

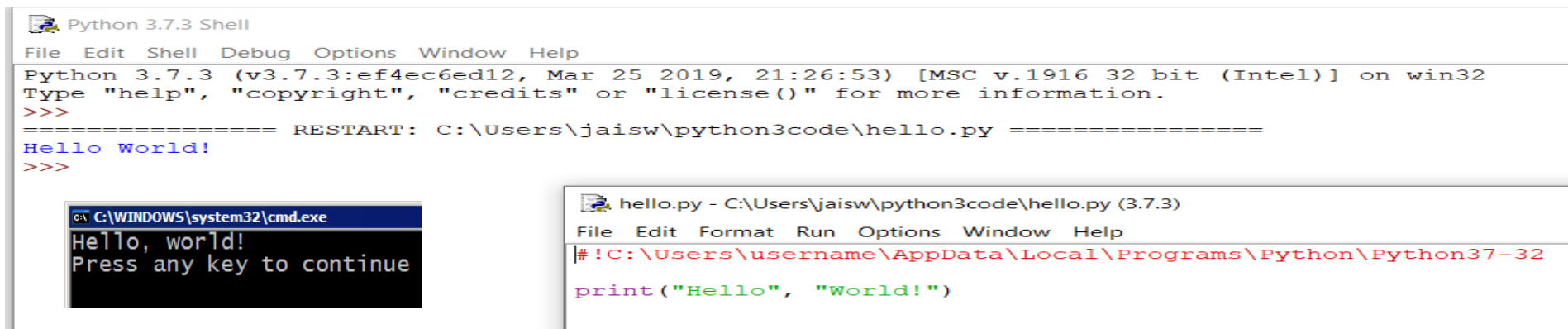
Introduction to Programming with Python

Recap



Programming basics

- **code** or **source code**: The sequence of instructions in a program.
- **syntax**: The set of legal structures and commands that can be used in a particular programming language.
- **output**: The messages printed to the user by a program.
- **console**: The text box onto which output is printed.
 - Some source code editors pop up the console as an external window, and others contain their own console window.



The image shows two overlapping windows. The top window is titled 'Python 3.7.3 Shell' and contains the following text:

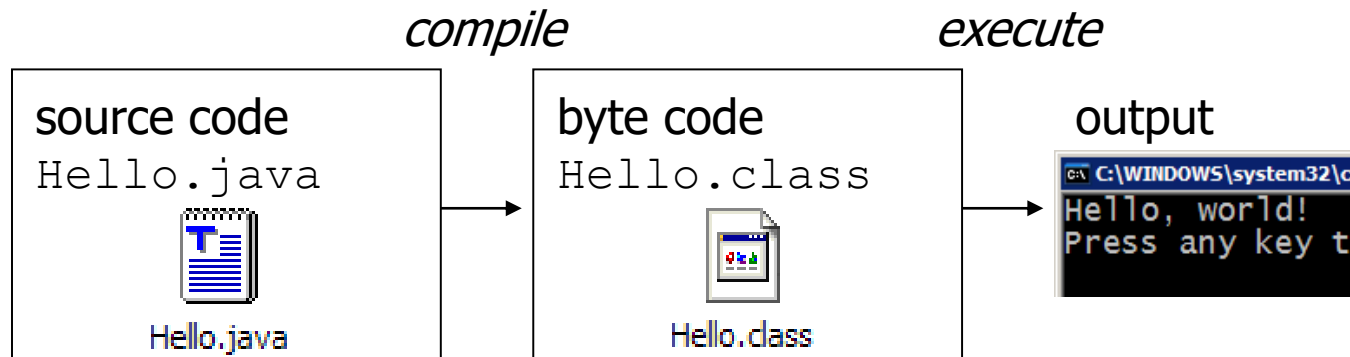
```
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\jaisw\python3code\hello.py =====
Hello World!
>>>
```

The bottom window is titled 'C:\WINDOWS\system32\cmd.exe' and contains the following text:

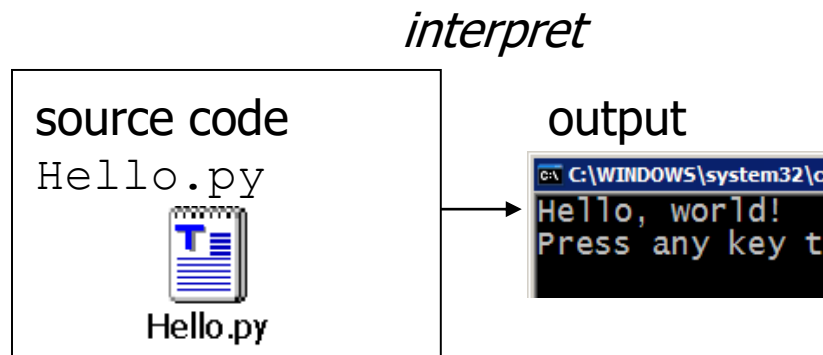
```
Hello, world!
Press any key to continue
```

Compiling and interpreting

- Many languages require you to *compile* (translate) your program into a form that the machine understands.



- Python is instead directly *interpreted* into machine instructions.



Expressions

- **expression:** A data value or set of operations to compute a value.

Examples: $1 + 4 * 3$
 42

- Arithmetic operators we will use:

$+$	$-$	$*$	$/$	addition, subtraction/negation, multiplication, division
$\%$				modulus, a.k.a. remainder
$**$				exponentiation

- **precedence:** Order in which operations are computed.

- $*$ $/$ $\%$ $**$ have a higher precedence than $+$ $-$

$1 + 3 * 4$ is 13

- Parentheses can be used to force a certain order of evaluation.

$(1 + 3) * 4$ is 16

Variables

- **variable:** A named piece of memory that can store a value.

- Usage:

- Compute an expression's result,
- store that result into a variable,
- and use that variable later in the program.



- **assignment statement:** Stores a value into a variable.

- Syntax:

name = value

- Examples:

$x = 5$

$\text{gpa} = 3.14$

x 5

gpa 3.14

- A variable that has been given a value can be used in expressions.

$x + 4$ is 9

- **Exercise:** Evaluate the quadratic equation for a given a , b , and c .

print

- `print` : Produces text output on the console.

- Syntax:

```
print "Message"
```

```
print Expression
```

- Prints the given text message or expression value on the console, and moves the cursor down to the next line.

```
print Item1, Item2, ..., ItemN
```

- Prints several messages and/or expressions on the same line.

- Examples:

```
print("Hello, world!")
```

```
age = 45
```

```
print("You have", 65 - age, "years until retirement")
```

Output:

```
Hello, world!
```

```
You have 20 years until retirement
```

input

- `input` : Reads a number from user input.
 - You can assign (store) the result of `input` into a variable.
 - Example:

```
age = input("How old are you? ")
print("Your age is", age)
print("You have", 65 - int(age), "years until retirement")
```

Output:

```
How old are you? 53
Your age is 53
You have 12 years until retirement
```
- **Exercise:** Write a Python program that prompts the user for his/her amount of money, then reports how many Xbox the person can afford, and how much more money he/she will need to afford an additional Xbox. Assume latest xBox one price \$ 450

99 Bottles

- 99 bottles of beer on the wall!
- 99 bottles of beer!
- Take one down
- And pass it around
- 98 bottles of beer on the wall!

- 98 bottles of beer on the wall!
- 98 bottles of beer!
- Take one down
- And pass it around
- 97 bottles of beer on the wall!

-
- 1 bottle of beer on the wall!
- 1 bottle of beer!
- Take it down
- And pass it around
- No more bottles of beer on the wall!

99 Bottles

```
for i in range(99, -1, -1):
    if i > 2:
        print ('{} bottles of beer on the wall!\n{} bottles of beer!\nTake one down\nAnd pass it around\n{} bottles of beer on the wall!\n\n'.format (i,i,i-1))
    elif i == 2:
        print ('{} bottles of beer on the wall!\n{} bottles of beer!\nTake one down\nAnd pass it around\n1 more bottle of beer on the wall!\n\n'.format
(i,i,))
    elif i == 1:
        print ('1 bottle of beer on the wall!\n1 bottle of beer!\nTake it down\nAnd pass it around\nNo more bottles of beer on the wall!')
print()
```

UI

What is a Module?

- Consider a module to be the same as a code library.
- A file containing a set of functions you want to include in your application.

- Example- How to create module
Save this code in a file named

File :mymodule.py

```
def greeting(name):  
    print("Hello, " + name)
```

- Example- How to use module

File : usemymodule.py

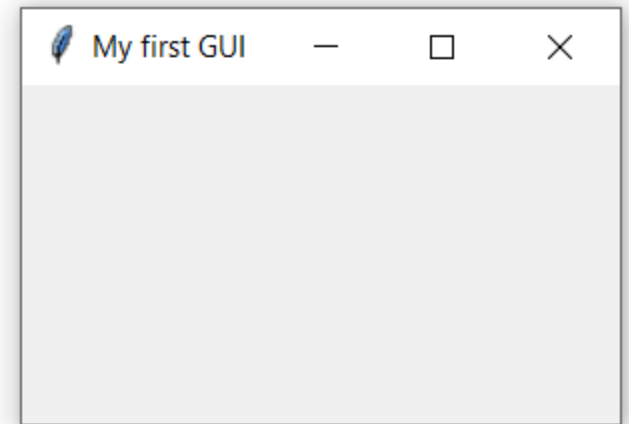
```
import mymodule as mm  
mm.greeting("Jonathan")
```

tkinter module ?

- Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

File :createwindow.py

```
import tkinter as tk
window = tk.Tk()
window.title("My first GUI")
window.geometry("800x600")
window.mainloop()
```



tkinter events ?

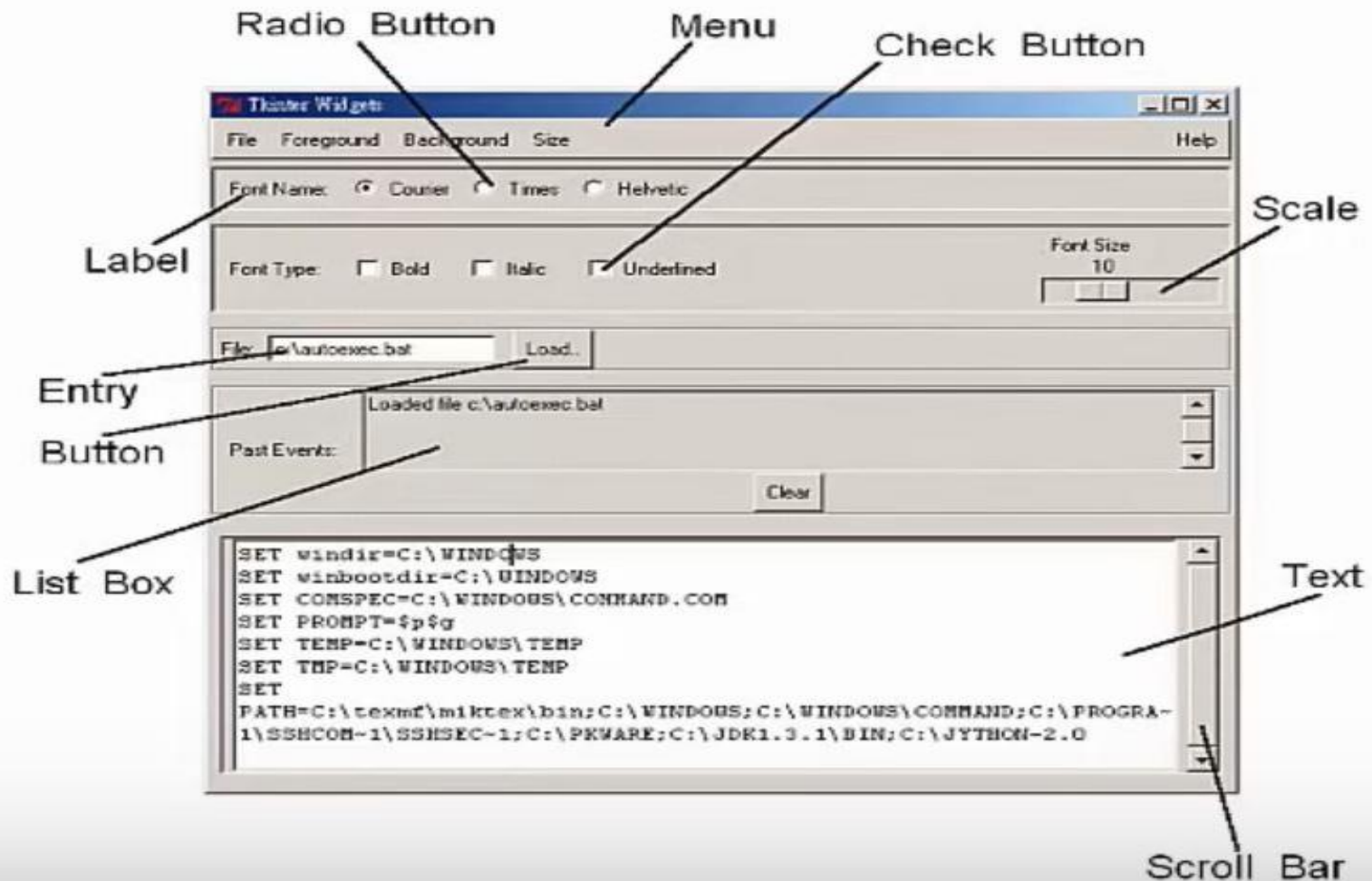
Tkinter Events and Binding

<Button-1>	- left mouse button	}	Mouse events
<Button-2>	- middle mouse button (on 3 button mouse)		
<Button-3>	- rightmost mouse button		
<B1-Motion>	- mouse moved with left button depressed		
<ButtonRelease-1>	- left button released		
<Double-Button-1>	- double click on button 1		
<Enter>	- mouse pointer entered widget	}	Keyboard events
<Leave>	- mouse pointer left the widget		
<FocusIn>	- Keyboard focus moved to a widget		
<FocusOut>	- Keyboard focus moved to another widget		
<Return>	- Enter key depressed		
<Key>	- A key was depressed		
<Shift-Up>	- Up arrow while holding Shift key		
<Configure>	- widget changed size or location		

tkinter widgets ?

Widget	Description
Button	Similar to a Label but provides additional functionality for mouse-overs, presses, and releases, as well as keyboard activity/events
Canvas	Provides ability to draw shapes (lines, ovals, polygons, rectangles); can contain images or bitmaps
Checkbutton	Set of boxes, of which any number can be "checked"
Entry	Single-line text field with which to collect keyboard input
Frame	Pure container for other widgets
Label	Used to contain text or images
LabelFrame	Combo of a label and a frame but with extra label attributes
Listbox	Presents the user with a list of choices from which to choose
Menu	Actual list of choices "hanging" from a Menubutton from which the user can choose
Menubutton	Provides infrastructure to contain menus (pulldown, cascading, etc.)
Message	Similar to a Label, but displays multiline text
PanedWindow	A container widget with which you can control other widgets placed within it
Radiobutton	Set of buttons, of which only one can be "pressed"
Scale	Linear "slider" widget providing an exact value at current setting; with defined starting and ending values
Scrollbar	Provides scrolling functionality to supporting widgets, for example, Text, Canvas, Listbox, and Entry
Spinbox	Combination of an entry with a button letting you adjust its value
Text	Multiline text field with which to collect (or display) text from user

tkinter widgets ?



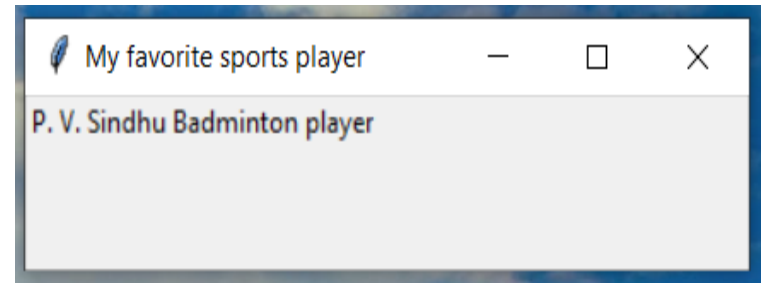
tkinter module-label ?

■ **Label**

A label is a class in Tkinter which is used to display text or image. It is a widget that the user just views but does not interact with.

File :label.py

```
import tkinter as tk
window=tk.Tk()
window.title("My favorite sports player")
window.geometry("300x200")
mylabel=tk.Label(text="P. V. Sindhu Badminton player")
mylabel.grid(column=0,row=0)
window.mainloop()
```



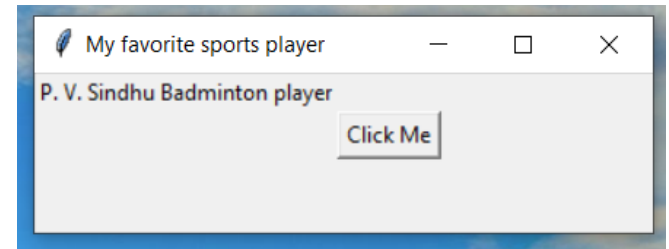
tkinter module -buttons?

■ Buttons

The Button widget is used to display the buttons in your application.

File :button.py

```
import tkinter as tk
window=tk.Tk()
window.title("My favorite sports player")
window.geometry("300x200")
mylabel=tk.Label(text="P. V. Sindhu Badminton player")
mylabel.grid(column=0,row=0)
button_name = tk.Button(window, text = "Click Me")
button_name.grid(column=3,row=3)
window.mainloop()
```



App1-You rang the doorbell

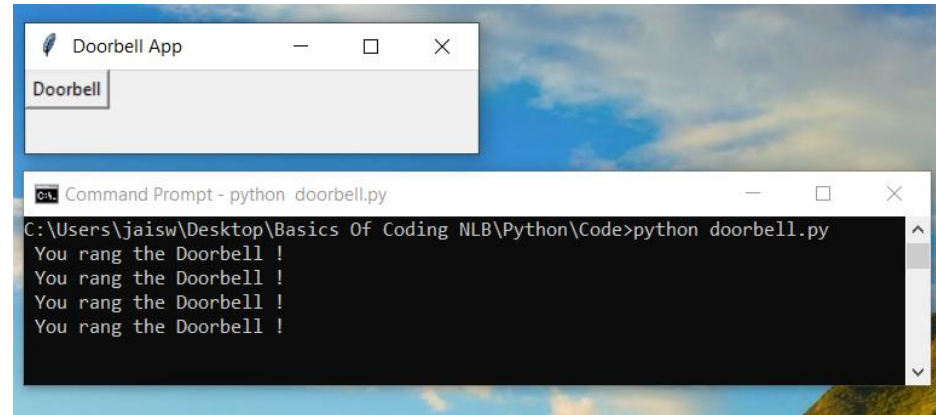
You rang the doorbell

Let's try an example to see how we can do these. We will create a doorbell app. When you click the button, it will print 'You rang the doorbell' on the window

parameter `<Button-1>` is the left click short key of the mouse.

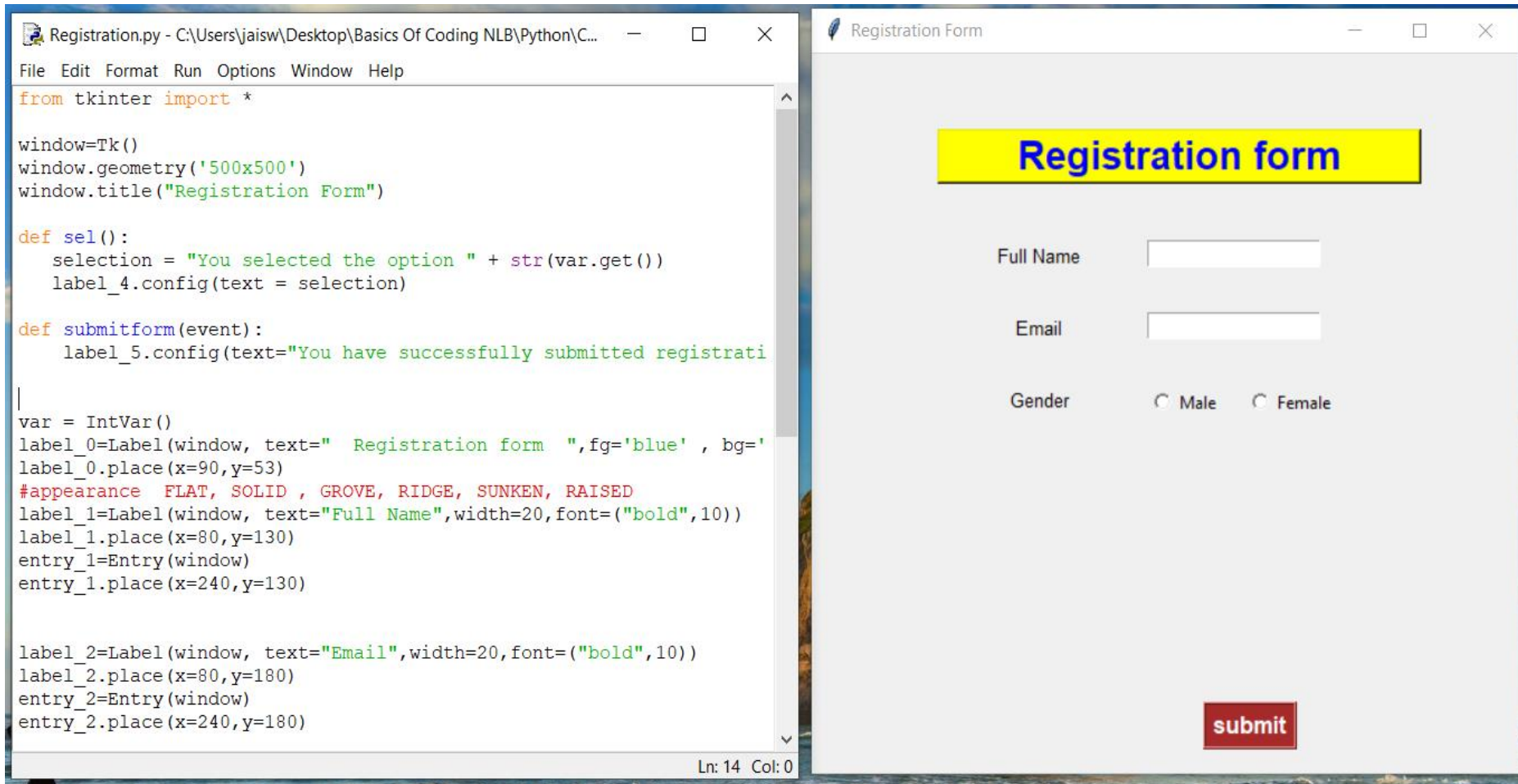
File :doorbell.py

```
import tkinter as tk
def doorbell(event):
    print(" You rang the Doorbell !")
window = tk.Tk()
window.title(" Doorbell App")
window.geometry("300x200")
mybutton = tk.Button(window, text = "Doorbell")
mybutton.grid(column=1,row=0)
mybutton.bind("<Button-1>",doorbell)
window.mainloop()
```



App2-Regisration Form

Registration Form



The image displays a Python Tkinter application for a registration form. It consists of two windows: a code editor on the left and the running application on the right.

Code Editor (Left Window):

```
Registration.py - C:\Users\jaisw\Desktop\Basics Of Coding NLB\Python\C...
File Edit Format Run Options Window Help

from tkinter import *

window=Tk()
window.geometry('500x500')
window.title("Registration Form")

def sel():
    selection = "You selected the option " + str(var.get())
    label_4.config(text = selection)

def submitform(event):
    label_5.config(text="You have successfully submitted registrati

|
var = IntVar()
label_0=Label(window, text="  Registration form  ",fg='blue' , bg='
label_0.place(x=90,y=53)
#appearance FLAT, SOLID , GROVE, RIDGE, SUNKEN, RAISED
label_1=Label(window, text="Full Name",width=20,font=("bold",10))
label_1.place(x=80,y=130)
entry_1=Entry(window)
entry_1.place(x=240,y=130)

label_2=Label(window, text="Email",width=20,font=("bold",10))
label_2.place(x=80,y=180)
entry_2=Entry(window)
entry_2.place(x=240,y=180)

Ln: 14 Col: 0
```

Running Application (Right Window):

The application window is titled "Registration Form". It features a yellow header bar with the text "Registration form" in blue. Below the header, there are three labels and their corresponding input fields:

- Full Name:** A text input field.
- Email:** A text input field.
- Gender:** Two radio buttons labeled "Male" and "Female".

At the bottom right of the window, there is a red button labeled "submit".

Registration Form

```
from tkinter import *

window=Tk()
window.geometry('500x500')
window.title("Registration Form")

def sel():
    selection = "You selected the option " + str(var.get())
    label_4.config(text = selection)

def submitform(event):
    label_5.config(text="You have successfully submitted registration form")

var = IntVar()
label_0=Label(window, text=" Registration form ",fg='blue' , bg='yellow', relief=RAISED, width=20,font=("arial",20,"bold"))
label_0.place(x=90,y=53)
#appearance FLAT, SOLID , GROVE, RIDGE, SUNKEN, RAISED
label_1=Label(window, text="Full Name",width=20,font=("bold",10))
label_1.place(x=80,y=130)
entry_1=Entry(window)
entry_1.place(x=240,y=130)

label_2=Label(window, text="Email",width=20,font=("bold",10))
label_2.place(x=80,y=180)
entry_2=Entry(window)
entry_2.place(x=240,y=180)
```

Registration Form

```
label_3=Label(window, text="Gender",width=20,font=("bold",10))
label_3.place(x=80,y=230)
r1=Radiobutton(window, text="Male", variable=var, value=1,command=sel)
r1.place(x=240,y=230)
r2=Radiobutton(window, text="Female", variable=var, value=2,command=sel)
r2.place(x=310,y=230)

label_4 = Label(window)
label_4.place(x=250,y=260)

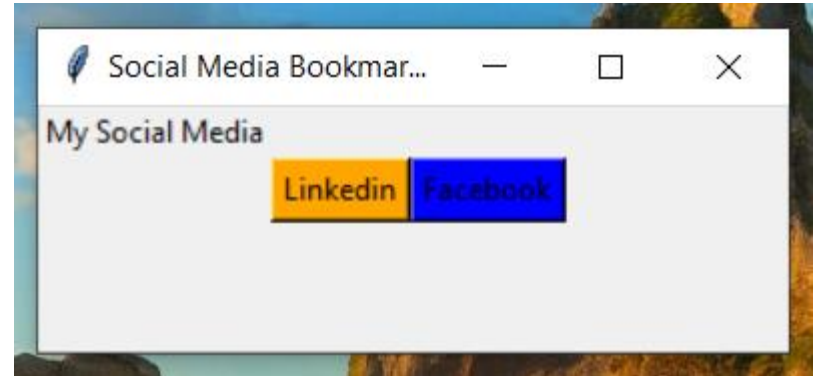
label_5 = Label(window)
label_5.place(x=100,y=360)

button_1=Button(window, text="submit", fg='white',bg='brown', relief=RIDGE, font=("arial",12,"bold"))
button_1.place(x=280, y=450)
button_1.bind("<Button-1>",submitform)

##GROVE, RIDGE, SUNKEN, RAISED
window.mainloop()
```


App3-My Social Media

My Social Media



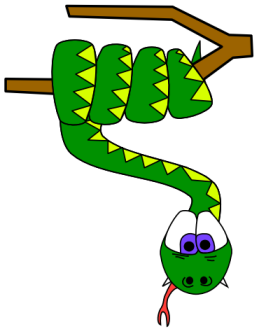
My Social Media

```
import tkinter as tk
import webbrowser
def linkedin(event):
    webbrowser.open_new_tab('https://www.linkedin.com/in/linkedinuserid/')
def facebook(event):
    webbrowser.open_new_tab('https://www.facebook.com/facebookuserid')
window=tk.Tk()
window.geometry("300x200")
window.title("Social Media Bookmark App")
label1=tk.Label(text="My Social Media")
label1.grid(column=0,row=0)

button1=tk.Button(window,text="Linkedin",bg="orange")
button1.grid(column=1,row=1)

button2=tk.Button(window,text="Facebook",bg="blue")
button2.grid(column=3,row=1)

button1.bind("<Button-1>",linkedin)
button2.bind("<Button-1>",facebook)
window.mainloop()
```



Repetition (loops) and Selection (if/else)

The for loop

- **for loop**: Repeats a set of statements over a group of values.

- Syntax:

```
for variableName in groupOfValues:  
    statements
```

- We indent the statements to be repeated with tabs or spaces.
- **variableName** gives a name to each value, so you can refer to it in the **statements**.
- **groupOfValues** can be a range of integers, specified with the `range` function.

- Example:

```
for x in range(1, 6):  
    print(x, "squared is", x * x)
```

Output:

```
1 squared is 1  
2 squared is 4  
3 squared is 9  
4 squared is 16  
5 squared is 25
```

range

- The `range` function specifies a range of integers:
 - `range(start, stop)` - the integers between **start** (inclusive) and **stop** (exclusive)
 - It can also accept a third value specifying the change between values.
 - `range(start, stop, step)` - the integers between **start** (inclusive) and **stop** (exclusive) by **step**

- Example:

```
for x in range(5, 0, -1):  
    print x  
print "Blastoff!"
```

Output:

```
5  
4  
3  
2  
1  
Blastoff!
```

- **Exercise:** How would we print the "99 Bottles of Beer" song?

Cumulative loops

- Some loops incrementally compute a value that is initialized outside the loop. This is sometimes called a *cumulative sum*.

```
sum = 0
for i in range(1, 11):
    sum = sum + (i * i)
print("sum of first 10 squares is", sum)
```

Output:

```
sum of first 10 squares is 385
```

- **Exercise:** Write a Python program that computes the factorial of an integer.

if

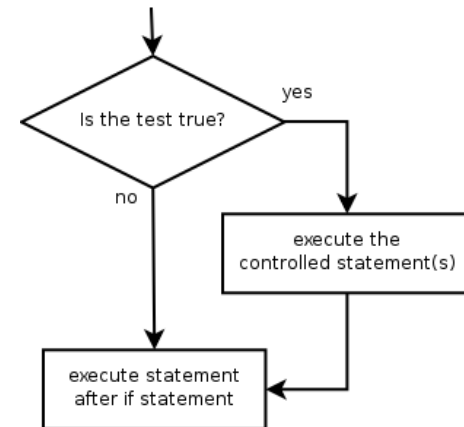
- **if statement:** Executes a group of statements only if a certain condition is true. Otherwise, the statements are skipped.

- Syntax:

```
if condition:  
    statements
```

- Example:

```
gpa = 3.4  
if gpa > 2.0:  
    print "Your application is accepted."
```



if/else

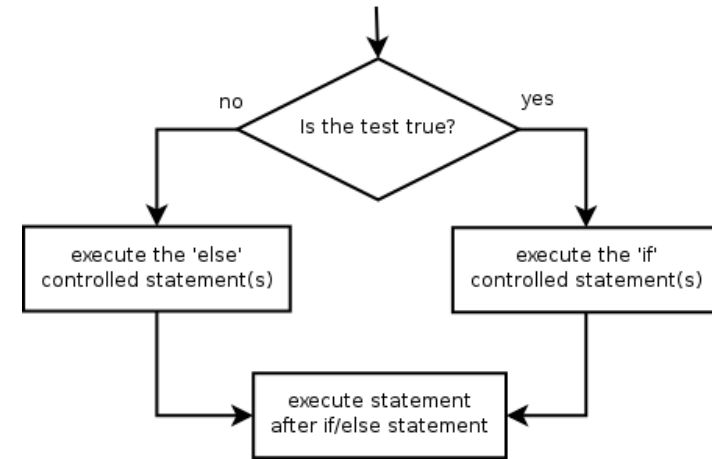
- **if/else statement:** Executes one block of statements if a certain condition is True, and a second block of statements if it is False.

- Syntax:

```
if condition:  
    statements  
else:  
    statements
```

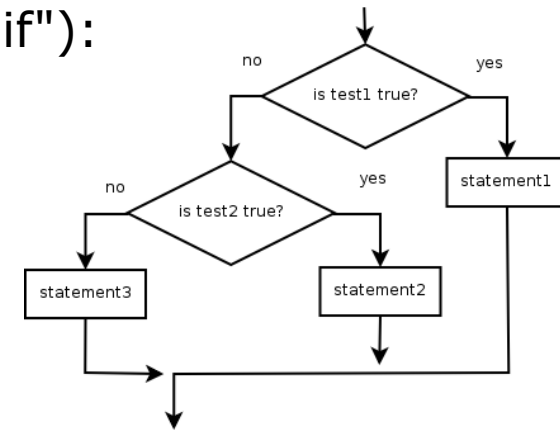
- Example:

```
gpa = 1.4  
if gpa > 2.0:  
    print "Welcome to Mars University!"  
else:  
    print "Your application is denied."
```



- Multiple conditions can be chained with `elif` ("else if"):

```
if condition:  
    statements  
elif condition:  
    statements  
else:  
    statements
```



while

- **while loop:** Executes a group of statements as long as a condition is True.
 - good for *indefinite loops* (repeat an unknown number of times)

- **Syntax:**

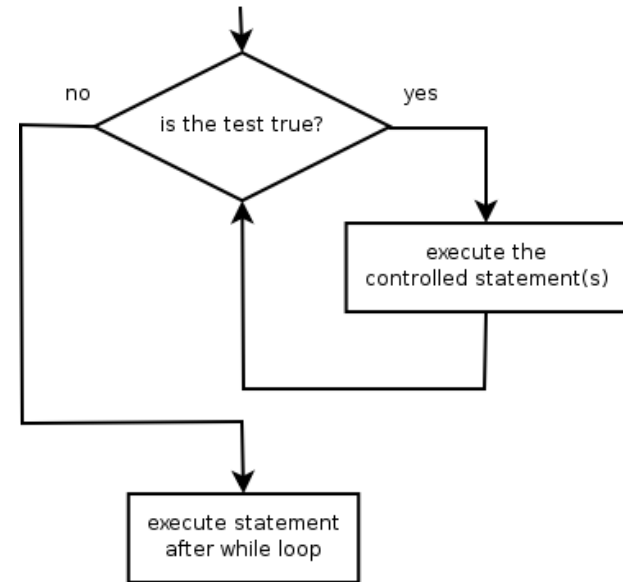
```
while condition:  
    statements
```

- **Example:**

```
number = 1  
while number < 200:  
    print number,  
    number = number * 2
```

- **Output:**

1 2 4 8 16 32 64 128



Logic

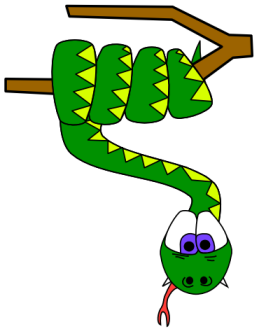
- Many logical expressions use *relational operators*:

Operator	Meaning	Example	Result
==	equals	<code>1 + 1 == 2</code>	True
!=	does not equal	<code>3.2 != 2.5</code>	True
<	less than	<code>10 < 5</code>	False
>	greater than	<code>10 > 5</code>	True
<=	less than or equal to	<code>126 <= 100</code>	False
>=	greater than or equal to	<code>5.0 >= 5.0</code>	True

- Logical expressions can be combined with *logical operators*:

Operator	Example	Result
and	<code>9 != 6 and 2 < 3</code>	True
or	<code>2 == 3 or -1 < 5</code>	True
not	<code>not 7 > 0</code>	False

- Exercise:** Write code to display and count the factors of a number.



Text and File Processing

Strings

- **string:** A sequence of text characters in a program.
 - Strings start and end with quotation mark " or apostrophe ' characters.
 - Examples:

```
"hello"  
"This is a string"  
"This, too, is a string.    It can be very long!"
```
- A string may not span across multiple lines or contain a " character.

```
"This is not  
a legal String."  
"This is not a "legal" String either."
```
- A string can represent characters by preceding them with a backslash.
 - \t tab character
 - \n new line character
 - \" quotation mark character
 - \\ backslash character
 - Example:

```
"Hello\tthere\nHow are you?"
```

Indexes

- Characters in a string are numbered with *indexes* starting at 0:

- Example:

```
name = "P. Diddy"
```

index	0	1	2	3	4	5	6	7
character	P	.		D	i	d	d	y

- Accessing an individual character of a string:

variableName [***index***]

- Example:

```
print name, "starts with", name[0]
```

Output:

```
P. Diddy starts with P
```

String properties

- `len(string)` - number of characters in a string (including spaces)
- `str.lower(string)` - lowercase version of a string
- `str.upper(string)` - uppercase version of a string

■ Example:

```
name = "Martin Douglas Stepp"  
length = len(name)  
big_name = str.upper(name)  
print big_name, "has", length, "characters"
```

Output:

```
MARTIN DOUGLAS STEPP has 20 characters
```

raw_input

- `raw_input` : Reads a string of text from user input.

- Example:

```
name = raw_input("Howdy, pardner. What's yer name? ")
print name, "... what a silly name!"
```

Output:

```
Howdy, pardner. What's yer name? Paris Hilton
Paris Hilton ... what a silly name!
```


Text processing

- **text processing:** Examining, editing, formatting text.
 - often uses loops that examine the characters of a string one by one
- A `for` loop can examine each character in a string in sequence.
 - Example:

```
for c in "booyah":  
    print c
```

Output:

```
b  
o  
o  
y  
a  
h
```

Strings and numbers

- `ord(text)` - converts a string into a number.
 - Example: `ord("a")` is 97, `ord("b")` is 98, ...
 - Characters map to numbers using standardized mappings such as *ASCII* and *Unicode*.
- `chr(number)` - converts a number into a string.
 - Example: `chr(99)` is "c"
- **Exercise:** Write a program that performs a rotation cypher.
 - e.g. "Attack" when rotated by 1 becomes "buubdl"

File processing

- Many programs handle data, which often comes from files.
- Reading the entire contents of a file:

```
variableName = open ("filename") .read()
```

Example:

```
file_text = open("bankaccount.txt").read()
```

Line-by-line processing

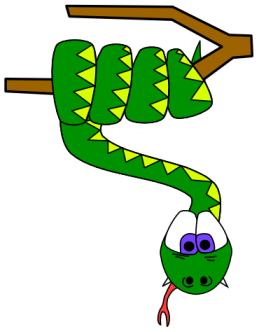
- Reading a file line-by-line:

```
for line in open("filename").readlines():  
    statements
```

Example:

```
count = 0  
for line in open("bankaccount.txt").readlines():  
    count = count + 1  
print "The file contains", count, "lines."
```

- **Exercise:** Write a program to process a file of DNA text, such as:
 ATGCAATTGCTCGATTAG
 - Count the percent of C+G present in the DNA.



Graphics

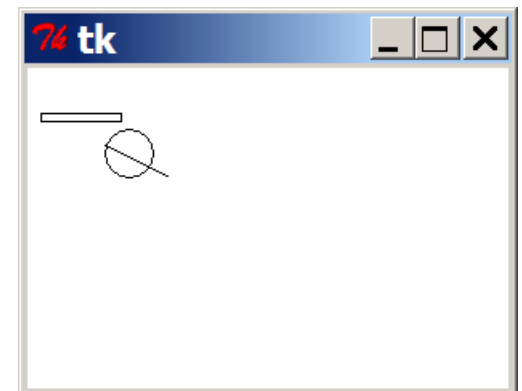
DrawingPanel

- To create a window, create a `drawingpanel` and its graphical pen, which we'll call `g` :

```
from drawingpanel import *  
panel = drawingpanel(width, height)  
g = panel.get_graphics()  
  
... (draw shapes here) ...  
  
panel.mainloop()
```

- The window has nothing on it, but we can draw shapes and lines on it by sending commands to `g` .
 - Example:

```
g.create_rectangle(10, 30, 60, 35)  
g.create_oval(80, 40, 50, 70)  
g.create_line(50, 50, 90, 70)
```



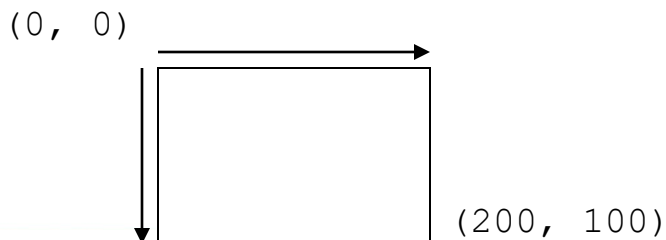
Graphical commands

Command	Description
<code>g.create_line(x1, y1, x2, y2)</code>	a line between (x1 , y1), (x2 , y2)
<code>g.create_oval(x1, y1, x2, y2)</code>	the largest oval that fits in a box with top-left corner at (x1 , y1) and bottom-left corner at (x2 , y2)
<code>g.create_rectangle(x1, y1, x2, y2)</code>	the rectangle with top-left corner at (x1 , y1), bottom-left at (x2 , y2)
<code>g.create_text(x, y, text="text")</code>	the given text at (x , y)

- The above commands can accept optional outline and fill colors.

```
g.create_rectangle(10, 40, 22, 65, fill="red", outline="blue")
```

- The coordinate system is y-inverted:



Drawing with loops

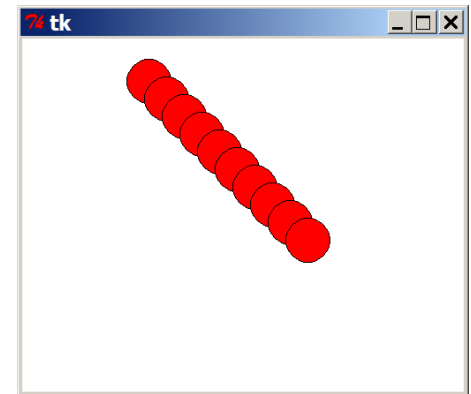
- We can draw many repetitions of the same item at different x/y positions with `for` loops.
 - The x or y assignment expression contains the loop counter, `i`, so that in each pass of the loop, when `i` changes, so does x or y.

```
from drawingpanel import *

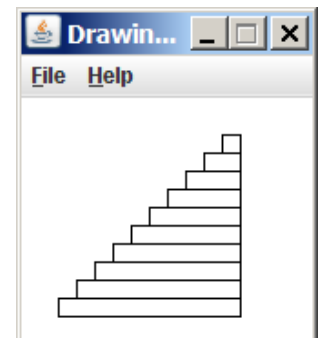
window = drawingpanel(500, 400)
g = window.get_graphics()

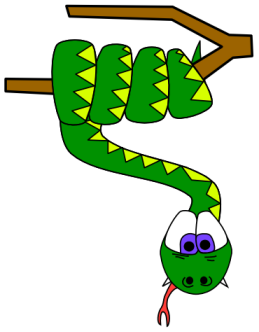
for i in range(1, 11):
    x = 100 + 20 * i
    y = 5 + 20 * i
    g.create_oval(x, y, x + 50, y + 50, fill="red")

window.mainloop()
```



- **Exercise:** Draw the figure at right.





What's Next?