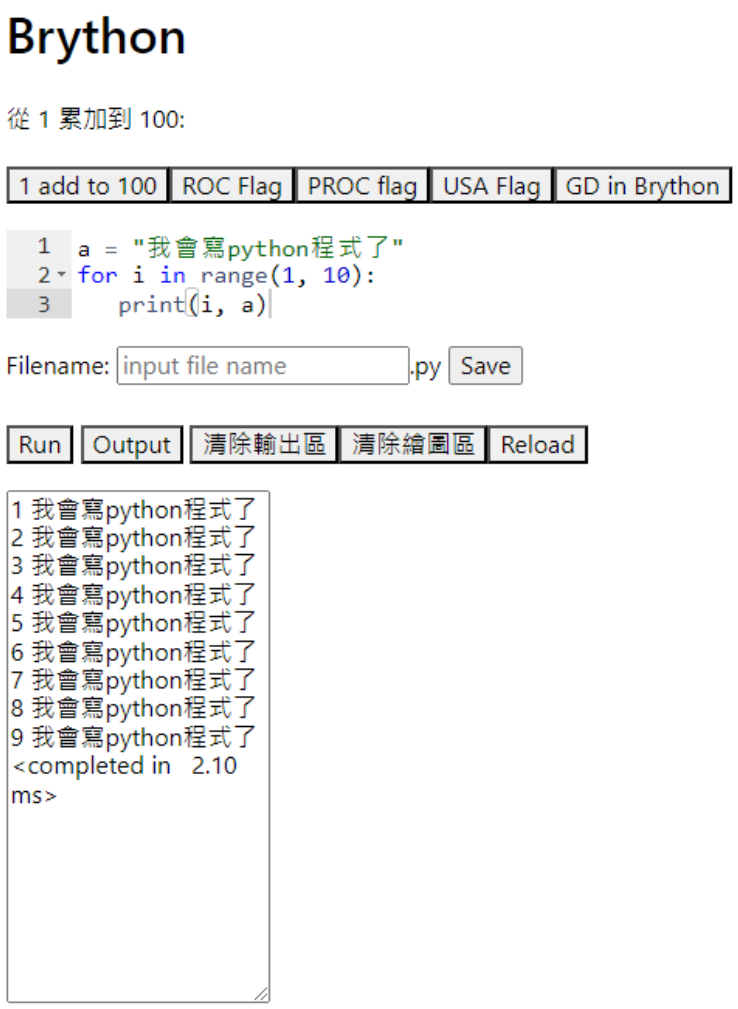
計算機實習報告

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。



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

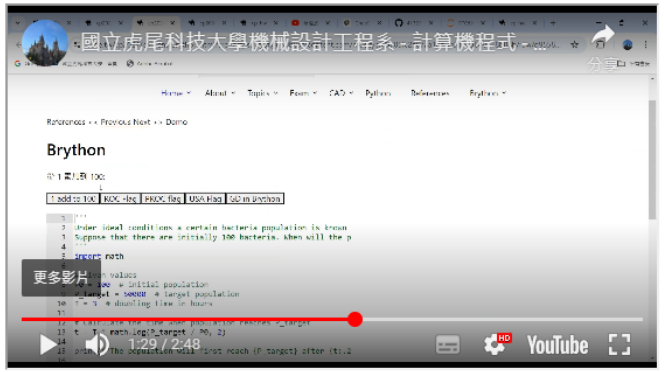
第一題

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

```python
import math

# 設定字元區域的大小 (11x11)
width, height = 11, 11
# 設定圓心位置
center_x, center_y = width // 2, height // 2

# 提示使用者輸入圓的半徑大小
try:
    radius = float(input("請輸入圓的半徑（建議5以內，過大可能
except ValueError:
    print("輸入錯誤，將使用預設半徑 5")
    radius = 5  # 預設半徑

# 遍歷整個字元區域
for y in range(height):
    for x in range(width):
        # 計算該位置到圓心的距離
        distance = math.sqrt((x - center_x) ** 2 + (y - cent
```

Filename: input file name .py Save

Run Output 清除輸出區 清除繪圖區 Reload

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

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

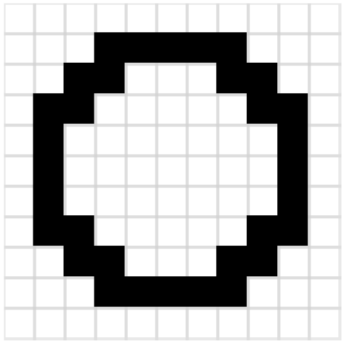
從 1 累加到 100:

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

```python
# 預設參數
center_x, center_y = 5, 5  # 圓心
radius = 4  # 預設圓的半徑
grid_width, grid_height = 11, 11  # 預設格子寬高

# 在 grid 中畫圓
def draw_circle():
    ctx.clearRect(0, 0, canvas.width, canvas.height)  # 清空
    grid(grid_width, grid_height, gs)  # 重新繪製格子
    for i in range(grid_width):
        for j in range(grid_height):
            # 計算 (i,j) 到圓心的距離
            dist = math.sqrt((i - center_x) ** 2 + (j - cent
            if abs(dist - radius) < 0.5:  # 若該點在圓邊界，
                fill(i * gs, j * gs, "black")

# 建立輸入框和按鈕
input_width = html.INPUT(type="number", value=grid_width, mi
input_height = html.INPUT(type="number", value=grid_height,
input_radius = html.INPUT(type="number", value=radius, min=1
```



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: input file name .py Save

Run | Output | 清除輸出區 | 清除繪圖區 | Reload

```
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: input file name .py Save

Run | Output | 清除輸出區 | 清除繪圖區 | Reload

```
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: input file name .py Save

Run Output 清除輸出區 清除繪圖區 Reload

```
/\_/\
>^.^<
 / \
(___)___
<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 add to 100 ROC Flag PROC flag USA Flag GD in Brython

```
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: input file name .py Save

Run Output 清除輸出區 清除繪圖區 Reload

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

```
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: input file name .py Save

Run Output 清除輸出區 清除繪圖區 Reload

```
['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: input file name .py Save

Run | Output | 清除輸出區 | 清除繪圖區 | Reload

```
['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

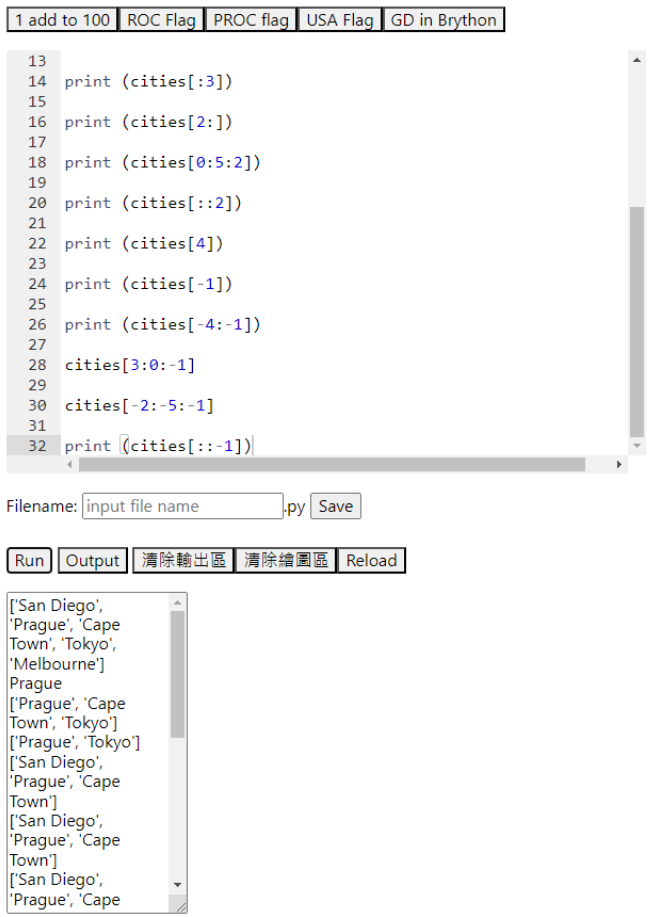
```
1  todays_menu = ["burger", "salad", "coke"]
2  print(todays_menu)
3
4  side_dish_index = todays_menu.index("salad")
5  print (side_dish_index)
6
7  todays_menu.pop(side_dish_index)
8  print (todays_menu)
9
10 todays_menu.insert(side_dish_index, "fries")
11 print (todays_menu)
```

Filename: input file name .py Save

Run | Output | 清除輸出區 | 清除繪圖區 | Reload

```
['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。



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。

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。

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

```
animals = ["giraffe", "penguin", "dolphin"]
print (animals)
# for each position in the list
for i in range (0, len(animals)):
    print ("-- Beginning of loop --")
    # print each element and its position
    print ("The element in position " + str(i) + " is " + anim
wanted_to_see = "penguin"
# for each position in the list
for i in range (0, len(animals)):
    # if the current animal is what you really wanted to see
    if animals[i] == wanted_to_see:
        # print out that that's the animal you really wanted t
        print ("I saw a " + animals[i] + " and I really wanted
    # if the current animal is not what you really wanted to s
    else:
        # just print out that you saw it
        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

```
    if len(accessories[i]) == 6:
        # print the element, its position, and its number of
        print ("The element " + accessories[i] + " is in pos
# for each position in the list
for i in range (len(accessories)):
    # if the length of the element is less than 6
    if len(accessories[i]) < 6:
        # print the element, its position, and its number of
        print ("The element " + accessories[i] + " is in pos
# defining the threshold
n_of_characters = 6
# for each position in the list
for i in range (len(accessories)):
    # if the length of the element is greater than the thres
    if len(accessories[i]) > n_of_characters:
        # print the element, its position, and its number of
        print ("The element " + accessories[i] + " is in pos
# defining the threshold
n_of_characters = 6
# for each position in the list
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:

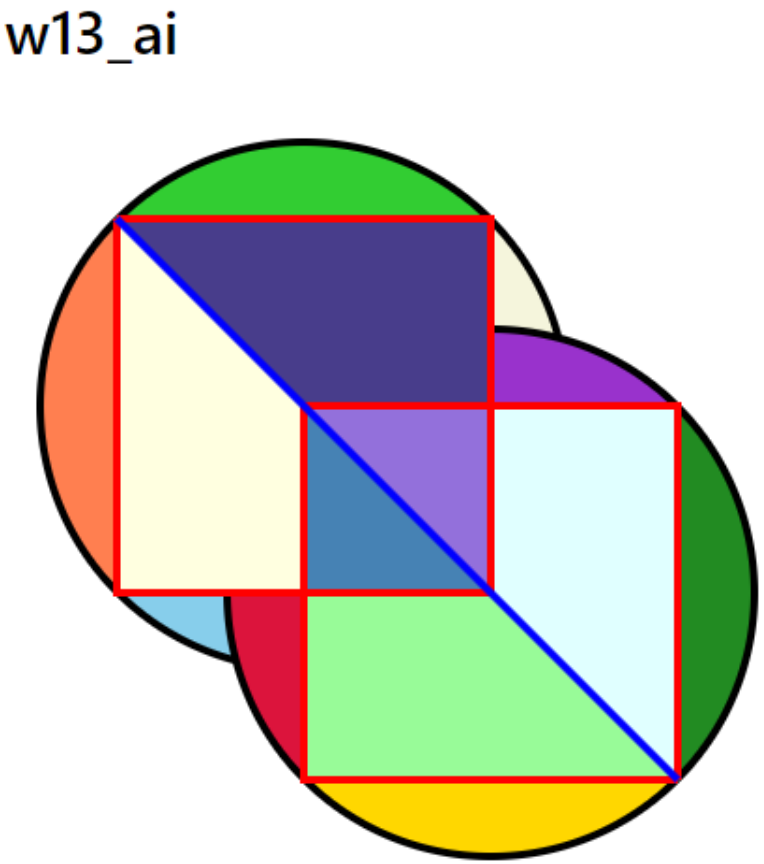
從browser模組中匯入html，接著建立canva元素，生成動態html並加入到網頁當中，隨後就可以開始用程式進行繪圖。

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_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。

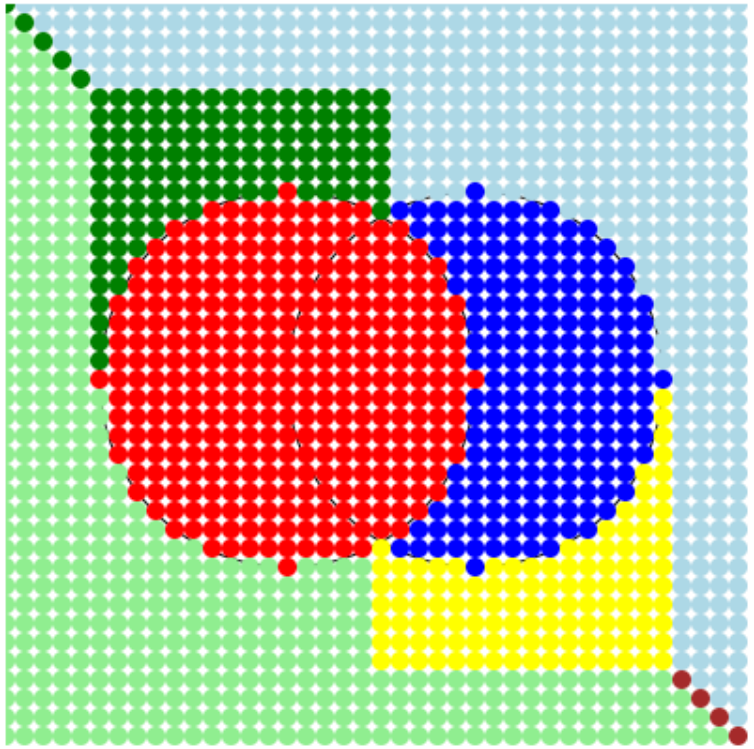
```
    # 掃描大繪點區域
    for x in range(0, canvas_width, step_size):  # x 從 0 到畫布寬度，步長為 step_size
        for y in range(0, canvas_height, step_size):  # y 從 0 到畫布高度，步長為 step_size

            # 判斷區域，並設定顏色
            if is_point_in_circle(x, y, *circle1_center, circle1_radius):
                region = "circle1"  # 在第一圓內部
            elif is_point_in_circle(x, y, *circle2_center, circle2_radius):
                region = "circle2"  # 在第二圓內部
            elif is_point_in_square(x, y, square1_x, square1_y, square1_size):
                region = "square1"  # 在第一正方形內部
            elif is_point_in_square(x, y, square2_x, square2_y, square2_size):
                region = "square2"  # 在第二正方形內部
            elif is_point_above_diagonal(x, y):
                region = "diagonal_above"  # 在對角線上方
            elif is_point_below_diagonal(x, y):
                region = "diagonal_below"  # 在對角線下方
            elif y < square1_y:
                if x < square1_x:
                    region = "top_left"  # 上左交界區域
                elif x >= square1_x + square1_size:
                    region = "top_right"  # 上右交界區域
                else:
                    region = "top"  # 上方區域
            elif y >= square1_y + square1_size:
                if x < square1_x:
                    region = "bottom_left"  # 下左交界區域
                elif x >= square1_x + square1_size:
                    region = "bottom_right"  # 下右交界區域
                else:
                    region = "bottom"  # 下方區域
            else:
                if x < square1_x:
                    region = "left"  # 左側區域
                elif x >= square1_x + square1_size:
                    region = "right"  # 右側區域
                else:
                    region = "top"  # 預設為上方區域

            # 設定圓點顏色
            ctx.fillStyle = colors[region]

            # 畫圓點
            ctx.beginPath()
```

w13_hw

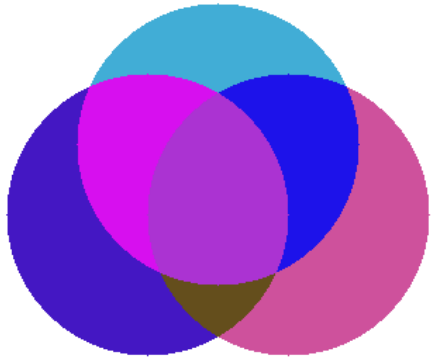


W14_ex:

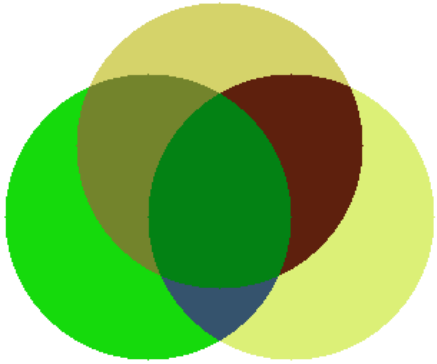
首先設定圓心座標及半徑，在進行掃描線填充，並進行填色，填色的部分也有設計，用random_color_generator這段函式，進行隨機的RGB選色，每一次刷新網頁都會是不一樣的顏色。

```
# 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

```
# 定義函數來計算和顯示結果
def calculate_sum():
    total = 0
    for i in range(1, 101):  # 從 1 遍歷到 100
        total += i
    print(f"總和是: {total}")  # 在控制台顯示結果

# 呼叫函數執行
calculate_sum()
```

Filename: input file name .py Save

Run Output 清除輸出區 清除繪圖區 Reload

```
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

總和s=5050

1 add to 100

```
def addto(start, end):
    """
    計算從 start 到 end 的整數總和。

    參數:
    - start: 起始值 (整數)
    - end: 結束值 (整數)

    回傳:
    - 總和 (整數)
    """
    total = sum(range(start, end + 1))  # 使用內建的 sum 函數
    return total

# 呼叫函式並印出結果
result = addto(1, 100)
print(f"累加的結果是: {result}")
```

Filename: input file name .py Save

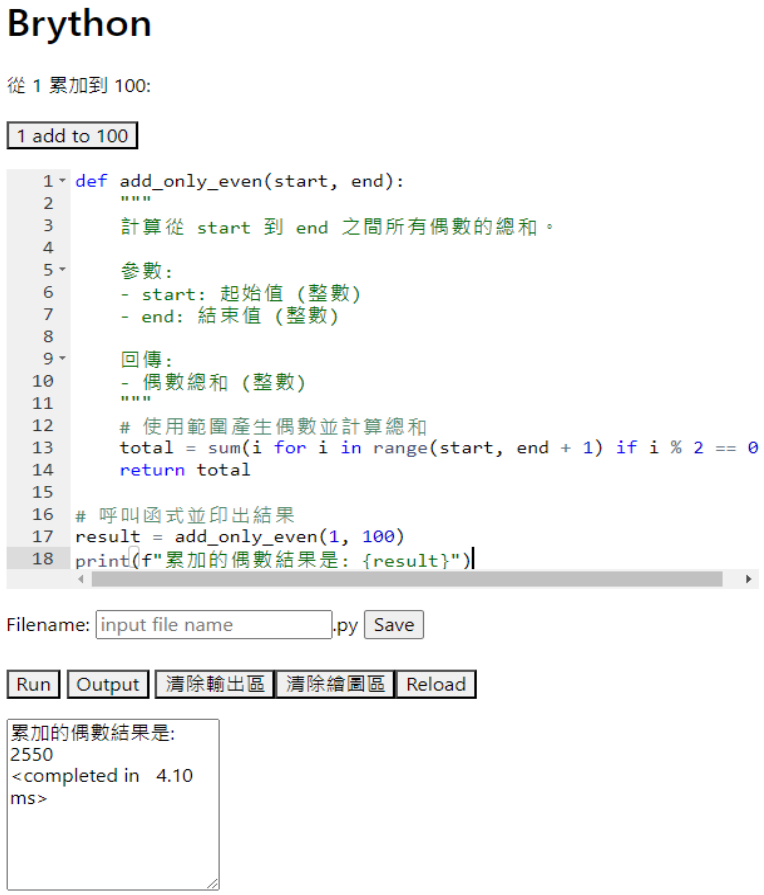
Run Output 清除輸出區 清除繪圖區 Reload

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

第三題:

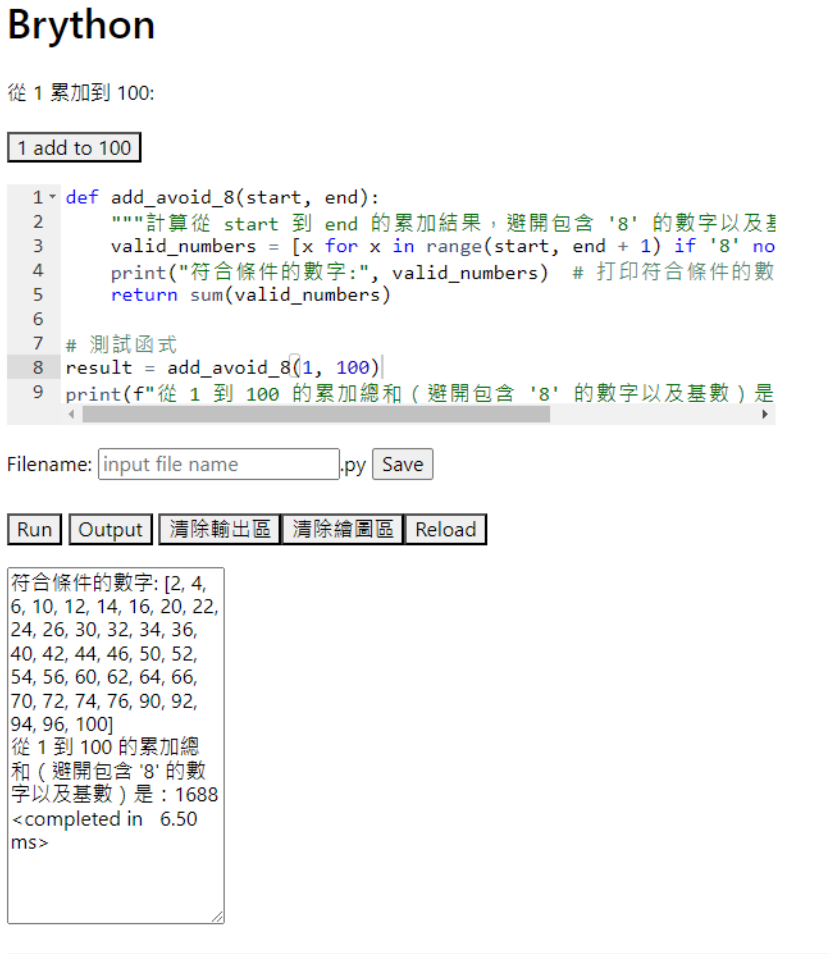
這題是要求只計算偶數，會使用if i % 2 == 0 這段程式來篩選當 I 是的餘數等於0時回傳運算結果。

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。



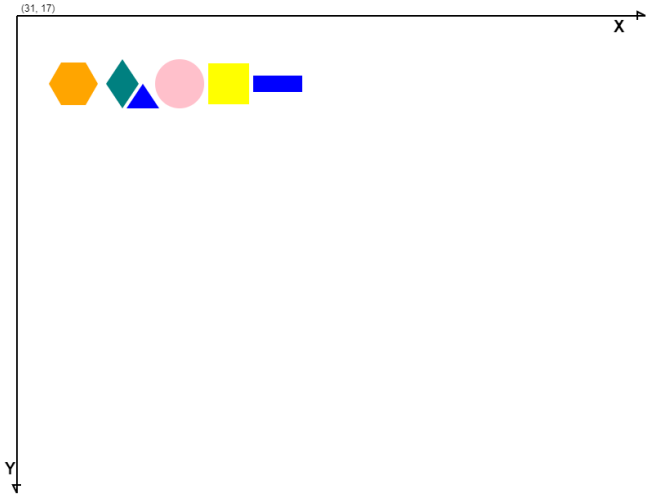
第四題:

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vaild_number這個變量。



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 I 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:

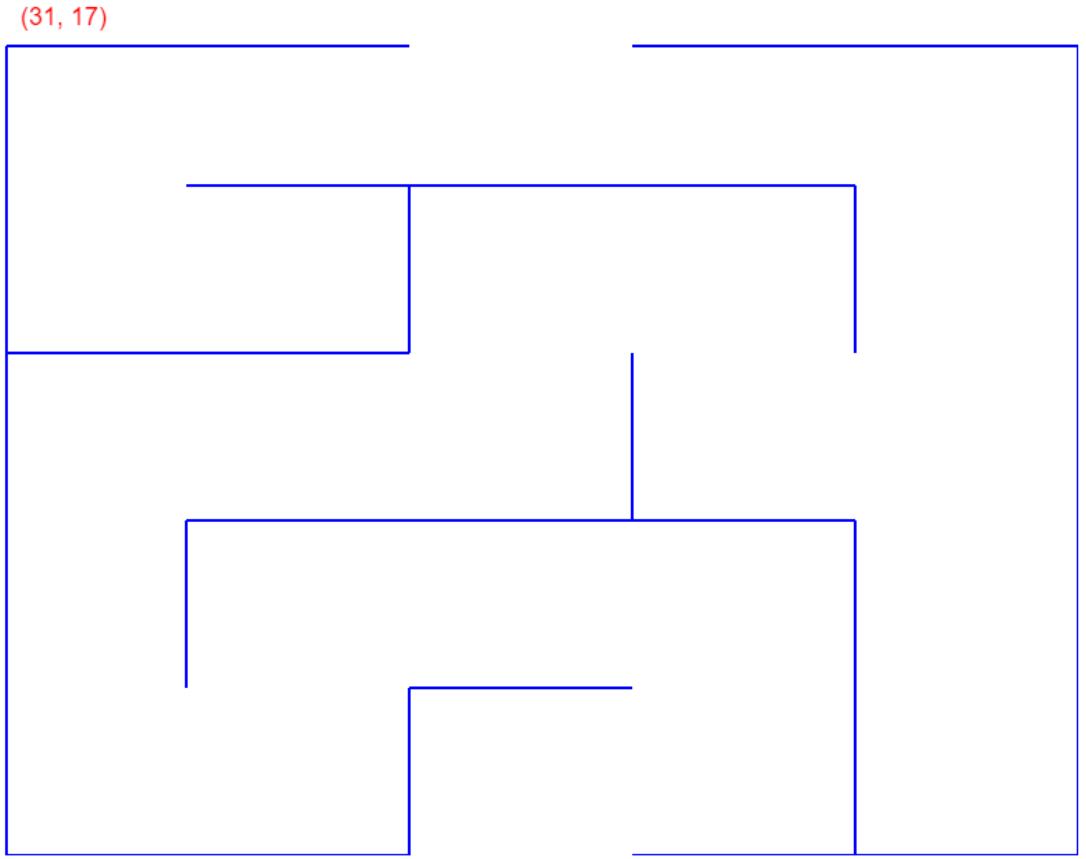
單純使用ctx.line還有ctx.moveTo兩個程式碼來做繪圖，抓大概的座標，一直重複拉線。

```
# 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
```