COMPSCI 101
Principles of Programming

Lecture 25 - Using the Canvas widget to draw rows and columns of shapes

---

Learning outcomes

▸ At the end of this lecture, students should be able to
  ▸ draw 2D shapes using characters
  ▸ draw 2D shapes on a Canvas

Drawing 2D shapes using Characters

▸ We write programs to draw 2D shapes using characters
  ▸ (e.g. asterisks)

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

▸ The way to conceptualize this is to think about the shape **as a sequence of rows** and to think carefully about **how to** describe the ith row, e.g. drawing a triangle.

▸ These kinds of problems will help you learn how to write loops by finding **appropriate formulas** to describe each iteration of the loop in terms of the iteration variable.

Printing a Row of characters

Example00.py   DEMO

▸ The following example prints only one row of '#' characters using a SINGLE for loop.

```
def print_row(number_of_cols):
    for j in range(number_of_cols):
        print('#', end="")
    print()
```

`###`

Print a new line character
(i.e. move to next line)

Printing Multiple Rows of Characters

▸ To create rows and columns of shapes we need nested loops

  ▸ That is, loops within loops to execute lines of code.

> Set up all the variables needed for the nested loop
>
> **for … in loop** which dictates how many rows:
>   Set everything up ready for drawing the row
>   **for … in loop** which handles one single row:
>   draw a single character
>   move to next line

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

  ▸ The first (outer) loop is looping through rows, the inner loop is looping through columns.

  ▸ As we go through each column of a given row, we print an asterisk. The result is that we can build any size rectangle we want.

---

1) Printing a Rectangle of Characters

▸ To print a rectangle, we need two parameters:

  ▸ number of rows = 4 rows
  ▸ number of columns = 3 columns

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

> Set up all the variables needed for the nested loop
>
> **for … in loop** …
>   **for … in loop** which handles one single row:
>   draw 3 asterisks
>   move to next line

  ▸ The outer for loop contains two statements:
    ▸ 1) inner for loop
    ▸ 2) print(): move cursor to the next line
  ▸ The inner for loop contains one statement:
    ▸ statement which prints a character

Example01.py

DEMO

1) Printing a Rectangle of Characters

▸ To print a rectangle, we need two parameters:

  ▸ number of rows = 4 rows
  ▸ number of columns = 3 columns

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

> Set up all the variables needed for the nested loop
>
> **for … in range** … 4 rows
>   **for … in range** … 3 columns
>   draw 1 asterisk
>   move to next line

```python
def print_square(number_of_rows, number_of_cols):
    for i in range(number_of_rows):
        for j in range(number_of_cols):
            print('*', end="")
        print()
```

---

2) Printing a right-angle Triangle

▸ To print a right-angle triangle, we need one parameter:

  ▸ number of rows = 4 rows

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

> Set up all the variables needed for the nested loop
>
> **for … in loop** … 4 rows
>   **for … in loop** which handles one single row:
>   if it is the first row, draw 1 asterisk
>   if it is the second row, draw 2 asterisks
>   if it is the ith row, draw i asterisks
>   move to next line

  ▸ The outer for loop contains two statements:
    ▸ 1) inner for loop
    ▸ 2) print(): move cursor to the next line
  ▸ The inner for loop contains one statement:
    ▸ statement which prints one or more character(s)

2) Printing a right-angle Triangle

Example02.py · DEMO

▸ To print a right-angle triangle, we need one parameter:

　▸ number of rows = 4 rows

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

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

Set up all the variables needed for the nested loop

for … in range … 4 rows
　　for … in range …
　　　　row = 0, number of columns = 1
　　　　row = 1, number of columns = 2
　　　　row = 2, number of columns = 3
　　move to next line

```
def print_right_angle_triangle(number_of_rows):
    for row in range(number_of_rows):
        for column in range(row+1):
            print('*', end="")
        print()
```

Exercise 1

▸ Task:

　▸ Complete the following code fragment to print …

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

```
def print_right_angle_triangle(number_of_rows):
    for row in range(number_of_rows):



        print()
```

Program skeleton

▸ All the programs in this lecture have the following code skeleton.

　▸ The draw_shapes() function is different for each exercise.

```
def main():
    root = Tk()
    root.title("My first Canvas")
    root.geometry("400x300+10+20")
    a_canvas = Canvas(root)
    a_canvas.config(background="pink")    #some colour
    a_canvas.pack(fill=BOTH, expand = True)
    draw_shapes(a_canvas)
    root.mainloop()

main()
```


Drawing 2D shapes on a Canvas

▸ In order to draw a 2D shape (e.g. multiples of squares) on a canvas, we need:

Size of the squares is 50 pixels by 50 pixels

　▸ The number of rows and number of columns

　▸ Size of each square (size=50)

　▸ Start point (x_margin, y_margin) = (20, 30)

　▸ Nested loops

　▸ Coordinates of the top left corner of each square

　　▸ Example:

　　　□ 1st (20, 30), (70, 30), (120,30) …

　　　□ 2nd (20, 80), (70,80), (120, 80)

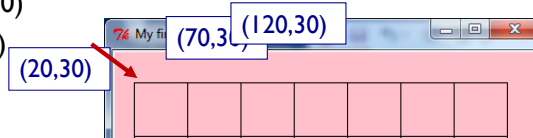　　　□ …

(120,30)

(70,30)

(20,30)

My first Canvas
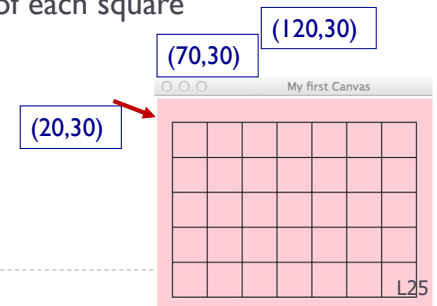
Example 3

▸ Let's look at ONE row of the shape FIRST:
  ▸ x = 20 (starts at 20 on each row)
    ▸ Coordinates of the first square: (20, 30, 70, 80)
    ▸ …Second square: (70, 30, 120, 80)
    ▸ …Third square(120, 30, 170, 80)

(120,30)
(70,30)
(20,30)

```
x_left = left_hand_side

for j in range(number_of_colums):
    rect = (x_left, y_down , x_left + size, y_down  + size)
    a_canvas.create_rectangle(rect)
    x_left += size
```

modify x-coordinate of the square in each iteration

---

Drawing … on a Canvas

▸ Now, we look at the entire shape. We need nested loops!
▸ The outer loop iterates number of rows.
  ▸ 1st row : coordinate of the top left corner: (20, **30**) and the next one is (70, **30**) and (120, 30) …
  ▸ 2nd row: coordinate of the top left corner: (20, **80**) and the next one is (70, **80**) and (120, 80) …
  ▸ 3rd row: : coordinate of the top left corner: (20, **130**) and the next one is (70, **130**) and (120, 130) …
  ▸ …

My first Canvas

(20,30)
(20,80)
(20,130)
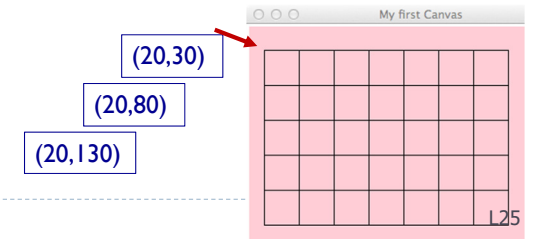
Drawing … on a Canvas

▸ We put them together:

Set up all the variables needed for the nested loop
for … in loop which dictates how many rows:
    Set everything up ready for drawing the row
    for … in loop which handles one single row:
        draw a single shape
        change the x value to move along the row
    change the y value ready for the next row down

Outer loop:

```
for i in range(number_of_rows):
  x_left = left_hand_side



    for j in range(number_of_columns):
        rect = (x_left, y_down, x_left + size, y_down + size)
        a_canvas.create_rectangle(rect)
        x_left += size

    y_down += size
```

reset the starting position of each row

Inner loop:

adjust the y coordinates

Example03.py

DEMO

Drawing … on a Canvas

reset the starting position of each row

y_down += size

y_down += size

(20,30) x_left = left_hand_side	(70, 30) x_left += size y no change	(120, 30) x_left += size y no change
(20,80) x_left = left_hand_side	(70, 80) x_left += size y no change	(120, 80) x_left += size y no change
(20,130) x_left = left_hand_side	(70, 130) x_left += size y no change	(120, 130) x_left += size y no change

▸ Algorithm:

Set up all the variables needed for the nested loop
for … in loop which dictates how many rows:
    Set everything up ready for drawing the row
    for … in loop which handles one single row:
        draw a single shape
        change the x value to move along the row
    change the y value ready for the next row down

▸ Consider the following code fragment:

```
def rectangular_grid(a_canvas):
    number_of_columns = 3
    number_of_rows = 4
    left_hand_side = 50
    y_down = 100
    size = 20
    for i in range(number_of_rows):
        x_left = left_hand_side         #position A
        for j in range(number_of_columns):
            rect = (x_left, y_down, x_left + size, y_down + size)
            a_canvas.create_rectangle(rect)
            x_left += size              #position B

        y_down += size
```

Example 4

▸ What should we do in order to draw the following shapes?

  ▸ First row:
    ▸ Fill, draw, fill, draw…
  ▸ Second row:
    ▸ Draw, fill, draw, fill …
  ▸ Third row
    ▸ Fill, draw, fill, draw…
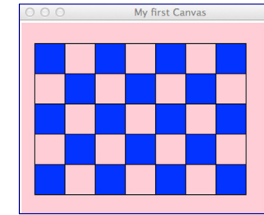
```
rect = (x_left, y_down, x_left + size, y_down + size)
a_canvas.create_rectangle(rect, fill="blue")
```
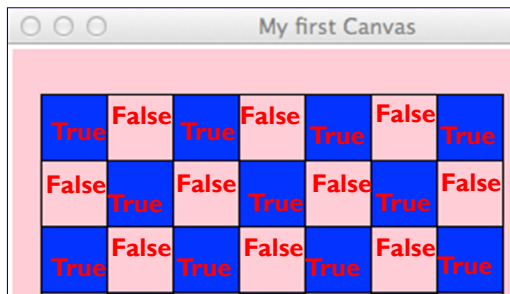
Command to create
the filled square

```
rect = (x_left, y_down, x_left + size, y_down + size)
a_canvas.create_rectangle(rect)
```

---

4) Drawing … on a Canvas

▸ Using a Boolean variable

  ▸ First row:
    ▸ True, False, True, False…
  ▸ Second row:
    ▸ False, True, False, True…
  ▸ Third row
    ▸ True, False, True, False…

---

4) Drawing … on a Canvas

▸ What is the output of the following code fragment?

```
is_filled = True
for i in range(5):
    print(is_filled, end=" ")
    is_filled = not is_filled
```

i	is_filled
	True
0	False
1	True
2	False
3	True
4	False

True False True False True

Drawing … on a Canvas

We put them together:

Outer loop:

Inner loop

```
x-margin, y-margin, width, height, first_in_row_filled=True
Set up all the variables needed for the nested loop
set up y-position
for … in loop which dictates how many rows:
    Set everything up ready for drawing the row
    set up x-positon, is_filled

    for … in loop which handles one single row:
        draw a single shape
        change the x value to move along the row
        modify the is_filled boolean
    change the y value ready for the next row down
    modify the first_in_row_filled boolean
```

Drawing … on a Canvas

Example04.py DEMO

Nested Loops:

```python
first_in_row_filled = True
for i in range(number_of_rows):
  x_left = left_hand_side
  is_filled = first_in_row_filled
  for j in range(number_in_row):
    rect = (x_left, y_down, x_left + size, y_down + size)
    if is_filled:
      a_canvas.create_rectangle(rect, fill="blue")
    else:
      a_canvas.create_rectangle(rect)
    x_left = x_left + size
    is_filled = not is_filled

  y_down = y_down + size
  first_in_row_filled = not first_in_row_filled
```
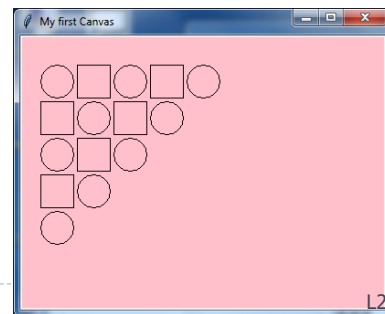

Example 5

Steps:

- 1st iteration of outer loop -> repeat 5 iterations in the inner loop
- 2nd iteration of outer loop -> repeat 4 iterations in the inner loop
- 3rd iteration of outer loop -> repeat 3 iterations in the inner loop
- 4th iteration of outer loop -> repeat 2 iterations in the inner loop
- 5th iteration of outer loop -> repeat 1 iteration in the inner loop

---

is_circle boolean

first_is_circle	is_circle				
True	True	False	True	False	True
False	False	True	False	True	
True	True	False	True		
False	False	True			
True	True				
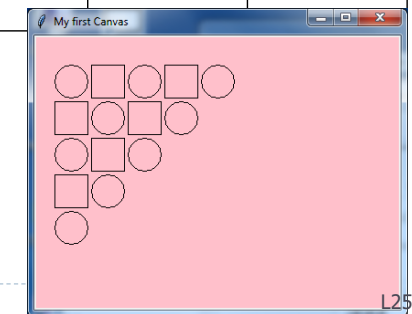
Exercise 2

▶ Draw the canvas

My first Canvas

gridlines are of
size 30 pixels

```python
def draw_shapes(a_canvas):
    number_of_rows = 6
    size = 30
    y_down = 0
    left_hand_side = size

    for number_along_row in range(1, number_of_rows + 1):
        x_left = left_hand_side

        for j in range(number_along_row):
            rect = (x_left + 2, y_down + 2, x_left + size - 2, y_down
                                                + size - 2)
            a_canvas.create_oval(rect, fill="blue")
            x_left = x_left + size * 2

        y_down = y_down + size
```