construction Industry

AIM:

Below is the table of the employment statistics in Hong Kong's construction industry from 2011 to 2020 (source: https://www.censtatd.gov.hk/tc/scode200.html by Census and Statistics Department, HKSAR).

Year	Number of Employees in Thousands	Share of the Employees in the Labour Force
2011	277.0	7.75 %
2012	290.1	7.93 %
2013	309.0	8.30 %
2014	309.7	8.27 %
2015	316.7	8.39 %
2016	328.4	8.67 %
2017	342.0	8.95 %
2018	351.6	9.09 %
2019	337.5	8.77 %
2020	310.0	8.47 %

Write a Python program that uses Matplotlib `Axes` class method `twinx` to produce a bar chart of the number of employees in Hong Kong's construction industry and a line plot of the percentage share of these employees in the labour force as a function of year on the same graph. You should label your graph with proper axis labels, title, and legends.

ALGORITHM:

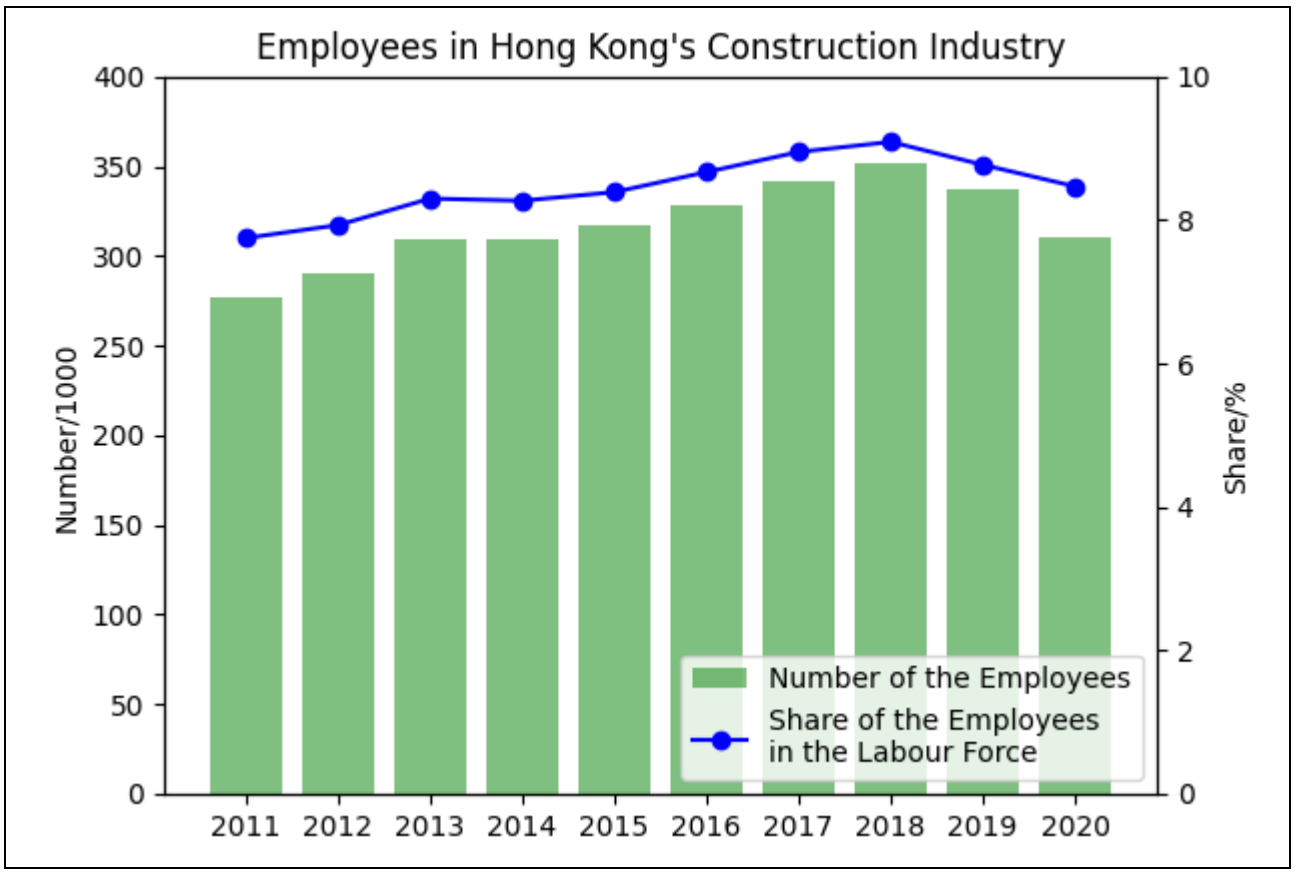
1. Start
2. Import the `numpy` module and the `matplotlib.pyplot` module
3. Set up the data for the plot
4. Produce a bar chart of the number of employees in Hong Kong's construction industry and a line plot of the percentage share of these employees in the labour force as a function of year on the same graph
5. End

(The program is shown on the next page.)

PROGRAM:

```
# Exercise 4: Employees in Hong Kong's Construction Industry
# Written by F K Chow, HKU
# Last update: 2022/4/4

# Import the numpy module and the matplotlib.pyplot module
import matplotlib.pyplot as plt
import numpy as np

# Set up the data for the plot
years = np.arange(2011, 2021)
number_in_K = np.array([277.0, 290.1, 309.0, 309.7, 316.7, 328.4,
                        342.0, 351.6, 337.5, 310.0])
pct_share = np.array([7.75, 7.93, 8.30, 8.27, 8.39, 8.67, 8.95, 9.09,
                      8.77, 8.47])

# Produce a bar chart of the number of employees in Hong Kong's
# construction industry and a line plot of the percentage share of
# these employees in the labour force as a function of year on the
# same graph
fig, ax_n = plt.subplots()
ax_s = ax_n.twinx()
bar = ax_n.bar(years, number_in_K, width=0.8, color="green", alpha=0.5,
               align="center", label="Number of the Employees")
line, = ax_s.plot(years, pct_share, 'b-o',
                  label='Share of the Employees\nin the Labour Force')
ax_n.set_ylim(0, 400)
ax_s.set_ylim(0, 10)
ax_n.set_xticks(years)
ax_n.set_ylabel('Number/1000')
ax_s.set_ylabel('Share/%')
ax_n.set_title('Employees in Hong Kong\'s Construction Industry')
plt.legend(handles=[bar, line], loc='lower right')
plt.show()
```

OUTPUT:


Employees in Hong Kong's Construction Industry
Number/1000
Share/%
400
350
300
250
200
150
100
50
0
10
8
6
4
2
0
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
Number of the Employees
Share of the Employees in the Labour Force