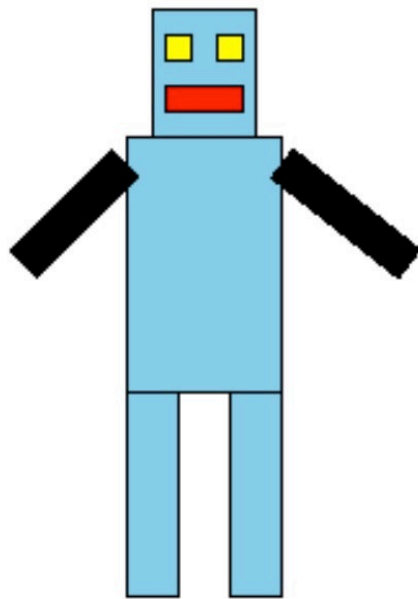
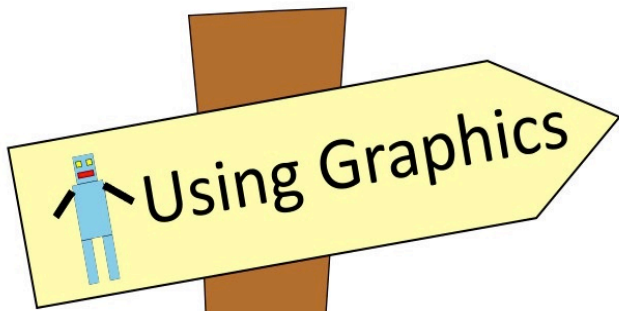


Creating with Python

Peter Kučera



```
min_number = int(entry1.get())
```

```
canvas.move('car',5,0)
```

```
button1=tkinter.Button(start,command='start')
```

```
canvas.bind('<Button-1>', click)
```

```
canvas.bind_all('<Up>', arrow_up)
```

```
elif y>300:
```

```
if y<200:
```

```
canvas.after(100, show)
```

```
def ball(x,y):
```

```
for i in range(10):
```

```
print(x)
```

```
x=i*10
```

```
canvas.create_oval
```

```
canvas.create_rectangle
```

```
import tkinter
```

```
canvas.create_line
```

Creating with Python

Using Graphics

Author © Peter Kučera

Design © Peter Kučera

Translation: Tomáš Jašek, Joshua Ruggiero

First published, 2016

Version number: 20161221

Publisher: Peter Kučera

Copyright © 2016 by Peter Kučera

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

For permission requests, write to the publisher at peter.kucera@gmail.com or via www.creatingwithpython.com.

Ordering Information:

Quantity sales. Special discounts are available on quantity purchases by corporations, associations, schools, and others. For details, contact the publisher at peter.kucera@gmail.com or via www.creatingwithpython.com.

Visit the author's website at www.creatingwithpython.com

ISBN 978-80-972537-3-8 (pdf)

ISBN 978-80-972537-4-5 (epub)

ISBN 978-80-972537-5-2 (mobi)

Contents

- [1 Introduction](#)
- [2 Installing the Python Environment and Starting IDLE](#)
- [3 Graphic Functions](#)
 - [3.1 Initializing the Canvas](#)
 - [3.2 Point Coordinates and Drawing Lines](#)
 - [3.3 Drawing Rectangles](#)
 - [3.4 Drawing Ellipses](#)
 - [3.5 Writing Text on the Canvas](#)
- [4 Variables and Random Values](#)
 - [4.1 Random Numbers in a Shell](#)
 - [4.2 Variables and Random Colors](#)
 - [4.3 Drawing Images at Random Positions](#)
- [5 Repeating Parts of the Program - For Loop](#)
 - [5.1 Repeated Execution of Statements](#)
 - [5.2 Using For Loops to Draw Regular Shapes](#)
- [6 Defining Functions](#)
- [7 Revision I](#)
- [8 Mouse and Keyboard Events](#)
 - [8.1 Left Mouse Button](#)
 - [8.2 Right Mouse Button](#)
 - [8.3 Keyboard Key Press Events](#)
- [9 Conditions](#)
- [10 Timer](#)
- [11 Buttons and Text Fields](#)
- [12 Moving Canvas Shapes](#)
- [13 Creating Simple Games](#)
 - [13.1 Reaction - Time Tester](#)
 - [13.2 Ball Catcher](#)
 - [13.3 Seeking Percentages](#)
- [14 Revision II](#)
- [15 Working with Text](#)
- [16 Exam - Knights' Race](#)
- [17 Notes](#)
 - [17.1 Using the Import Command](#)
 - [17.2 Random Colors](#)
 - [17.3 Text Field \(Entry\)](#)
- [18 Bibliography](#)

1 Introduction

About Python

Python was created in 1991 by Guido van Rossum, a Dutch programmer. Being a big fan of Monty Python's Flying Circus, Guido van Rossum named his newly created language Python.

At present, Python is a popular language and its popularity is continually growing. It is a modern programming language that supports different programming paradigms. Python is freeware and open-source. It is used by CERN, Google, Facebook, YouTube, Mozilla and others. It runs on different platforms, e. g. Linux, Windows or Mac. Python is taught at numerous top universities as the first programming language (MIT, Berkeley, etc.)

The purpose of this e-book

There are several approaches to the study of programming (e. g. a text-based interface, a graphical user interface, a mathematical context or Turtle geometry). The unique feature of this e-book is its use of graphics to teach programming to beginners.

The e-book is illustrative, containing many practical tasks using the graphic environment of the Tkinter library. Its content covers a basic programming course. We start by learning graphical shapes, and using a loop and conditions. Later we learn to react to mouse and keyboard events. Finally, we are able to create animations and simple games. At the end of the e-book, you can find a final exam to test your knowledge and skills.

Author's publications can be found and purchased here:

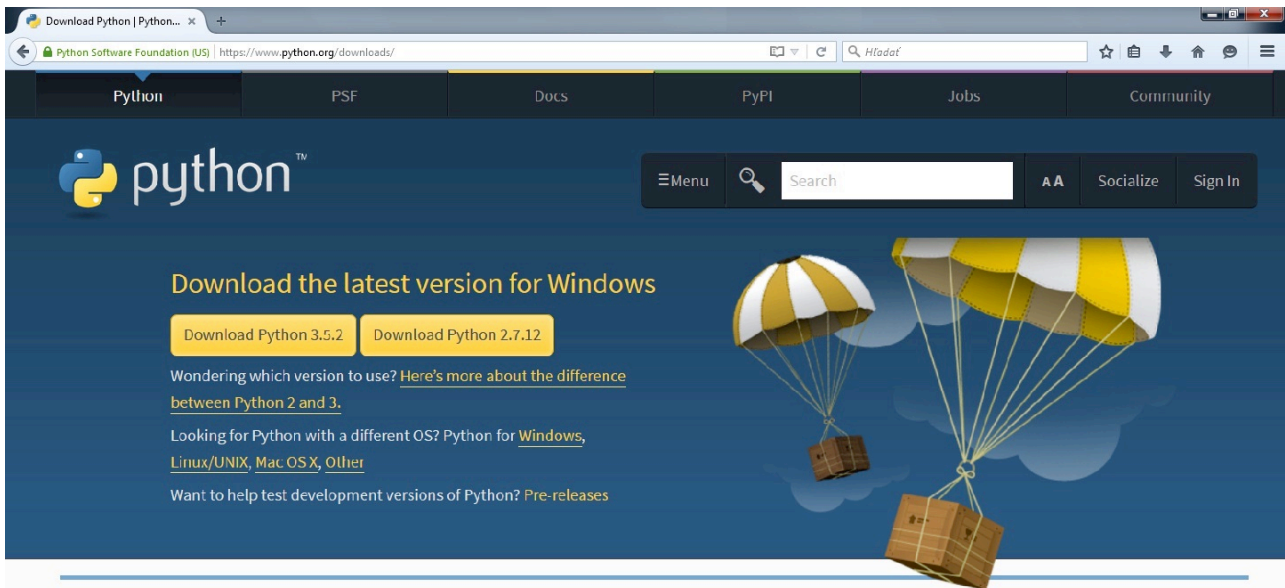
<http://www.creatingwithpython.com>

<https://www.facebook.com/creatingwithpython>

We wish you good luck with Python :)
Author

2 Installing the Python Environment and Starting IDLE

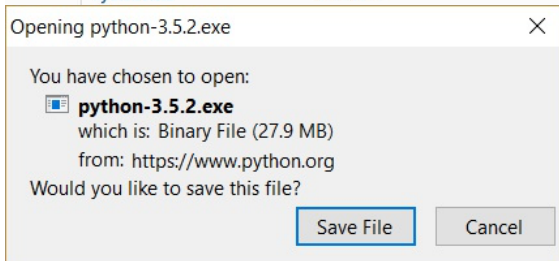
Download Python version 3.5.2 (or any other version starting with the number 3) from <http://www.python.org>. A 64-bit version is needed in case you have a 64-bit operating system. That can be easily verified in your system settings.



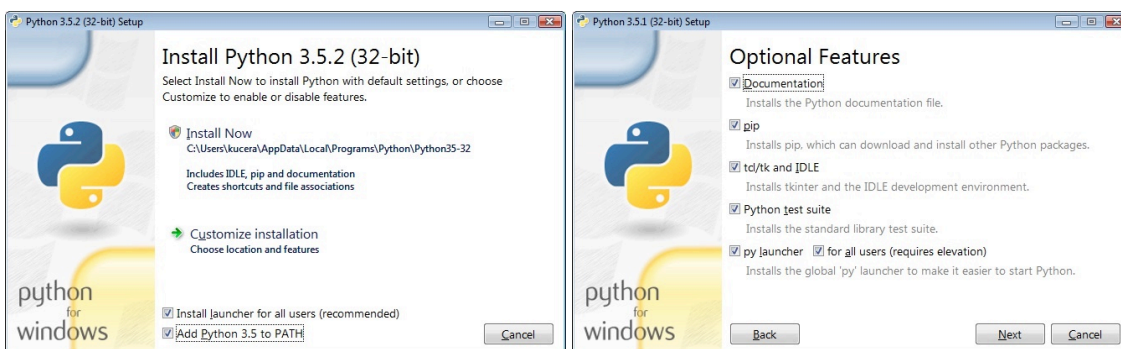
Looking for a specific release?

Python releases by version number:

Release version	Release date	Click for more	
Python 3.4.5	2016 06 27	Download	Release Notes
Python 3.5.2	2016-06-27	Download	Release Notes

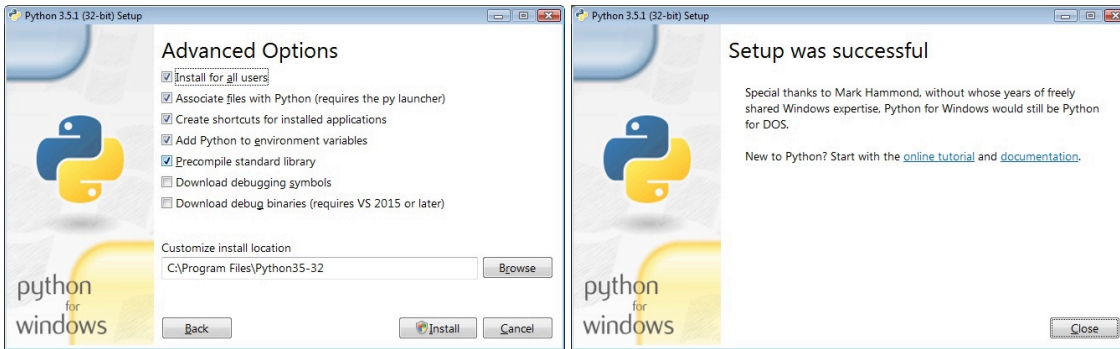


During the installation process, tick the "Customize installation" and "Add Python 3.5 PATH" checkboxes.

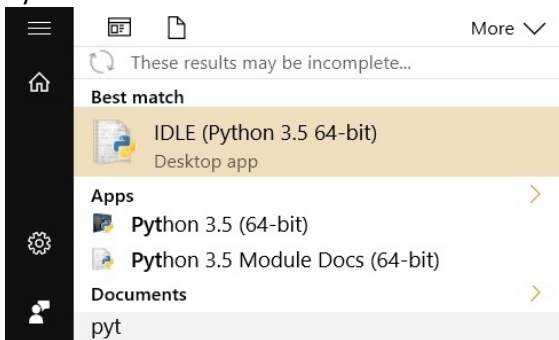


In the "Advanced Options" window, tick "Install for all users" if needed.

Note: School networks where each student uses their own account will most likely need this option. If you are not sure, please consult the person responsible for installing new software on the computers at your institution.



Python 3.5.2 is now installed. It can be started using the IDLE icon.



This opens up a Python Shell window. Python Shell is an interactive environment, which means it can execute commands right after they are typed in. Our programs will, however, be so long that it would be uncomfortable to enter each line into the shell separately. That is why we will use the editing mode to write our program, in a way similar to editing a regular text document. To get into editing mode, click File > New File in the shell window. The new window which has just opened up is the one used for writing a program. We can save the program using CTRL-S as usual or File > Save As from the menu. It is important to save the program to a file with a .py extension to enable syntax highlighting in editing mode. To run our empty program, press F5. Python outputs information about the program that has been started to a shell window. Running the program also resets all Python settings.

3 Graphic Functions

3.1 Initializing the Canvas

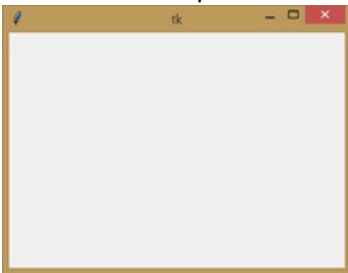
After starting IDLE and creating a new file, we are ready to start typing our first program.


In order to draw something on the screen, we need to create a window with a component called a canvas. To work with the canvas, we have to import a module that contains a definition of what the canvas is. In this book we will be using a `tkinter` module to work with canvas. To import this module, start your program with the statement `import tkinter`. After importing the module, we can start creating the canvas we will later use to draw our shapes on: `canvas = tkinter.Canvas()`. We use the statement `canvas.pack()` to show the canvas in a new application window.

This is what our program looks like so far:

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
```

Now we are ready to run our program. Press F5 to run the program. After a short delay, an empty window like this one shows up:

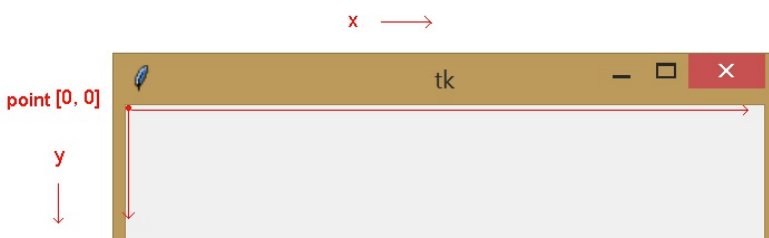


The window is empty because we have not drawn any shape on the canvas. Click  to close the window as usual.

3.2 Point Coordinates and Drawing Lines

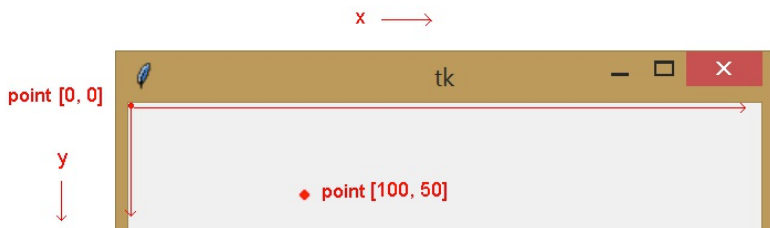
Various shapes like lines, rectangles or ellipses can be drawn on the canvas.

To draw a shape on the canvas, we need to specify its position using the coordinates of one or more points (depending on the shape).



Each point's position consists of 2 coordinates: x and y. The x coordinate determines the distance of the point from the left edge of the canvas, while the y coordinate determines the distance of the point from the top of the canvas. Coordinates are always written as two numbers.

The x coordinate is always the first number and the y coordinate the second.



Tasks:

1. What are the coordinates of a point located 20 pixels to the left from a point with the coordinates [100, 50]?
2. What are the coordinates of a point located 20 pixels to the right from a point with the coordinates [100, 50]?
3. What are the coordinates of a point located 20 pixels up from a point with the coordinates [100, 50]?
4. What are the coordinates of a point located 20 pixels down from a point with the coordinates [100, 50]?
5. What is the shared property of all points located to the left or to the right of [100,50]?

To draw a line from the point [10, 100] to the point [200, 100], we would use `canvas.create_line(10, 100, 200, 100)`. This function also allows us to draw lines through more than just 2 points.

We can specify more points by adding the coordinates of those points to the statement as follows:

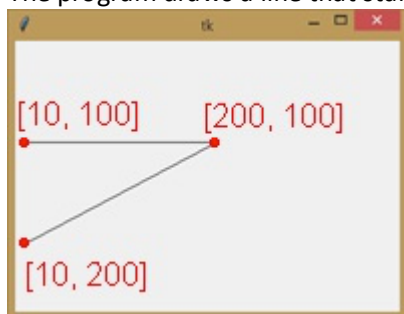
```
canvas.create_line(10, 100, 200, 100, 10, 200)
```

This statement extends the previous line to the additional point [10, 200].

After adding the statement that draws a line, our program looks like this:

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_line(10, 100, 200, 100, 10, 200)
```

The program draws a line that starts at point [10, 100], continues to [200, 10] and ends at [10, 200]:



Exercise:

- 1 Modify the program above to create a triangle.

While drawing a line, we can define the width of the line:

```
canvas.create_line(10, 100, 200, 100, width=5)
```

The statement above draws a line that is 5 pixels thick.

We can also change the color of the line:


```
canvas.create_line(10, 100, 200, 100, fill='red')
```

We can also combine both decorations to draw a thick red line like this:

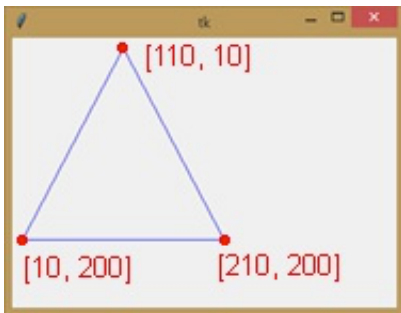
```
canvas.create_line(10, 100, 200, 100, fill='red', width=5)
```

or this:

```
canvas.create_line(10, 100, 200, 100, width=5, fill='red')
```

Both programs draw a red line which is 5 pixels thick. The parameter `width` can be used to set the thickness by a numeric value. Color can be set using the `fill` parameter, which accepts the name of the color enclosed in apostrophes. This supports more colors, for example:

```
'white', 'black', 'red', 'blue', 'yellow', 'green', 'maroon',  
'orange', 'gray', 'skyblue', 'violet', 'fuchsia', 'olive'.
```



There are multiple ways to draw the triangle above:

```
import tkinter  
canvas = tkinter.Canvas()  
canvas.pack()  
canvas.create_line(110, 10, 10, 200, fill='blue')  
canvas.create_line(10, 200, 210, 200, fill='blue')  
canvas.create_line(210, 200, 110, 10, fill='blue')
```

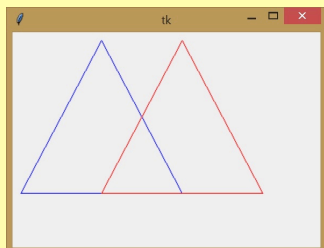
or

```
import tkinter  
canvas = tkinter.Canvas()  
canvas.pack()  
canvas.create_line(110, 10, 10, 200, 210, 200, 110, 10, fill='blue')
```

Exercises:

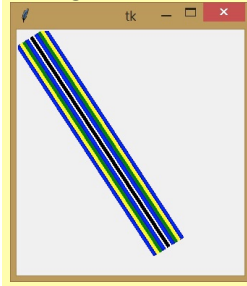
2 Try to come up with more ways to draw the triangle above.

3 Use red to draw an identical triangle next to the original blue one. The final drawing should look like this:



4

Change the color and thickness of the lines to draw the following pattern:



5

Draw the letters L, T, H and Z using lines.

6

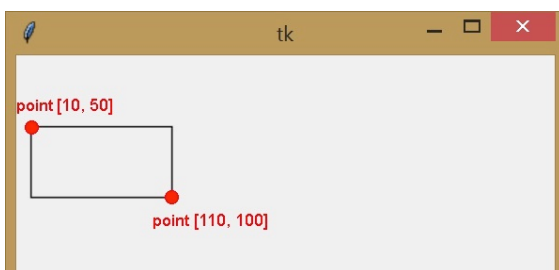
Draw a rectangle using lines.

3.3 Drawing Rectangles

Tasks:

6. We already know how to draw a rectangle using lines. What was the number of points (or numeric parameters) we had to pass to the `create_line` function to get a rectangle?
7. Is there a way to specify a rectangle using fewer points? What is the minimal number of points we have to know to specify a rectangle? Give arguments.

Similarly to lines, the canvas has a function to draw a rectangle, too. The following statement: `canvas.create_rectangle(10, 50, 110, 100)` draws a rectangle specified by points [10, 50] and [110, 100], like the one in the picture:



Tasks:

8. What is the width (in pixels) of the rectangle in the picture above?
9. What is the height (in pixels) of the rectangle in the picture above?
10. Does the following statement draw the same rectangle?
`canvas.create_rectangle(110, 100, 10, 50)`
11. Is there any other set of coordinates that draws the same rectangle in the same position?
12. What does this statement draw?
`canvas.create_rectangle(10, 50, 110, 50)`
13. What does this statement draw?
`canvas.create_rectangle(10, 50, 10, 50)`
14. What are the dimensions of the rectangles from Tasks 12 and 13?

Exercises:

7

Start your program with a rectangle using this statement:

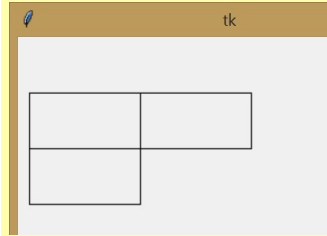
```
canvas.create_rectangle(10, 50, 110, 100).
```

Draw another rectangle of equal size to the right of the previous rectangle.

These 2 rectangles should touch, but not overlap.

8

Draw another rectangle of equal size below the leftmost rectangle from the previous exercise to create an arrangement like this one:



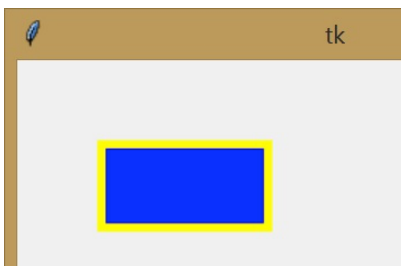
Tasks:

15. Do we need to work out all the coordinates of all the points of all the rectangles from exercises 7 and 8?
16. Think about adding a new rectangle to the regular grid-like arrangement, like we did in exercises 7 and 8. Are there any points in the new rectangle shared with points of the rectangles that already exist?

Similarly to drawing lines, various parameters can be used to change the appearance of a rectangle. However, the fill parameter works differently. This time, a fill parameter sets the inner color, filling the rectangle. It does not change the outline color as one might think. To set the outline color, we can add an outline parameter to our statement like this:

```
canvas.create_rectangle(50, 50, 150, 100, fill='blue', outline='yellow', width=5)
```

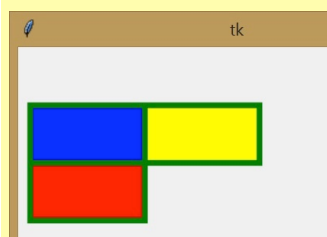
The statement above draws a blue rectangle with a 5 pixel thick yellow outline. Remember, that the order of the decoration parameters (fill, outline, width) is not important.



Exercises:

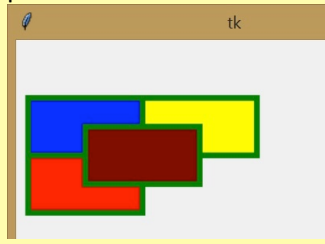
9

Take the rectangles from exercise 8 and set their parameters so that they look like the picture below. Use blue, yellow and red as fill colors, give it a green outline color and set the outline to be 5 pixels thick for all rectangles.



10

Add a fourth rectangle with the same size and a center at the point where the outlines of all the other rectangles intersect. Use brown to fill the rectangle and use the same outline properties as in the previous exercises.

**Tasks:**

17. What happens if we mix the decoration parameters with point coordinates as in the following statement?

```
canvas.create_rectangle(outline='yellow', 50, 50, 150, 100, fill='blue')
```

18. In exercise 10, we drew four rectangles using four statements. Is the order of those statements important? What would happen if we drew the brown rectangle before all the other rectangles? Which statements can be swapped to get the same result?

19. What happens if we set the outline parameter like this: `outline=''`?

20. What happens if we set the fill parameter like this: `fill=''`?

21. What happens if both fill and outline parameters are set like this: `fill='', outline=''`?

22. How many rectangles are we going to see after executing the following statements?

a) `canvas.create_rectangle(120, 100, 170, 150)`

`canvas.create_rectangle(100, 100, 150, 150)`

`canvas.create_rectangle(150, 100, 200, 150)`

`canvas.create_rectangle(100, 150, 150, 200)`

b) `canvas.create_rectangle(120, 100, 170, 150, fill='white')`

`canvas.create_rectangle(100, 100, 150, 150, fill='')`

`canvas.create_rectangle(150, 100, 200, 150, fill='green')`

`canvas.create_rectangle(100, 150, 150, 200, fill='', outline='')`

Exercises:**11**

Draw the flags of: a) Poland, b) France, c) Germany, d) Hungary, e) Switzerland and f) Israel.

12

Use at most two functions to draw the flag of Latvia.

**13**

Draw the flag of Greece.

**14**

Daniel wanted to draw a French flag and wrote this program. Try to find the mistake in his program:

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_rectangle(10, 50, 110, 300, fill='blue', outline='')
canvas.create_rectangle(60, 50, 160, 300, fill='white', outline='')
canvas.create_rectangle(110, 50, 210, 300, fill='red', outline='')
```

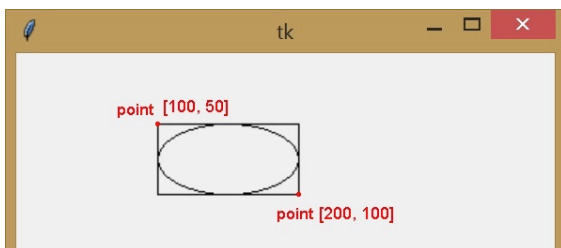
3.4 Drawing Ellipses

We already know that we can use the statement `canvas.create_rectangle(100, 50, 200, 100)` to draw a rectangle with a left top point at [100, 50] and a right bottom one at [200, 100]. If we use the same coordinates and change the function `canvas.create_rectangle` to `canvas.create_oval`, we get an ellipse.

The last statement of this program uses the same coordinates with a different function:

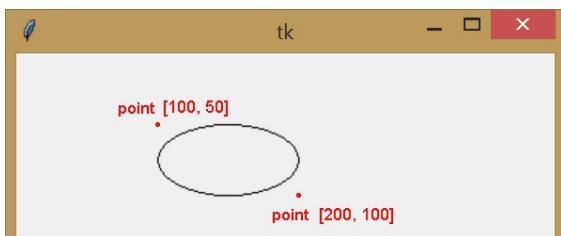
```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_rectangle(100, 50, 200, 100)
canvas.create_oval(100, 50, 200, 100)
```

The program above draws both a rectangle and an ellipse. As we can see in the picture below, the coordinates of the points which specify the ellipse are not inside of the ellipse. These points specify a rectangle, in which the ellipse is drawn. The rectangle is a minimum binding rectangle of the ellipse. The drawn ellipse is also the biggest ellipse that fits inside the binding rectangle.



`create_rectangle` statement can be removed to draw just the ellipse:

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_oval(100, 50, 200, 100)
```



The knowledge of binding rectangles is especially useful while working out the coordinates for an ellipse. We can start by drawing a binding rectangle at the coordinates and once we get the coordinates right, we can change the function to `create_oval`.

Tasks:

23. We can now draw a rectangle in one statement. Do we need a special function to draw a square? How can we tell that the coordinates passed to `create_rectangle` will draw a square?

24. Can we use `canvas.create_oval` to draw circles? Do we need a special function to draw circles? How can we tell that the coordinates passed to `create_oval` will draw a circle?

We might want to draw more shapes in a row on one canvas. Both the size of the canvas and its background color can be customized upon its creation. To change the size of the canvas, we can use the `width` and `height` parameters, which change the width and the height of the canvas. To change the background color, use the `bg` parameter.

```
import tkinter
canvas = tkinter.Canvas(bg='blue', width=800, height=600)
canvas.pack()
```

Exercises:

15 Draw another ellipse of equal size next to the ellipse drawn by the statement
`canvas.create_oval(100,50,200,100)`

16 Use different fill and outline colors for both ellipses from the previous exercise and also change their outline thickness.

17 Draw the flag of Japan.

18 Draw a snowman.

19 Draw a symmetric face with eyes, a nose and a mouth:



20 Add glasses to the face from the previous exercise.

21 Draw a flag with a red half-moon:



22 Draw a character with a hat:



23 Draw a colorful dart target.

3.5 Writing Text on the Canvas

To output some text on our canvas, we can use the `create_text` function. Let us start by looking at an example:

```
canvas.create_text(100, 70, text='Hello')
```

The statement above draws the text "Hello" at the coordinates [100, 70]. The text is aligned so that the coordinates are in its exact horizontal and vertical center. This point is also the center of the bounding rectangle of the text. The text parameter contains the text to be written on the canvas.

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_text(100, 70, text='Hello')
```



It is also possible to decorate text written on the canvas. For example, we can use a different font. That is what the font parameter is for. For example, we can specify `font='Arial 70 bold'`. However, the font parameter can only handle font names containing a single word. For example, 'Comic Sans' or 'Times New Roman' are impossible to enter this way. Advanced facilities would have to be used to specify font names that contain more words.

The following program:

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_text(100, 70, text='Hello')
canvas.create_text(200, 50, text='Python', font='Arial 70 bold')
```

draws these texts:



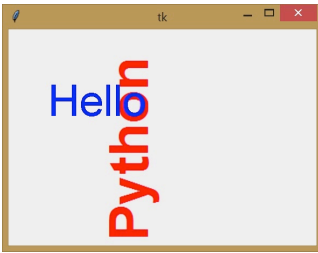
The well-known `fill` parameter can be used in the same way as in the previous functions to change the color of the text. The text can also be rotated using the `angle` parameter (Note: this feature is not available on the Mac OS).

```
canvas.create_text(100, 70, text='Hello', fill='blue')
canvas.create_text(200, 50, text='Python', font='Arial 70 bold', fill='red')
```



```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_text(150, 150, text='Python', font='Arial 50 bold', fill='red',
angle=90)
canvas.create_text(130, 90, text='Hello', fill='blue', font='Arial 40')
```

We have just rotated the "Python" text by 90 degrees. Setting a text angle rotates the text around its center (coordinates specified in create_text). The default direction for a rotation angle is counter-clockwise, but we can easily achieve a clockwise rotation using negative values. A 180-degree angle would rotate the text upside down.



Exercises:

24 Draw these european traffic signs:



25 Draw these european traffic signs:

