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
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 $i^{th}$ 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.
The following example prints only one row of ‘#’ characters using a SINGLE for loop.

```python
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
1) Printing a Rectangle of Characters

To print a rectangle, we need two parameters:

- number of rows = 4 rows
- number of columns = 3 columns

```python
# 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()
```

DEMO
Example01.py L257
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

```plaintext
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 i\textsuperscript{th} 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

- To print a right-angle triangle, we need one parameter:
  - number of rows = 4 rows

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

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
```

DEMO

Example02.py

L25
Exercise 1

Task:

Complete the following code fragment to print ...

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

print()
```

Case 1:

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

Case 2:

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

L25
All the programs in this lecture have the following code skeleton.

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

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

- The number of rows and number of columns
- Size of each square (size=50)
- Start point \((x_{\text{margin}}, y_{\text{margin}}) = (20, 30)\)
- Nested loops
- Coordinates of the top left corner of each square

Example:
- 1\(^{st}\) \((20, 30), (70, 30), (120,30) \ldots\)
- 2\(^{nd}\) \((20, 80), (70,80), (120, 80)\)
- ...
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)

```python
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
```

modify x-coordinate of the square in each iteration
Now, we look at the entire shape. We need nested loops!

The outer loop iterates number of rows.

1\(^{st}\) row: coordinate of the top left corner: \((20, 30)\) and the next one is \((70, 30)\) and \((120, 30)\) …

2\(^{nd}\) row: coordinate of the top left corner: \((20, 80)\) and the next one is \((70, 80)\) and \((120, 80)\) …

3\(^{rd}\) row: coordinate of the top left corner: \((20, 130)\) and the next one is \((70, 130)\) and \((120, 130)\) …

…
Drawing ... on a Canvas

- We put them together:

```
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
```

Outer loop:
- 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

Inner loop:
- reset the starting position of each row

```
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
```

- adjust the y coordinates
Drawing ... on a Canvas

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
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…

```python
rect = (x_left, y_down, x_left + size, y_down + size)
a_canvas.create_rectangle(rect, fill="blue")
```

```
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?

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

<table>
<thead>
<tr>
<th>i</th>
<th>is_filled</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>True</td>
</tr>
<tr>
<td>4</td>
<td>False</td>
</tr>
</tbody>
</table>

True False True False True
We put them together:

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-position, 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
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

<table>
<thead>
<tr>
<th>first_is_circle</th>
<th>is_circle</th>
<th>is_circle</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
| True           | True      | True      | }
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
def draw_shapes(a_canvas):
    number_of_rows = 5
    left_hand_side = 0
    y_down = 0
    size = 50
    first_is_circle = True

    for number_to_do in range(1, number_of_rows + 1):
        x_left = left_hand_side
        is_circle = first_is_circle
        for j in range(number_to_do):
            rect = (x_left + 3, y_down + 3, x_left + size - 3, y_down + size - 3)

            if is_circle:
                a_canvas.create_oval(rect, fill="blue")
            else:
                a_canvas.create_rectangle(rect)

            x_left = x_left + size * 2
            is_circle = not is_circle

    y_down = y_down + size
    first_is_circle = not first_is_circle