Day – 10

Database programming with Python

Easy Shop…

- Easy shop want to store their Suppliers, Items and Quotations details in a database application and manipulate information's stored in database.

Suppliers	Supplier ID	Supplier Name	Supplier Contact No	Supplier Email ID

Item	Item Code	Item Type	Description	Price	Reorder Level	Quantity on hand	Category

Quotation	Quotation ID	Supplier ID	Item Code	Quoted Price	Quotation Date	Quotation Status

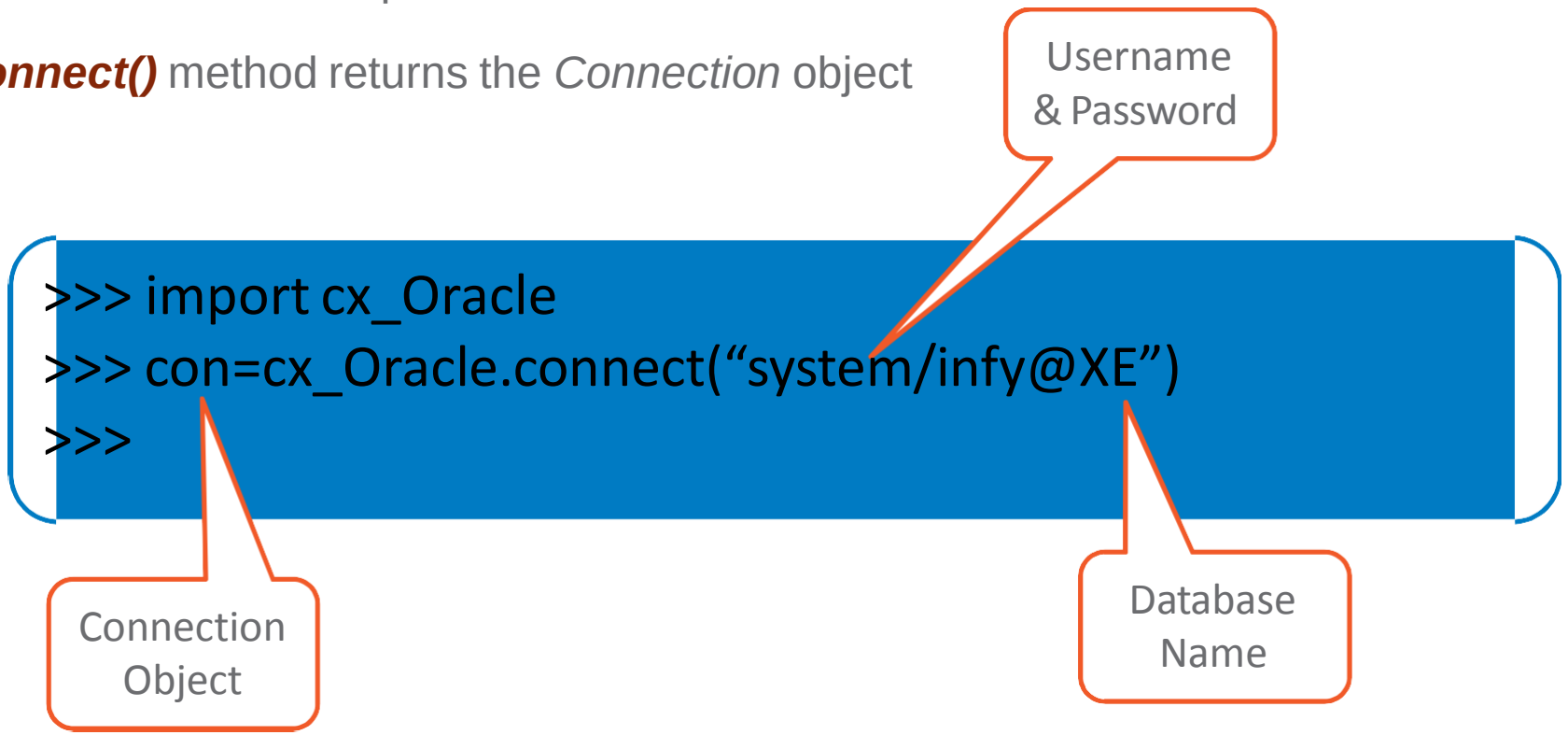
Database – Modules

- Database Modules for different database applications used as back end data storage.
  - **Cx_Oracle** for Oracle database
  - **Pydb2** for DB2 database
  - **MySQLdb** for MySQL database
  - **Adodbapi** for Microsoft SQL Server
  - **PyPyODBC** for Microsoft Access
  - **mxODBC** for Teradata

Download and import cx_Oracle in Python

Connect to Oracle Database

- **Connect()** - to establish connection with database application using its database name, username and password.
- ***Connect()*** method returns the *Connection* object



Disconnect Database

- Connected Database must be terminated by end of the program using close() method.
  - **ConnectionObject.close()**

```
>>> import cx_Oracle
>>> con=cx_Oracle.connect("system/infy@XE")
>>>
>>>
>>>
>>> con.close()
```

Disconnect Connection

Commit the changes done in database

- Commit() used to save the changes made in database otherwise the changes will not be reflected in database.
  - **ConnectionObject.commit()**

```
>>> import cx_Oracle
>>> con=cx_Oracle.connect("system/infy@XE")
>>>
>>>
>>> con.commit()
>>> con.close()
```

Save changes in Database

Rollback the changes done in database

- Rollback() used to reverse the changes made in database
  - **ConnectionObject.rollback()**

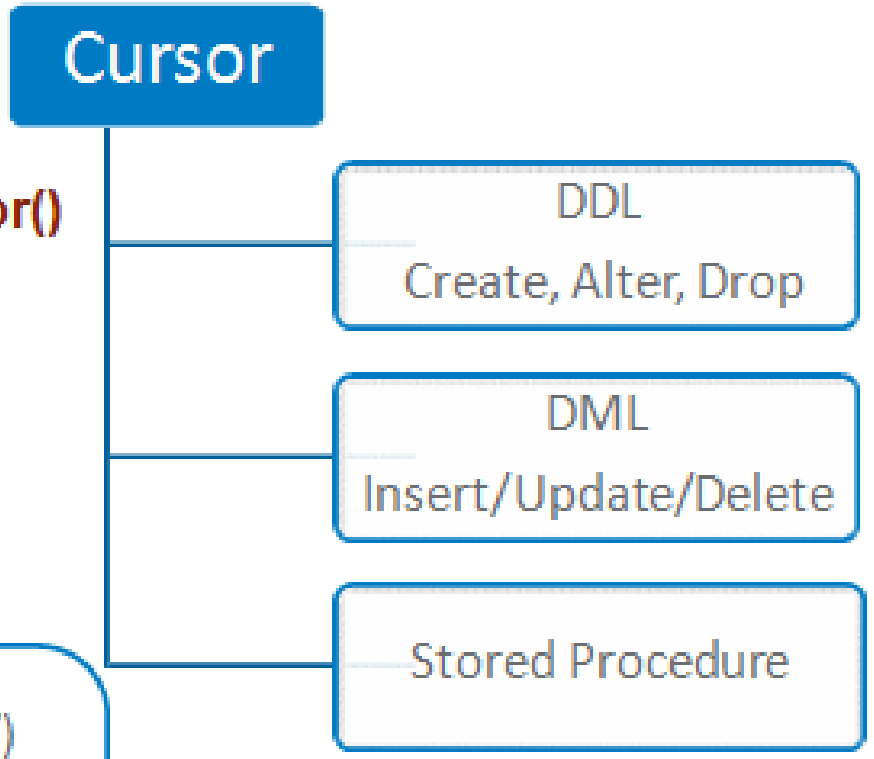
```
>>> import cx_Oracle
>>> con=cx_Oracle.connect("system/infy@XE")
>>>
>>>
>>> con.rollback()
>>> con.close()
```

Rollback changes in Database

Cursor for performing Operation in Database

- Cursor() - we need a Cursor object to do database operations.

CursorObject = ConnectionObject.Cursor()




```
>>> import cx_Oracle
>>> con=cx_Oracle.connect("system/infy@XE")
>>> cur=con.cursor()
>>>
>>> con.rollback()
>>> con.close()
```

Cursor Methods

Execute(Query)

Execute() methods is used to execute the SQL Commands.

- CursorObject.execute(Query, Bindvariable)

Fetchall()

Fetches all the remaining rows from the result-set and returns as a *list* of *tuples*.
If no rows remaining in the result-set, it returns an empty *list*.

Fetchmany(x)
X=arraysize

Fetches x rows out of total number of remaining row.
If x value > count of remaining number of rows, returns all the remaining rows
If invoked without any args, it returns arraysize number of rows. Default is 50

Fetchone()

Fetches single row from result-set and returns it as a tuple.
If no more rows are available, it returns None

Cursor Methods (Cont…)

Execute(Query)

Execute() – Create Query

```
import cx_Oracle
con=cx_Oracle.connect(" system/infy@XE ")
cur=con.cursor()
cur.execute("""CREATE TABLE supplier(
            supplierid VARCHAR2(6) PRIMARY KEY,
            suppliername VARCHAR2(30),
            suppliercontactno varchar2(15),
            supplieremailid VARCHAR2(30)
            )
                        """)
con.close()
```

Cursor Methods (Cont…)

Execute(Query)

Execute() – Insert Query

```
import cx_Oracle
con=cx_Oracle.connect("
system/infy@XE ")  cur=con.cursor()
cur.execute("""INSERT INTO supplier VALUES ('S1001','Giant Store','203-
237-2079',  'rachel1@easy.com')""")
cur.execute("""INSERT INTO supplier VALUES  ('S1002','EBATs','115-340-
2345','ebats@easy.com')""")
cur.execute("""INSERT INTO supplier VALUES ('S1003','Shop Zilla','203-
123-3456',  'shopzilla@easy.com')""")
cur.execute("""INSERT INTO supplier VALUES ('S1004','VV Electronics','115-
340-6756',  'vvelectronics@easy.com')""")
con.close()
```

Cursor Methods (Cont…)

Execute(Query)

Execute() – Update Query

```
import cx_Oracle
con=cx_Oracle.connect("
system/infy@XE ")  cur=con.cursor()
cur.execute("""UPDATE supplier SET
supplieremailid='batse@easy.com' where  supplierid='S1002'""")
con.close()
```

Cursor Methods (Cont…)

Execute(Query)

Execute() – Delete Query

```
import cx_Oracle
con=cx_Oracle.connect("
system/infy@XE ")  cur=con.cursor()
cur.execute("""delete from supplier where
supplierid='S1003"""")  con.close()
```

Cursor Methods (Cont…)

Execute(Query)

Execute() – Select Query

```
import cx_Oracle
con=cx_Oracle.connect("
system/infy@XE ")  cur=con.cursor()
cur.execute("""Select * from
supplier""")  con.close()
```

Cursor Methods (Cont…)

Execute(Query)

Fetchall() - Fetch Records

```
import cx_Oracle
con=cx_Oracle.connect("system/infy@XE")
cur=con.cursor()
cur.execute("""Select * from supplier""")
print (cur.fetchall())
con.commit()
con.close()
```

Output

```
>>>
[('S1001', 'Giant Store', '203-237-2079', 'rachel1@easy.com'),
('S1002', 'EBATs', '115-340-2345', 'ebats@easy.com'), ('S1003',
 'Shop Zilla', '203-123-3456', 'shopzilla@easy.com'), ('S1004',
 'VV Electronics', '115-340-6756', 'vvelectronics@easy.com')]
```

Cursor Methods (Cont…)

Execute(Query)

Fetchall() - Fetch Records based on criteria:

```
import cx_Oracle
con=cx_Oracle.connect("system/infy@XE")
cur=con.cursor()
cur.execute("""Select * from supplier where supplierid='S1001'""")
print (cur.fetchall())
con.commit()
con.close()
```

Output

```
>>>
[('S1001', 'Giant Store', '203-237-2079', 'rachel1@easy.com')]
```

Cursor Methods (Cont…)

Execute(Query)

Fetchmany() – Fetch n Records from resultset:

```
import cx_Oracle
con=cx_Oracle.connect("system/infy@XE")
cur=con.cursor()
cur.execute("""Select supplierid from supplier""")
print (cur.fetchmany(2))
print (cur.fetchmany(1))
print (cur.fetchmany(2))
con.commit()
con.close()
```

Output

```
>>>
[('S1001',), ('S1002',)]
[('S1003',)]
[('S1004',)]
```

Cursor Methods (Cont…)

Execute(Query)

Fetchone() – Fetch one record from resultset:

```
import cx_Oracle
con=cx_Oracle.connect("system/infy@XE")
cur=con.cursor()
cur.execute("""Select supplierid,suppliername from supplier""")
print (cur.fetchone())
print (cur.fetchone())
print (cur.fetchone())
con.commit()
con.close()
```

Output

```
>>>
('S1001', 'Giant Store')
('S1002', 'EBATs')
('S1003', 'Shop Zilla')
```

Day -11

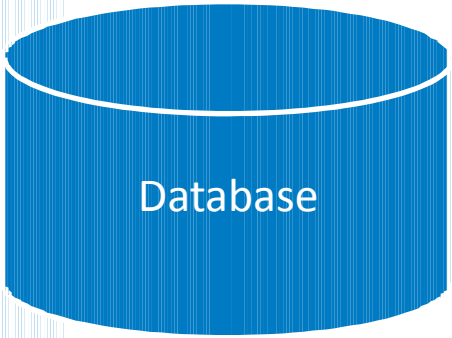
GUI and CGI Programming

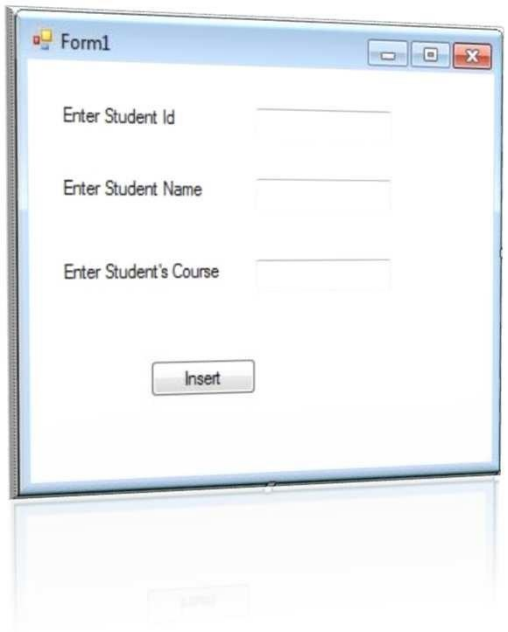
GUI Programming

- Graphical User Interface Programming using Python is:

Easy Shop…

- Easy shop want to design a front end application for their end user to interact with the information's stored in the database.





GUI Toolkit

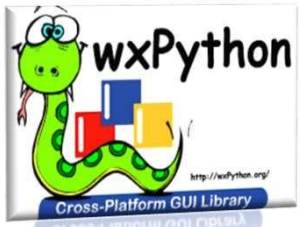
- GUI tool kit basically contains various:

Containers
Frame, window etc.,

Basic Controls
Buttons, Textboxes, Checkboxes etc.,

List Controls

- There are many tool kits available for Python, like:



Hello World in Tkinter

- Let us design our first Welcome Tkinter program with GUI

```
from tkinter import *

root = Tk()

w = Label(root, text="Welcome to Tkinter!")
w.pack()

root.mainloop()
```

Output

Hello World in Tkinter (Continued …)

- Let us modify our first Welcome to Python program with OOP

```
from tkinter import Tk, Label, Button
class MyFirstGUI:
    def __init__(self, root):
        self.root = root
        root.title("GUI with Label and Button")

        self.label = Label(root, text="This is our first GUI!")
        self.label.pack()

        self.greet_button = Button(root, text="Greet", command=self.greet)
        self.greet_button.pack()

    def greet(self):
        print("Greetings!")

root = Tk()
my_gui = MyFirstGUI(root)
root.mainloop()
```

Importing Label and Button from Tkinter

Creating a class with constructor and root is the Tk window

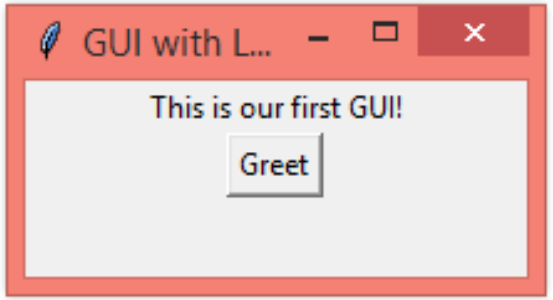
Title for the Parent Window

Label is added with Text

Button is added with text and command that call greet function

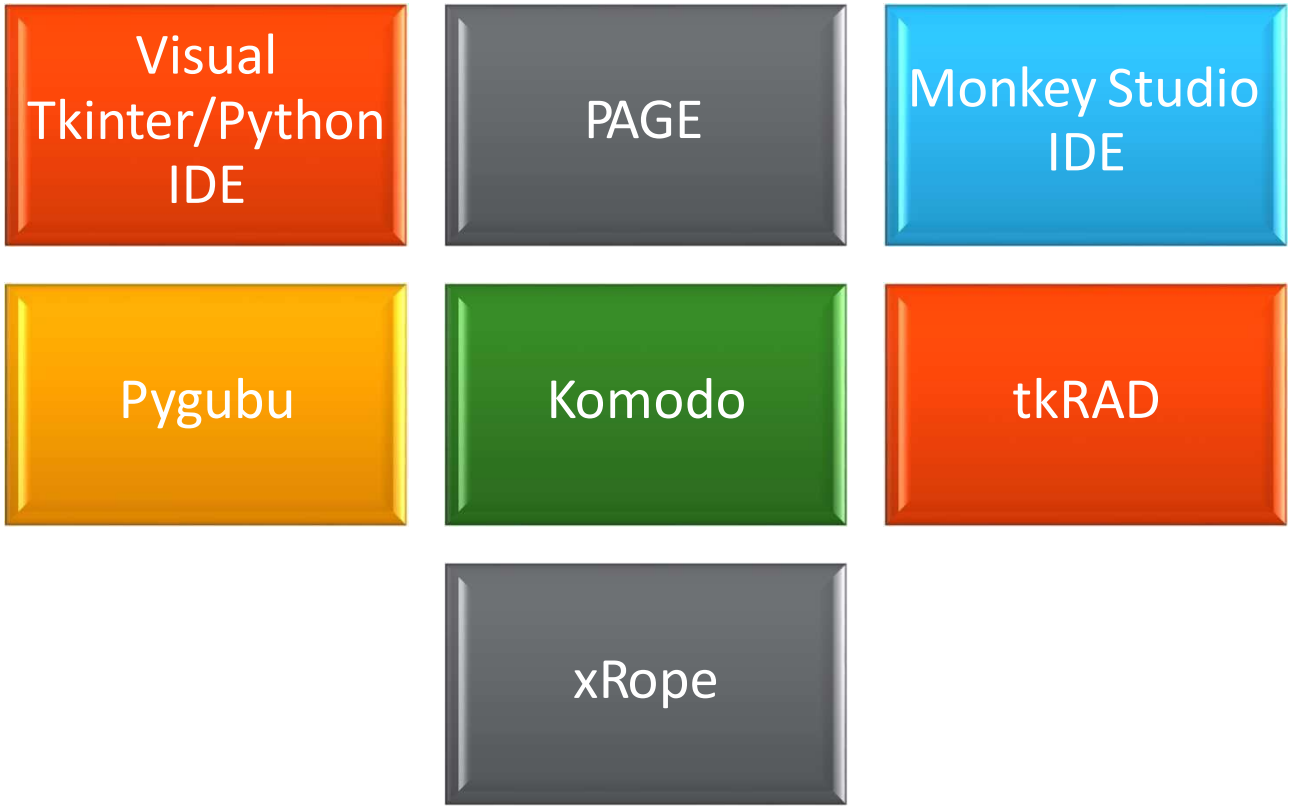
Object Creation for Class

Calling Mainloop

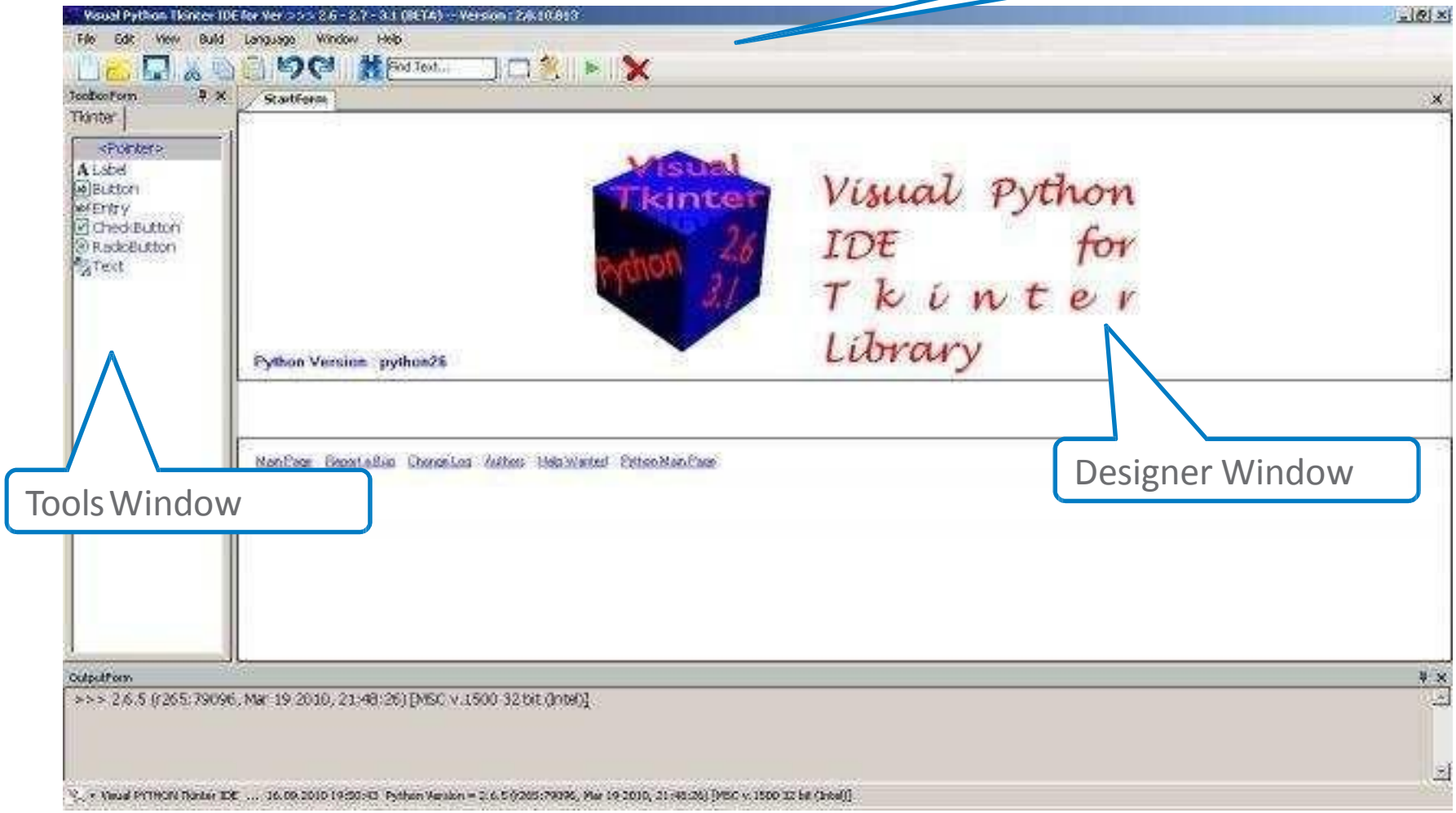


Python GUI builder?

- There are many GUI builders and IDE's available for Python GUI programming out of which Visual Python is the feature rich IDE.



Visual Tkinter IDE- familiarizing



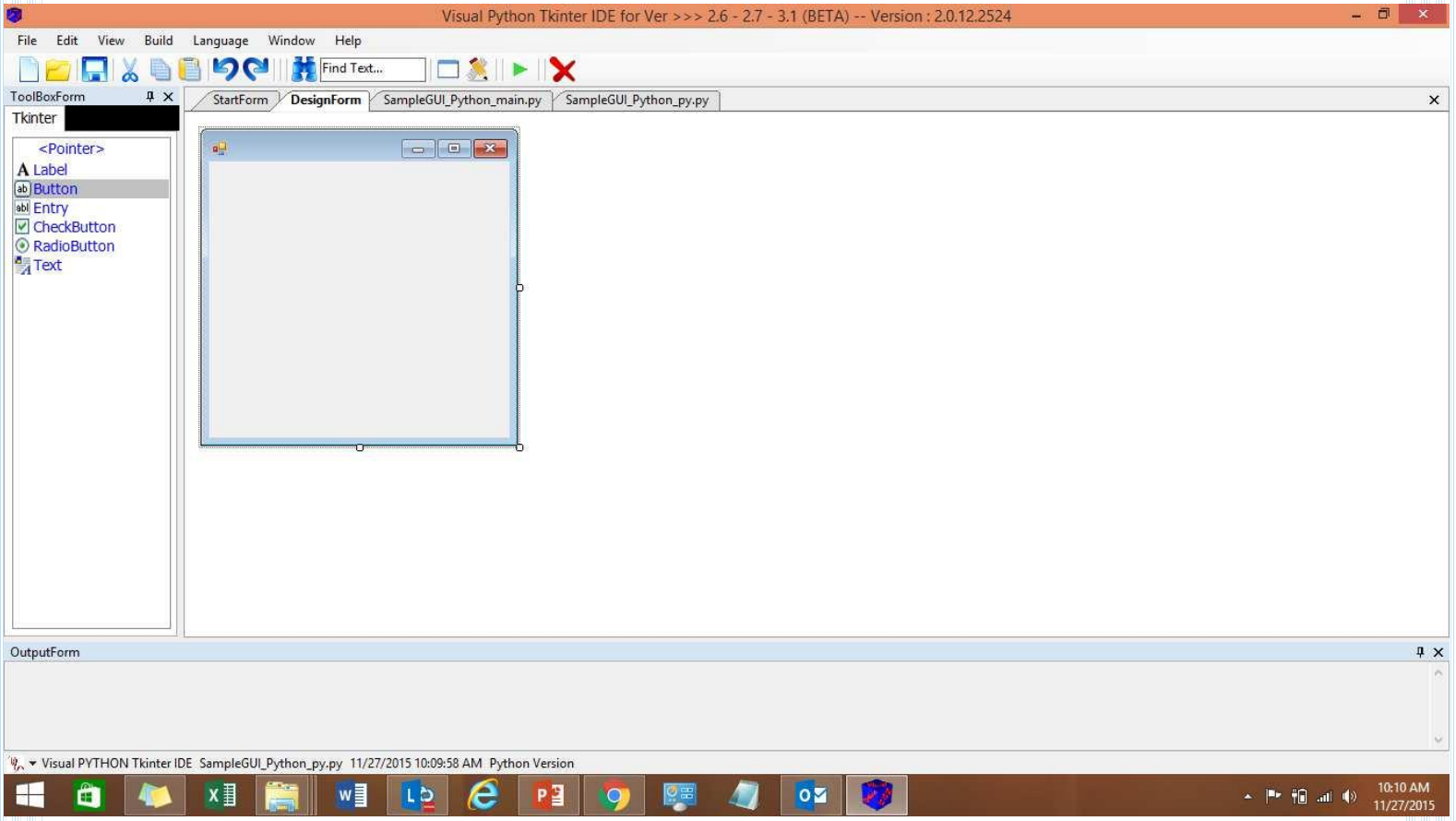
Visual Tkinter IDE– familiarizing (Continued)

Visual Tkinter is divided into three windows namely

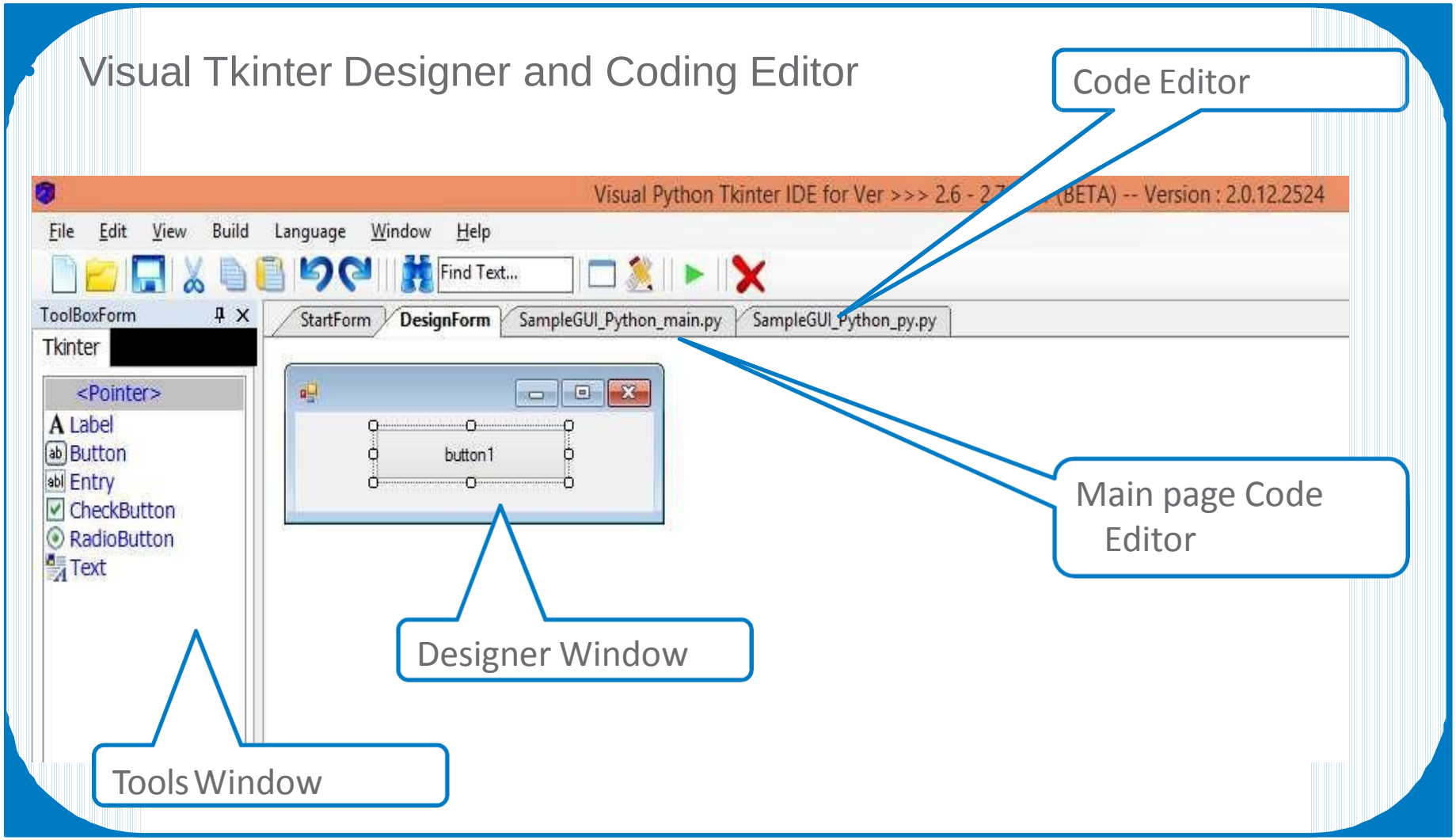
- **Designer Window** – Properties of various controls can be set in this window
- **Tools Window** – All controls, layouts, inputs etc., will be available
- **Code Editor** – The place where we code the application

Hello World Again… (Continued)

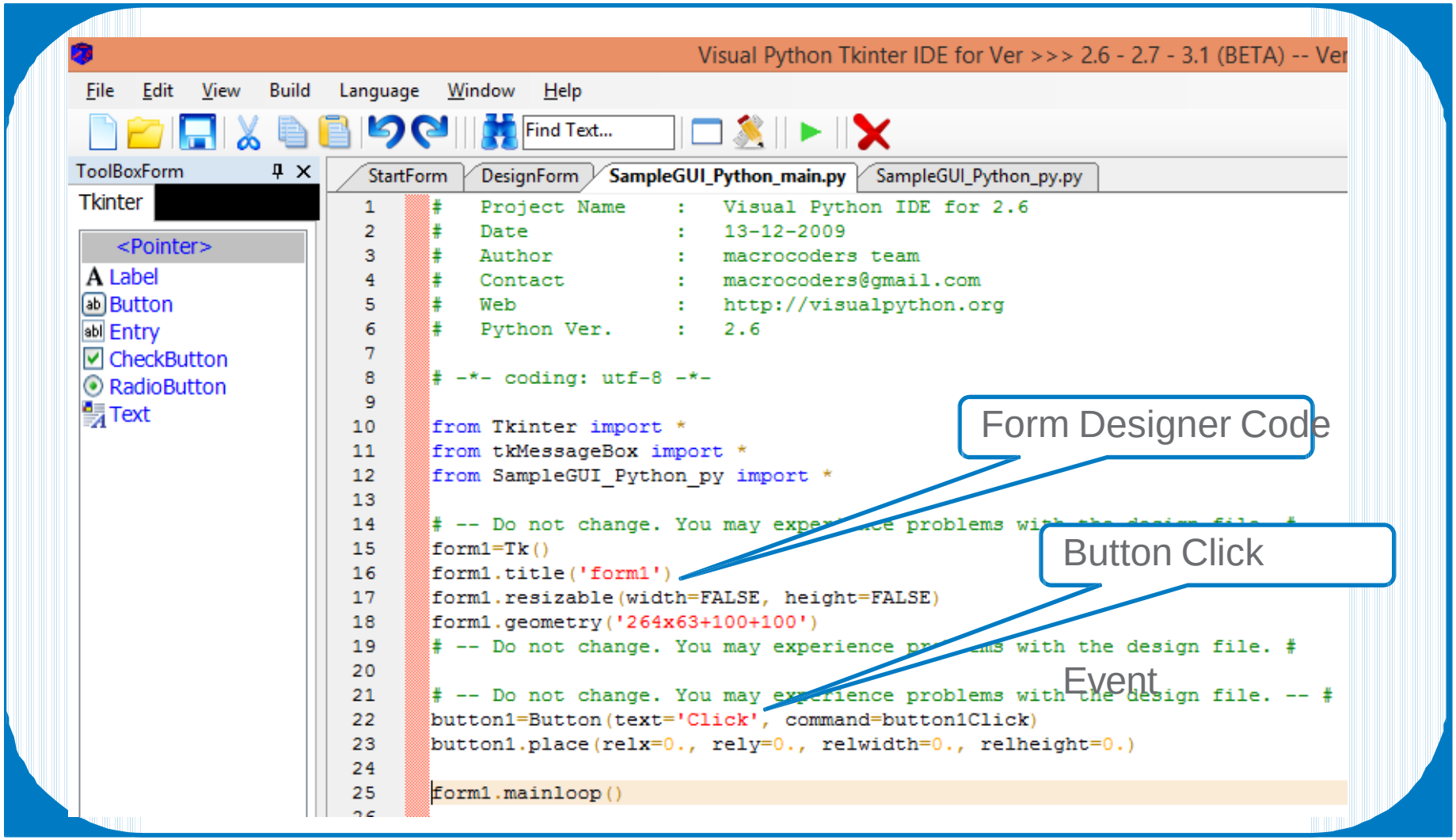
Your New Python GUI project using Visual Tkinter will now look like



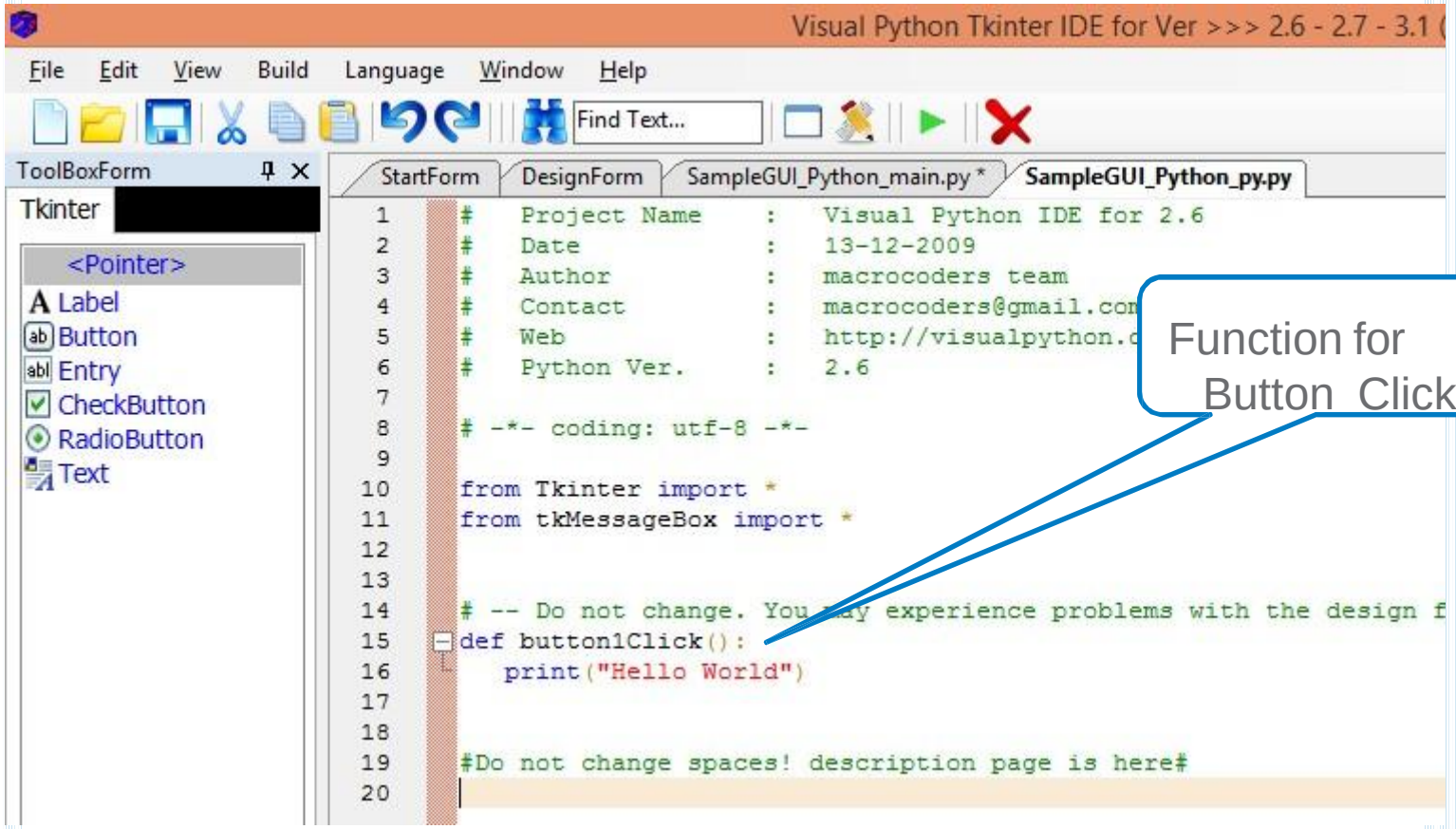
Hello World Again… (Continued)



Hello World Again… (Continued)

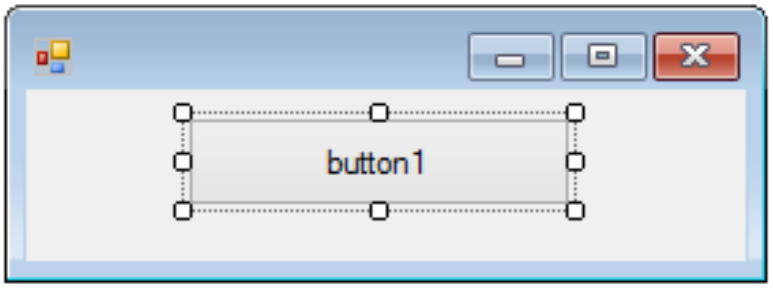


Hello World Again… (Continued)



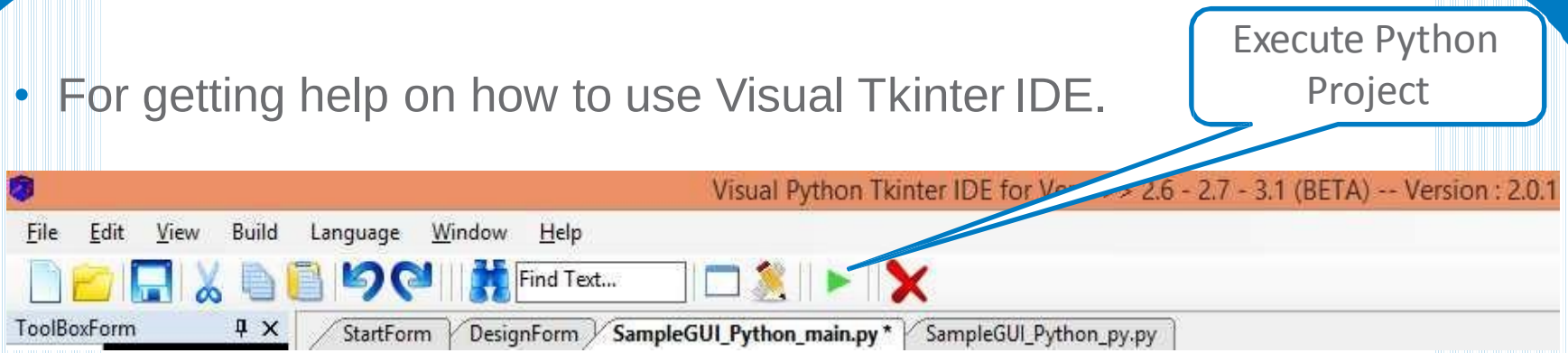
Hello World Again… (Continued)

- Our HelloWorld GUI program is ready without writing a single line of code.

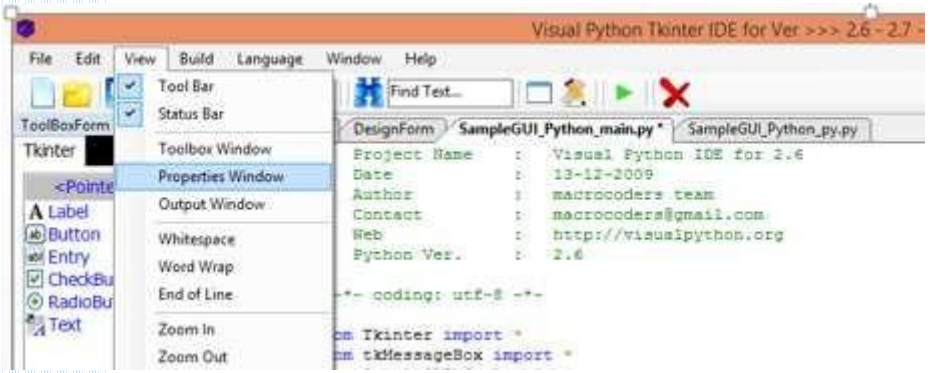


Help in Visual Tkinter

- For getting help on how to use Visual Tkinter IDE.



- Property Window



Thank you