Numerical Method

Lab01

1. RockPaperScissors

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import random


# In[2]:


cmp = random.randint(1, 3)


# In[3]:


if(cmp == 1):
    cmp = 'r'
elif(cmp == 2):
    cmp = 'p'
else:
    cmp = 's'


# In[4]:


l = 0
w = 0
t = 0


# In[5]:


print('Welocome to ROCK, PAPER, SCISSORS game!')


# In[6]:


while True:
    player = input('Enter your move: (r)ock (p)aper (s)cissors')
    if(player == cmp):
        print('It is a tie!')
        t += 1
    elif(player == 'r'):
        if(cmp == 'p'):
            print('You lose!')
            l += 1
        else:
            print('You Win!')
            w += 1
    elif(player == 'p'):
        if(cmp == 's'):
            print('You lose!')
            l += 1
        else:
            print('You Win!')
            w += 1
    elif(player == 's'):
        if(cmp == 'r'):
            print('You lose!')
            l += 1
        else:
            print('You Win!')
            w += 1
```

```
    if(w == 1):
        break
    else:
        print('You have ', t,' ties and ' , l, 'losses' )
```

2. cos

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import math


# In[2]:


x = float(input('Please input x '))


# In[3]:


N = 2
cos = 1
sign = 1
while(N <= 24):
    sign = -sign
    cos += sign * x**N / math.factorial(N)
    N += 2

print(cos)
```

Lab02

1. map

```
#!/usr/bin/env python
# coding: utf-8

# In[65]:


class Map:
    def __init__(self, n, p):
        """
        initialize by n
        """
        self.n = n
        self.p = p

        if p == 1:
            """
            construct square inside the Map
            """
            self.matrix = [['*'] * n for i in range(n)]      #先全部放*
            for i in range(1, n-1):                          #第一跟最後都是***填滿，但中間是*ooo*之類的，所以range start from 1
                self.matrix[i][1:-1] = ['o'] * (n - 2)
        else:
            self.matrix = [[' '] * n for j in range(n)]

    def display(self):
        """
        display the map
        """
        for i in self.matrix:
            print(i)
```

```
# In[68]:


my_map = Map(5, 1)
my_map.display()
```

2. matrix

```
#!/usr/bin/env python
# coding: utf-8

# In[53]:


import random
from copy import deepcopy


class Matrix:

    def __init__(self, nrows, ncols):
        """Construct a (nrows X ncols) matrix"""
        self.nrows = nrows
        self.ncols = ncols

        self.matrix = []

        for i in range(nrows):
            r = []
            for j in range(ncols):
                r.append(random.randint(0, 10))                  #random int from 0-10所以用randint

            self.matrix.append(r)
        pass

    def add(self, m):
        """return a new Matrix object after summation"""
        # Check two matrix are in the same size

        if(self.nrows != m.nrows or self.ncols != m.ncols):
            print('Error')
            return None

        # add self.matrix with m.matrix
        result = deepcopy(self)                                  #copy result form self

        for i in range(self.nrows):
            for j in range(self.ncols):
                result.matrix[i][j] += m.matrix[i][j]            #每一個元素相加

        pass

        return result

    def sub(self, m):
        """return a new Matrix object after substraction"""
        # Check two matrix are in the same size

        if(self.nrows != m.nrows or self.ncols != m.ncols):
            print('Error')
            return None

        # substraction self.matrix from m.matrix
        result = deepcopy(self)                                  #copy result form self

        for i in range(self.nrows):
            for j in range(self.ncols):
                result.matrix[i][j] -= m.matrix[i][j]            #每一個元素相減

        pass

        return result

    def mul(self, m):
        """return a new Matrix object after multiplication"""
        # Check if the nrows of m is the same as ncols of self

        if(self.ncols != m.nrows):
```

```
            print('Error')
            return None

        # Multiply self.matrix and m.matrix
        result = Matrix(self.nrows, m.ncols)
        pass

        for i in range(self.nrows):
            for j in range(m.ncols):
                temp = 0                                        #定義一個要暫存的值的變數且每輪要重設
                for k in range(m.ncols):
                    temp = self.matrix[i][k] * m.matrix[k][j]   #將乘完的值丟進temp暫存
                result.matrix[i][j] = temp                      #將temp的值丟進result中

        return result

    def transpose(self):
        """return a new Matrix object after transpose"""
        # Tranpose
        result = Matrix(self.nrows, self.ncols)                 #此處的self = mul的result所以不用再算一次mul
        pass

        for i in range(self.nrows):
            for j in range(self.ncols):
                result.matrix[j][i] = self.matrix[i][j]         #row col互換

        return result

    def display(self):
        """Display the content in the matrix"""

        for i in self.matrix:                                   #將整個矩陣print出來
            print(i)

        pass


# In[54]:


a_rows = int(input("Enter A matrix's rows:"))
a_cols = int(input("Enter A matrix's cols:"))
print("Matrix A({}, {}):".format(a_rows, a_cols))
A = Matrix(a_rows, a_cols)
A.display()

b_rows = int(input("Enter B matrix's rows:"))
b_cols = int(input("Enter B matrix's cols:"))
print("Matrix B({}, {}):".format(b_rows, b_cols))
B = Matrix(b_rows, b_cols)
B.display()

print("="*10, 'A + B', "="*10)
result = A.add(B)
if result is not None:
    result.display()

print("="*10, 'A - B', "="*10)
result = A.sub(B)
if result is not None:
    result.display()

print("="*10, 'A * B', "="*10)
result = A.mul(B)
if result is not None:
    result.display()

print("="*5, "the transpose of A*B", "="*5)
result = result.transpose()
if result is not None:
    result.display()
```

Lab03

1. LineProp

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


# define parameter for plt
linewidths = [0.5, 1.0, 2.0, 4.0]
linestyles = ['-', '--', '-.', ':']
markers = ['+', 'o', '*', 's', '.', '1', '2', '3', '4']
markersizecolors = [(4, "white"), (8, "red"), (12, "yellow"), (16, "lightgreen")] #tuple in list


# In[2]:


import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-5, 5, 5)
y = np.ones_like(x)                                     # return an array of 1 with the same shape and type as x.
def axes_settings(fig, ax, title, ymax):
    ax.set_xticks([])                                   # No xticks
    ax.set_yticks([])                                   # No yticks
    ax.set_ylim(0, ymax+1)
    ax.set_title(title)

fig, axes = plt.subplots(1, 4, figsize=(16,3))             # create a figure with 1 row X 4 columns subfig
# Line width
for n, linewidth in enumerate(linewidths):                 # enumerate returns index and value
    axes[0].plot(x, y + n, color="blue", linewidth=linewidth)  #線粗為題目定義的
axes_settings(fig, axes[0], "linewidth", len(linewidths))

# Line style
for n, linestyle in enumerate(linestyles):
    axes[1].plot(x, y + n, color =  "blue", linewidth = 3, linestyle = linestyles[n])     #線粗自訂，線的形式則為題目規定的
axes_settings(fig, axes[1], "linestyle", len(linestyles))


# Marker
for n, marker in enumerate(markers):
    axes[2].plot(x, y + n, color =  "blue", linewidth = 0, markersize = 4, marker = markers[n])     #不需要線所以linewidth = 0，makersize
axes_settings(fig, axes[2], "makers", len(markers))

# marker size/color
for n, (size, color) in enumerate(markersizecolors):
    axes[3].plot(x, y + n, color =  "blue", marker = 'o', linewidth = 0, markersize = size, markerfacecolor = color)    #不需要線所以line
axes_settings(fig, axes[3], "markersizecolors", len(markersizecolors))



# In[ ]:




# In[ ]:
```

2.

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
def least_squares_fit(x, y):
    x_ave = np.mean(x)
    y_ave = np.mean(y) # np.mean() return mean value of given array
    # [hint] np.dot: inner product for 1D array; matrix multiplication for 2D array
    num = np.sum((x - x_ave) * (y - y_ave))   #分子 = (xi - x_avg)(yi - y_avg)
    den = np.sum((x - x_ave) ** 2)            #分母 = (x - x_avg ^ 2)
    beta1 = num / den
    beta0 = y_ave - beta1 * x_ave

    return beta0, beta1
```

```
    pass


# In[6]:


#for testing your function of computing beta

#report beta0 and beta1
X = np.array([1, 2, 3, 4])
Y = np.array([9, 13, 14, 18])
beta0, beta1 = least_squares_fit(X, Y)
print('From home-made linear regression model')
print('beta0 =', beta0)
print('beta1 =', beta1)


# In[38]:


#plot the line with matplotlib

import matplotlib.pyplot as plt

#1. [hint]plot points by plt.scatter (parameter setting: c=marker color='blue', s=marker size=20)
plt.scatter(X, Y, c = 'blue', s = 20)        #散布圖
#2. [hint]plot line by plt.plot(parameter setting: lw=linewidth=3)
n = len(X)
x_fit = np.arange(X.min(), Y.min(), n)
y_fit = beta0 + beta1 * x_fit
plt.plot(x_fit, y_fit, linewidth = 3, color = 'red', markersize = 20)       #Fitting line

plt.show()


# In[ ]:
```

Lab04

```
#!/usr/bin/env python
# coding: utf-8

# ### import the module you need first

# In[17]:
import numpy as np


# ## step1.
# #### Construct a numpy array represent the equations.
# #### the coefficients,answers and augmented matrix of each equation need to be stored separately.

# In[18]:


"""
example:

5y-7z = 7        | 0 5 -7 7 |
4x-z = 8      -> | 4 0 -1 8 |
x-y+z = 9        | 1 -1 -1 9 |
"""

coe = np.array([[0, 5, -7], [4, 0, -1], [1, -1, -1]])          #係數
ans = np.array([7, 8, 9])                                       #答案

augmented_matrix = np.column_stack((coe, ans)).astype(float)  #增廣矩陣，且有可能是float

for i, row in enumerate(augmented_matrix):                    #列出矩陣
    equation = "  ".join([f"{coe[i][j]}{var}" for j, var in enumerate(["x", "y", "z"]) if coe[i][j] != 0])
    print(f"{equation} = {ans[i]}  |  {row}")


# ## step2.
# ### Before doing gauss elimination,
# ### we need to check if the first element([0,0]) of the augmented matrix is zero or not.
```

```
# ### If the first element is zero,
# ### we need to find another row whose first element isn't zero,and change them.

#test:step2
#print the new augment matrix

if augmented_matrix[0][0] == 0:                 # check if the first element([0,0]) of the augmented matrix is zero or not.
    for i in range(1, len(augmented_matrix)):    # find the first element in row != 0
        if augmented_matrix[i][0] != 0:
            augmented_matrix[[0, i]] = augmented_matrix[[i, 0]]    #swap
            break

print("The new augmented matrix:")
for i in augmented_matrix:
    print(i)


# ## step3. gauss elimination
#
# #### print the matrix after  gauss elimination

# In[20]:


nrows, ncols = augmented_matrix.shape

for i in range(nrows):
    pivot = augmented_matrix[i][i]                          #即將要被消除=0的主對角線的數
    for j in range(i+1, nrows):
        factor = augmented_matrix[j][i] / pivot             #找出它跟前幾列的factor
        augmented_matrix[j] -= factor * augmented_matrix[i]


print(augmented_matrix)


# ## step4. LU decomposition(bonus)
#
# ## if you don't want to submit bouns,you don't need to do  step4 and step5

# In[21]:


#print L and U

n = coe.shape[0]      #n = how many rows
L = np.zeros((n, n))
U = np.zeros((n, n))
P = np.eye(n)

for i in range(n):
    pivot_row = i + np.argmax(np.abs(coe[i:, i])) #Find pivot
    coe[[i, pivot_row]] = coe[[pivot_row, i]]     #Swap rows
    L[[i, pivot_row]] = L[[pivot_row, i]]
    P[[i, pivot_row]] = P[[pivot_row, i]]
    L[i][i] = 1                                   #diagonal elements = 1 in L
    for j in range(i, n):                         #Doolittle Algorithm
        U[i][j] = coe[i][j] - sum(L[i][k] * U[k][j] for k in range(i))
    for j in range(i+1, n):
        L[j][i] = (coe[j][i] - sum(L[j][k] * U[k][i] for k in range(i))) / U[i][i]

print(L)
print(U)


# ## step5. check the ansewer of LU(bonus)
#
# please use the function in scipy.linalg to check your answer
#
# the documentatin of scipy.linalg :
# https://docs.scipy.org/doc/scipy/reference/linalg.html

# In[22]:


import scipy.linalg as la


# In[23]:
```

```
P, L, U = la.lu(coe)
print(L)
print(U)
```

Lab05

```
#!/usr/bin/env python
# coding: utf-8

# In[9]:


import numpy as np
from copy import deepcopy


# ##### Write the function for check the matrix is diagonally dominant

# In[10]:


def check_diagonally(a):
    # Find diagonal coefficients
    # diag = ...
    diag = np.diag(np.abs(a))                       #|aii|

    # Find row sum without diagonal coefficients
    # off_diag = ...
    off_diag = np.sum(np.abs(a), axis = 1) - diag     #axis = 1 -> row, sum of row of each rows

    if np.all(diag > off_diag):
        print("matrix is diagonally dominant")
    else:
        print("NOT diagonally dominant")


# ##### Wirte the funtion for Gauss-Seidel method

# In[16]:


def Gauss_Seidel(a, x, y, it, epsilon):
    converged = False
    x_old = np.zeros(a.shape[0])
    print("Iteration results")
    print(" k,    x1,      x2,       x3 ")
    # You should to do...
    # Use the for loop to complete the iterative process
        # check if it is smaller than threshold
        # assign the latest x value to the old value
    for i in range(it):                                                 #the most iteration number
        for j in range(a.shape[0]):                                     #how many rows
            sum = 0
            for k in range(a.shape[1]):                                 #how many cols
                if k != j:                                              #avoid diagonal elements
                    sum += a[j][k] * x[k]                               #Gauss–Seidel method
            x[j] = (y[j] - sum) / a[j][j]
        print(f" {i + 1}, {x[0]:.4f}, {x[1]:.4f}, {x[2]:.4f}")
        if np.linalg.norm(x - x_old) < epsilon:                         #check if it is convergent
            converged = True
            break
        x_old = x.copy()                                                # assign the latest x value to the old value

    if converged:
        print('Converged')
    else:
        print("Didn't converged" )


    return x


# # Sample 1

# In[17]:


a = np.array([[5, -1, -3], [2, 9, 3], [2, 4, 8]])
```

```
check_diagonally(a)


# In[18]:


a = np.array([[5, -1, -3], [2, 9, 3], [2, 4, 8]])
x = np.zeros(a.shape[0])
y = np.array([14, 5, -8])

x = Gauss_Seidel(a, x, y, it=50, epsilon=0.0001)
print('')
print('Check')
print('my solve:',x)
x = np.linalg.solve(a, y)
print('np solve:',x)


# # Sample 2

# In[14]:


a = np.array([[12, 3, -5, 2], [1, 7, 3, 1], [3, 7, 13, -2], [-2, 2, 5, 20]])

check_diagonally(a)


# In[15]:


a = np.array([[12, 3, -5, 2], [1, 7, 3, 1], [3, 7, 13, -2], [-2, 2, 5, 20]])
x = np.zeros(a.shape[0])
y = np.array([10, 6, 3, 2])

x = Gauss_Seidel(a, x, y, it=50, epsilon=0.0001)
print('')
print('Check')
print('my solve:',x)
x = np.linalg.solve(a, y)
print('np solve:',x)
```

Lab06

```
#!/usr/bin/env python
# coding: utf-8

# In[84]:


import numpy as np
from numpy.linalg import eig
from numpy.linalg import inv


# In[85]:


def inverse_normalize(x):
    # factor fac is the maximum absoulute value of x
    fac = abs(x).max()
    if fac == abs(x.min()):          #check if x最大的絕對值是否在x中是負數
        fac = fac * -1
    x_n = x / fac

    return fac, x_n


# In[86]:


def inverse_power_method(a, x):
    # The subtraction of the lambda_0 and the lambda_1 must be less than 1e-30
    lambda_0 = 1
    a_inv = inv(a)                           # define a_inv is the inverse of the matrix a
```

```
    iter = 1000                                 # iteration time
    y = 0

    for i in range(iter):
        x = np.dot(a_inv, x)                    #A^-1 dot x0
        lambda_1, x = inverse_normalize(x)
        if(abs(lambda_0 - lambda_1) < 10 ** (-30)):
            break
    lambda_1 = 1 / lambda_1

    return lambda_1, x


# ### Sample 1

# In[87]:


x = np.array([1, 1])
a = np.array([[0, 2],[2, 3]])
lambda_1, x = inverse_power_method(a, x)
print("The Minimum Eigenvalue:", lambda_1)
print("Eigenvector:", x)


# In[88]:


# compare with numpy

a = np.array([[0, 2],[2, 3]])

value, vector = eig(a)
print("E-value:", value)
print("E-vector:\n", vector)


# ### Sample 2

# In[89]:


x = np.array([1, 1, 1])
a = np.array([[1, 5, 2],[2, 4, 3],[2, 1, 6]])
lambda_1, x = inverse_power_method(a, x)
print("The Minimum Eigenvalue:", lambda_1)
print("Eigenvector:", x)


# In[90]:


# compare with numpy

a = np.array([[1, 5, 2],[2, 4, 3],[2, 1, 6]])

value, vector = eig(a)
print("E-value:", value)
print("E-vector:\n", vector)
```

Lab06

```
#!/usr/bin/env python
# coding: utf-8

# In[6]:


import numpy as np
import matplotlib.pyplot as plt


# In[7]:


def my_central_diff(y, h):
```

```
    # y : compute function
    # h : step size

    n = len(y)
    d = np.zeros(n)                         #creat a zero matrix which the len is y

    for i in range(1, n - 1):
        d[i] = (y[i+1] - y[i-1]) / (2 * h)   # f'(xi) = (f(xj+1) - f(xj-1))/2h

    return d


# In[8]:


# step size
h = 0.1
# define grid
x = np.arange(0, 2*np.pi, h)
# compute function
y = np.sin(x)
# compute vector of forward differences
central_diff = my_central_diff(y, h)[1:-1]
# compute corresponding grid
x_diff = x[1:-1]
# compute exact solution
exact_solution = np.cos(x_diff)

# Plot solution
plt.figure(figsize = (12, 8))
plt.plot(x_diff, central_diff, "-", label = "Finite difference approximation", lw=8)
plt.plot(x_diff, exact_solution, label = "Exact solution", lw=4)
plt.legend()
plt.show()

# Compute max error between
# numerical derivative and exact solution
max_error = max(abs(exact_solution - central_diff))
print('The maximum error is', max_error)


# ### ---Bonus---

# In[9]:


# define step size
h = 1
# define number of iterations to perform
iterations = 15
# list to store our step sizes
step_size = []
# list to store max error for each step size
max_error = []


for i in range(iterations):
    # halve the step size
    h /= 2
    # ...to be continued
    # define grid
    x = np.arange(0, 2*np.pi, h)
    # compute function
    y = np.sin(x)
    # compute vector of forward differences
    central_diff = my_central_diff(y, h)[1:-1]
    # compute corresponding grid
    x_diff = x[1:-1]
    # compute exact solution
    exact_solution = np.cos(x_diff)
    step_size.append(h)
    max_error.append(max(abs(exact_solution - central_diff)))   # define the max error between exact sol and central sol

# produce log-log plot of max error versus step size
plt.figure(figsize = (12, 8))
plt.loglog(step_size, max_error, "v", markersize=14)
plt.xlabel('step size', fontsize=20)
plt.ylabel('max error', fontsize=20)
plt.show()
```

Lab08

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
import pandas as pd
import csv
import flet as ft


# In[2]:


# 讀資料

ticket_df = pd.read_csv('tickets.csv', header=None)
ticket = ticket_df.values
ticket_df


# In[3]:


# 南下的組合

southbound_dict = {}
for i in range(len(ticket[0])):
    for j in range(i+1, len(ticket[0])-1):
        station_1 = str(ticket[0][i+1])
        station_2 = str(ticket[0][j+1])
        southbound_dict[(station_1, station_2)] = (ticket[j+1][i+1], ticket[i+1][j+1])


# In[4]:


# 北上的組合

northbound_dict = {}
for i in range(len(ticket[0])):
    for j in range(i+1, len(ticket[0])-1):
        station_1 = str(ticket[0][-i-1])
        station_2 = str(ticket[0][-j-1])
        northbound_dict[(station_1, station_2)] = (ticket[-i-1][-j-1], ticket[-j-1][-i-1])


# In[5]:


# 站名

station = []
for i in range(1, len(ticket[0])):
    station.append(str(ticket[0][i]))


# In[6]:


# 這個function只是給大家感受一下GUI大概會怎麼被使用的樣子。接收到2個input，然後output票價的結果

def THSR_fare():
    start_station = input('Enter your start station:')
    end_station = input('Enter your end station:')
    if southbound_dict.get((start_station, end_station)) is not None:
        print(f'Southbound from {start_station} to {end_station}')
        print(f'The ticket fare of Standard Car is: {southbound_dict[start_station, end_station][0]}')
        print(f'The ticket fare of Business Car is: {southbound_dict[start_station, end_station][1]}')
    elif northbound_dict.get((start_station, end_station)) is not None:
        print(f'Northbound from {start_station} to {end_station}')
        print(f'The ticket fare of Standard Car is: {northbound_dict[start_station, end_station][0]}')
        print(f'The ticket fare of Business Car is: {northbound_dict[start_station, end_station][1]}')
    else:
        print(f'The destination from {start_station} to {end_station} is not found')
```

```
# In[7]:


THSR_fare()
# input 1 is Taipei
# input 2 is Tainan
# output:
# Southbound from Taipei to Tainan
# The ticket fare of Standard Car is: 1,350
# The ticket fare of Business Car is: 2,230


# ## 開始GUI介面實作

# ###### 大家不一定要照著下面的模板實作，只要能夠做出相同功能、相似尺寸的GUI就好

# In[8]:


def main(page: ft.Page):
    global start_station, end_station
    page.scroll = "always"


    # GUI的排版
    page.title = "Taiwan High Speed Rail Fare System"
    page.window_width = 750
    page.window_height = 600
    page.horizontal_alignment = ft.CrossAxisAlignment.CENTER

    button_width = 150
    button_height = 75

    start_station = ""
    end_station = ""

    # 按鈕的function
    def start_station_click(aaa):
        # 使用global變數定義start_station讓其他function也能夠使用到被更新後的同樣參數
        # 可以使用aaa.control.data抓到站名
        global start_station
        start_station = aaa.control.data
        pass

    def end_station_click(aaa):
        global end_station
        end_station = aaa.control.data
        pass

    def result_click(aaa):
        # 可參考THSR_fare()這個function去實作這邊
        fare = [0, 0]
        if southbound_dict.get((start_station, end_station)) is not None:
            fare[0] = southbound_dict[start_station, end_station][0]
            fare[1] = southbound_dict[start_station, end_station][1]
        elif northbound_dict.get((start_station, end_station)) is not None:
            fare[0] = northbound_dict[start_station, end_station][0]
            fare[1] = northbound_dict[start_station, end_station][1]
        else:
            print(f'The destination from {start_station} to {end_station} is not found')
        result_text = ft.Text(f"From {start_station} to {end_station}\nThe frare is for standard car is {fare[0]}\nThe frare is for busines
        page.add(result_text)
        pass

    # ------建立物件------
    start_text = ft.Text("Please select a start station", size=18)
    end_text = ft.Text("Please select a end station", size=18)

    # /// for start station view ///
    # 可以一個一個建立出按鈕，但也可以透過for loop 去建立出 4(column)*3(row)的樣子
    # 使用station這個參數可以抓到各個站名
    # 使用這個函數實作，參數可以自己調整，除了width
    # ft.ElevatedButton(text=f"{station[i]}", data=f"{station[i]}", width=150, on_click=start_station_click)
    num_row = 3
    num_col = 4

    start_button = []
    # 創建所有站的buttons
    for i in range(len(station)):
```

```
        button_text = station[i]
        start_button.append(ft.ElevatedButton(text = button_text, data = button_text, width = 150, on_click = start_station_click))
    #呈現4 * 3 的排列
    start_grid = ft.GridView(start_button, runs_count = 4, child_aspect_ratio = 4)


    # /// for end station view ///
    end_button = []
    # 創建所有站的buttons
    for i in range(len(station)):
        button_text = station[i]
        end_button.append(ft.ElevatedButton(text = button_text, data = button_text, width = 150, on_click = end_station_click))
    #呈現4 * 3 的排列
    end_grid = ft.GridView(end_button, runs_count = 4, child_aspect_ratio = 4)


    # /// for result view ///
    # 使用以下函數實作
    # ft.ElevatedButton(text=f"Calculate the fare", width=630, on_click=result_click)
    # ft.Text("")
    result_button = ft.ElevatedButton(text = "Calculates the fare", width = 630, on_click = result_click)
    result_text = ft.Text("", size = 8)


    # ------將物件進行排版------
    page.add(start_text,
             start_grid,
             end_text,
             end_grid,
             result_button,
             result_text
            )

ft.app(target=main)
```

Midterm Mock Exam

```
# %%
import numpy as np
a = [1, 3, 5 ,7]
type(a)
-------------------------------------------------
list

# %%
b = np.array([1, 3, 5 ,7])
type(b)
-------------------------------------------------
numpy.ndarray

# %%
def normalize(x):
    fac = abs(x).max()
    x_n = x / x.max()
    return fac, x_n

x = np.array([1, 1, 1])
a = np.array([[2, 1, 2], [1, 3, 2], [2, 4, 1]])
for i in range(3):
    x = np.dot(a, x)
    lambda_1, x = normalize(x)
    print(lambda_1)
    print(x)


print('Eigenvalue:', lambda_1)
print('Eigenvector:', x)
-------------------------------------------------
7
[0.71428571 0.85714286 1.        ]
5.857142857142857
[0.73170732 0.90243902 1.        ]
6.073170731707317
[0.7188755  0.89558233 1.        ]
Eigenvalue: 6.073170731707317
Eigenvector: [0.7188755  0.89558233 1.
```

```
import numpy as np
from numpy.linalg import cond, matrix_rank

def Gauss_Elimination(a, y):
    if(a.shape[0] != a.shape[1]):
        print("Error\n")

    if(a.shape[0] != y.shape[0] or y.shape[1] > 1):
        print("Error\n")

    n = len(y)
    i = 0
    j = i - 1
    x = np.zeros(n)

    augmented_matrix = np.concatenate((a, y), axis = 1, dtype = float)
    print(f"Initial matrix : \n{augmented_matrix}\n")

    while i < n:
        # check it won't divide by zero
        if(augmented_matrix[i][i] == 0):
            #swap
            for k in range(1, len(augmented_matrix)):
                if augmented_matrix[k][i] != 0:
                    augmented_matrix[[i, k]] = augmented_matrix[[k, i]]
        for j in range(i + 1, n):
            fac = augmented_matrix[j][i] / augmented_matrix[i][i]
            augmented_matrix[j] -= fac * augmented_matrix[i]
            print(f"The new augmented matrix : \n{augmented_matrix}")
        i = i + 1

    # back substitution
    x[n - 1] = augmented_matrix[n - 1][n] / augmented_matrix[n - 1][n - 1]

    for k in range(n - 2, -1, -1):
        x[k] = augmented_matrix[k][n]
        for j in range(k + 1, n):
                x[k] = x[k] - augmented_matrix[k][j] * x[j]
        x[k] = x[k] / augmented_matrix[k][k]

    print(f"x: {x}")

a = np.array([[0,-2,6],[12,4,-7],[0,4,1]]).astype('float')
y = np.array([[7],[5],[3]]).astype('float')
x = Gauss_Elimination(a, y)
```

```
import numpy as np

# normalize the resulting vector
def normalize(x):
    fac = abs(x).max()
    # check if abs in x is nagetive or not
    if fac == abs(x.min()):
        fac *= -1
    x_n = x / x.max()
    return fac, x_n

def Power_Method(a, x):
    lamba_0 = 1
    iter = 100
    y = 0

    for i in range(iter):
        x = np.dot(a, x)
        # normalize the resulting vector in each iteration
        lambda_1, x = normalize(x)
    return lambda_1, x

x = np.array([1, 1])
a = np.array([[2, 3],[3, -1]])
Eigenvalue, Eigenvector = Power_Method(a, x)
print("Eigenvalue:", Eigenvalue)
print("Eigenvector:", Eigenvector)
```