**# H\_5\_1.py 功能:能夠計算出每一年的薪資所得、支出額、存款餘額**

# 設定參數

salary = 30000 #薪資

expenses = 19000 #支出額

age = 25 #年齡

age\_future = 28 #未來年齡

salary\_rate = 0.03 #薪資成長率

exp\_rate = 0.02 #支出額成長率

Deposit = 0 #存款餘額起始值為0

# 使用迴圈計算每一年的薪資所得、支出額、存款餘額

for n in range(age,age\_future,1):

FV\_salary = salary \* (1+salary\_rate)

FV\_expenses = expenses \* (1+exp\_rate)

Deposit = Deposit + (FV\_salary - FV\_expenses)\*12

salary = FV\_salary # 將調整過的薪資覆蓋先前的薪資

expenses = FV\_expenses # 將調整過的支出額覆蓋先前的支出額

# 將結果印出來

print('薪資所得 : %d' %(round(FV\_salary,0)))

print('支出額 : %d' %(round(FV\_expenses,0)))

print('存款餘額 : %d' %(round(Deposit,0)))

**# H\_5\_2.py 功能:計算購屋資金需要多久時間可以達成，使用while迴圈撰寫**

# 設定參數

PV = 10000 # 每個月投資金額

r = 0.12 # 投報率

n = 1 # 期數

FV = 0 # 設定終值起始為0

while FV < 3000000:

FV = (FV + PV) \* (1+(r/12))

n += 1

print('小芳 %d 年後就可以買到房子' %(round(n/12,0)))

**# H\_5\_3.py 功能:輸入兩個陣列，並且完成加減乘除後輸出顯示**

# 儲存在A的陣列

matrix1\_1 = []

matrix1\_2 = []

matrix1\_3 = []

# 儲存在B的陣列

matrix2\_1 = []

matrix2\_2 = []

matrix2\_3 = []

# 存放相加後的陣列

matrix3\_1 = []

matrix3\_2 = []

matrix3\_3 = []

# 存放相減後的陣列

matrix4\_1 = []

matrix4\_2 = []

matrix4\_3 = []

# 存放相乘後的陣列

matrix5\_1 = []

matrix5\_2 = []

matrix5\_3 = []

# 存放相除後的陣列

matrix6\_1 = []

matrix6\_2 = []

matrix6\_3 = []

# 陣列A輸入

for r in range(0,3,1):

n = int(input('請輸入A陣列的(0,%d) : ' %(r)))

matrix1\_1.insert(r,n)

for r in range(0,3,1):

n = int(input('請輸入A陣列的(1,%d) : ' %(r)))

matrix1\_2.insert(r,n)

for r in range(0,3,1):

n = int(input('請輸入A陣列的(2,%d) : ' %(r)))

matrix1\_3.insert(r,n)

# 陣列B輸入

for r in range(0,3,1):

n = int(input('請輸入B陣列的(0,%d) : ' %(r)))

matrix2\_1.insert(r,n)

for r in range(0,3,1):

n = int(input('請輸入B陣列的(1,%d) : ' %(r)))

matrix2\_2.insert(r,n)

for r in range(0,3,1):

n = int(input('請輸入B陣列的(2,%d) : ' %(r)))

matrix2\_3.insert(r,n)

# 計算相加的迴圈

for r in range(0,3,1):

matrix3\_1.insert(r,matrix1\_1[r] + matrix2\_1[r])

matrix3\_2.insert(r,matrix1\_2[r] + matrix2\_2[r])

matrix3\_3.insert(r,matrix1\_3[r] + matrix2\_3[r])

print('-----分隔線-----')

print(matrix3\_1)

print(matrix3\_2)

print(matrix3\_3)

# 計算相減的迴圈

for r in range(0,3,1):

matrix4\_1.insert(r,matrix1\_1[r] - matrix2\_1[r])

matrix4\_2.insert(r,matrix1\_2[r] - matrix2\_2[r])

matrix4\_3.insert(r,matrix1\_3[r] - matrix2\_3[r])

print('-----分隔線-----')

print(matrix4\_1)

print(matrix4\_2)

print(matrix4\_3)

# 計算相乘的迴圈

for r in range(0,3,1):

matrix5\_1.insert(r,matrix1\_1[r] \* matrix2\_1[r])

matrix5\_2.insert(r,matrix1\_2[r] \* matrix2\_2[r])

matrix5\_3.insert(r,matrix1\_3[r] \* matrix2\_3[r])

print('-----分隔線-----')

print(matrix5\_1)

print(matrix5\_2)

print(matrix5\_3)

# 計算相除的迴圈

for r in range(0,3,1):

matrix6\_1.insert(r,round(matrix1\_1[r] / matrix2\_1[r],2))

matrix6\_2.insert(r,round(matrix1\_2[r] / matrix2\_2[r],2))

matrix6\_3.insert(r,round(matrix1\_3[r] / matrix2\_3[r],2))

print('-----分隔線-----')

print(matrix6\_1)

print(matrix6\_2)

print(matrix6\_3)

**# H\_5\_4.py 功能:輸入數字後判斷是否為11的倍數**

num\_even = 0 # 儲存偶數位數字暫存

num\_odd = 0 # 儲存奇數位數字暫存

number = str(input('請輸入數字 : '))

l = len(number) # 判斷輸入數字之長度

x = int(number) # 轉換成數值型態

for n in range(l,0,-1):

y = x//(10\*\*(n-1)) # 計算奇偶位數字

x = x - (y\*(10\*\*(n-1)))

if n%2 == 0: #判斷若是偶數位數字則儲存在偶數位暫存，反之存奇數位暫存

num\_even = num\_even + y

else:

num\_odd = num\_odd + y

# 判斷是否為11的倍數

if abs(num\_even - num\_odd) == 0 or (abs(num\_even - num\_odd))%11 == 0:

print('此數為11的倍數')

else:

print('此數不是11的倍數')

**# H\_5\_5.py 功能:輸入頭和腳的數量，並判斷出有多少馴鹿及聖誕老人**

# 輸入頭及腳的數量

head = int(input('請輸入頭的數量 : '))

foot = int(input('請輸入腳的數量 : '))

# 計算馴鹿和聖誕老人的數量

reindeer = (foot/2) - head

Santa = head - reindeer

# 將結果顯示出來

print('聖誕老人有 : %d 位' %(Santa))

print('馴鹿有 : %d 隻' %(reindeer))

**# H\_5\_6.py 功能:輸入金額後，判斷水果有幾種不同的買法**

N = int(input('請輸入金額 : '))

if N >= 100 and N <= 1000: # 判斷金額是否有輸入錯誤

# N//5的部分是如果全部金額都買這項水果的最大值，當成是迴圈的末端

for x in range(5,N//5,5):

for y in range(3,N//6,3):

for z in range(2,N//10,2):

if (x\*5) + (y\*6) + (z\*10) == N :

print('x = %d y = %d z = %d' %(x,y,z))

#將最後一組的組合儲存起來，方便後續判斷是否沒有解

d = x

e = y

f = z

else:

print('輸入金額必須介在100~1000之間')

# 判斷如果沒有組合時，輸出 No Solution

if N - ((d\*5) + (e\*6) + (f\*10)) != 0 and N >= 100 and N <= 1000:

print('No Solution') N = int(input('請輸入金額 : '))

**# H\_5\_7.py 功能:輸入兩個人間的距離，並計算出何時會相遇**

# 輸入距離(公尺)

x = int(input('請輸入距離 : '))

# 轉換單位

y = 100 \* x # 將公尺轉換成公分

t = y / 176.2 # 每經過一秒兩人的差距就少了176.2cm，所以以相距的距離下去除以176.2便可得知多少秒之後可以相遇

if t < 60: # 判斷若在60秒內相遇的顯示方式

print('兩人在 %d 秒 會相遇' %(t))

elif t >= 60 and t < 3600 : # 判斷若超過分鐘單位但不滿小時的顯示方式

m = t//60

s = round(((t/60) - m),1)\*60

print('兩人在 %d 分 %d 秒 會相遇' %(m,s))

else: # 判斷超過小時的顯示方式

h = t//3600

s = ((t/3600) - h)\*3600

m = s//60

s1 = round(((s/60) - m),1)\*60

print('兩人在 %d 時 %d 分 %d 秒 會相遇' %(h,m,s1))

**# H\_5\_8.py 功能:輸入開始及結束的停車時間，計算出停車費用並顯示**

# H\_5\_8.py 功能:輸入開始及結束的停車時間，計算出停車費用並顯示

start = [] # 儲存開始的時間

end = [] #儲存結束的時間

start = str(input('開始停車時間 : '))

end = str(input('結束停車時間 : '))

# 將輸入字串的以"點"做分割

start = start.split("點")

end = end.split("點")

#區分早上、下午與小時數

start\_list = [start[0][:2],start[0][2:]]

end\_list = [end[0][:2],end[0][2:]]

#轉換為24小時制

if start\_list[0] == "下午":

start\_hour = int(start\_list[1])+12

else:

start\_hour = int(start\_list[1])

if end\_list[0] == "下午":

end\_hour = int(end\_list[1])+12

else:

end\_hour = int(end\_list[1])

#取出分鐘之數字，並轉換為整數

start\_min = int(start[1].replace("分",""))

end\_min = int(end[1].replace("分",""))

# 轉換成分鐘單位，並處理隔夜的狀況

if start\_hour > end\_hour:

minute = (end\_hour - start\_hour + 24) \* 60 + (end\_min - start\_min)

else:

minute = (end\_hour - start\_hour) \* 60 + (end\_min - start\_min)

# 判斷停車時間並計算費用

if minute <= 120 : # 兩小時以內

cost = (minute // 30) \* 30

elif minute >= 120 and minute <= 240 : # 兩小時以內但不超過四小時

cost = 120 + ((minute-120) // 30) \* 40

else : #超過四小時

cost = 120 + 160 + ((minute-240) // 30) \* 60

print('停車費用 : %d' %(cost))

**# H\_5\_9.py 功能:輸入西曆年，判斷是否為閏年**

# 輸入西曆年

year = int(input('請輸入西曆年 : '))

# 判斷是否為閏年

if year % 4 == 0 and year % 100 == 0 and year % 400 != 0 :

print('%d 年是閏年' %(year)) # 判斷結果為閏年，顯示出來

else:

print('%d 年不是閏年' %(year)) # 判斷結果不為閏年，顯示出來