

# 計算機實習報告

41323117 41323115 41323114

沈宇揚 李昱揚 李俊穎

# W7:

首先給a一個變數，在創建一個for in range迴圈，最後會輸出兩個值，1~10還有重複的a變數。

## Brython

從 1 累加到 100:

1 add to 100 ROC Flag PROC flag USA Flag GD in Brython

```
1 a = "我會寫python程式了"  
2 for i in range(1, 10):  
3     print(i, a)
```

Filename: .py

```
1 我會寫python程式了  
2 我會寫python程式了  
3 我會寫python程式了  
4 我會寫python程式了  
5 我會寫python程式了  
6 我會寫python程式了  
7 我會寫python程式了  
8 我會寫python程式了  
9 我會寫python程式了  
<completed in 2.10  
ms>
```

# W10:

算是一個測驗:總共有三題，第一題是測驗我們是否會使用除了brython以外的程式執行軟體。

## w10

### 第一題



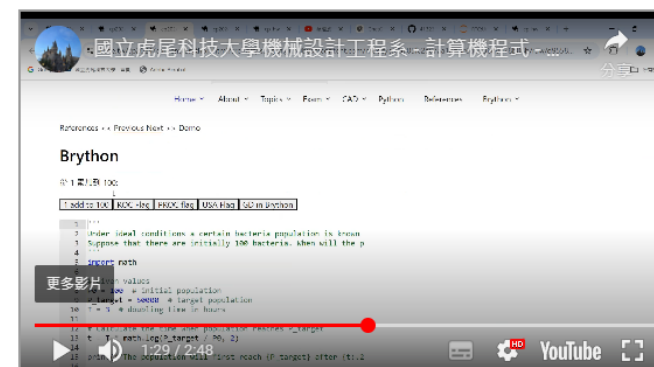
第二題:分辨正確和錯誤的命名方式，讓我們認識在各種執行程式軟體裡會跳出何種警訊。

### 第二題



第三題:真正讓數學進入程式，只要將需要的數值，或著電腦計算完後的數值，丟入公式裡就能得出答案。

第三題 心得:發現了一個非常好用的程式碼，只要將其中的數據換成自己需要的，就可以像一台專屬科目的計算機，非常方便。



# W11\_hw:

第一題:設定邊界>設定圓心位置>給予半徑一個浮點數的變量>檢查是否在邊界上(是:給\*字號, 否:空格)>是否換行>完成後結束

```
1 import math
2
3 # 設定字元區域的大小 (11x11)
4 width, height = 11, 11
5 # 設定圓心位置
6 center_x, center_y = width // 2, height // 2
7
8 # 提示使用者輸入圓的半徑大小
9 try:
10     radius = float(input("請輸入圓的半徑 (建議5以內, 過大可能無法繪圖): "))
11 except ValueError:
12     print("輸入錯誤, 將使用預設半徑 5")
13     radius = 5 # 預設半徑
14
15 # 遍歷整個字元區域
16 for y in range(height):
17     for x in range(width):
18         # 計算該位置到圓心的距離
19         distance = math.sqrt((x - center_x) ** 2 + (y - center_y) ** 2)
20         if distance <= radius:
21             print('*', end='')
```

Filename: .py

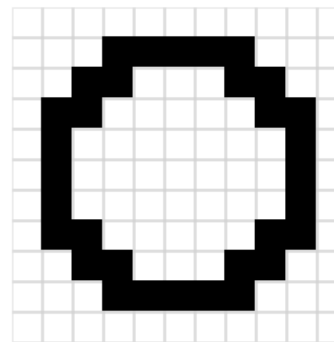
```
*****
*  *
*  *
*  *
*  *
*  *
*  *
*  *
*  *
*  *
*****
```

第二題:建立畫布(設定長寬)>設置參數>設定輸入框和按鈕>讀取數值>計算座標>填色

## Brython

從 1 累加到 100:

```
36 # 預設參數
37 center_x, center_y = 5, 5 # 圓心
38 radius = 4 # 預設圓的半徑
39 grid_width, grid_height = 11, 11 # 預設格子寬高
40
41 # 在 grid 中畫圓
42 def draw_circle():
43     ctx.clearRect(0, 0, canvas.width, canvas.height) # 清空
44     grid(grid_width, grid_height, gs) # 重新繪製格子
45     for i in range(grid_width):
46         for j in range(grid_height):
47             # 計算 (i,j) 到圓心的距離
48             dist = math.sqrt((i - center_x) ** 2 + (j - center_y) ** 2)
49             if abs(dist - radius) < 0.5: # 若該點在圓邊界,
50                 fill(i * gs, j * gs, "black")
51
52 # 建立輸入框和按鈕
53 input_width = html.INPUT(type="number", value=grid_width, min=1, max=100)
54 input_height = html.INPUT(type="number", value=grid_height, min=1, max=100)
55 input_radius = html.INPUT(type="number", value=radius, min=1, max=100)
56
```



# W12:

給n的變量為5，放入for in range的迴圈中，range(n) 生成的是一個序列，從0到n-1，共n個數字，所以會執行5次並且從0開始，如果想要從1開始，也可以更改為(1, n+1)

1 add to 100 ROC Flag PROC flag USA Flag GD in Brython

```
1 n = 5
2 space = "_"
3 ...
4 for i in range(n): # 對於每一行
5     # 印出空格
6     print(space * (n - i - 1), end=' ')
7     ...
8     ""
9 for i in range(n): # 對於每一行
10    # 印出空格
11    print(space * (n - i - 1), end=' ')
12    ""
13 for i in range(1, n+1): # 對於每一行
14    # 印出空格
15    #print(space * (n - i - 1), end=' ')
16    print("目前的數字為 " + str(i))
17
```

Filename: .py

```
目前的數字為 1
目前的數字為 2
目前的數字為 3
目前的數字為 4
目前的數字為 5
<completed in 3.00
ms>
```

## Brython

從 1 累加到 100:

1 add to 100 ROC Flag PROC flag USA Flag GD in Brython

```
1 n = 5
2 space = "_"
3 ...
4 for i in range(n): # 對於每一行
5     # 印出空格
6     print(space * (n - i - 1), end=' ')
7     ...
8     ""
9 for i in range(n): # 對於每一行
10    # 印出空格
11    print(space * (n - i - 1), end=' ')
12    ""
13 for i in range(n): # 對於每一行
14    # 印出空格
15    #print(space * (n - i - 1), end=' ')
16    print("目前的數字為 " + str(i))
17
```

Filename: .py

```
目前的數字為 0
目前的數字為 1
目前的數字為 2
目前的數字為 3
目前的數字為 4
<completed in 1.50
ms>
```

# W12\_hw:

## 第一題:

所運用到的程式語法為print，執行後會顯示你輸入在print後的字串

```
1 print (" /\_/\ ")
2 print (" >^.^< ")
3 print (" / \ ")
4 print ([" ( _ ) "])
```

Filename: .py

```
 /\_/\
>^.^<
 / \
( _ )
<completed in 2.90
ms>
```

## 第二題:

使用到的語法是input及print，執行後，他會出現提示框，讓你輸入自己想要的字串，再將字串放到設計好的對話當中。

從 1 累加到 100:

```
1 name = input("What's your name?")
2
3 favorite_food = input ("What is your favorite food?")
4
5 print ("Hi! My name is " + name)
6 print ("My favorite food is " + favorite_food)
7 print (name + "'s favorite food is " + favorite_food)
```

Filename: .py

```
Hi! My name is 沈宇揚
My favorite food is 蛋餅
沈宇揚's favorite food
is 蛋餅
<completed in
31518.80 ms>
```

## 第三題:

同樣是input和print，還有用到if，else的語法，若是有做出設計者預設的答案，就會做出不同的回應。

```
1 programming_books = ["Learn python", "Python for all", "Intro
2 print (programming_books)
3
4 wanted_book = input("Hi! What book would you like to buy?")
5 print (wanted_book)
6
7- if wanted_book in programming_books:
8     print ("Yes, we sell it!")
9- else:
10    print (["Sorry, we don't sell that book"])
```

Filename: .py

```
['Learn python',
'Python for all', 'Intro
to python']
波西傑克森
Sorry, we don't sell
that book
<completed in
13083.70 ms>
```

## 第四題：

有用到input、print、if、else、in的語法，利用了購物車的概念，將需要購買的東西用in語法增加在list中，也可以刪除不要的東西。

從 1 累加到 100:

1 add to 100 ROC Flag PROC flag USA Flag GD in Brython

```
1 shopping_list = ["carrots", "chocolate", "olives"]
2 print (shopping_list)
3
4 new_item = input("What else do I have to buy?")
5
6 if new_item in shopping_list:
7     print (new_item + " is/are already in the shopping list")
8     print (shopping_list)
9 else:
10    shopping_list.append(new_item)
11    print (shopping_list)
12
13 item_to_remove = input("What do I have to remove?")
14
15 if item_to_remove in shopping_list:
16    shopping_list.remove(item_to_remove)
17    print (shopping_list)
18 else:
19    print (item_to_remove + " is/are not in the list")
20    print (shopping_list)
```

Filename: .py

```
['carrots', 'chocolate',
'olives']
['carrots', 'chocolate',
'olives', 'book']
['carrots', 'olives',
'book']
<completed in |
15373.80 ms>
```

## 第五題：

新增了.index、.pop、.insert語法.index是用來搜尋，如果沒有會跳出error，pop則是移除列表中的索引，若()中是空白，則會刪除最後一個索引。

## Brython

從 1 累加到 100:

1 add to 100 ROC Flag PROC flag USA Flag GD in Brython

```
1 todays_menu = ["burger", "salad", "coke"]
2 print(todays_menu)
3
4 side_dish_index = todays_menu.index("salad")
5 print (side_dish_index)
6
7 todays_menu.pop(side_dish_index)
8 print (todays_menu)
9
10 todays_menu.insert(side_dish_index, "fries")
11 print (todays_menu)
```

Filename: .py

```
['burger', 'salad',
'coke']
1
['burger', 'coke']
['burger', 'fries', 'coke']
<completed in 2.30
ms>
```

## 第六題：

主要在教學列表的使用方式，

【】內第一個數字是起點，第二個數字是終點，第三個數字是每多少輸出一次。

```
1 add to 100 ROC Flag PROC flag USA Flag GD in Brython
13
14 print (cities[:3])
15
16 print (cities[2:])
17
18 print (cities[0:5:2])
19
20 print (cities[::-2])
21
22 print (cities[4])
23
24 print (cities[-1])
25
26 print (cities[-4:-1])
27
28 cities[3:0:-1]
29
30 cities[-2:-5:-1]
31
32 print (cities[::-1])
```

Filename:  .py

```
['San Diego',
'Prague', 'Cape
Town', 'Tokyo',
'Melbourne']
Prague
['Prague', 'Cape
Town', 'Tokyo']
['Prague', 'Tokyo']
['San Diego',
'Prague', 'Cape
Town']
['San Diego',
'Prague', 'Cape
Town']
['San Diego',
'Prague', 'Cape
```

## 第七題：

增加了del的語法，是一個用於刪除物件、變數、列表元素、字典項目等關鍵字，類似加入購物車後，不想要購買就按下刪除觸發del的這個語法。

### Brython

從 1 累加到 100:

```
1 add to 100 ROC Flag PROC flag USA Flag GD in Brython
1 senses = ["eyes", "nose", "ears", "tongue", "skin"]
2 print (senses)
3 senses[1] = "smell"
4 print (senses)
5 senses[3:5] = ["taste", "touch"]
6 print (senses)
7 senses[0:3:2] = ["sight", "hearing"]
8 print (senses)
9 planets = ["Mercury", "Mars", "Earth", "Neptune"]
10 print (planets)
11 planets = planets + ["Jupiter"]
12 print (planets)
13 planets = planets[0:2] + ["Venus"] + planets[2:5]
14 print (planets)
15 planets = planets[:5] + ["Uranus", "Saturn"] + planets[5:]
16 print(planets)
17 house = ["kitchen", "dining room", "living room", "bedroom",
18 print (house)
19 del house[1]
20 print (house)
21
```

Filename:  .py

```
['eyes', 'nose', 'ears',
'tongue', 'skin']
['eyes', 'smell', 'ears',
'tongue', 'skin']
['eyes', 'smell', 'ears',
'taste', 'touch']
['sight', 'smell',
'hearing', 'taste',
'touch']
['Mercury', 'Mars',
'Earth', 'Neptune']
['Mercury', 'Mars',
'Earth', 'Neptune',
```

## 第八題：

使用了for... in range的語法，從列表中選出要的字串來表示。

### Brython

從 1 累加到 100:

```
1 add to 100 ROC Flag PROC flag USA Flag GD in Brython
1 "Luca", "Daisy", "Juhan"]
2 "burgers", "tacos", "pizza"]
3 names are:")
4
5 0,4):
6 " + str(index))
7 " + friends[index])
8 te dishes are:")
9
10 0,4):
11 " + str(index))
12 + dishes[index])
13 0,4):
14 d " + friends[index] + "'s favorite dish is " + dishes[index])
```

Filename:  .py

```
My friends' names
are:
['Geetha', 'Luca',
'Daisy', 'Juhan']
index: 0
friend: Geetha
index: 1
friend: Luca
index: 2
friend: Daisy
index: 3
friend: Juhan
Their favorite dishes
are:
['sushi', 'burgers',
'tacos', 'pizza']
```



## 第九題：

用到了for... loop，else的語法，操作列表list進行迴圈處理，使用if...else來檢查條件成立，來判斷是不是我們想要的東西。

### Brython

從 1 累加到 100:

1 add to 100 ROC Flag PROC flag USA Flag GD in Brython

```
1 animals = ["giraffe", "penguin", "dolphin"]
2 print (animals)
3 # for each position in the list
4 for i in range (0, len(animals)):
5     print ("-- Beginning of loop --")
6     # print each element and its position
7     print ("The element in position " + str(i) + " is " + anim
8 wanted_to_see = "penguin"
9 # for each position in the list
10 for i in range (0, len(animals)):
11     # if the current animal is what you really wanted to see
12     if animals[i] == wanted_to_see:
13         # print out that that's the animal you really wanted t
14         print ("I saw a " + animals[i] + " and I really wanted
15     # if the current animal is not what you really wanted to s
16     else:
17         # just print out that you saw it
18         print ("I saw a " + animals[i])
```

Filename: input file name .py Save

Run Output 清除輸出區 清除繪圖區 Reload

```
['giraffe', 'penguin', 'dolphin']
-- Beginning of loop --
The element in position 0 is giraffe
-- Beginning of loop --
The element in position 1 is penguin
-- Beginning of loop --
The element in position 2 is dolphin
I saw a giraffe
I saw a penguin and I really wanted to see it!
I saw a dolphin
<completed in 3.80 ms>
```

## 第十題：

用for迴圈查看list列表並根據是否條件成立並進行輸出。

### Brython

從 1 累加到 100:

1 add to 100 ROC Flag PROC flag USA Flag GD in Brython

```
10 if len(accessories[i]) == 0:
11     # print the element, its position, and its number of
12     print ("The element " + accessories[i] + " is in pos
13 # for each position in the list
14 for i in range (len(accessories)):
15     # if the length of the element is less than 6
16     if len(accessories[i]) < 6:
17         # print the element, its position, and its number of
18         print ("The element " + accessories[i] + " is in pos
19 # defining the threshold
20 n_of_characters = 6
21 # for each position in the list
22 for i in range (len(accessories)):
23     # if the length of the element is greater than the thres
24     if len(accessories[i]) > n_of_characters:
25         # print the element, its position, and its number of
26         print ("The element " + accessories[i] + " is in pos
27 # defining the threshold
28 n_of_characters = 6
29 # for each position in the list
30 for i in range (len(accessories)):
```

Filename: input file name .py Save

Run Output 清除輸出區 清除繪圖區 Reload

```
['belt', 'hat', 'gloves', 'sunglasses', 'ring']
The element belt is in position 0
The element hat is in position 1
The element gloves is in position 2
The element sunglasses is in position 3
The element ring is in position 4
The element gloves is in position 2 and it has 6 characters
The element belt is in position 0 and it has less than 6
characters
The element hat is in position 1 and it has less than 6
characters
The element ring is in position 4 and it has less than 6
characters
```

# W13ai:

從browser模組中匯入html，接著建立canva元素，生成動態html並加入到網頁當中，隨後就可以開始用程式進行繪圖。

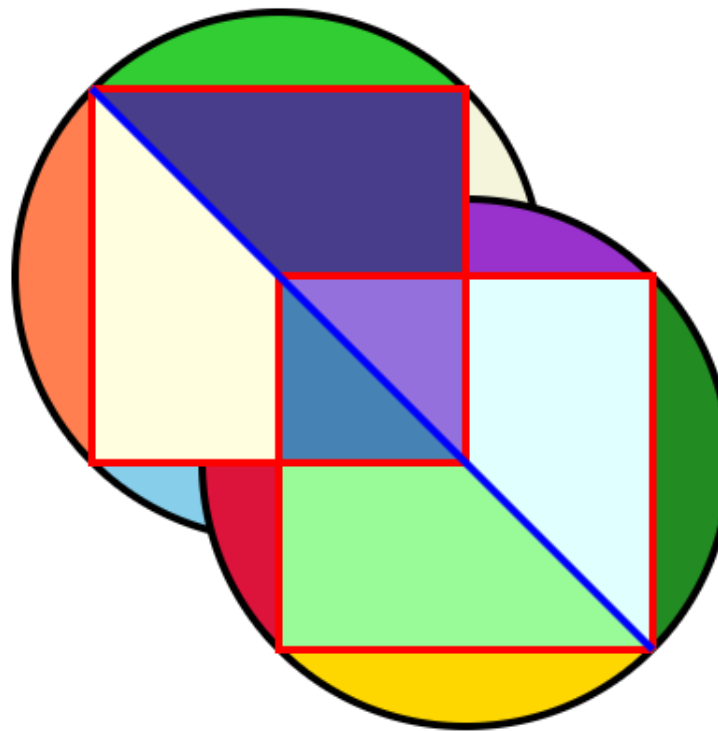
## Brython

從 1 累加到 100:

1 add to 100 | ROC Flag | PROC flag | USA Flag | GD in Brython

```
1 from browser import html
2 from browser import document as doc
3 import math
4
5 canvas = html.CANVAS(width=500, height=500)
6 brython_div = doc["brython_div1"]
7 brython_div <= canvas
8
9 ctx = canvas.getContext("2d")
10 ctx.lineWidth = 4
11 |
12 # 混色的
13 ctx.globalCompositeOperation = "screem"
14
15 # 黑邊圓*2 (每個畫四個邊邊)
16 ctx.strokeStyle = 'black'
17
18 # 第一個圓
19 ctx.beginPath()
20 ctx.arc(160, 160, 141, 0.25 * math.pi, 0.75 * math.pi)
```

w13\_ai

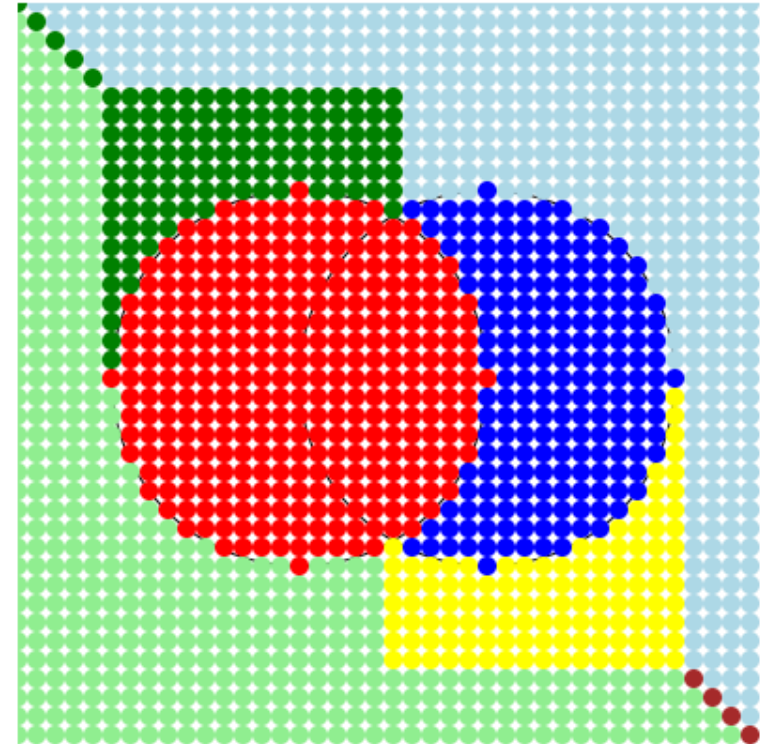


# W13\_hw:

這段程式的主要目的是，根據特定的規則判斷畫布上每個點所屬的區域，並將這些點填上對應的顏色，從而呈現不同的區域分類。

```
99 # 繪畫大點區域
100 for x in range(0, canvas_width, step_size): # x 從 0 到畫布寬度，步長為 step_size
101     for y in range(0, canvas_height, step_size): # y 從 0 到畫布高度，步長為 step_size
102
103         # 判斷區域，並設定顏色
104         if is_point_in_circle(x, y, *circle1_center, circle1_radius):
105             region = "circle1" # 在第一圓內部
106         elif is_point_in_circle(x, y, *circle2_center, circle2_radius):
107             region = "circle2" # 在第二圓內部
108         elif is_point_in_square(x, y, square1_x, square1_y, square1_size):
109             region = "square1" # 在第一正方形內部
110         elif is_point_in_square(x, y, square2_x, square2_y, square2_size):
111             region = "square2" # 在第二正方形內部
112         elif is_point_above_diagonal(x, y):
113             region = "diagonal_above" # 在對角線上方
114         elif is_point_below_diagonal(x, y):
115             region = "diagonal_below" # 在對角線下方
116         elif y < square1_y:
117             if x < square1_x:
118                 region = "top_left" # 左上交界區域
119             elif x >= square1_x + square1_size:
120                 region = "top_right" # 右上交界區域
121             else:
122                 region = "top" # 上方區域
123         elif y >= square1_y + square1_size:
124             if x < square1_x:
125                 region = "bottom_left" # 左下交界區域
126             elif x >= square1_x + square1_size:
127                 region = "bottom_right" # 右下交界區域
128             else:
129                 region = "bottom" # 下方區域
130         else:
131             if x < square1_x:
132                 region = "left" # 左側區域
133             elif x >= square1_x + square1_size:
134                 region = "right" # 右側區域
135             else:
136                 region = "top" # 預設為上方區域
137
138         # 設定圓點顏色
139         ctx.fillStyle = colors[region]
140
141         # 畫圓點
142         ctx.beginPath()
```

w13\_hw

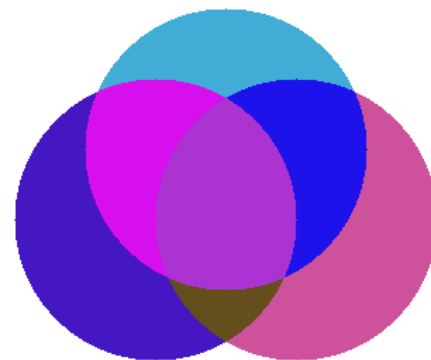


# W14\_ex:

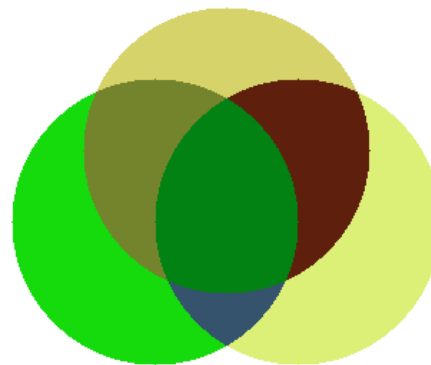
首先設定圓心座標及半徑，在進行掃描線填充，並進行填色，填色的部分也有設計，用random\_color\_generator這段函式，進行隨機的RGB選色，每一次刷新網頁都會是不一樣的颜色。

```
# 定義一個隨機顏色生成函式
def random_color_generator():
    """
    生成一個隨機的 RGB 顏色字符串。
    返回的顏色是 "rgb(r, g, b)" 格式，其中 r, g, b 是 0 到 255 之間的隨機整數。
    """
    r = random.randint(0, 255) # 隨機生成紅色成分 (0-255)
    g = random.randint(0, 255) # 隨機生成綠色成分 (0-255)
    b = random.randint(0, 255) # 隨機生成藍色成分 (0-255)
    return f"rgb({r}, {g}, {b})" # 返回隨機顏色字符串
```

w14\_ex



w14\_ex



# W15:

## 題目一:

`def calculate_sum()`: 是一個函數定義的起始部分 `()` (空括號): 表示該函數不接受任何參數  
初始化總和: `total = 0`, 用來存儲累加結果  
`for` 循環遍歷從 1 到 10 的數字, 並將每個數字累加到 `total` 中。

從 1 累加到 100:

1 add to 100

```
1 # 定義函數來計算和顯示結果
2 def calculate_sum():
3     total = 0
4     for i in range(1, 101): # 從 1 遍歷到 100
5         total += i
6     print(f"總和是: {total}") # 在控制台顯示結果
7
8 # 呼叫函數執行
9 calculate_sum()
```

Filename: .py

```
總和是: 5050
<completed in 2.20
ms>
```

## 題目二:

這段程式碼套用了等差數列當中求和的公式

$$n*(n+1)/2=s$$

$$n=100$$

$$\text{總和 } s=5050$$

1 add to 100

```
1 def addto(start, end):
2     """
3     計算從 start 到 end 的整數總和。
4
5     參數:
6     - start: 起始值 (整數)
7     - end: 結束值 (整數)
8
9     回傳:
10    - 總和 (整數)
11    """
12    total = sum(range(start, end + 1)) # 使用內建的 sum 函數
13    return total
14
15 # 呼叫函式並印出結果
16 result = addto(1, 100)
17 print(f"累加的結果是: {result}")
```

Filename: .py

```
累加的結果是: 5050
<completed in 2.90
ms>
```

### 第三題:

這題是要求只計算偶數，會使用`if i % 2 == 0` 這段程式來篩選當 `i` 是的餘數等於0時回傳運算結果。

`Return`則是讓每一次的運算結果回傳到`sum`這段函式並進行累加，最後傳到`total`這個變量。

#### Brython

從 1 累加到 100:

1 add to 100

```
1 def add_only_even(start, end):
2     """
3     計算從 start 到 end 之間所有偶數的總和。
4
5     參數:
6     - start: 起始值 (整數)
7     - end: 結束值 (整數)
8
9     回傳:
10    - 偶數總和 (整數)
11    """
12    # 使用範圍產生偶數並計算總和
13    total = sum(i for i in range(start, end + 1) if i % 2 == 0)
14    return total
15
16 # 呼叫函式並印出結果
17 result = add_only_even(1, 100)
18 print(f"累加的偶數結果是: {result}")
```

Filename:  .py

```
累加的偶數結果是:
2550
<completed in 4.10
ms>
```

### 第四題:

這題要求避開8和奇數，先用`range`函式，列出數字，如果數字包含8和奇數才會用`sum`函式累加結果並回傳到`valid_number`這個變量。

#### Brython

從 1 累加到 100:

1 add to 100

```
1 def add_avoid_8(start, end):
2     """計算從 start 到 end 的累加結果，避開包含 '8' 的數字以及基數
3     valid_numbers = [x for x in range(start, end + 1) if '8' not in str(x)]
4     print("符合條件的數字:", valid_numbers) # 打印符合條件的數字
5     return sum(valid_numbers)
6
7 # 測試函式
8 result = add_avoid_8(1, 100)
9 print(f"從 1 到 100 的累加總和 (避開包含 '8' 的數字以及基數) 是: {result}")
```

Filename:  .py

```
符合條件的數字: [2, 4, 6, 10, 12, 14, 16, 20, 22, 24, 26, 30, 32, 34, 36, 40, 42, 44, 46, 50, 52, 54, 56, 60, 62, 64, 66, 70, 72, 74, 76, 90, 92, 94, 96, 100]
從 1 到 100 的累加總和 (避開包含 '8' 的數字以及基數) 是: 1688
<completed in 6.50
ms>
```

# W16\_Exam1:

## 流程順序:

檢查是否獲取元素(預設已獲取)>建立CANVA元素>給予元素id名稱並建立畫布高度及深度>使用document <= canvas 將新建的 Canvas 元素插入到 document>取得 Canvas 並設定繪圖上下文>開始繪圖



這是用文字標示起點座標的程式碼

ctx.font:指字體樣式和大小

ctx.fillStyle:指填充顏色

Ctx.fillText(1,2,3):1指的是文字，2是X座標，3是Y座標

```
# Label origin
```

```
ctx.font = "12px Arial"
```

```
ctx.fillStyle = "black"
```

```
ctx.fillText("(31, 17)", origin_x + 5, origin_y - 5)
```

六邊形程式碼:

Hex\_x,Hex\_y:原點座標

Ctx.beingpath:開始路徑

For l in range(6):循環6次

angle:計算角度

X,Y:計算X,Y座標,30是圓的半徑

Ctx.moveTo:移動到新的原點

Ctx.lineTo:從原點畫一條線到新座標

closePath:關閉路徑

Ctx.fill:填充顏色

```
# Hexagon (Orange)
```

```
hex_x, hex_y = 100, 100
```

```
ctx.beginPath()
```

```
for i in range(6):
```

```
    angle = i * (2 * math.pi / 6)
```

```
    x = hex_x + 30 * math.cos(angle)
```

```
    y = hex_y + 30 * math.sin(angle)
```

```
    if i == 0:
```

```
        ctx.moveTo(x, y)
```

```
    else:
```

```
        ctx.lineTo(x, y)
```

```
ctx.closePath()
```

```
ctx.fillStyle = "orange"
```

```
ctx.fill()
```

菱形程式碼:

```
# Diamond (Teal)
```

```
ctx.beginPath()
```

```
ctx.moveTo(160, 70)
```

```
ctx.lineTo(180, 100)
```

```
ctx.lineTo(160, 130)
```

```
ctx.lineTo(140, 100)
```

```
ctx.closePath()
```

```
ctx.fillStyle = "teal"
```

```
ctx.fill()
```

三角形程式碼:

```
# Triangle (Teal)
```

```
ctx.beginPath()
```

```
ctx.moveTo(165, 130)
```

```
ctx.lineTo(205, 130)
```

```
ctx.lineTo(185, 100)
```

```
ctx.closePath()
```

```
ctx.fillStyle = "blue"
```

```
ctx.fill()
```

圓形程式碼:

```
# Circle (Pink)
```

```
ctx.beginPath()
```

```
ctx.arc(230, 100, 30, 0, 2 * math.pi)
```

```
ctx.fillStyle = "pink"
```

```
ctx.fill()
```

正方形程式碼：

```
# Square (Yellow)
ctx.beginPath()
ctx.rect(265, 75, 50, 50)
ctx.fillStyle = "yellow"
ctx.fill()
```

長方形程式碼：

```
# Rectangle (Blue)
ctx.beginPath()
ctx.rect(320, 90, 60, 20)
ctx.fillStyle = "Blue"
ctx.fill()
```



# W16\_Exam2:

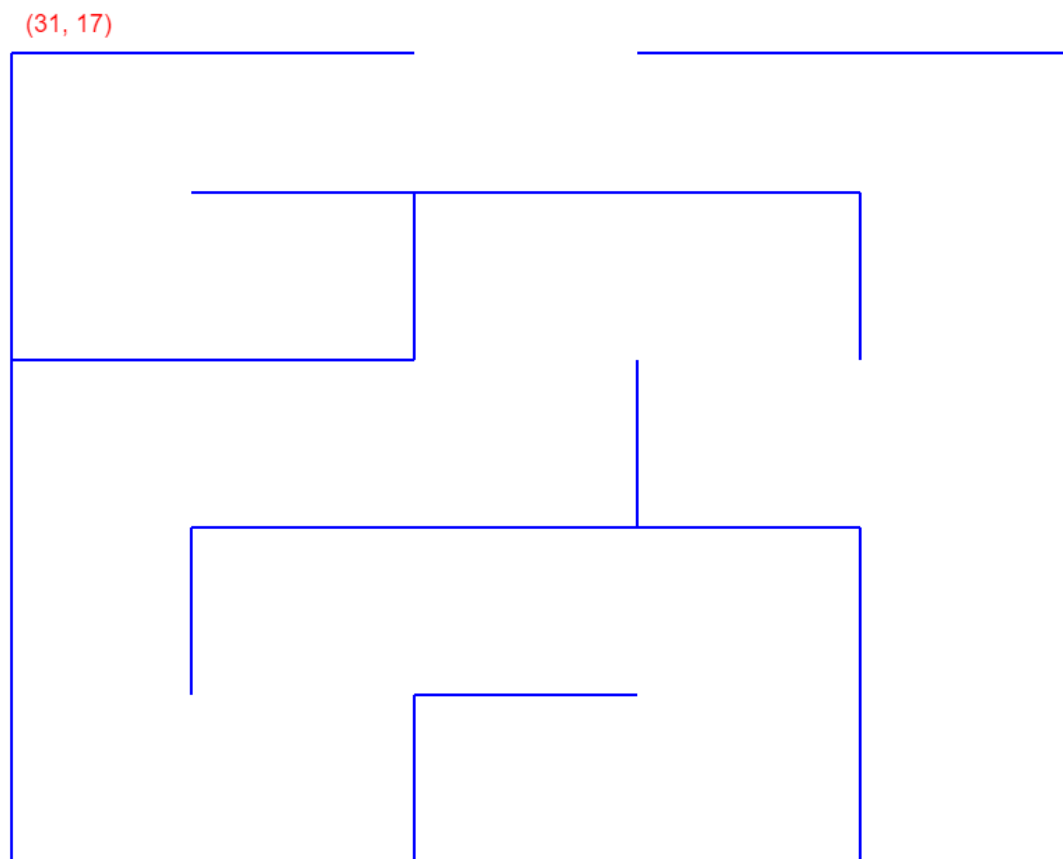
單純使用ctx.line還有ctx.moveTo兩個程式碼來做繪圖，抓大概的座標，一直重複拉線。

```
# Draw X

ctx.beginPath()
ctx.moveTo(origin_x, origin_y) # 起始點
ctx.lineTo(320, origin_y)      # 第一段線
ctx.moveTo(480, origin_y)      # 跳過空白部分
ctx.lineTo(800, origin_y)      # 第二段線
ctx.strokeStyle = "blue"
ctx.lineWidth = 2
ctx.stroke()

ctx.beginPath()
ctx.moveTo(160, 150) # 起始點
ctx.lineTo(640, 150) # 第一段線
ctx.strokeStyle = "blue"
ctx.lineWidth = 2
ctx.stroke()

ctx.beginPath()
ctx.moveTo(origin_x, 270) # 起始點
ctx.lineTo(320, 270)      # 第一段線
ctx.strokeStyle = "blue"
ctx.lineWidth = 2
ctx.stroke()
```



# W16\_Exam3:

重複拉線，抓座標點去畫線。

```
ctx.beginPath()
ctx.moveTo(800, origin_y) # 起始點
ctx.lineTo(800, 630)      # 第一段線
ctx.strokeStyle = "blue"
ctx.lineWidth = 2
ctx.stroke()
```

```
ctx.beginPath()
ctx.moveTo(400, origin_y) # 起始點
ctx.lineTo(400, 100)      # 第一段線
ctx.lineTo(720, 100)
ctx.lineTo(720, 320)
ctx.lineTo(560, 320)
ctx.lineTo(560, 220)
ctx.lineTo(400, 220)
ctx.lineTo(400, 320)
ctx.lineTo(80, 320)
ctx.lineTo(80, 580)
ctx.lineTo(240, 580)
ctx.lineTo(240, 460)
ctx.lineTo(560, 460)
ctx.lineTo(560, 580)
ctx.lineTo(400, 580)
ctx.lineTo(400, 630)
ctx.strokeStyle = "red"
ctx.lineWidth = 2
```

