**BÀI 1.1.**

**n = int(input('Nhap mot so nguyen: '))**

**while n < 0 or n >= 10000:**

**print('n thuoc [0, 9999]. Moi nhap lai: ')**

**n = int(input('Nhap mot so nguyen: '))**

**N = n//1000**

**T = (n % 1000)//100**

**C = (n % 100)//10**

**D = n % 10**

**print("{} nghin {} tram {} chuc {} don vi".format(N, T, C, D))**

**BÀI 1.2.**

**cach Euclidean: ", D)**

**print("Khoang cach Manhatan: ", M)**

**print("Khoang** import math

X1 = float(input('X1: '))

Y1 = float(input('Y1: '))

X2 = float(input('X2: '))

Y2 = float(input('Y2: '))

D = math.sqrt((X1-X2)\*\*2 + (Y1-Y2)\*\*2)

M = math.fabs(X2-X1) + abs(Y2-Y1)

C = 1 - (X1\*X2 + Y1\*Y2)/(math.sqrt(X1\*\*2+Y1\*\*2)\*math.sqrt(X2\*\*2+Y2\*\*2))

print("Khoang**cach Cosin : ", C)**

**BÀI 1.3.**

**import math**

**a = float(input("a = "))**

**b = float(input("b = "))**

**c = float(input("c = "))**

**if a==0:**

**if b==0 and c==0:**

**print("Pt bac 1 - vo so nghiem")**

**elif b==0 and c !=0:**

**print("PT bac 1 - Vo nghiem")**

**else:**

**print("Pt bac 1 co nghiem: X = ", -c/b)**

**else:**

**delta = b\*b - 4\*a\*c**

**if delta<0:**

**print("Vo nghiem")**

**elif delta==0:**

**print("Nghiem kep: ", -b/(2\*a))**

**else:**

**print("X1 = ", (-b + math.sqrt(delta))/(2\*a))**

**print("X2 = ", (-b - math.sqrt(delta))/(2\*a))**

**BÀI 1.4.**

**import math**

**x = float(input("x="))**

**n = int(input("n="))**

**P = 0.0**

**if n%2 == 0:**

**P = 2016\*x**

**T = x**

**M = 1**

**for i in range(1, n):**

**T \*= x**

**M \*= 3**

**P += T/M**

**print("P = %0.3f" %P)**

**BÀI 1.5. License Plate**

**import math  
a = int(input("a = "))  
isNT = True  
#Check so nguyen to  
for i in range(2, a):  
 if a % i == 0:  
 isNT = False  
  
#Check doi xung  
isDX = True  
n = 6  
while a // (10 \*\* n) == 0: n = n-1**

**k = int((n + 1) / 2)  
for i in range(1, k + 1):  
 if a % 10 == a//(10\*\*n):  
 a = a % (10\*\*n) // 10  
 n = n - 2  
 else:  
 isDX = False  
#In ket qua  
if isNT and isDX:  
 print("So hop le")  
else:  
 print("So khong hop le !")**

**Đáp án trong trường hợp giả sử số n luôn có 6 chữ số:**

**import math  
a = int(input("a = "))  
isNT = True  
#Check so nguyen to  
for i in range(2, a):  
 if a % i == 0:  
 isNT = False  
  
#Check doi xung  
isDX = False  
if a % 10 == a // 100000:  
 a = (a % 100000) // 10  
 if a % 10 == a // 1000:  
 a = (a % 1000) // 10  
 if a % 10 == a // 10:  
 isDX = True  
#In ket qua  
if isNT and isDX:  
 print("So hop le")  
else:  
 print("So khong hop le !")**

**BÀI 2.1. License Plate - Upgrade**

**def isPrim(n):  
 if n > 1:  
 for i in range(2, n): *# có thể cho chạy tới sqrt(n)*  
 if n % i == 0:  
 return False  
 return True  
 return False  
  
  
def isSymmetry(n):  
 k = 8 # tìm k là số chữ số của n  
 while n//(10 \*\* k) == 0: k = k – 1**

**t = (k+1)//2  
 for j in range(t):  
 if n % 10 == n // (10 \*\* k):  
 n = n % (10 \*\* k) // 10  
 k = k - 2  
 else:  
 return False  
 return True  
  
  
S = int(input("Start number: "))  
E = int(input("End number : "))  
while E <= S:  
 print("E > S please !")  
 E = int(input("End number : "))  
TOTAL = 0  
for i in range(S, E+1):  
 if isPrim(i) and isSymmetry(i):  
 TOTAL += i  
print("TOTAL: ", TOTAL)**

**BÀI 2.2. Module**

**File rate.py**

**USD = 23000  
EUR = 26000  
RUB = 170  
JPY = 190**

**File counter.py**

**import rate  
  
def toUSD(n):  
 return n / rate.USD  
  
def toEUR(n):  
 return n / rate.EUR  
  
def toRUB(n):  
 return n / rate.EUR  
  
def toJPY(n):  
 return n / rate.JPY**

**File main.py**

**import counter  
mile = float(input("Nhập số dặm bay: "))  
uint\_price = float(input("Số tiền/ dặm: "))  
k = mile\*uint\_price  
print("Số tiền phải trả là", k, "VND")  
print("=", counter.toUSD(k), "USD")  
print("=", counter.toEUR(k), "EUR")  
print("=", counter.toRUB(k), "RUB")  
print("=", counter.toJPY(k), "JPY")**

**BÀI 2.3. Package**

**Sinh viên tự tổ chức chương trình trong bài 2.2 thành các package (xem slide bài giảng).**

**BÀI 2.4. Points**

**File point.py**

**from math import sqrt  
def getPoint():  
 x = float(input('Nhập tọa độ x: '))  
 y = float(input('Nhập tọa độ y: '))  
 return x, y  
  
def distance(x1, y1, x2, y2):  
 return sqrt((x1 - x2) \*\* 2 + (y1 - y2) \*\* 2)**

**File counter.py**

**from point import distance  
def checkMax(x1, y1, x2, y2, x3, y3):  
 d1 = distance(x1, y1, 0, 0)  
 d2 = distance(x2, y2, 0, 0)  
 d3 = distance(x3, y3, 0, 0)  
 m = d1  
 if d2 > m: m = d2  
 if d3 > m: m = d3  
 if m == d1: return "Điểm xa O nhất: A"  
 if m == d2: return "Điểm xa O nhất: B"  
 return "Điểm xa O nhất: C"  
  
def checkMin(x1, y1, x2, y2, x3, y3):  
 d1 = distance(x1, y1, 0, 0)  
 d2 = distance(x2, y2, 0, 0)  
 d3 = distance(x3, y3, 0, 0)  
 m = d1  
 if d2 < m: m = d2  
 if d3 < m: m = d3  
 if m == d1: return "Điểm gần O nhất: A"  
 if m == d2: return "Điểm gần O nhất: B"  
 return "Điểm gần O nhất: C"  
  
def arcage(x1, y1, x2, y2, x3, y3):  
 d1 = distance(x1, y1, x2, y2)  
 d2 = distance(x2, y2, x3, y3)  
 d3 = distance(x3, y3, x1, y1)  
 if d1 + d2 > d3 and d2 + d3 > d1 and d3 + d1 > d2:  
 return d1 + d2 + d3  
 return 0**

**File main.py**

**import counter  
import point  
  
print("Nhập tọa độ điểm A:")  
x1, y1 = point.getPoint()  
print("Nhập tọa độ điểm B:")  
x2, y2 = point.getPoint()  
print("Nhập tọa độ điểm C:")  
x3, y3 = point.getPoint()  
  
k = counter.checkMin(x1, y1, x2, y2, x3, y3)  
print(k)  
q = counter.checkMax(x1, y1, x2, y2, x3, y3)  
print(q)  
print("Diện tích tam giác ABC: ", counter.arcage(x1, y1, x2, y2, x3, y3))**

**BÀI 2.5. Vector**

**Module 1: cal.py (đặt trong MyPackage.calculation)**

**from math import sqrt  
  
def add(x1, x2, x3, y1, y2, y3):  
 return x1+y1, x2+y2, x3 + y3  
  
def minus(x1, x2, x3, y1, y2, y3):  
 return x1-y1, x2-y2, x3-y3  
  
def ox(x1, x2, x3):  
 return sqrt((x1\*\*2 + x2\*\*2 + x3\*\*2))  
  
def invert(x1, x2, x3):  
 return -x1, -x2, -x3**

**Module 2: dis.py (đặt trong MyPackage.display)**

**def vecdisplay(x1, x2, x3, name):  
 print(name, "(", x1, ",", x2, ",", x3,")")  
  
def getvector(name):  
 a = float(input(name + "1: "))  
 b = float(input(name + "2: "))  
 c = float(input(name + "3: "))  
 return a, b, c**

**Chương trình chính: main.py**

**from Mypackage.calculation import cal  
from Mypackage.display import dis  
#Nhập dữ liệu: là hai véc tơ x và y  
x1, x2, x3 = dis.getvector("X")  
y1, y2, y3 = dis.getvector("Y")  
  
#Tính tổng và hiệu hai véc tơ  
z1, z2, z3 = cal.add(x1, x2, x3, y1, y2, y3)  
t1, t2, t3 = cal.minus(x1, x2, x3, y1, y2, y3)  
# Hiển thị hai véc tơ ban đầu và hai véc tơ kết quả  
dis.vecdisplay(x1, x2, x3, "X")  
dis.vecdisplay(y1, y2, y3, "Y")  
dis.vecdisplay(z1, z2, z3, "Z")  
dis.vecdisplay(t1, t2, t3, "T")  
#Tính khoảng cách từ tâm O  
print("OZ = ", cal.ox(z1, z2, z3))  
print("OT = ", cal.ox(t1, t2, t3))  
  
#Lấy đối xứng qua gốc tọa độ  
z1, z2, z3 = cal.invert(z1, z2, z3)  
t1, t2, t3 = cal.invert(t1, t2, t3)  
#Hiển thị kết quả lấy đối xứng  
dis.vecdisplay(z1, z2, z3, "-Z")  
dis.vecdisplay(t1, t2, t3, "-T")**

**BÀI 3.1. Vector Lib**

**Module myvector.py**

**def vecinput():  
 n = int(input("n="))  
 a = [0]\*n  
 for i in range(n):  
 a[i] = int(input('a[{}] = '.format(i)))  
 return a**

**#cách 2 dùng append**

**def vecinput():  
 n = int(input("n="))  
 a = []  
 for i in range(n):  
 a.append(int(input('a[{}] = '.format(i))))  
 return a**

**#Dùng để nhập nhiều mảng khác nhau: thêm biến name  
def** vecinput(name):  
 n = int(input(**"Kích thước:"**))  
 a = [0]\*n  
 **for** i **in** range(n):  
 a[i] = int(input(**'{}[{}] = '**.format(name, i)))  
 **return** a

**#chú ý lúc gọi, ví dụ: a = vecinput(‘a’),**

**#b = vecinput(‘b’)**

**def vecinsert(a, p, v):  
 if p < 0 or p > len(a):  
 print("Position error !")  
 else:  
 a.insert(p, v)  
 return a  
  
def vecdel(a, v):  
 if a.count(v) > 0:  
 a.remove(v) #chỉ xóa được 1 phần tử**

**#chú ý: nếu xóa hết: thay if bằng while**

**return a**

**def vecsum(a):  
 total = 0.0  
 for i in a[:]:  
 total += i  
 return total  
  
def vecadd(a, b):  
 res = []  
 if len(a) == len(b):  
 for i in range(len(a)):  
 res.append(a[i] + b[i])  
 return res**

**File Main.py**

**from myvector import \***

**a = vecinput()  
print(vecsum(a))  
a = vecinsert(a, 2, 100)  
print(a)  
a = vecdel(a, 3)  
print(a)  
b = vecinput()  
c = vecadd(a, b)  
print(c)**

**BÀI 3.2. Matrix from list**

**def listinput():  
 n = int(input("n="))  
 a = [0]\*n  
 for i in range(n):  
 a[i] = int(input('a[{}] = '.format(i)))  
 return a  
  
def matrixfromlist(a):  
 n = int(input("n="))  
 m = int(input("m="))  
 b = []  
 if n\*m > len(a):  
 print('Can not create matrix !')  
 else:  
 for i in range(n): *#có n dòng trong ma trận* k = a[i\*m:i\*m + m]   
 b.append(k) *#thêm dòng k vào ma trận b*  
 return b**

***#chương trình chính***

**a = listinput()  
print(a)  
b = matrixfromlist(a)  
print("matrix", b)**

**BÀI 3.3. Merge lists**

**def listinput():  
 n = int(input("n="))  
 a = [0]\*n  
 for i in range(n):  
 a[i] = int(input('a[{}] = '.format(i)))  
 return a  
  
def mergelist(a, b):  
 c = []  
 if len(a) <= len(b):  
 for i in range(len(a)):  
 c.append(a[i])  
 c.append(b[i])  
 for t in range(i+1, len(b)):  
 c.append(b[t])  
 else:  
 for i in range(len(b)):  
 c.append(a[i])  
 c.append(b[i])  
 for t in range(i+1, len(a)):  
 c.append(a[t])  
 return c**

**#Chú ý: Sinh viên có thể cải tiến hàm mergelist**

**a = listinput()  
b = listinput()  
c = mergelist(a, b)  
print(c)**

**BÀI 3.4. Merge sorted list**

**listinput():  
 n = int(input("n="))  
 a = [0]\*n**def  **for i in range(n):  
 a[i] = int(input('a[{}] = '.format(i)))  
 return a  
  
def mergesortedlist(a, b):  
 c = []  
 i = 0 #i chạy trên a  
 j = 0 #j chạy trên b  
 while i < len(a) and j < len(b):  
 if a[i] < b[j]:  
 c.append(a[i])  
 i = i + 1  
 else:  
 c.append(b[j])  
 j = j + 1**

**if i < len(a):   
 for t in range(i, len(a)):  
 c.append(a[t])  
 if j < len(b):  
 for t in range(j, len(b)):  
 c.append(b[t])  
 return c  
  
a = listinput() #chú ý: a cần được sắp tăng  
b = listinput() #chú ý: b cần được sắp tăng  
c = mergesortedlist(a, b)  
print(c)**

**BÀI 3.5. Tuple from list (mảng các xâu)**

**def strlistinput():  
 n = int(input("n="))  
 a = ['0']\*n  
 for i in range(n):  
 a[i] = input('a[{}] = '.format(i))  
 return a  
  
def tuplefromlist(a):  
 b = tuple(a)  
 return b  
  
def numcount(a):  
 d = 0  
 for x in a[:]:  
 if x.isnumeric():  
 d += 1  
 return d  
  
  
a = strlistinput()  
b = tuplefromlist(a)  
print(b)  
print(type(b))  
print("Numerical form = ", numcount(b))**

**BÀI 3.5. Tuple from list (mảng các từ)**

**def strtotuple**():#*nhập một xâu và trả về 1 tuple* a = tuple(input(**'Nhập một xâu: '**).split())  
 **return** a

**def numcount**(a):#*Đếm số phần tử dạng số*  
 d = 0  
 **for** x **in** a[:]:  
 **if** x.isnumeric():  
 d += 1  
 **return** d  
  
a = **strtotuple**()  
print(a)  
print(type(a))  
print(**"Numerical form = "**, **numcount**(a))

**BÀI 3.6. Tuple from numbers**

**a = ('123', '456', '789', '012')  
b = tuple(float(x) for x in a[:])  
print(b)  
TOTAL = 0  
for x in b[:]:  
 TOTAL += x  
print("AVERAGE: ", TOTAL/len(b))**

**BÀI 3.7. Set**

**a = {2020601001, 2020601002, 2020601003, 2020601004}  
b = {2020601003, 2020601004, 2020601005, 2020601006}  
print(a)  
print(b)  
  
print('Sinh viên đăng ký tại cả hai bàn:')  
print(a & b)  
print('Danh sách tổng hợp:')  
print(a | b)  
print('Danh sách sinh viên chỉ đăng ký bàn 1:')  
print(a - b)**

**BÀI 3.8. Dictionary**

**def dictcount(d):  
 count = 0  
 for x in d.values():  
 if 2.5 <= float(x) <= 3.5:  
 count += 1  
 return count  
  
  
*#Xóa mục thỏa mãn điều kiện: Trích ra các item không thỏa mãn điều kiện*  
def dictremove(d):  
 R = {}  
 for k in d.keys():  
 if d[k] >= 2.0:  
 R[k] = d[k]  
 return R  
  
  
a = {2020601001: 2.9,**

**2020601002: 2.5,**

**2020601003: 2.4,**

**2020601004: 1.5}**

**print(a)  
print("Số điểm [2.5, 3.5] = ", dictcount(a))  
x = int(input('Nhập mã sinh viên mới: '))  
y = float(input("Nhập điểm của sinh viên: "))  
a[x] = y  
print("Sau khi bổ sung: ", a)  
a = dictremove(a)  
print("Sau khi xóa: ", a)**

**BÀI 3.9. String**

**S = input("Nhập một câu: ")  
Q = 'ha'  
if S.find(Q) >= 0:  
 print("Xâu '", S, "' có chứa '", Q, "'")  
else:  
 print("Xâu '", S, "' không chứa '", Q, "'")  
  
P = S + Q  
print("Xâu ghép: ", P)  
P = P.replace('Ha', 'Ba')  
print("Xâu thay thế: ", P)  
  
L = P.split()  
D = {}  
for i in range(len(L)):  
 D[i] = L[i]  
print(D)**

**BÀI 3.10. Skill**

**def dictcreate():  
 D = {'n': 1500,  
 'CLUSTERS': 3,  
 'ITER': 1000,  
 'METHOD': 'DCA CLUSTERING',  
 'MEASURE': 'EUCLIDEAN',  
 'YEARS': 9,  
 'MAX': 200}  
 return D  
  
def createset(D):  
 s = set()  
 for val in D.values():  
 s.add(val)  
 return s  
  
def createlist(D):  
 l = []  
 for val in D.values():  
 l.append(val)  
 return l  
  
D = dictcreate()  
print("Dictionary ban đầu: ", D)  
D['MEASURE'] = 'MANHATAN'  
print("Thông số METHOD = ", D['METHOD'])  
D['LOSS FUNCTION'] = 'SOFT MAX'  
print('Sau khi bổ sung: ', D)  
  
del D['YEARS']  
print('Sau khi xóa: ', D)  
  
S = input('Nhập một khóa S: ')  
if S in D.keys():  
 print('Key ', S, 'đã tồn tại')  
else:  
 print('Key ', S, 'chưa có')  
  
s = createset(D)  
print('SET: ', s)  
  
l = createlist(D)  
print('LIST: ', l)**

**BÀI 4.1. File export (export data to file)**

**def matrixinput():  
 n = int(input("n="))  
 m = int(input("m="))  
 a = []  
 for i in range(n):  
 k = [0]\*m  
 for j in range(m):  
 k[j] = float(input('a[{}][{}] = '.format(i, j)))  
 a.append(k)  
 return a  
  
  
def writefile(a):  
 f = open('D:/MATRIX.txt', mode='w')  
 f.write(str(len(a)) + ' ') #ghi số dòng  
 f.write(str(len(a[0])) + '\n') #ghi số cột  
  
 for i in range(len(a)):  
 for j in range(len(a[0])):  
 f.write(str(a[i][j]) + ' ')  
 f.write('\n')  
 f.close()**

**a = matrixinput()  
print(a)  
writefile(a)**

**BÀI 4.2. Reading**

**def readtoscreen(filename):  
 with open(filename, mode='r', encoding='utf-8') as f:  
 n, m = f.readline().split()  
 print(n, m)  
 n = int(n)  
 for i in range(n):  
 print(f.readline(), end = '')**

**def readtovariable(filename):  
 a = []  
 n = m = 0  
 with open(filename, mode='r', encoding='utf-8') as f:  
 n, m = f.readline().split()  
 n = int(n)  
 m = int(m)   
 for i in range(n):  
 k = list(map(float, f.readline().split()))  
 a.append(k)  
 return a, n, m**

**readtoscreen('D:/MATRIX.txt')**

**a, n, m = readtovariable('D:/MATRIX.txt')  
print(n)  
print(m)  
print(a)**

**BÀI 4.3. Real set**

**def readtovariable(filename):  
 a = []  
 n = m = 0  
 with open(filename, mode='r', encoding='utf-8') as f:  
 n, m = f.readline().split()  
 n = int(n)  
 m = int(m)  
 for i in range(n):  
 k = f.readline().split()  
 for i in range(len(k)):  
 k[i] = float(k[i])  
 a.append(k)  
 return a, n, m**

**def columnavg(a, n, m, i):  
 if i >= m:  
 print('Data with ', m, 'columns !')  
 return  
 else:  
 total = 0.0  
 for j in range(n):  
 total += a[j][i]  
 return total/n**

**a, n, m = readtovariable('D:/iris.txt')  
print(n)  
print(m)  
print(a)**

**col = int(input("Column = "))  
k = columnavg(a, n, m, col)  
print("AVG = ", k)**

**BÀI 4.4. Directory**

**import os  
import shutil as st  
  
os.mkdir("D:/BAI4")  
st.copy('D:/iris.txt', 'D:/BAI4')  
os.rename('D:/BAI4/iris.txt', 'D:/BAI4/iris.dat')  
#chuyển tới thư mục BAI4  
os.chdir('D:/BAI4')  
#show nội dung thư mục  
files = os.listdir()  
for file in files:  
 print(file)  
#xóa file  
os.remove('iris.dat')  
#xóa thư mục rỗng  
os.chdir('D:/')  
st.rmtree('BAI4')**

**BÀI 4.5. Skill**

**def readtoscreen(filename):  
 with open(filename, mode='r', encoding='utf-8') as f:  
 n, m = f.readline().split()  
 print(n, m)  
 n = int(n)  
 for i in range(n):  
 print(f.readline(), end = '')  
  
  
def readtovariable(filename):  
 a = []  
 n = m = 0  
 with open(filename, mode='r', encoding='utf-8') as f:  
 n, m = f.readline().split()  
 n = int(n)  
 m = int(m)  
 a = []  
 for i in range(n):  
 k = f.readline().split()  
 for i in range(len(k)):  
 k[i] = float(k[i])  
 a.append(k)  
 return a, n, m  
  
  
def countzero(a):  
 count = 0  
 for i in range(len(a)):  
 for j in range(len(a[0])):  
 if a[i][j] == 0:  
 count += 1  
 return count**

**def columnavg(a, col):  
 total = 0.0  
 for i in range(len(a)):  
 total += a[i][col]  
 return total/len(a)  
  
  
def replace(a):  
 for col in range(len(a[0])):  
 t = columnavg(a, col)  
 for row in range(len(a)):  
 if a[row][col] == 0:  
 a[row][col] = t  
 return a**

**def create2file(a, filename1, filename2):  
 with open(filename1, mode='w', encoding='utf-8') as f1:  
 f1.write(str(100) + ' ')  
 f1.write(str(len(a[0])) + '\n')  
 for i in range(100):  
 for j in range(len(a[0])):  
 f1.write(str(a[i][j]) + ' ')  
 f1.write('\n')  
 with open(filename2, mode='w', encoding='utf-8') as f2:  
 f2.write(str(len(a)-100) + ' ')  
 f2.write(str(len(a[0])) + '\n')  
 for i in range(101, len(a)):  
 for j in range(len(a[0])):  
 f2.write(str(a[i][j]) + ' ')  
 f2.write('\n')**

**readtoscreen('D:/image.data')  
a, n, m = readtovariable('D:/image.data')  
print(n)  
print(m)  
print(a)  
print('Số phần tử 0: ', countzero(a))  
a = replace(a)  
print(a)  
create2file(a, 'D:/image1.data', 'D:/image2.data')**

**import os  
import shutil  
os.mkdir('D:/BAI45')  
shutil.copy('D:/image1.data', 'D:/BAI45')  
shutil.copy('D:/image2.data', 'D:/BAI45')  
os.remove('D:/image1.data')  
os.remove('D:/image2.data')**

**BÀI 5.1. Array creating**

**import numpy as np**

**#1  
n = int(input("n="))  
a = np.random.rand(n)  
print("a = ", a)  
print('Số chiều của a: ', a.ndim)  
print('Kích thước mỗi chiều: ', a.shape)  
print('Độ dài của a: ', len(a))  
print('Kích thước 1 phần tử: ', a.itemsize)  
print('Kiểu của các phần tử: ', a.dtype)**

**#2  
b = np.linspace(1, n, 100)  
print('Mảng b:\n', b)  
print('Số chiều của b: ', b.ndim)  
print('Kích thước mỗi chiều: ', b.shape)  
print('Độ dài của b: ', len(b))  
print('Kích thước 1 phần tử: ', b.itemsize)  
print('Kiểu của các phần tử: ', b.dtype)  
#3  
c = np.arange(2, 201, 2)  
print('Mảng c:\n', c)  
#4  
d = np.ones(100)  
print('Mảng d:\n', d)  
#5  
e = np.zeros(100)  
print('Mảng e:\n', d)  
#6  
h = np.random.randn(100)  
print('Mảng h:\n', h)  
#7  
m = int(input('m='))  
k = np.ones((n, m))  
print('Mảng k:\n', k)  
#8  
p = np.eye(n)  
print('Mảng p:\n', p)  
#9  
q = np.diag(a)  
print('Mảng q:\n', q)**

**BÀI 5.2. Simple operations**

**import numpy as np**

**#1 NHẬP MẢNG  
def vecinput(n, name):  
 print("Nhập mảng ", name)  
 a = np.random.rand(n)  
 for i in range(n):  
 a[i] = float(input(name + "[{}] = ".format(i)))  
 return a  
  
n = int(input("n="))  
a = vecinput(n, 'a')  
b = vecinput(n, 'b')**

**#2, 3 TÍNH TOÁN SỐ HỌC ĐƠN GIẢN  
print('Mảng a: ', a)**

**print('Mảng b: ', b)  
c = a + b  
d = a - b  
e = a \* b  
f = a / b  
print('Mảng c: ', c)  
print('Mảng d: ', d)  
print('Mảng e: ', e)  
print('Mảng f: ', f)**

**#4 SỬ DỤNG CÁC HÀM/PHƯƠNG THỨC TRÊN MẢNG  
print('Tổng của c: ', c.sum())  
print('Max của c : ', c.max())  
print('Min của c : ', c.min())**

**#5 SLICING TRÊN MẢNG  
k = c[::2]  
print('Các phần tử có chỉ số chẵn trong c: ', k)  
print('Tổng = ', k.sum())**

**#6 RESHAPE TỪ 1D THÀNH 2D  
try:  
 t = int(input('Nhập số dòng của ma trận: '))  
 if n % t != 0: raise ValueError** *#nếu không thể reshape* **k = c.reshape(t, n//t)  
 print('Ma trận thu được: \n', k)  
except:  
 print(ValueError, ': Không thể reshape !')**

**BÀI 5.3. Calculations**

**import numpy as np**

**#1 HÀM NHẬP MẢNG  
def vecinput(n, name):  
 print("Nhập mảng ", name)  
 a = np.random.rand(n)  
 for i in range(n):  
 a[i] = float(input(name + "[{}] = ".format(i)))  
 return a**

**#2 HÀM RESHAPE TỪ 1D SANG 2D  
def to2D(a, n):  
 try:  
 t = int(input('Nhập số dòng của ma trận: '))  
 if n % t != 0: raise ValueError  
 k = a.reshape(t, n//t)  
 return k  
 except:  
 print(ValueError, ': Không thể reshape !')**

**return  
  
n = int(input("n="))  
a = vecinput(n, 'a')  
a = to2D(a, n)**

**#3**

**if a != ‘None’:**

**print('Mảng a:\n', a)  
  
if len(a[0]) > 1: #Nếu có từ 2 cột trở lên**

**b = a[:, 0]  
 c = a[:, 1]  
 b = np.reshape(b, -1)  
 c = np.reshape(c, -1)  
  
 print('Cột thứ nhất b: ', b)  
 print('Cột thứ hai c: ', c)  
  
 d = np.concatenate((b, c))  
 print('Mảng d: ', d)  
  
 k = np.where(d > 1)  
 print('Vị trí các phần tử lớn hơn 1 trong c: ', k[0])**

**d = np.sort(d, kind='heapsort')   
 print('c đã sắp: ', d)**

**k = int(input('Nhập vào giá trị cần chèn:'))  
vt = np.searchsorted(d, k)  
print('Vị trí thích hợp để chèn: ', vt)  
  
d = np.insert(d, vt, k)  
print('Mảng d sau khi chèn: ', d)**

**BÀI 5.4. Data Preparation**

**Chú ý: Đây là bài thực hành tổng hợp, sinh viên cần đầu tư thời gian để làm trước ở nhà (hoặc làm lại sau khi đi thực hành).**

**import numpy as np  
  
#1  
def readfile(name):  
 with open(name, mode='r', encoding='utf-8') as f:  
 n, m = f.readline().split()  
 n = int(n)  
 m = int(m)  
 a = []  
 for i in range(n):  
 k = f.readline().split()  
 a.append(k)  
 a = np.array(a)  
 return a, n, m  
  
#2  
def arrsplit(a, n, m):  
 x = a[:, :m-1]  
 y = a[:, m-1]  
 y = np.reshape(y, -1)  
 y = y.astype(int)  
 return x, y  
  
#3  
def countval(y):  
 k = np.bincount(y)  
 d = np.count\_nonzero(k)  
 print('Số phần tử khác nhau: ', d)  
 for i in range(len(k)):  
 if k[i] != 0:  
 print('Phần tử ', i, ' xuất hiện ', k[i], ' lần.')  
  
  
#4  
def columnaverage(x):  
 b = np.zeros(len(x[0]))  
  
 for i in range(len(x[0])):  
 k = x[:, i]  
 k = np.reshape(k, -1)**

**d = 0  
 for t in range(len(k)):  
 if k[t] != '?':  
 b[i] += float(k[t])**

**else:**

**d = d + 1  
 b[i] /= (len(x) – d)   
 return b  
  
#5  
def replacemissingvalue(x, tb):  
 n = len(x)  
 m = len(x[0])  
 for j in range(m):  
 for i in range(n):  
 if x[i, j] == '?':  
 x[i, j] = str(tb[j])  
 x = x.astype(float)  
 return x  
  
#6  
def writefile(x, y, filename1, filename2):  
 with open(filename1, mode='w', encoding='utf-8') as f1:  
 for i in range(len(x)):  
 for j in range(len(x[0])):  
 f1.write(str(x[i, j]) + ' ')  
 f1.write('\n')  
 with open(filename2, mode='w', encoding='utf-8') as f2:  
 for i in range(y.size):  
 f2.write(str(y[i]) + '\n')  
  
#7  
def traintestsplit(x, y, tyle):  
 n = len(x) # n= số dòng (150 dòng)  
 n\_train = int(n\*tyle) # n\_train: số dòng của tệp train  
 n\_test = n - n\_train # n\_test: số dòng của tệp test  
 x\_test = []  
 y\_test = []  
 for i in range(n\_test):  
 pos = np.random.randint(0, n-1)#lấy 1 vị trí ngẫu nhiên (pos)  
 k = list(x[pos, :]) #Lấy ra bản ghi tại vị trí pos  
 x\_test.append(k) #Đưa bản ghi vào k  
 y\_test.append(y[pos]) #Lấy lớp tương ứng  
 x = np.delete(x, pos, 0) #Xóa bản ghi khỏi x  
 y = np.delete(y, pos, 0) #Xóa cả trên mảng y   
 n = len(x) #cập nhật lại kích thước của x  
 x\_train = np.array(x)  
 y\_train = np.array(y)  
 x\_test = np.array(x\_test)  
 y\_test = np.array(y\_test)  
 print('Đã phân chia xong !')  
 return x\_train, y\_train, x\_test, y\_test  
  
#1  
a, n, m = readfile('D:/DATA54.txt')**

**#2  
x, y = arrsplit(a, n, m)  
print('Mảng x:\n', x)  
print('Mảng y:\n', y)**

**#3  
countval(y)**

**#4  
tb = columnaverage(x)  
print('Trung bình các cột: ', tb)**

**#5  
x = replacemissingvalue(x, tb)  
print(x)**

**#6  
writefile(x, y, 'D:/x.dat', 'D:/y.dat')  
#7  
x\_train, y\_train, x\_test, y\_test = traintestsplit(x, y, 0.7)  
writefile(x\_train, y\_train, 'D:/x\_train.txt', 'D:/y\_train.txt')  
writefile(x\_test, y\_test, 'D:/x\_test.txt', 'D:/y\_test.txt')  
#8  
print('Kích thước của x\_train:', x\_train.shape)  
print('Kích thước của x\_test :', x\_test.shape)  
print('Kích thước của y\_train:', y\_train.shape)  
print('Kích thước của y\_test :', y\_test.shape)  
#9  
x = np.reshape(x, -1)  
print('X đã chuyển về 1 chiều: ', x)  
print('Số phần tử của x: ', x.size)**

**BÀI 6.1. Simple chart**

**import matplotlib.pyplot as mp  
import numpy as np  
x = np.array([1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5])  
y = np.array([20, 30, 10, 40, 20, 30, 20, 10, 40, 50])  
  
mp.subplot(1,2,1)  
mp.plot(x, y)  
x = np.linspace(-10, 10, 100)  
y = 3\*x\*x\*x - 3\*x\*x + 3\*x - 3  
mp.subplot(1,2,2)  
mp.plot(x, y)  
mp.show()**

**BÀI 6.2. Chart decor**

**import matplotlib.pyplot as mp  
import numpy as np  
mp.figure(facecolor='gray')  
mp.gcf().canvas.set\_window\_title('BÀI THỰC HÀNH SỐ 6')**

**x = np.linspace(-10,10, 100)  
y = np.sin(x)  
font = {'family':'consolas','color':'blue','size':14}  
mp.title("ĐỒ THỊ Y = SIN(X)")  
mp.xlabel("Trục X", loc='right', fontdict=font)  
mp.ylabel("Trục Y", loc='top', fontdict=font)**

**mp.plot(x, y, marker='\*', ms=10, mec='r', mfc='y')**

**mp.savefig('D:/FIG.png')  
mp.show()**

**BÀI 6.3. Scatter chart**

**import matplotlib.pyplot as mp  
import numpy as np  
with open('D:/DATA63.txt') as f:  
 x = np.array(f.readline().split()).astype(float)**

**y = np.array(f.readline().split()).astype(float)**

**mp.title('SCATTER CHART')  
mp.xlabel("Trục x", loc='right')  
mp.ylabel("trục y", loc = 'top')  
mp.grid(axis='y', color='g', linestyle=':', linewidth=1.5)  
mp.scatter(x, y, color='r', s=y\*2)  
mp.show()**

**BÀI 6.4. Bar, Histogram, Pie, Subplot**

**import matplotlib.pyplot as mp  
import numpy as np**

**def readdata(filename):  
 f = open(filename, mode='r')  
 x = np.array(f.readline().split()).astype(float)**

**y = np.array(f.readline().split()).astype(float)**

**f.close()  
 return x, y**

**x, y = readdata('D:/DATA63.txt')  
mp.suptitle('BAR-HIST-PIE CHART')**

**mp.subplot(1, 3, 1)  
mp.title('BAR CHART')**

**mp.grid(axis='y', color ='g', linestyle =':', linewidth =1.5)  
mp.bar(x, y, color='r')**

**mp.subplot(1, 3, 2)  
mp.title('HISTOGRAM CHART')**

**mp.grid(axis='y', color ='g', linestyle =':', linewidth =1.5)  
mp.hist(y, color='b')  
  
mp.subplot(1, 3, 3)  
mp.title('PIE CHART')  
mp.pie(y)**

**mp.show()**

**BÀI 6.5. Pie chart**

**import matplotlib.pyplot as mp  
import numpy as np**

**label = ['Nhãn', 'Xoài', 'Dứa', 'Chuối', 'Cam', 'Bưởi']  
data = np.array([5800, 3200, 1200, 1700, 2400, 980])  
explode = [0.2, 0, 0.3, 0.2, 0.1, 0]  
color = ( "orange", "cyan", "brown", "grey", "indigo", "beige")  
wp = { 'linewidth' : 1, 'edgecolor' : "green" }#độ rộng và màu của viền**

**mp.title('BIỂU ĐỒ SẢN LƯỢNG 2022')  
mp.pie(data, labels=label, explode=explode, colors=color, wedgeprops=wp,  
 shadow = True, autopct = '%.2f')  
mp.show()**

**BÀI THU HOẠCH CUỐI KHÓA**

import matplotlib.pyplot as plt  
import numpy as np  
def read(filename):  
 try:  
 with open(filename, mode='r') as f:  
 x = []  
 while True:  
 line = f.readline()  
 if not line: break  
 x.append(line.split())  
 return np.array(x)  
 except:  
 print('File not found !')  
 return  
  
  
def distance(x1, x2):  
 T = 0.0  
 if len(x1) == len(x2):  
 for i in range(len(x1)):  
 T += (x1[i] - x2[i])\*\*2  
 return T  
 else:  
 print('Distance calculation error !')  
 return  
  
def distancetoall(x\_train, test):  
 dis = np.zeros(len(x\_train))  
 for i in range(len(x\_train)):  
 dis[i] = distance(x\_train[i], test)  
 return dis  
  
  
  
def predict(dis, label):  
 m = min(dis[np.where(dis > 0)])  
 minpos = np.where(dis == m)[0][0]  
 return label[minpos]  
  
#đọc dữ liệu từ hai tệp lên  
filename1 = 'D:/KDDCUP.data'  
filename2 = 'D:/KDDCUP.test'  
print('Đang đọc tệp', filename1, '.......', end='')  
x\_train = read(filename1)  
label = x\_train[:, len(x\_train[0])-1]  
x\_train = x\_train[:, 4:len(x\_train[0])-1]  
x\_train = x\_train.astype(float)  
print('Done !')  
print('\t',filename1, x\_train.shape)  
  
print('Đang đọc tệp', filename2, '.......', end='')  
x\_test = read(filename2)  
x\_test = x\_test[:, 4:len(x\_test[0])-1]  
x\_test = x\_test.astype(float)  
print('Done !')  
print('\t', filename2, x\_test.shape)  
  
  
pos = int(input('Nhập bản ghi muốn dự báo lớp: '))  
if pos < 0 or pos >= len(x\_test):  
 print('\tVị trí bản ghi không hợp lệ !')  
else:  
 print('\tVị trí bản ghi hợp lệ !')  
 print('Đang tính khoảng cách..............', end='')  
 test = x\_test[pos]  
 dis = distancetoall(x\_train, test)  
 print('Done !')  
  
 print('Lớp của dữ liệu: ', predict(dis, label))