# Python 程式設計作業

## 範圍： 函式的應用

## 銘傳大學電腦與通訊工程系

| | |
|---|---|
| 班　　級 | 電通四乙 |
| 姓　　名 | 陳昱叡 |
| 學　　號 | 04052474 |
| 作業成果 | 應繳作業共 7 題，前 3 題每題 20 分，後 4 題每題 10 分，滿分為 100 分<br>我共完成 7 題，應得 100 分 |
| 授課教師 | 陳慶逸 |

■ 請確實填寫自己寫完成題數，並且計算得分。填寫不實者(如上傳與作業明顯無關的答案，或是計算題數有誤者)，本次作業先扣 50 分。

■ 確實填妥封面的內容，完成後請上傳 pdf 檔。

**【立即練習】**：試實作一個函式 hyberCub()，當使用者所輸入的兩個參數為 h 和 w 時，程式會交錯使用「+」和「-」列印一個長寬分別為 h 和 w 的長方形。若是函式沒有輸入參數時，則函式會以預設值:h = 6, w = 7 來處理。

例如:

輸入 hyberCub(8, 3)，則回傳:

```
+-+-+-
+-+-+-
+-+-+-
```

輸入 hyberCub()，則回傳:

```
+-+-+-
+-+-+-
+-+-+-
+-+-+-
+-+-+-
+-+-+-
+-+-+-
```

**程式碼：**

```
def hyberCub(w='6',h='7'):
    w=int(w)
    h=int(h)
    for x in range(0,h):
        for y in range(0,w):
            if y%2==0:
                print("+",end="")
            else:
                print("-",end="")
        print("")
w=input("輸入w:")
h=input("輸入h:")
if w=="" and h=="":
    hyberCub()
else:
    hyberCub(w,h)
```

**執行結果擷圖：**

In [15]:

```
def hyberCub(w='6',h='7'):
    w=int(w)
    h=int(h)
    for x in range(0,h):
        for y in range(0,w):
            if y%2==0:
                print("+",end="")
            else:
                print("-",end="")
        print("")
w=input("輸入w:")
h=input("輸入h:")
if w=="" and h=="":
    hyberCub()
else:
    hyberCub(w,h)
```

```
輸入w:8
輸入h:3
+-+-+-+-
+-+-+-+-
+-+-+-+-
```

In [16]:
```
def hyberCub(w='6',h='7'):
    w=int(w)
    h=int(h)
    for x in range(0,h):
        for y in range(0,w):
            if y%2==0:
                print("+",end="")
            else:
                print("-",end="")
        print("")
w=input("輸入w:")
h=input("輸入h:")
if w=="" and h=="":
    hyberCub()
else:
    hyberCub(w,h)
```

```
輸入w:
輸入h:
+-+-+-
+-+-+-
+-+-+-
+-+-+-
+-+-+-
+-+-+-
+-+-+-
```

**【立即練習】**: 遞迴的目的在於分割問題，或將問題縮小以逐步降低問題的難度。一個問題如果可以拆解成規模較小但是性質和原問題完全一樣的問題的時候，就可以考慮用遞迴函式來處理。舉例來說，m * n 的乘法運算我們便可以將之寫成 m + m * (n – 1)，因為 m * (n – 1)要比 m * n 容易一點，在實作一個遞迴函式 multiply(m ,n)時，我們重點在於計算出 x = m * (n – 1)，因為 m + x 便是計算結果了。

例如:

輸入 multiply(6, 5)，則回傳 30

輸入 multiply(7, 9)，則回傳 63

```
def multiply(m,n):
    if n==1:
        return m
    return m+multiply(m,n-1)

m=int(input('輸入m:'))
n=int(input('輸入n:'))
print("輸入 multiply(",m,",",n,")")
print(multiply(m,n))
```

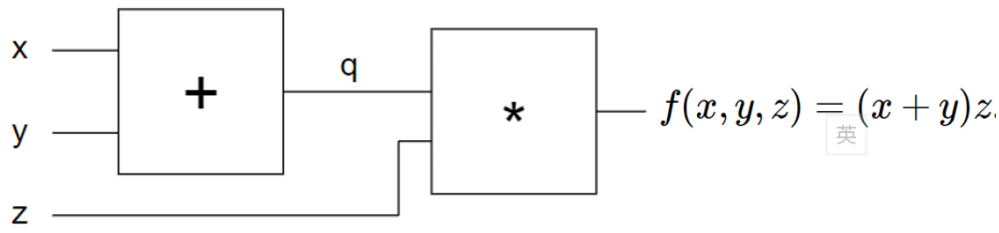
執行結果擷圖：

```
In [14]: def multiply(m,n):
             if n==1:
                 return m
             return m+multiply(m,n-1)

         m=int(input('輸入m:'))
         n=int(input('輸入n:'))
         print("輸入 multiply(",m,",",n,")")
         print(multiply(m,n))

         輸入m:6
         輸入n:5
         輸入 multiply( 6 , 5 )
         30
```

In [15]:
```
def multiply(m,n):
    if n==1:
        return m
    return m+multiply(m,n-1)

m=int(input('輸入m:'))
n=int(input('輸入n:'))
print("輸入 multiply(",m,",",n,")")
print(multiply(m,n))
```

```
輸入m:7
輸入n:9
輸入 multiply( 7 , 9 )
63
```

**【立即練習】**:請以 Lambda 函式來實現實數相加(forwardAddGate(x, y))以及實數相乘(forwardMultGate(x, y))兩個函式，整體電路(forwardCircuit(x,y,z))則以一般函式來撰寫:




例如:

輸入 forwardCircuit(-2, 5, -4)，則回傳-12

輸入 forwardCircuit(-4, 3, -5)，則回傳 5

```
forwardAddGate =lambda x,y:x+y
forwardMultGate =lambda q,z:q*z
def forwardCircuit(x, y, z):
    q=forwardAddGate(x,y)
    ans=forwardMultGate(q,z)
    return ans
x=int(input('輸入x:'))
y=int(input('輸入y:'))
z=int(input('輸入z:'))
print("輸入 forwardCircuit(",x,",",y,",",z,")")
print("回傳:",forwardCircuit(x,y,z))
```

執行結果擷圖：

In [15]:
```
forwardAddGate =lambda x,y:x+y
forwardMultGate =lambda q,z:q*z
def forwardCircuit(x, y, z):
    q=forwardAddGate(x,y)
    ans=forwardMultGate(q,z)
    return ans
x=int(input('輸入x:'))
y=int(input('輸入y:'))
z=int(input('輸入z:'))
print("輸入 forwardCircuit(",x,",",y,",",z,")")
print("回傳:",forwardCircuit(x,y,z))
```

```
輸入x:-2
輸入y:5
輸入z:-4
輸入 forwardCircuit( -2 , 5 , -4 )
回傳: -12
```

In [16]:
```
forwardAddGate =lambda x,y:x+y
forwardMultGate =lambda q,z:q*z
def forwardCircuit(x, y, z):
    q=forwardAddGate(x,y)
    ans=forwardMultGate(q,z)
    return ans
x=int(input('輸入x:'))
y=int(input('輸入y:'))
z=int(input('輸入z:'))
print("輸入 forwardCircuit(",x,",",y,",",z,")")
print("回傳:",forwardCircuit(x,y,z))
```

```
輸入x:-4
輸入y:3
輸入z:-5
輸入 forwardCircuit( -4 , 3 , -5 )
回傳: 5
```

EX 1: 試實作一個函式 max_mul()，當我們給定一個內容皆為整數的串列 l 時，函式的回傳值是串列中連續兩個相鄰元素乘積的最大值。

例如:

輸入 max_mul([2, -1, 7, 9, 23, 2])， 則輸出 207

輸入 max_mul([2, -1, 7, 9, 23, 25])， 則輸出 575

輸入 max_mul([-22, -18, 7, 9, 23, 2])， 則輸出 396

```
def max_mul(l):
    first=l[0]*l[1]
    for i in range(0,len(l)):
        if i+1<len(l):
            nextone=l[i]*l[i+1]
        if first<nextone:
            first=nextone
    return first
print("max_mul([2, -1, 7, 9, 23, 2])")
print("輸出:",max_mul([2, -1, 7, 9, 23, 2]))
print("max_mul([2, -1, 7, 9, 23, 25])")
print("輸出:",max_mul([2, -1, 7, 9, 23, 25]))
print("max_mul([-22, -18, 7, 9, 23, 2])")
print("輸出:",max_mul([-22, -18, 7, 9, 23, 2]))
```

執行結果擷圖：

In [9]:
```
def max_mul(l):
    first=l[0]*l[1]
    for i in range(0,len(l)):
        if i+1<len(l):
            nextone=l[i]*l[i+1]
        if first<nextone:
            first=nextone
    return first
print("max_mul([2, -1, 7, 9, 23, 2])")
print("輸出:",max_mul([2, -1, 7, 9, 23, 2]))
print("max_mul([2, -1, 7, 9, 23, 25])")
print("輸出:",max_mul([2, -1, 7, 9, 23, 25]))
print("max_mul([-22, -18, 7, 9, 23, 2])")
print("輸出:",max_mul([-22, -18, 7, 9, 23, 2]))
```

```
max_mul([2, -1, 7, 9, 23, 2])
輸出: 207
max_mul([2, -1, 7, 9, 23, 25])
輸出: 575
max_mul([-22, -18, 7, 9, 23, 2])
輸出: 396
```

EX 2: 試實作一個函式 longest_str()，當我們給定一個內容皆為字串(str)的串列 l 時，函式的回傳值是串列中長度最長的字串。若長度相同，則回傳 index 較小的字串。

例如:

輸入 longest_str(['apple', 'python', 'plasma']),則輸出'python'

輸入 longest_str(['apples', 'python', 'plasma']),則輸出'apples'

輸入 longest_str(['Jane', 'Peter', 'stephanie']), 則輸出'stephanie'

```
def longest_str(l):
    first=len(l[0])
    ayour_ans=l[0]
    for i in range(0,len(l)):
        if first<len(l[i]):
            first=len(l[i])
            ayour_ans=l[i]
    return ayour_ans
print("輸入longest_str(['apple', 'python', 'plasma'])")
print("輸出:",longest_str(['apple', 'python', 'plasma']))
print("輸入longest_str(['apples', 'python', 'plasma'])")
print("輸出:",longest_str(['apples', 'python', 'plasma']))
print("輸入longest_str(['Jane', 'Peter', 'stephanie'])")
print("輸出:",longest_str(['Jane', 'Peter', 'stephanie']))
```

執行結果擷圖：

```
In [5]: def longest_str(l):
            first=len(l[0])
            ayour_ans=l[0]
            for i in range(0,len(l)):
                if first<len(l[i]):
                    first=len(l[i])
                    ayour_ans=l[i]
            return ayour_ans
        print("輸入longest_str(['apple', 'python', 'plasma'])")
        print("輸出:",longest_str(['apple', 'python', 'plasma']))
        print("輸入longest_str(['apples', 'python', 'plasma'])")
        print("輸出:",longest_str(['apples', 'python', 'plasma']))
        print("輸入longest_str(['Jane', 'Peter', 'stephanie'])")
        print("輸出:",longest_str(['Jane', 'Peter', 'stephanie']))

        輸入longest_str(['apple', 'python', 'plasma'])
        輸出: python
        輸入longest_str(['apples', 'python', 'plasma'])
        輸出: apples
        輸入longest_str(['Jane', 'Peter', 'stephanie'])
        輸出: stephanie
```

EX 3: 試實作一個函式 score_threshold()，函式的輸入參數為整數所構成的串列 l 和門檻值 t；當我們給定一個內容皆為整數的串列 l,以及一個整數門檻值 t 時，函式的回傳值是一個判斷原始串列中每個值是否大於等於門檻 t 的結果(true 或是 false)。

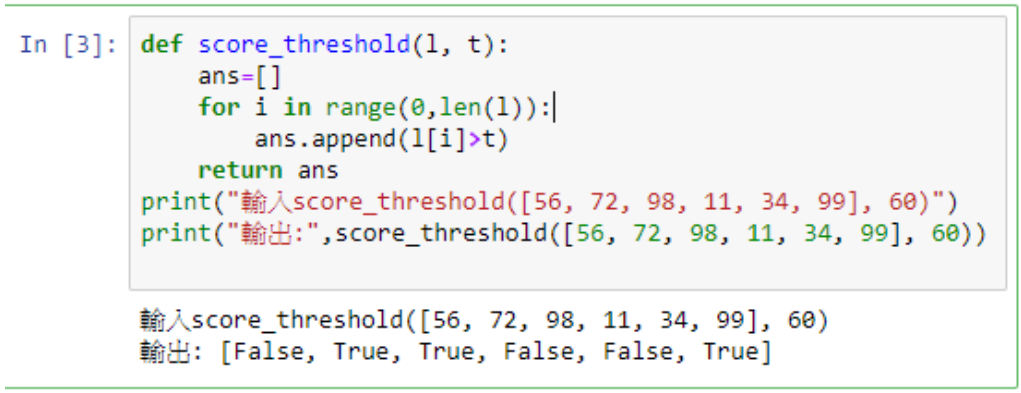
例如:

輸入 score_threshold([56, 72, 98, 11, 34, 99], 60)

輸出[False, True, True, False, False, True]

```
def score_threshold(l, t):
    ans=[]
    for i in range(0,len(l)):
        ans.append(l[i]>t)
    return ans
print("輸入score_threshold([56, 72, 98, 11, 34, 99], 60)")
print("輸出:",score_threshold([56, 72, 98, 11, 34, 99], 60))
```

**執行結果擷圖：**




EX 5: 試實作一個函式 Insert_str()，它能在輸入的括號符號中插入指定的字串。

例如:

輸入 insert_str('[[]]', 'Python')，輸出[[Python]]

輸入 insert_str('{{}}', 'C++')，輸出{{C++}}

輸入 insert_str('<<>>', 'Java')，輸出<<Java>>

```
def insert_str(str, word):
    return  str[0:2]+word+str[2:]
print("輸入insert_str('[[]]', 'Python')")
print("輸出:",insert_str('[[]]', 'Python'))
print('')
print("輸入insert_str('{{}}', 'C++')")
print("輸出:",insert_str('{{}}', 'C++'))
print('')
print("輸入insert_str('<<>>', 'Java')")
print("輸出:",insert_str('<<>>', 'Java'))
```

執行結果擷圖：

```
In [3]: def insert_str(str, word):
            return  str[0:2]+word+str[2:]
        print("輸入insert_str('[[]]', 'Python')")
        print("輸出:",insert_str('[[]]', 'Python'))
        print('')
        print("輸入insert_str('{{}}', 'C++')")
        print("輸出:",insert_str('{{}}', 'C++'))
        print('')
        print("輸入insert_str('<<>>', 'Java')")
        print("輸出:",insert_str('<<>>', 'Java'))

        輸入insert_str('[[]]', 'Python')
        輸出: [[Python]]

        輸入insert_str('{{}}', 'C++')
        輸出: {{C++}}

        輸入insert_str('<<>>', 'Java')
        輸出: <<Java>>
```