CompSci 101 - Assignment 5

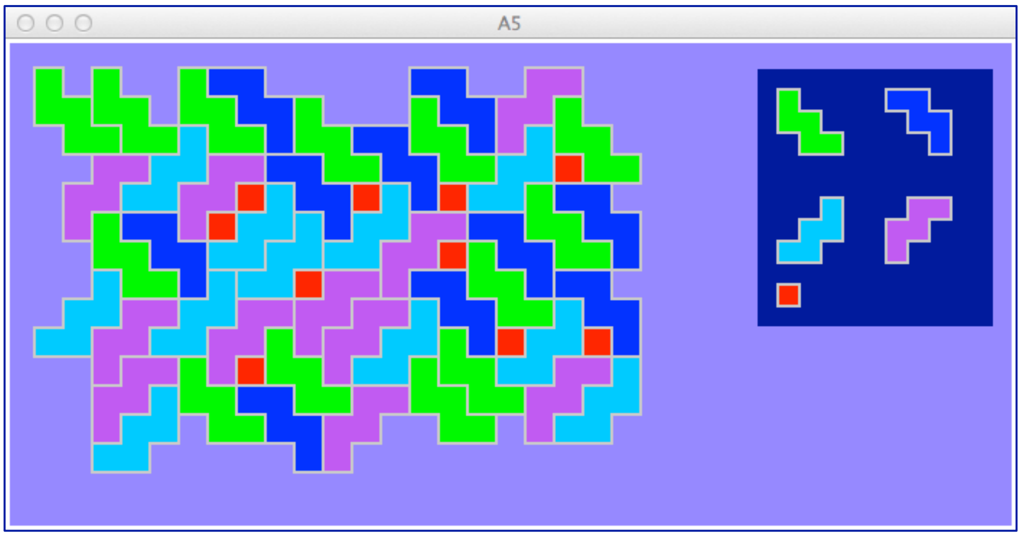
Due: 4:30pm, Saturday 24th October 2015.

Worth: This assignment is marked out of 10 and is worth 2% of your final mark. One mark out of ten is given for the style of your code (docstring, variable names, code).

Topic covered: Drawing a grid of shapes inside a Canvas object.

TESSELATIONS

A tesselation is an arrangement of shapes closely fitted together, often of polygons in a repeated pattern without gaps (non-overlapping shapes). For this assignment you are required to complete two functions of a program which produces a tesselation using five different tile shapes. The completed program produces the following window:



VERY IMPORTANT:

- In your program, there must not be any variables used outside any of the functions.
- To run this program you need to create a folder containing both your program file and the file, **`TileMap.txt`**. Download the `TileMap.txt` file from the CompSci 101 Assignments web page:
  https://www.cs.auckland.ac.nz/courses/compsci101s2c/assignments/
- Your program must include a docstring at the top of the file containing your name, UPI, ID number and a description of the program.
- Copy the skeleton code at the end of this document into your program before you start this assignment.

Submit the file containing your program using the Assignment Dropbox (you do not need to submit the `TileMap.txt` file):
https://adb.auckland.ac.nz/Home/

Your program file should be named 'YourUPIA5.py', e.g., afer023A5.py.

Initially the program produces a canvas (the window area) showing a grid of empty squares and an empty blue rectangle on the right hand side of the canvas.

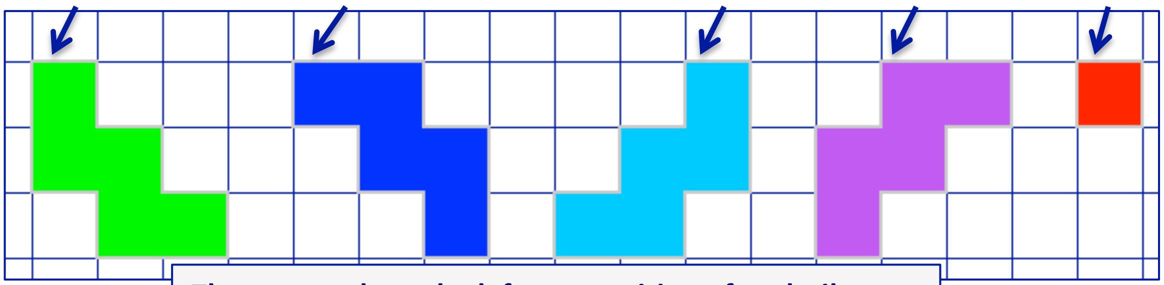
ASSIGNMENT 4 PART 1 (1 MARK):

Currently the title bar of the program window displays "A5". Add your upi to the title bar of the program, i.e., the title bar should display the string, "A5 – yourUPI", e.g., A5 – afer023

Assignment 4 Part 2 (5 marks) – The `draw_tile()` function:

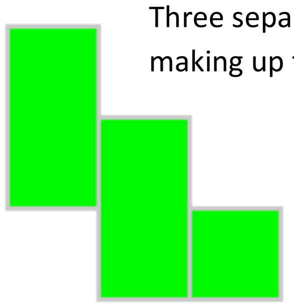
The five different tiles which are used in this tesselation are shown below. Note that the fifth tile (the red one in the diagram) is really a 'cheat' tile because it is used to fill any gaps in this tesselation :-)

Each tile has a type, a number from 1 to 5. Going from left to right, the green tile is type 1, the blue tile is type 2, the light blue tile is type 3, the purple tile is type 4 and the red tile is type 5. Each tile covers five squares of the grid (except for the red one which covers 1 square).



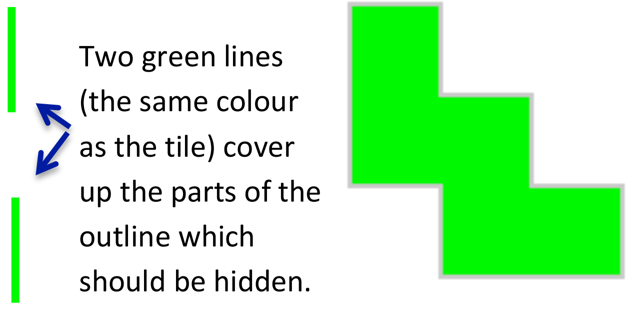
The arrows show the left, top position of each tile.

This part of this assignment requires you to write the code which draws each of these tiles. Each tile is made up of rectangles and the whole tile should have a "grey" outline of width 2. However, the problem with this is that the tile then has short grey lines cutting through the tile where the rectangles meet. In order to hide this part of the outline, cover the grey lines with a line (of width 2) of the same colour as the tile. The following diagrams (using type 1 tile) show you what is meant by this.



Three separate rectangles making up the type 1 tile.

The three rectangles are outlined and parts of the outline are visible inside the tile.



Two green lines (the same colour as the tile) cover up the parts of the outline which should be hidden.

The `draw_five_tiles()` function makes calls to the `draw_tile()` function which draws each of the tiles:

```
draw_tile(a_canvas, tile_type, left_value, top_value, size)
```

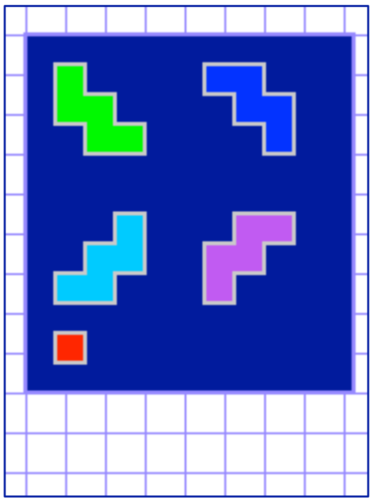
The `draw_tile()` function is passed five parameters:

- `a_canvas`: the canvas on which the tile is drawn,
- `tile_type`: the type number of the tile,
- `left`: the left position (how far from the left of the canvas) of the top-left position of the tile,
- `top`: the y position (how far down the canvas) of the top-left position of the tile,
- `size`: the size of each of the five squares making up the tile area.

The colour used to fill each tile is the colour at the index given by the `tile_type` number, e.g., type 1 tile has the colour given by `colours[1]`.

Complete the `draw_tile()` function. This function draws one of the five tiles. The parameter, `tile_type`, indicates which of the five tiles is drawn when this function is called. For example,

```
draw_tile(a_canvas, 1, 100, 60, 10)
```



draws a tile of type 1, in a green colour, with a top left position at 100, 60 and with a size of 10 (i.e., covering five grid squares of 10 pixels by 10 pixels).

Once you have coded this function correctly you will see the five tiles displayed inside the blue rectangle on the right hand side of the canvas area.

Assignment 4 Part 3 (3 marks):

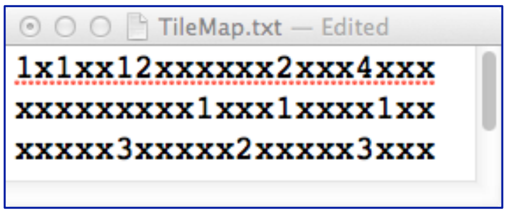
The file, `TileMap.txt`, contains lines of symbols:

- an 'x' letter – means that the grid square does not require any drawing of a tile,
- a number – means that a tile of that type number should be drawn starting at the top left position of the grid square.

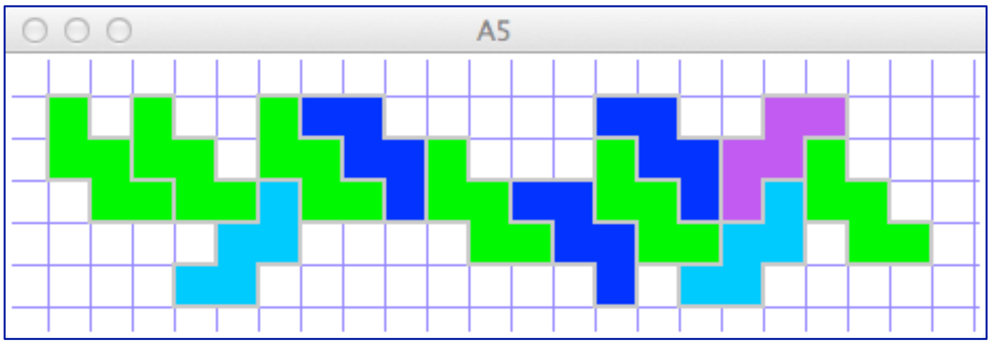
For example the line: `'1x1xx12xxxxxx2xxx4xxx'` means that in the row there should be:

a tile of type 1, followed by one grid space which does not require any tile drawing, followed by a tile of type 1, followed by 2 grid spaces which do not require tile drawing, followed by a tile of type 1, followed by a tile of type 2, followed by 6 grid spaces which do not require tile drawing, followed by a tile of type 2, followed by 3 non tile drawing grid squares, followed by a tile of type 4, and, finally, followed by 3 non tile drawing grid squares.

For example, the three lines of the file, `TileMap.txt`, shown on the right should display the following pattern in the canvas area.



```
1x1xx12xxxxxx2xxx4xxx
xxxxxxxxx1xxx1xxxx1xx
xxxxx3xxxxx2xxxxx3xxx
```



Complete the `process_single_line()` function. This function is passed a line of symbols (a string) and the function organises the drawing of one row of the tesselation, i.e., it should make calls to the `draw_tile()` function, 'telling' it to draw tiles in the correct positions along each row. Below is the function header:

```
def process_single_line(a_canvas, line_of_pattern, left, top, size):
```

This function is passed five parameters:

- `a_canvas`: the canvas on which the tiles are drawn,
- `line_of_pattern`: a string showing the sequence of tiles which should start along this row,
- `left`: the left position (how far from the left of the canvas) of the first square of the row,
- `top`: the y position (how far down the canvas) of the first square of the row,
- `size`: the size of each of the five grid squares making up the tiles.

At the end of the assignment, you may uncomment the line of code in the `main()` function:

```
#a_canvas.config(background="SlateBlue1")
```

Now you will see a SlateBlue1 coloured background and this 'hides' the grid lines.

THE PROGRAM SKELETON:

The skeleton of the program for this assignment is shown below. Two of the functions need to be completed by you. The rest of the program should not be changed except for two small changes, one required in Part 1 of the assignment and one when you have completed the assignment code. Copy the following code into your program.

```
from tkinter import *

# ------Draws one of the five different tiles.------
def draw_tile(a_canvas, tile_type, left, top, size):
    colours = ["yellow","green","blue"," deepskyblue", "purple","red","orange","cyan"]
    #This function needs to be completed

# ------Process each symbol from a single line (string). ------
def process_single_line(a_canvas, line_of_pattern, left, top, size):
    pass    #This function needs to be completed

# ------Organise the processing of the pattern. ------
def process_pattern(a_canvas, size):
    left = size
    top = size
    list_of_lines = get_list_of_pattern_lines("TileMap.txt")
    for line_string in list_of_lines:
        process_single_line(a_canvas, line_string, left, top, size)
        top += size

# ------Get the list of lines (strings) from the file. ------
def get_list_of_pattern_lines(filename):
    file_to_read = open(filename, "r")
    file_info = file_to_read.read()
    lines_list = file_info.split("\n")
    file_to_read.close()
    return lines_list

# ------Draws the five tiles on the right side of the canvas. ------

def draw_five_tiles(a_canvas, left, top, size):
    size = size * 3 // 4
    large_rect = (left, top, left + size * 11, top + size * 12)
    a_canvas.create_rectangle(large_rect, fill="Blue4",outline="SlateBlue1",width = 2)
    left_size_multiply = [0, 1, 6, 3, 7, 1]
    down_size_multiply = [0, 1, 1, 6, 6, 10]

    for tile_type in range(1, 6):
        left_value = left + size * left_size_multiply[tile_type]
        top_value = top + size * down_size_multiply[tile_type]
        draw_tile(a_canvas, tile_type, left_value, top_value, size)

# ------Draws the blue background grid lines of the given size. ------
def draw_grid(a_canvas, size, right_hand_side, bottom):
    for row in range(size, bottom, size):
        a_canvas.create_line(-1, row, right_hand_side + 1, row, fill="SlateBlue1")
    for col in range(size, right_hand_side, size):
        a_canvas.create_line(col, -1, col, bottom + 1, fill="SlateBlue1")
```

```
# ------main function. ------
def main():
    size = 20
    canvas_width = 700
    canvas_height = 340
    root = Tk()
    root.title("A5")
    geometry_string = str(canvas_width)+"x"+str(canvas_height)+"+10+20"
    root.geometry(geometry_string)
    a_canvas = Canvas(root)

    #a_canvas.config(background="SlateBlue1") #Uncomment when you have finished
    a_canvas.pack(fill=BOTH, expand = True) #Canvas fills the whole window
    #Draw the light blue background grid lines
    draw_grid(a_canvas, size, canvas_width, canvas_height)

    process_pattern(a_canvas, size)
    draw_five_tiles(a_canvas, canvas_width - size * 3 // 4 * 12, size, size)

    root.mainloop()

main()
```

Include this program in a module (file), named 'YourUPIA5.py', e.g., afer023A5.py.