TKinterDesigner Tutorial

Author	Honghaier
Version	V1.3.1
Date	2020-05-15

GitHub: https://github.com/honghaier-game/TKinterDesigner.git

What is TKinterDesigner ?

TKinterDesigner is a Python-based development tool for developing Python application projects based on a small interface.

What are the functions of TKinterDesigner?

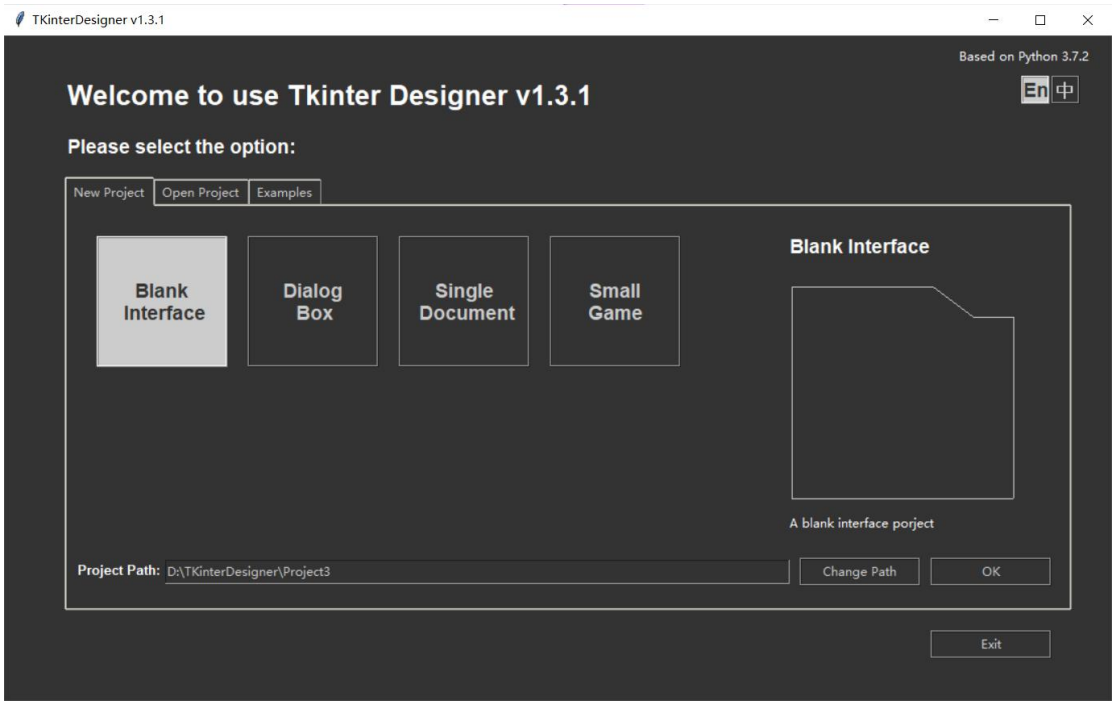
TKinterDesigner V1.3 currently includes the following nine functions:

1. Project management: create and open the project.
2. File management: For the project to create a form, file creation and resource import.
3. Interface design: Design the TKinter interface.
4. Control settings: perform basic attribute editing for the controls.
5. Variable binding: Bind variables to TKinter controls.
6. Event response: Establish a mapping between events and functions for TKinter controls.
7. Logic writing: logical processing of event functions.
8. Compile and run: call Python command to compile and run the project.
9. Pack EXE: Call Python command to pack the project EXE.
10. Custom module import: Import and call the customized module.

TKinterDesigner function explanation：

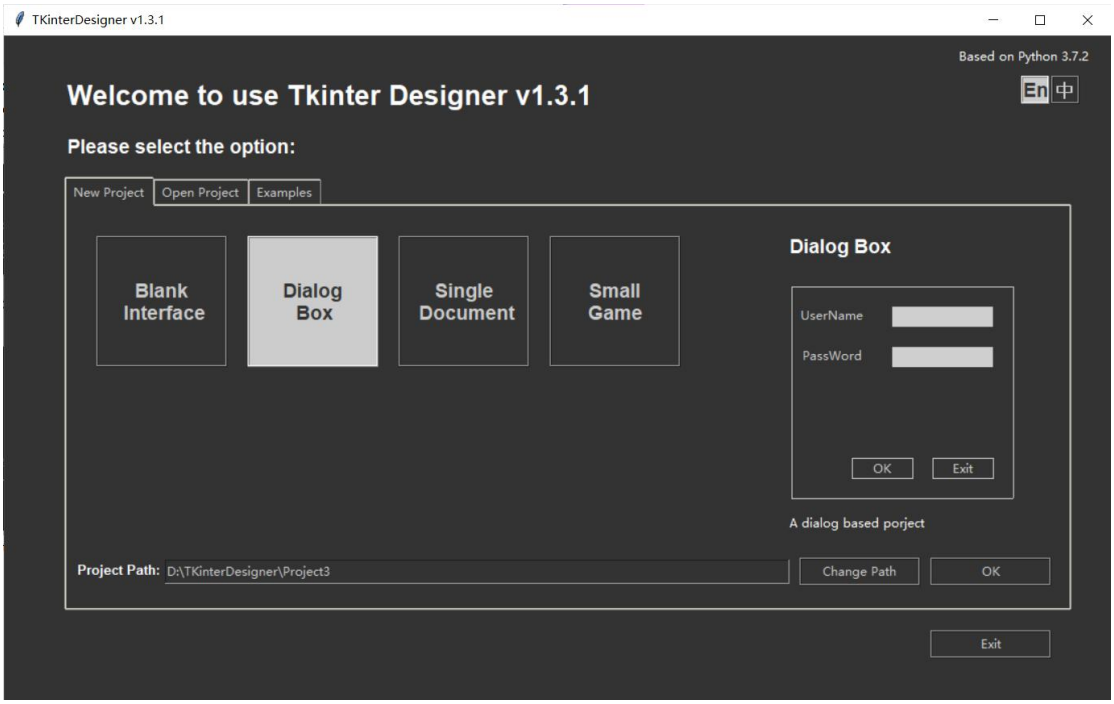
1. Project Management

Double-click to start TKinterDesigner.exe. The first entry is the project management interface. You can select the language to be used according to the need in the upper right corner. Here we choose English.

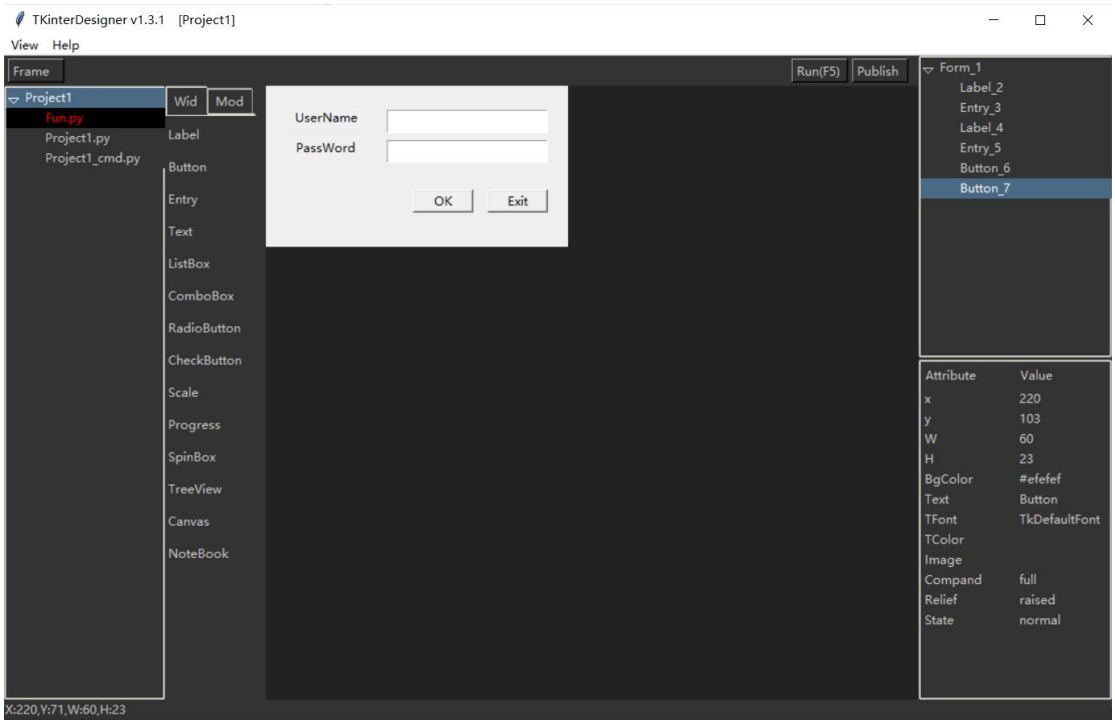


In this interface, I provide tabs with three options:

（1）**New Project:** Three templates for **blank interface items**, **dialog interface items** , **single-document interface items** and **small game** are provided. You only need to select the corresponding project and click "OK" to complete the establishment of a project. If you need to change the project path, you can click "Change Path" to modify it.。

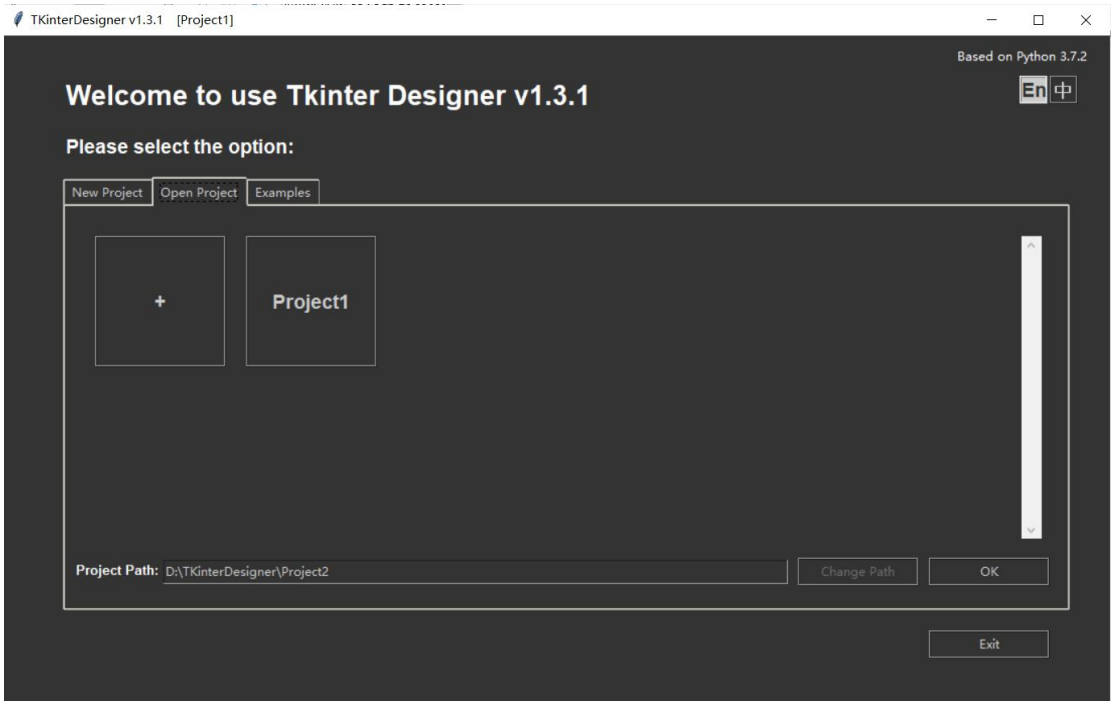


Here we select the dialog interface item and click "OK"。

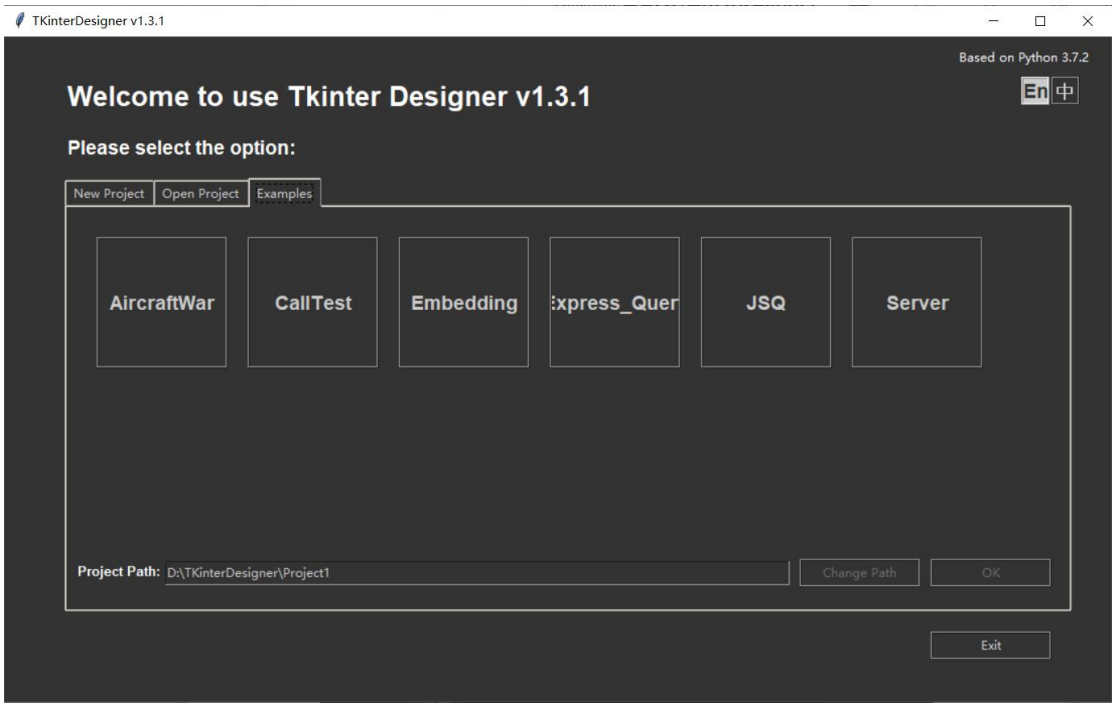


After the project is successfully created, we will immediately enter the main design interface for project development, and we can return to the project management console through the close button in the upper right corner.

（2）**Open Project:** All the projects we created will be displayed in this panel list. We only need to select the required items and click "OK" to enter the project. The first button shown as a plus sign is used to open an item that is not in the list.

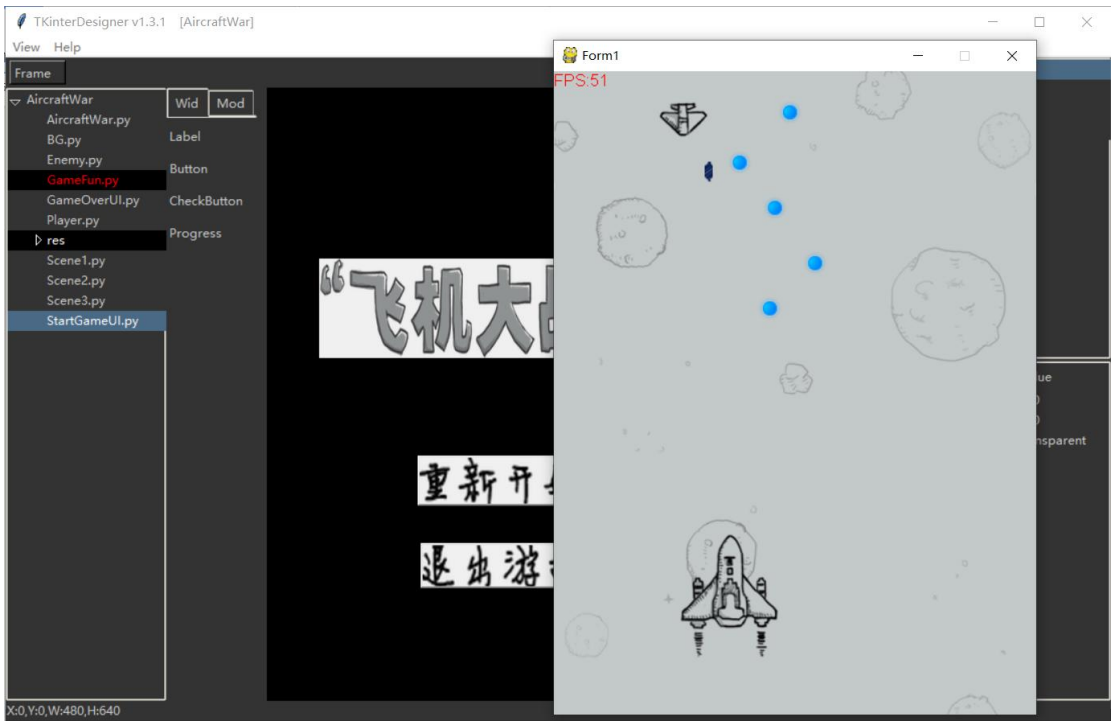


（3）**Examples:** I provide some small cases as a reference, developers can open to learn in order to understand the framework and implementation of some similar small projects.

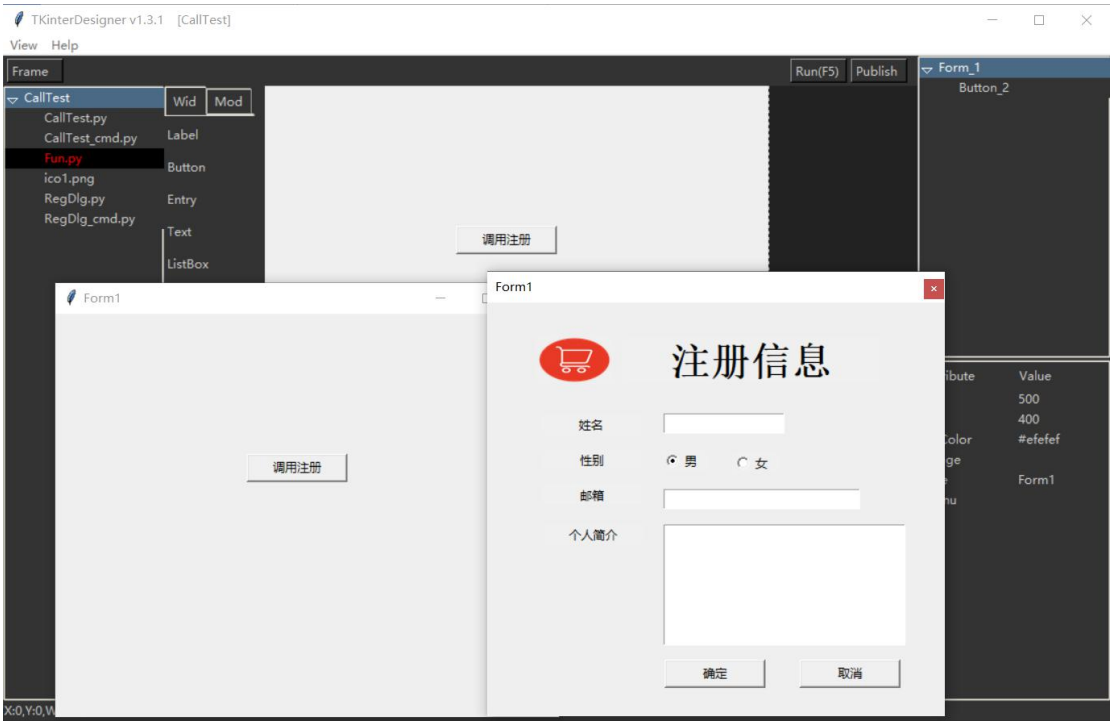


Example projects include:

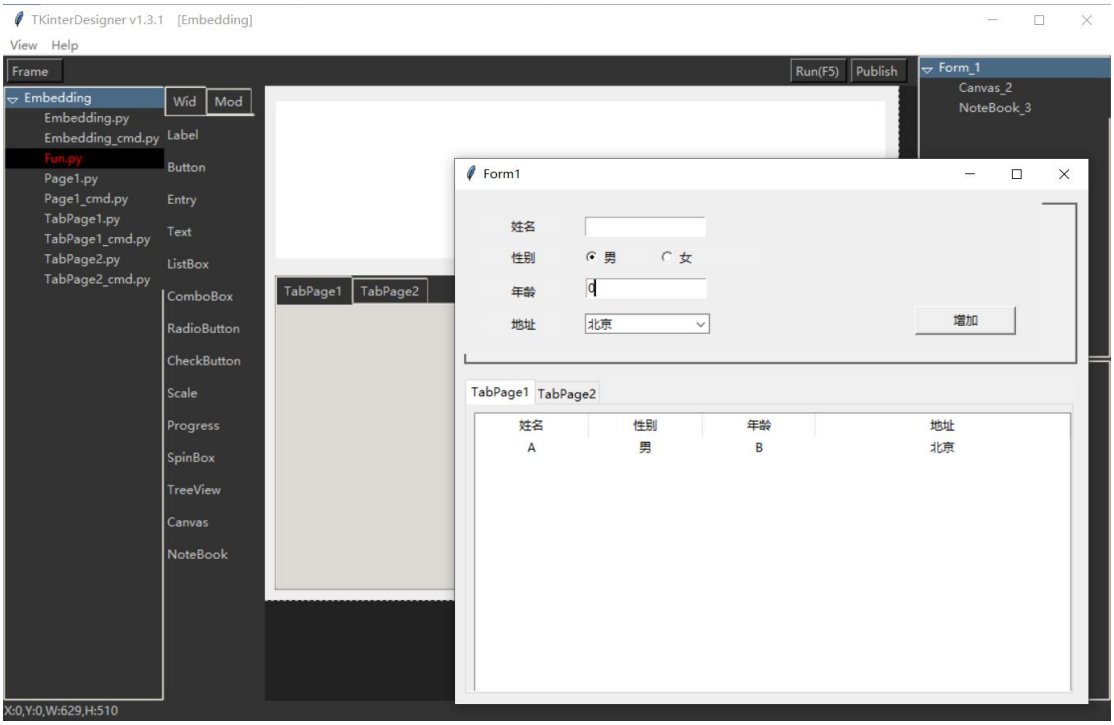
AircraftWar: Aircraft Wars, demonstrates a simple framework for developing 2D games using the pygame library, and encapsulates pygame to make it have scenes, layers, characters, interfaces and other modules.



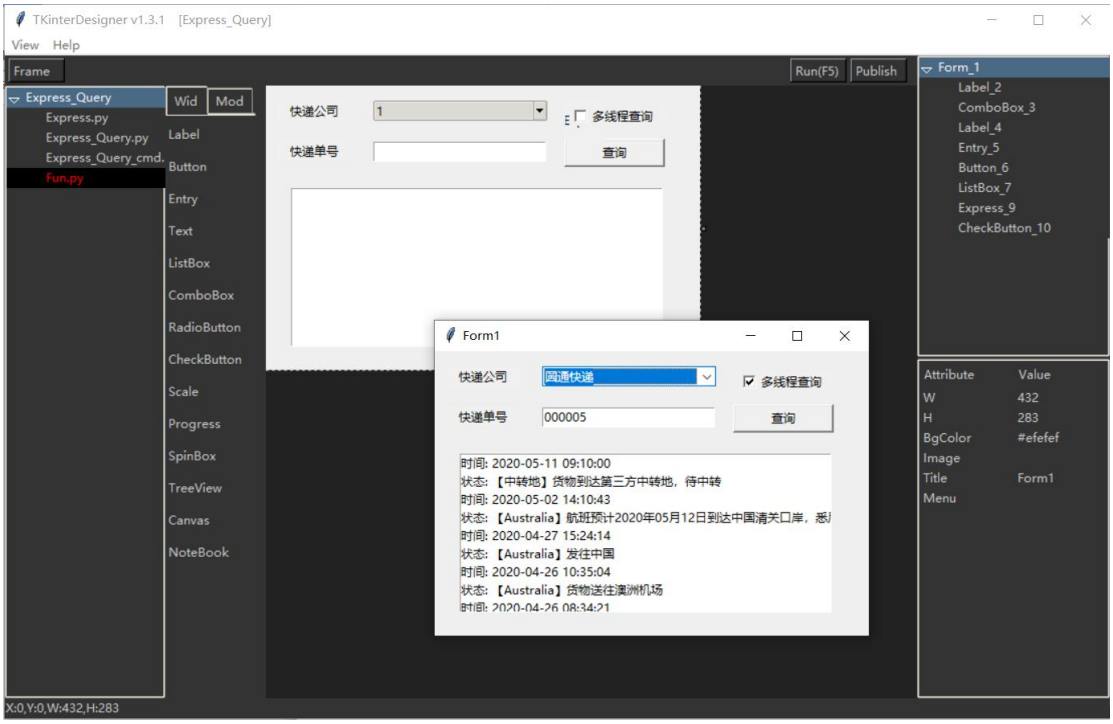
CallTest: Call test, demonstrates how to call another interface program in a dialog box, and return the data results.



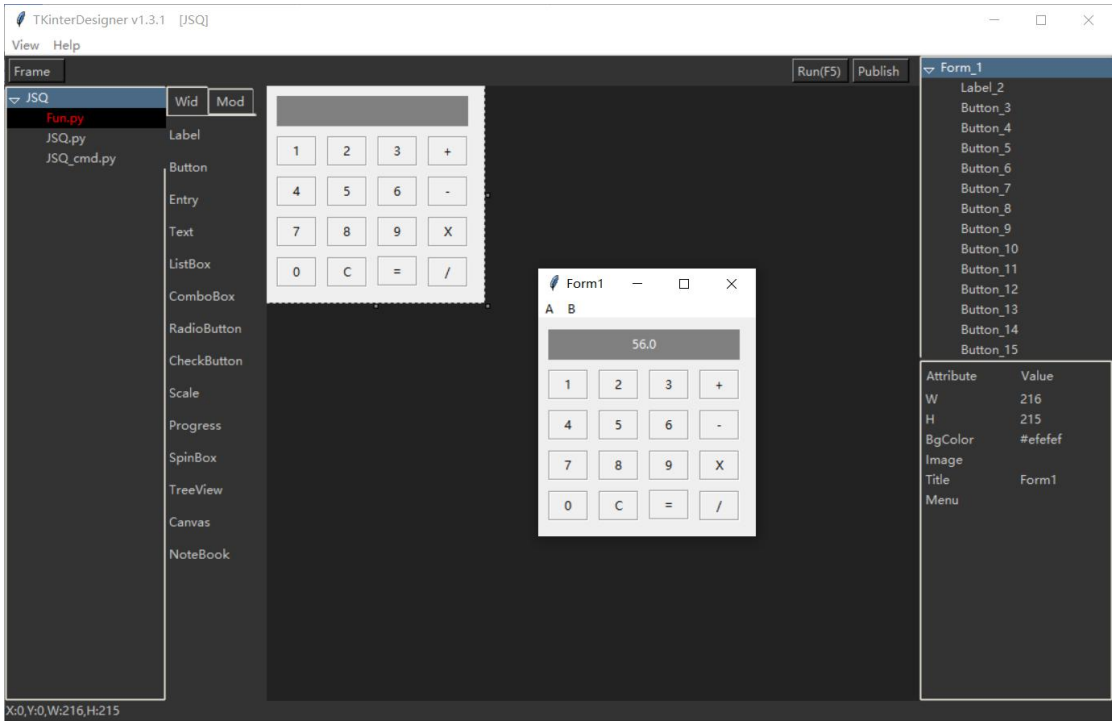
EmbEdding: the embedding of the interface, demonstrates how to embed other dialog boxes in an interface, it can make your interface development work more flexible.



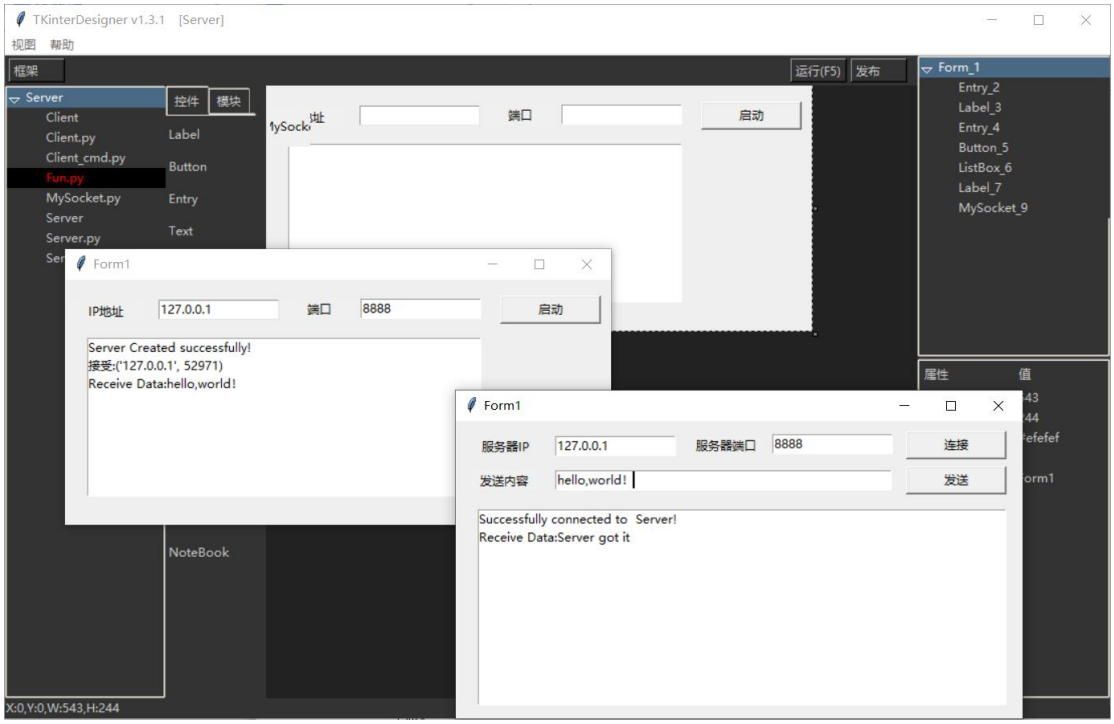
Express_Query: Express query tool, demonstrates how to query network information through the use of custom module classes, and can use multi-threading.



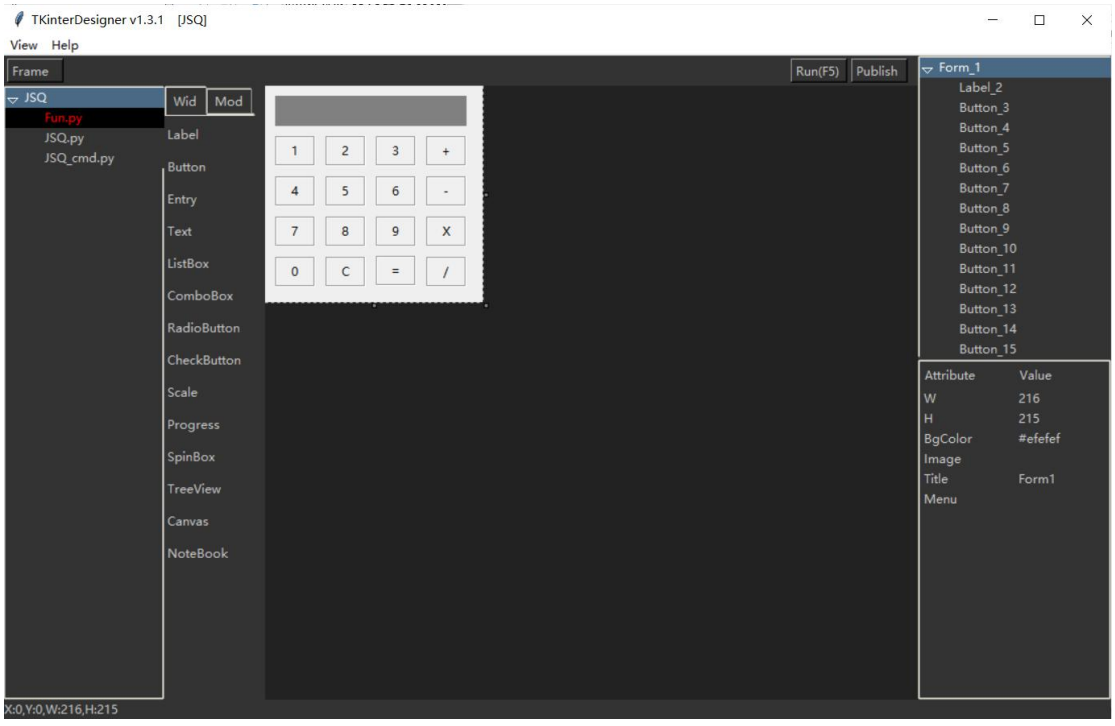
JSQ: A small calculator that demonstrates the use of control variable binding.



Server: A small socket message sending and receiving program that demonstrates how to use a custom SOCKET module to communicate messages.



For example, if we select "JSQ" and then click "OK", you will see a calculator interface project:



In this main design interface, we can see that its layout is:

（1）**The main menu at the top：**1. View: including the grid and adsorption function used in the design, you can also quickly call through Ctrl + G, Ctrl + D. 2. Help: Some useless information, if you need to find me, you can take a look.

（2）**Shortcut button below the main menu：** The "frame" button can show or hide the frame structure tree, the "run" button can quickly run the project for testing, and the "release" button can be used to package and release the project as an EXE program.

（3）**frame structure tree：** Including the list of all files in the project, remember: you can also add a form interface, or add a Python file, or import a resource file from the menu that pops up with the right mouse button. If you are designing the interface, this frame structure tree will affect your viewing window space, you can click the "frame" button to show or hide it.

（4）**Control and module list selection area on the left：** For the commonly used controls required for interface design, I have listed here. Although not all the controls have been covered, I believe that they will gradually enrich with the update. The module selection area is used to import a custom module In the project, there are some custom module classes and use cases in the project, such as “Express_Query” or “ChatServer”, you can take a look, it only needs to have certain design constraints. I will explain in detail in the seventh part of the function.

（5）**Central design preview area**:In the main view area of interface design, you can drag the various interface controls you need here and place and stretch them, and build the interface by WYSIWYG.

（6）**The list tree of all controls in the current interface** :The tree structure in the upper right corner lists all the controls in the current interface. You can click the corresponding tree item to select the corresponding control or delete it from the pop-up menu with the right mouse button.

（7）**Control property editing area：** The list box in the lower right corner lists all the attribute lists corresponding to the currently selected control. You can double-click the corresponding attribute item to modify it here.

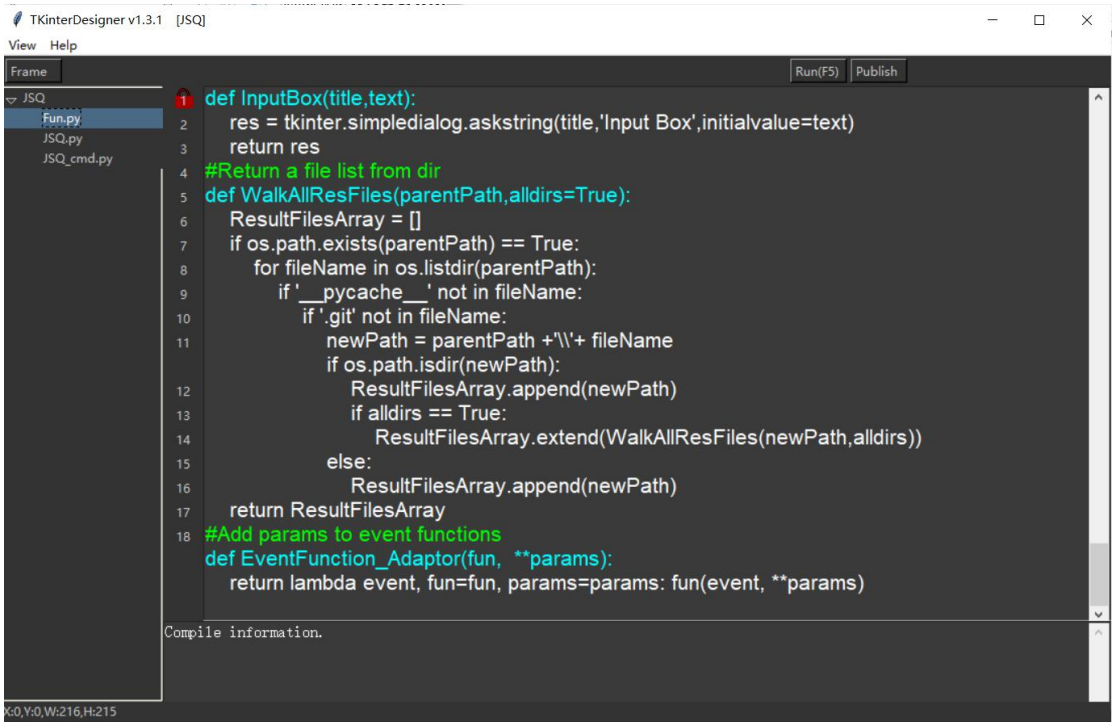
（8）**Information text at the bottom：** Display the position and size information of the current control.

1. Document management

The frame structure tree lists the files in the current project. Taking the dialog box project as an example, we can see that there is a root node item named after the project name under the frame structure tree, and there are three file items under it:

1. 1.Fun.py: This is a public function library file that provides access to functions such as access to controls and control variables.
2. 2.Project1.py: This is a Python file for the main interface of the project, providing code support for the basic layout of the interface.
3. 3.Project1_cmd.py: This is the logic file of the main interface of the project, which provides code support for the logic of the interface. The event function of the control is mainly coded here.

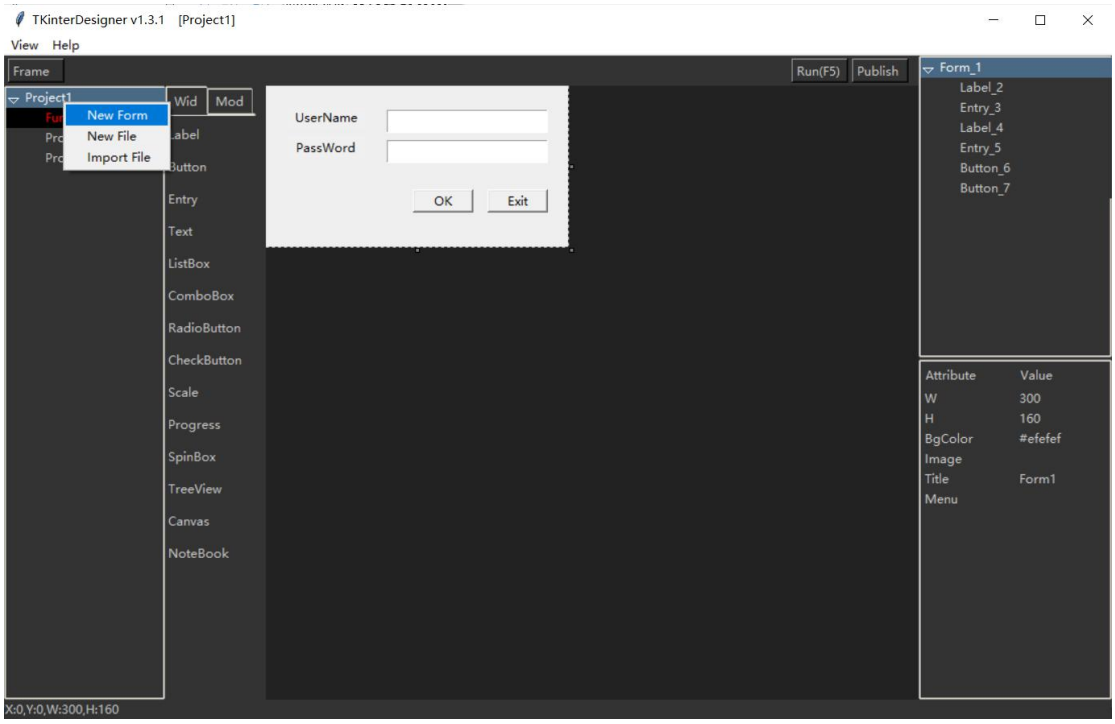
When we click on a Fun.py or Project1_cmd.py, it will become the code area in the main view area:



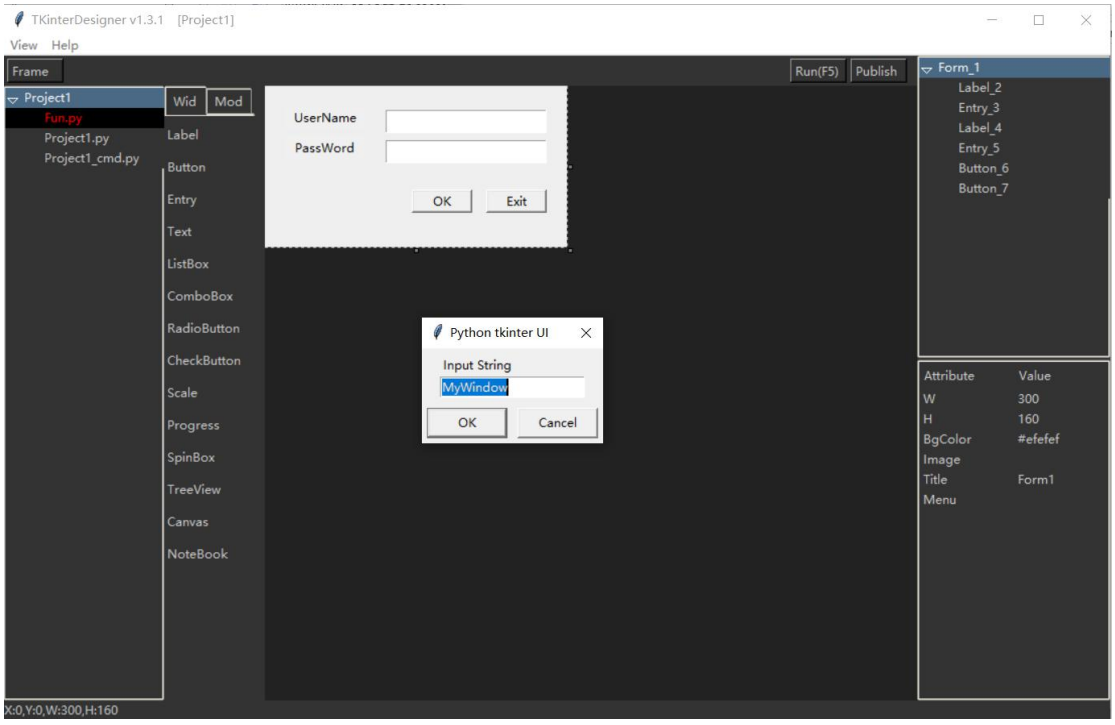
The file Fun.py is a function library provided by the tool, and it is not allowed to be modified directly, so here you can see that there is a small red lock at the position of line 1, which cannot be edited in the text box.

Because the software is focused on using the form designer rather than modifying the code of the form interface, when you click Project1.py, the main viewport displays the interface, and for the logic code and function library code, this is the hope that the developer A lot of logic code is written, modified and adjusted, so a code text area and an information output window are displayed at this time, which is convenient to modify the code and view the output when compiling and running.

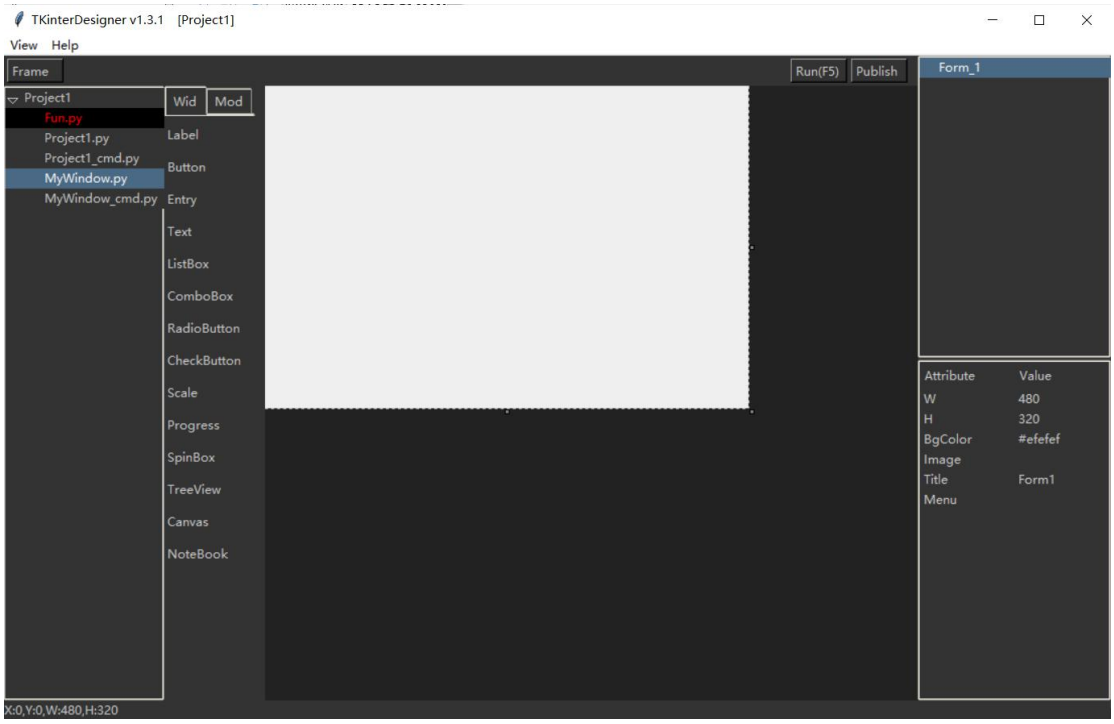
If we want to create multiple windows in the project, we can add a new interface in the frame structure tree. For example, we open the newly created dialog interface item and right-click on the left frame structure tree.



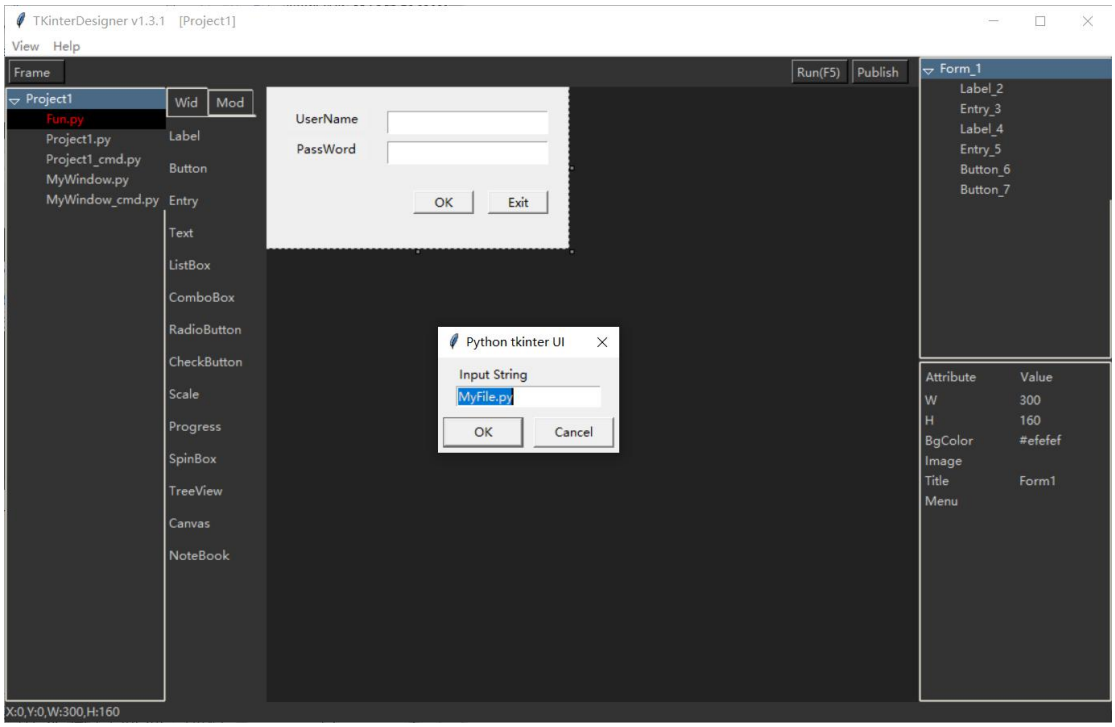
In the pop-up menu, click "New Form", here we can see a new pop-up dialog box, we can enter the name of the new form, and then click "OK".



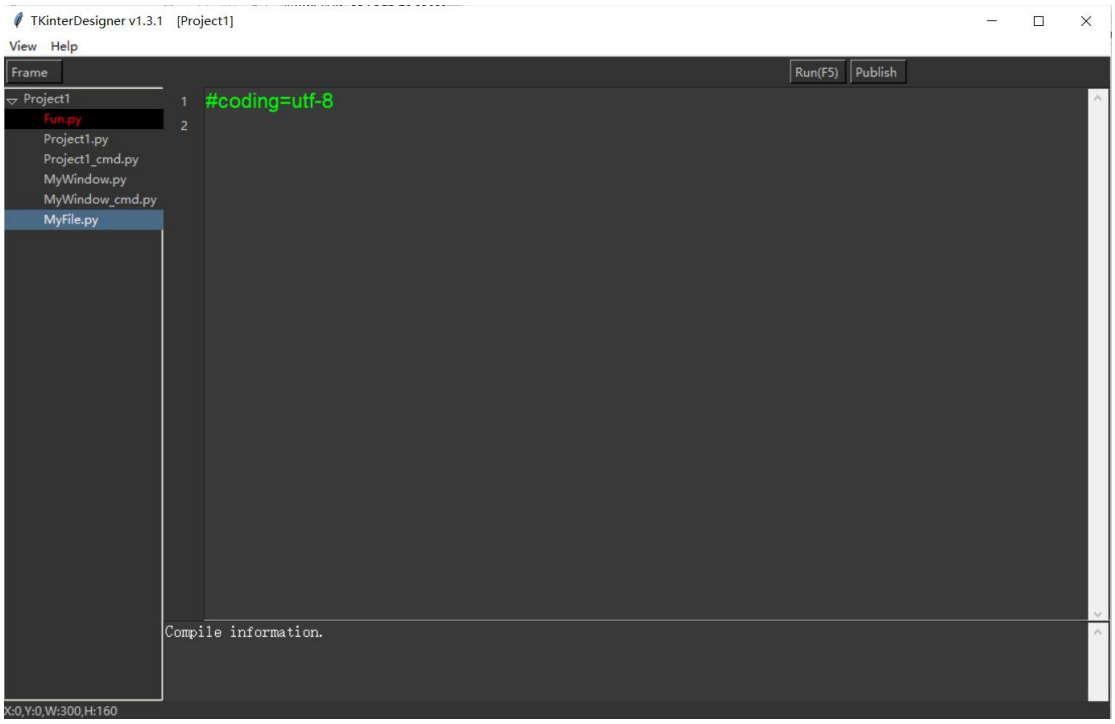
After clicking "OK", we can see a new form, including MyWindow.py and MyWindow_cmd.py two files, respectively corresponding to MyWindow form layout and logic implementation.



If we want to add our own logic code, we can click "New File" in the pop-up menu item in the right-click of the frame structure tree, enter the name of the new file, and we can create a new Python file.



After clicking "OK", you can see the new file code, then you can start writing code.



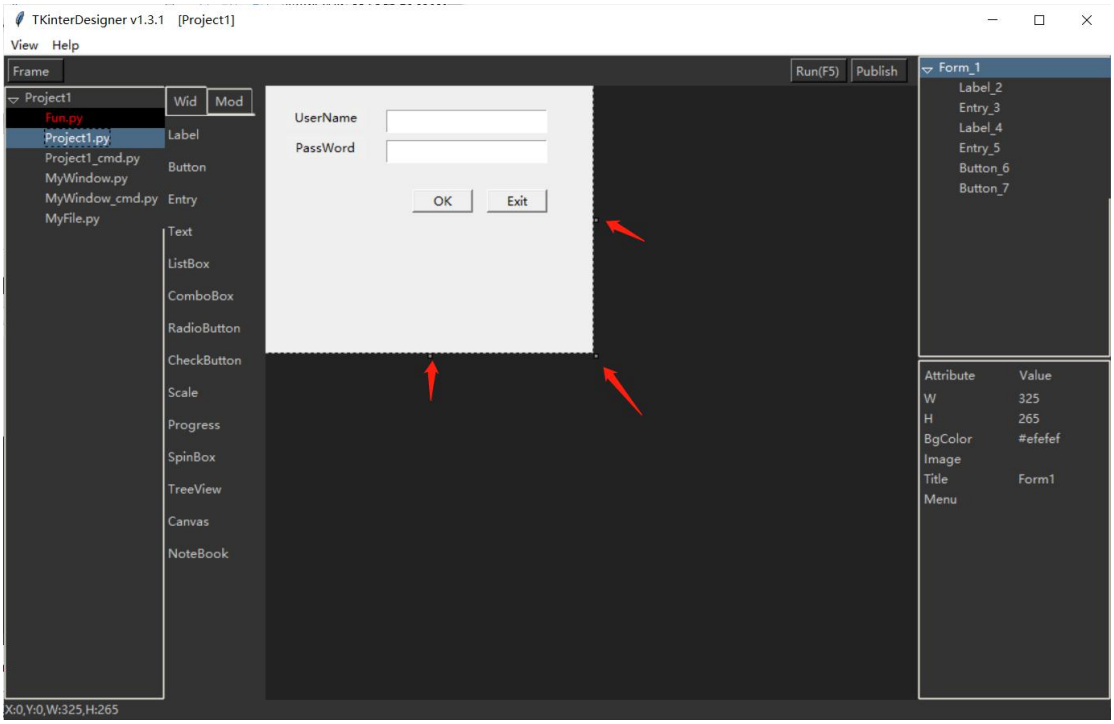
Sometimes, you may need some pictures, sounds, or other format file resources to put into the project. Here, you can also select and import it by clicking "Import Resources" in the right-click menu item of the frame structure tree.

Finally, if you want to delete one of the files, you need to right-click the corresponding file item to pop up the menu item and click "Delete File". After confirmation, you can delete the file.

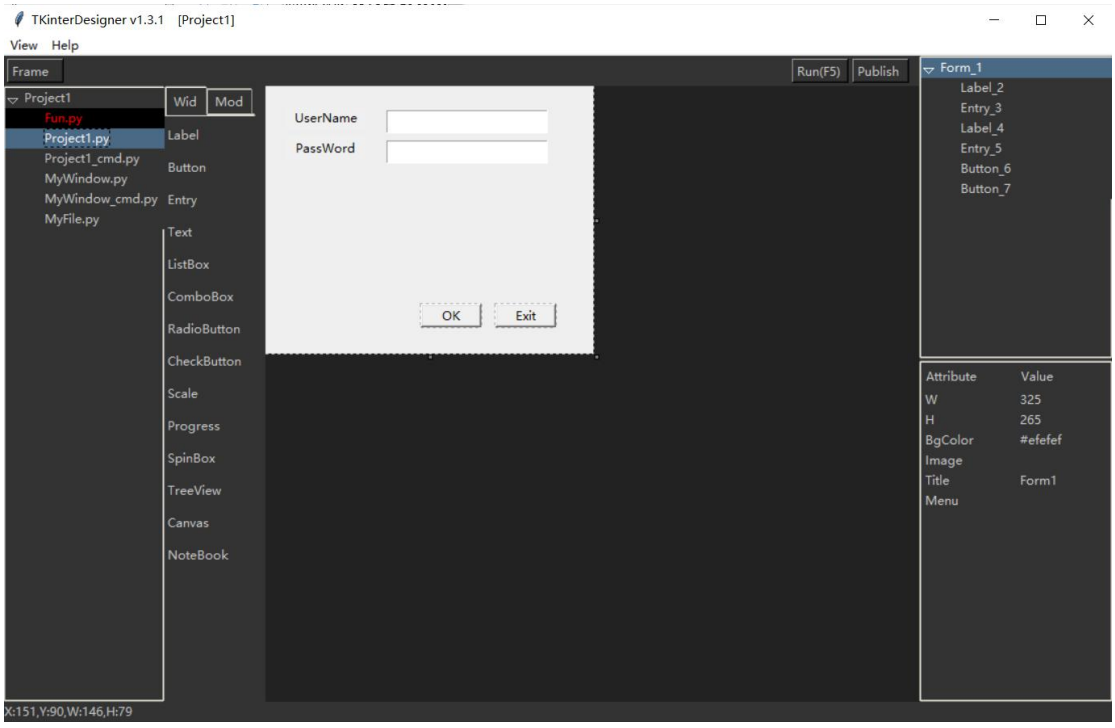
2. Interface design

First, we click Project1 or Project1.py to enter the interface design area, and then we can start the interface design. For example, we want to add gender options, occupation classification, and whether we are married to this basic account, password input interface. , We need to add some new controls, including two RadioButton, a ComboBox, a CheckButton and the required Label text. These are very commonly used controls and are quite representative.

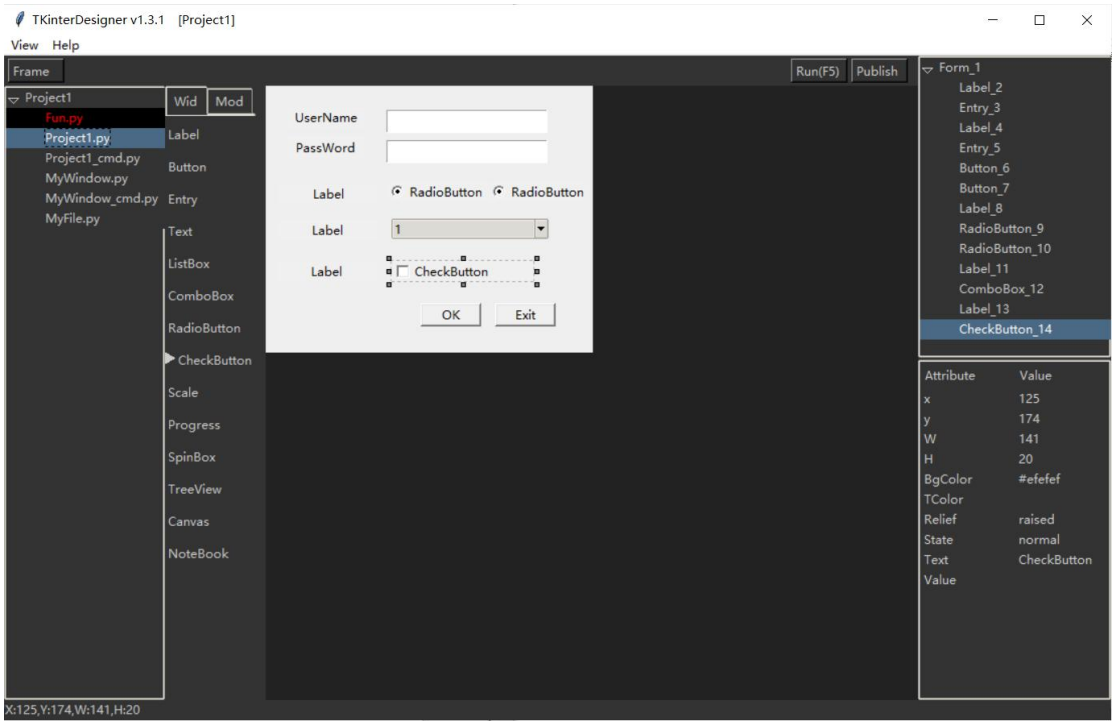
We must first expand the main form, because its size is not enough, then you can click the control tree item "Form_1" in the upper right corner, or directly click on the form interface in the design area, we can see the form Around the dotted line, a gray-white drag block appears at the midpoint of the vertex and the edge. We can use the mouse to click the drag block in the lower right corner and drag it to the appropriate size.



After we are done, we can drag the "OK" and "Exit" buttons directly to the appropriate position.

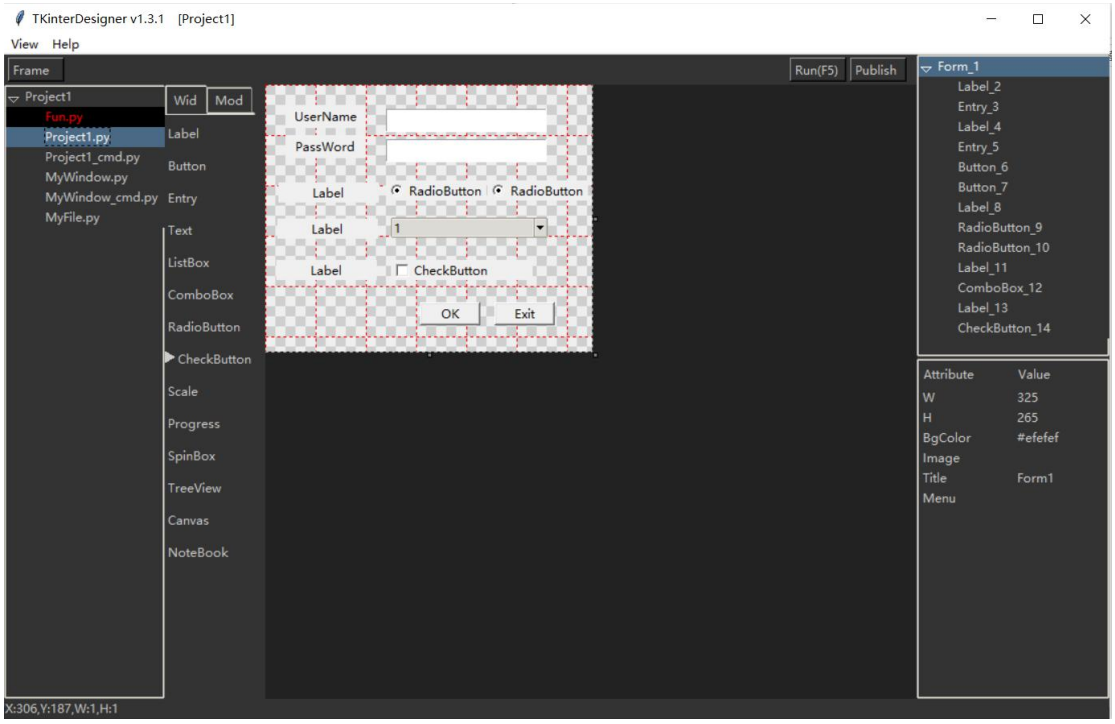


Now we can select the required controls one by one from the list of controls on the left and drag and drop them into the form.



Here is a little trick, if you need to create the same control repeatedly, you can directly select it on a control and drag it with the mouse while pressing the ALT key, you can directly copy out a control for you to drag operating.

If you feel that the position is not well aligned, you can select "Grid" and "Absorbent" under the "View" item in the main menu, or you can quickly call display or cancel by Ctrl + G, Ctrl + D It is in units of 10 pixels, which is convenient for you to drag and align after sucking.

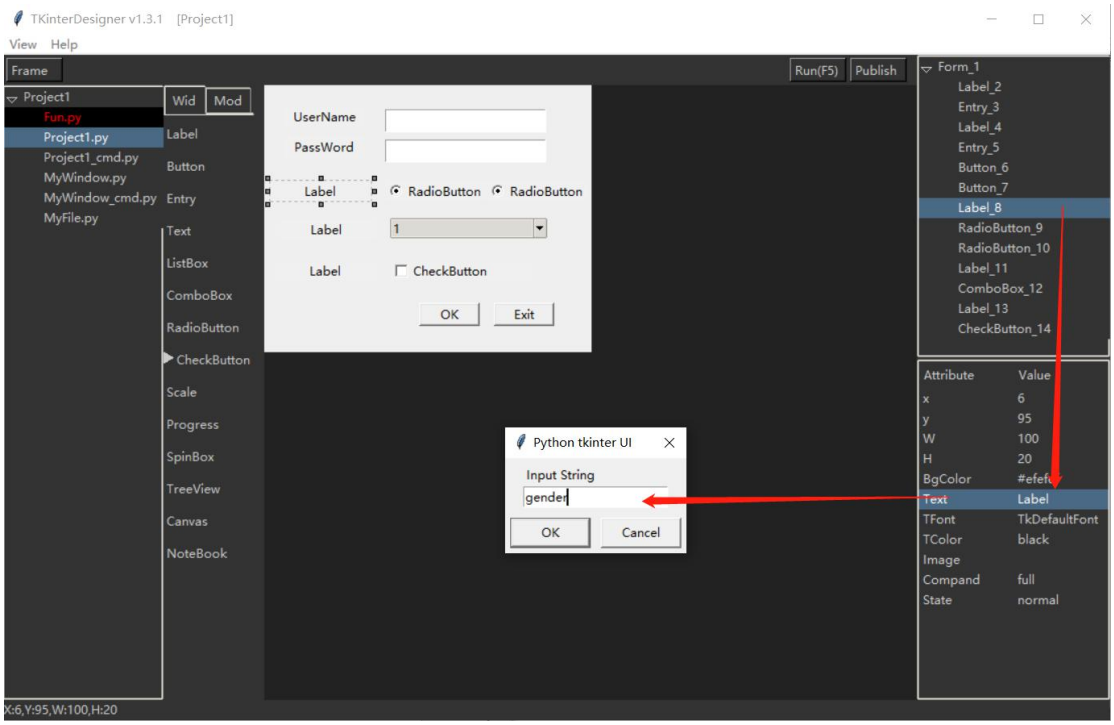


Ok, now that we have completed the creation of the required controls, is it very simple?

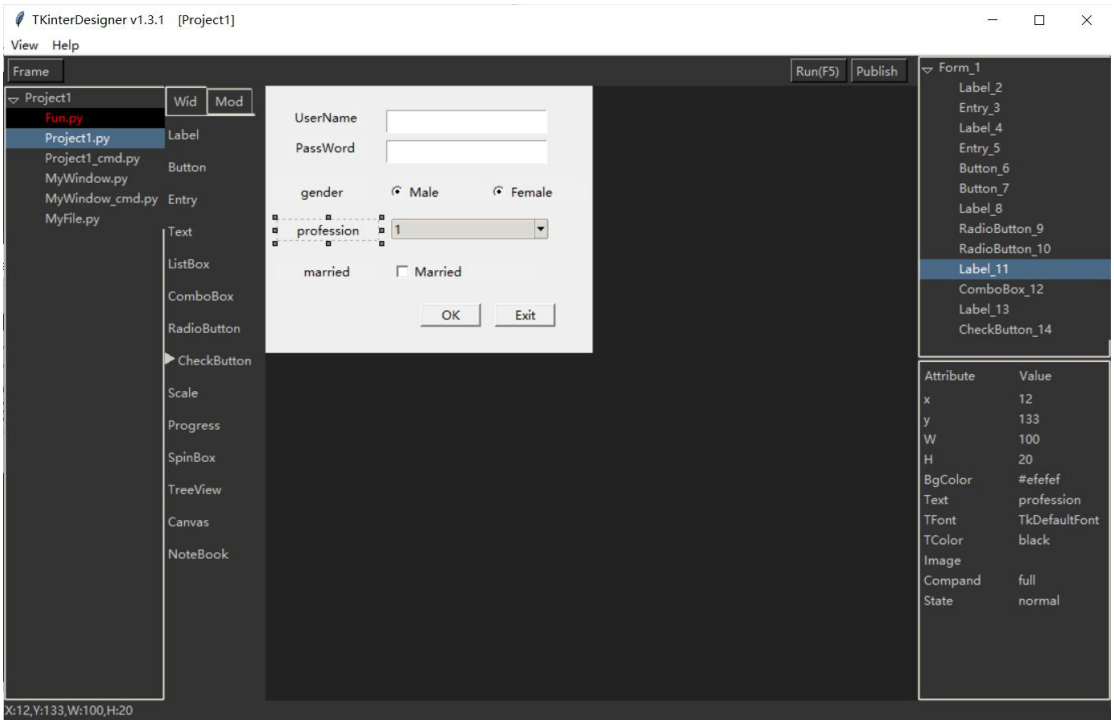
Let's set these controls.

3. Control settings

Select the Label before the gender radio button, and then we find the "text" property in the property box on the right, double-click it, enter "gender" in the pop-up dialog box, and click "OK" to proceed to the text of the corresponding Label change.

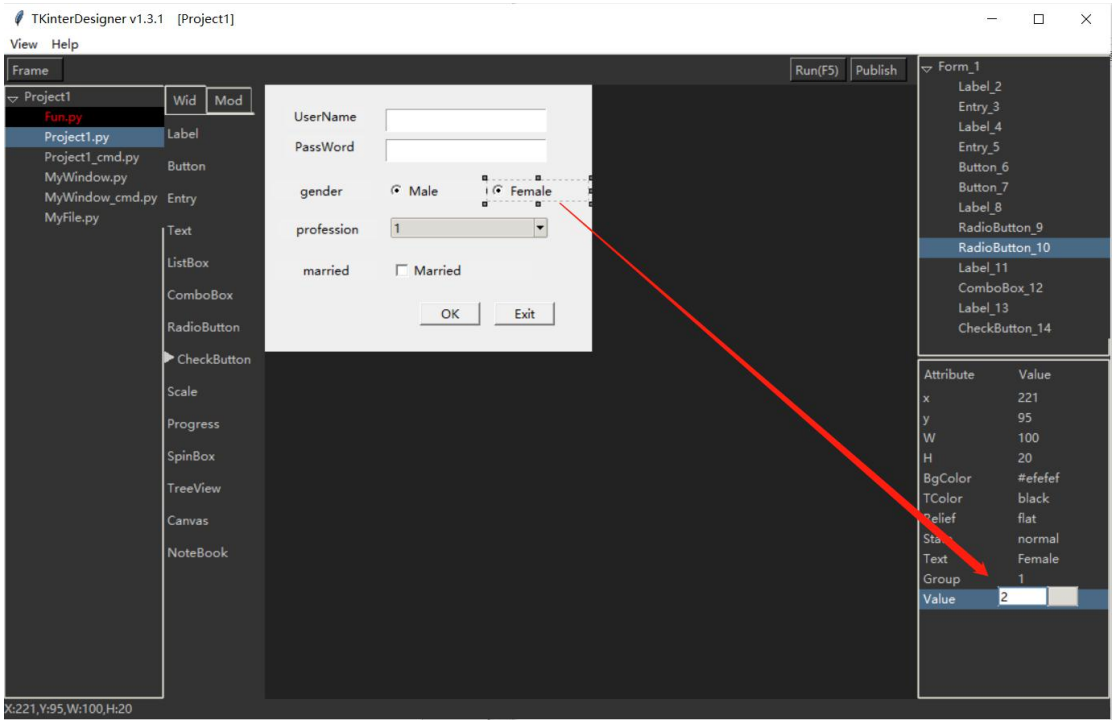


So, soon we completed all Label, two RadioButton text and CheckButton text settings.



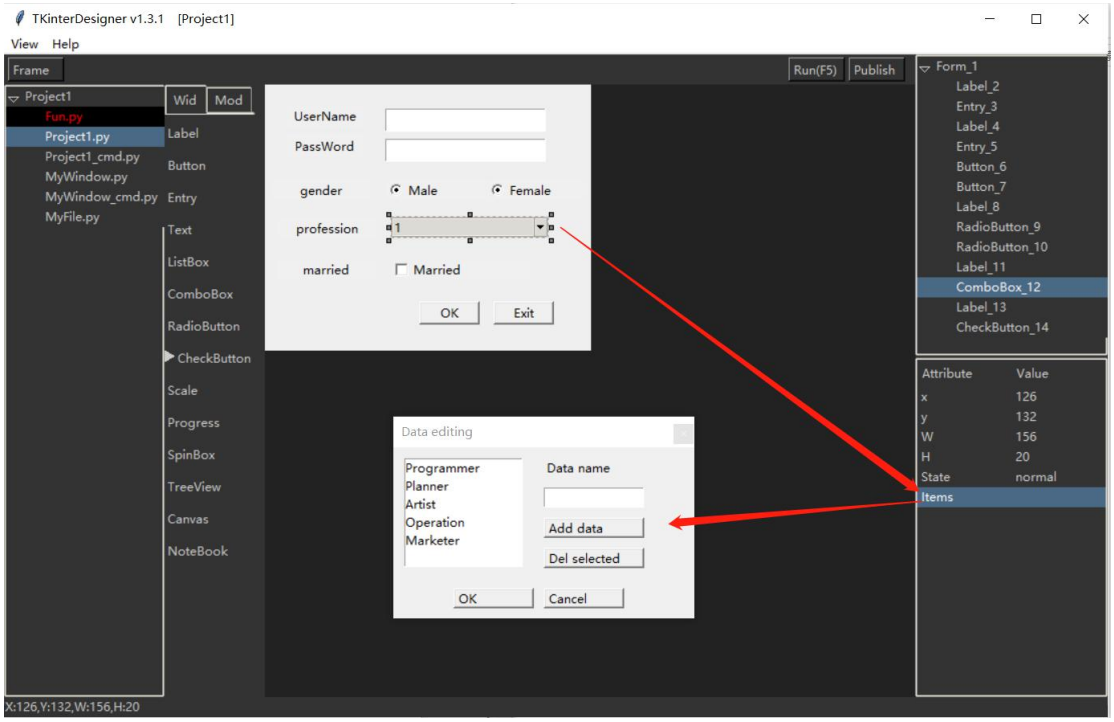
there are many attributes that can be set, for example, you can modify the background color and text color, you can also add pictures to mix, or modify the font, etc.

Of course, it is impossible for gender to be both male and female at the same time. Such bisexuality does not conform to our orientation, right? We select the RadioButton with the text "Female", in the group and value column, double-click the value item, change it to 2 in the input box, and then click the button, then we can see the correct RadioButton grouping.

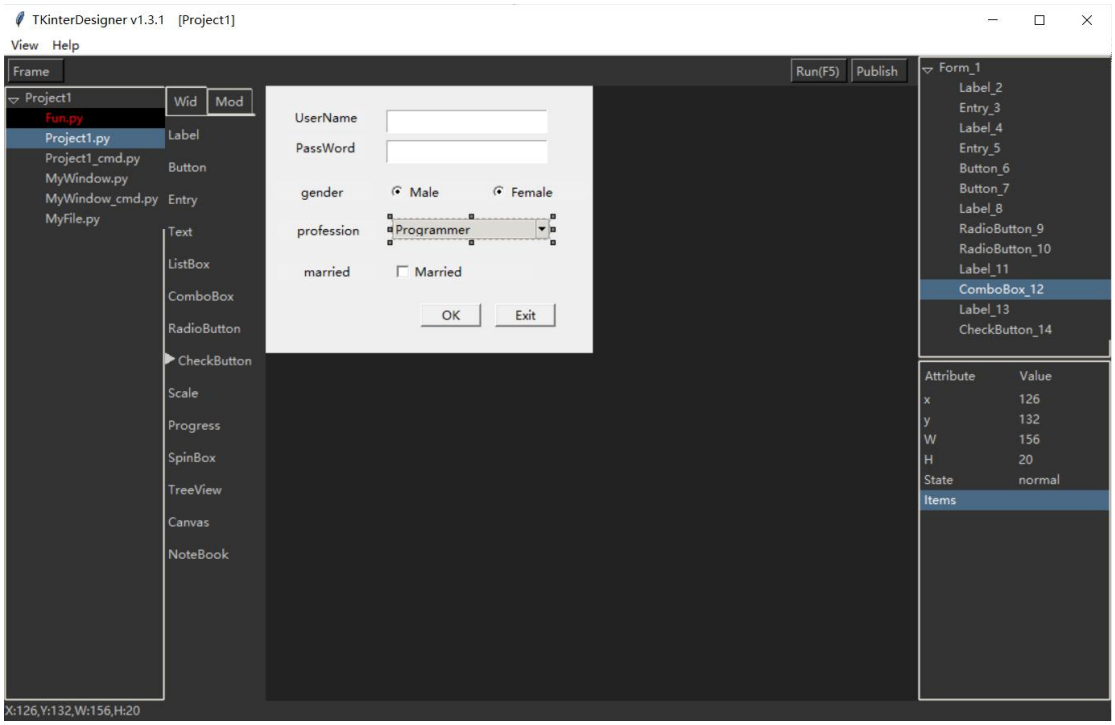


How is this going? First of all, we need to be clear that there can be many RadioButtons in an interface. Some of them may be for one option, such as gender, and some for another option, such as living in several areas of the city. These two parts need Grouped into two groups, and they need to have a unique value in each group to distinguish. Therefore, when the two radio buttons representing "Male" and "Female" use the default group number of 1, only need to modify the value of "Female" corresponding to the RadioButton to 2.

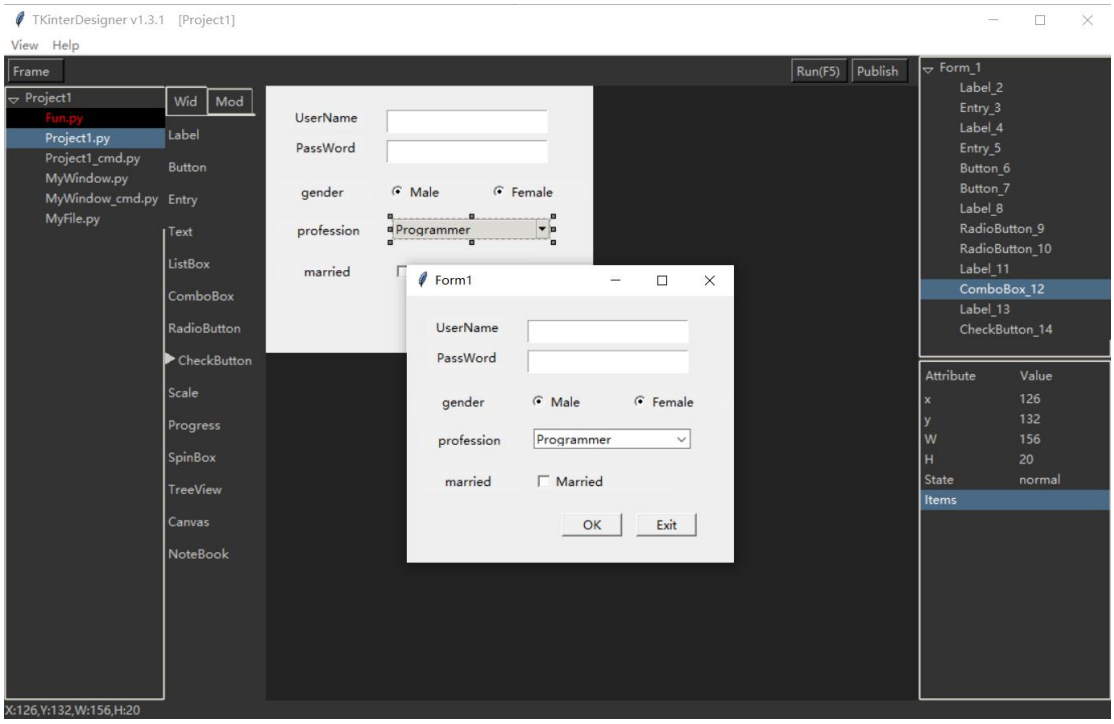
Below we add career options for input, we select ComboBox, find "data item" in its property box, double-click, in the pop-up data item editing area dialog box, you can edit the data item corresponding to ComboBox.



For example, we enter "programmer", "Planner", "Artist", "Operations", "Marketer" five data, click "OK", we can see that the ComboBox has become what you want.



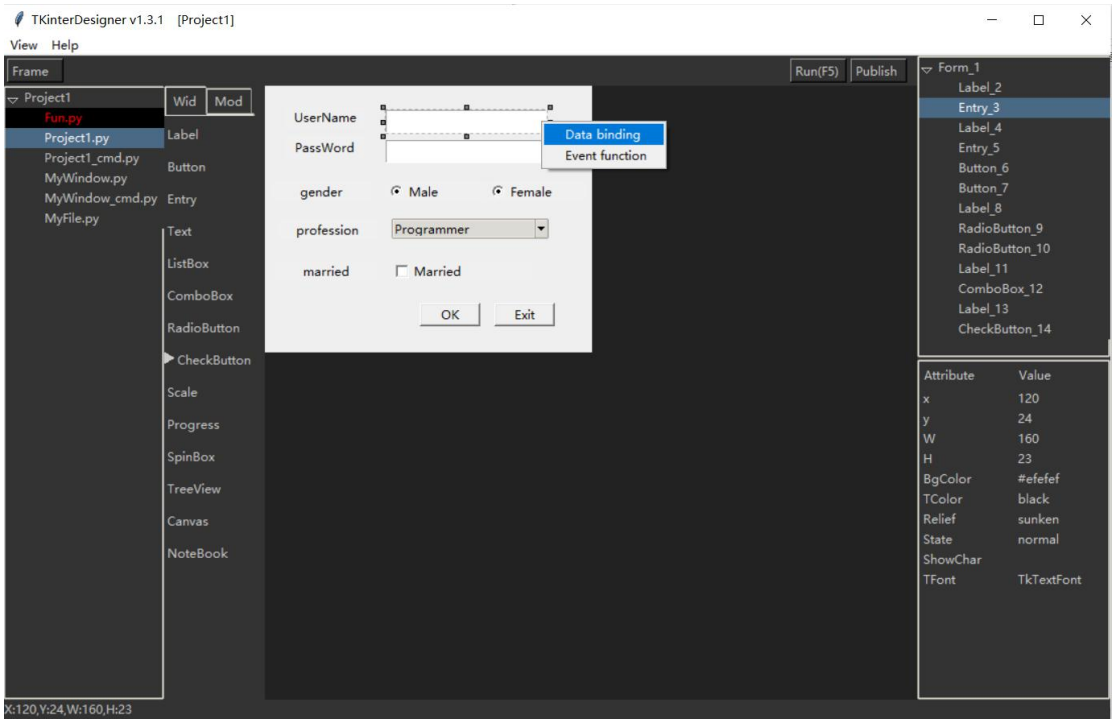
OK, click "Run" or press F5, this will automatically save the design and code, and compile and run the results.



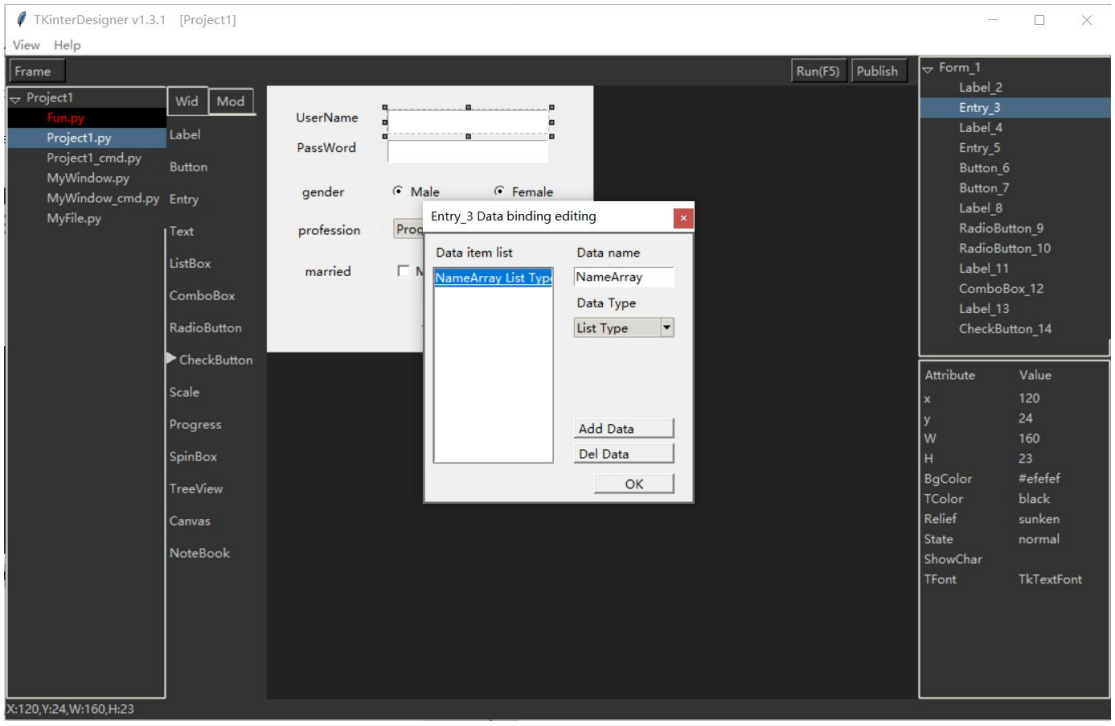
This example demonstrates the property design and use of several common controls. You can try other controls yourself, and I will not repeat them here.

4. Variable binding

In development, we often need to store some data, maybe just a simple result storage, or maybe the input value of the control. For example, in the development of the example project "calculator", we bind a temporary variable storage for the Label of the displayed data. Value to facilitate operations of addition, subtraction, multiplication and division. In the above example, suppose that when clicking "OK" to determine that the account number is the same as the last input value, we pop up a dialog box prompting "Account is used", we can add a custom variable binding for the account.



We right-click on the input box corresponding to the account, click "variable binding" in the pop-up menu, in the pop-up dialog box, we enter the name of the data item to be bound "NameArray", select "list "Type, if we are using numeric type or string type, we can see that there is a" map to 'text' "option box, click this item will mean that this variable will be set when calling Fun.setUIData function to set At the same time update the variable to the text of the Label or Entry control. Only one variable is allowed for the same control. If this item is not selected, you can create multiple variables for a control. We don't need to click here if we don't need it.



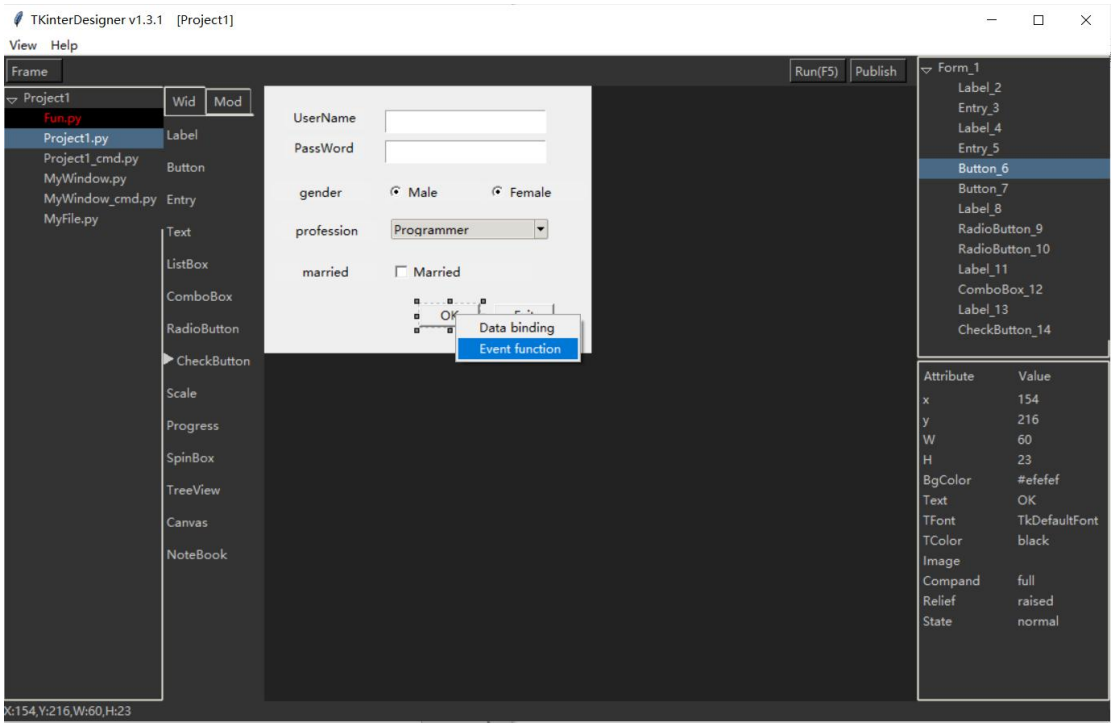
After the creation is complete, the input box will have a binding list variable, you can get it at

any time through Fun.getUIData (className,'Entry_3', 'NameArray').

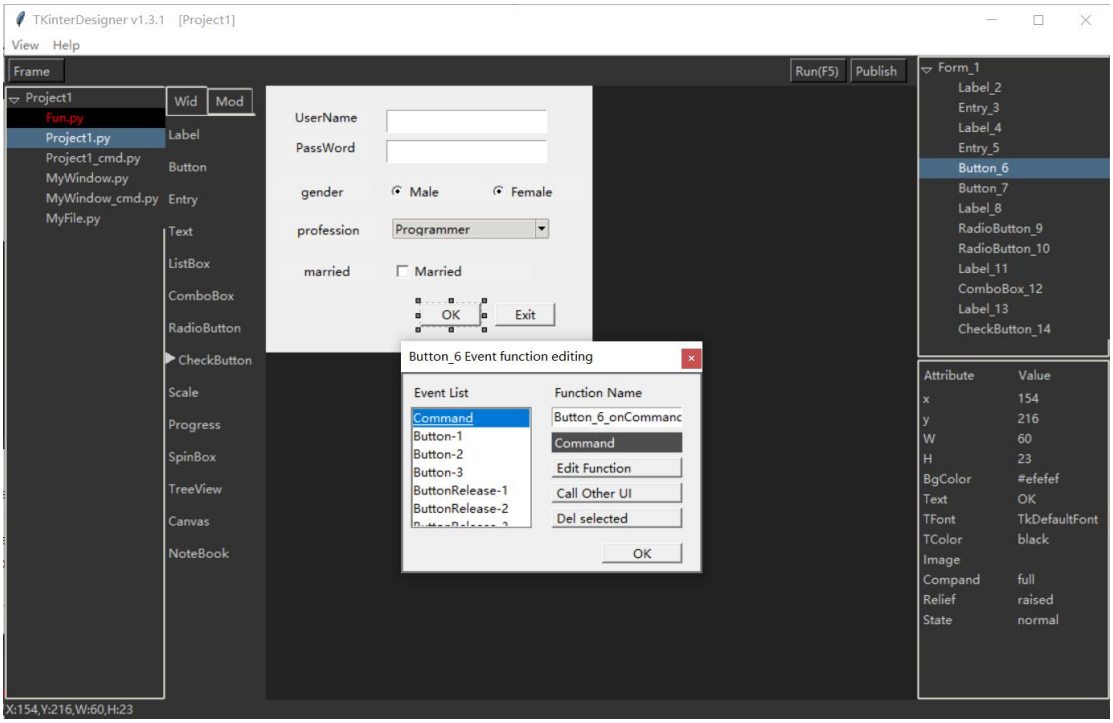
Next, let's deal with making related judgments when clicking the "OK" button. This requires adding a Command event map for the "OK" button.

5. Event Response

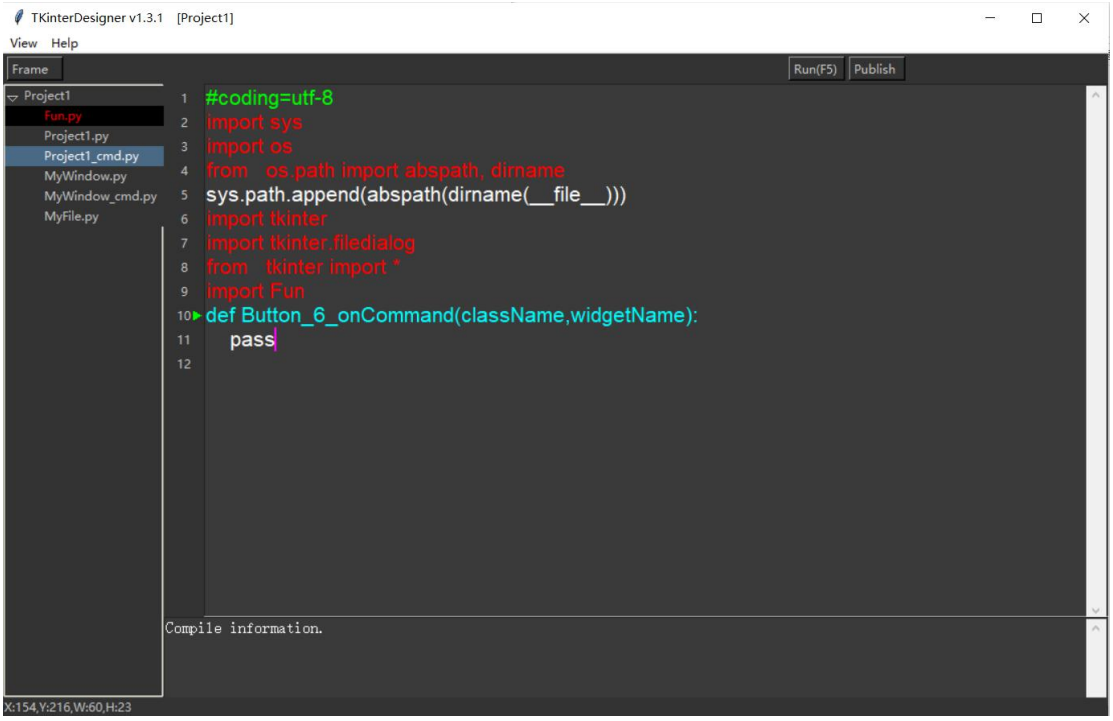
The so-called event response is to make a function mapping for the events that can be bound to the control, so that when the corresponding event is triggered, the set function is called.



We right-click on the "OK" button and select "Event Response" in the pop-up menu.。

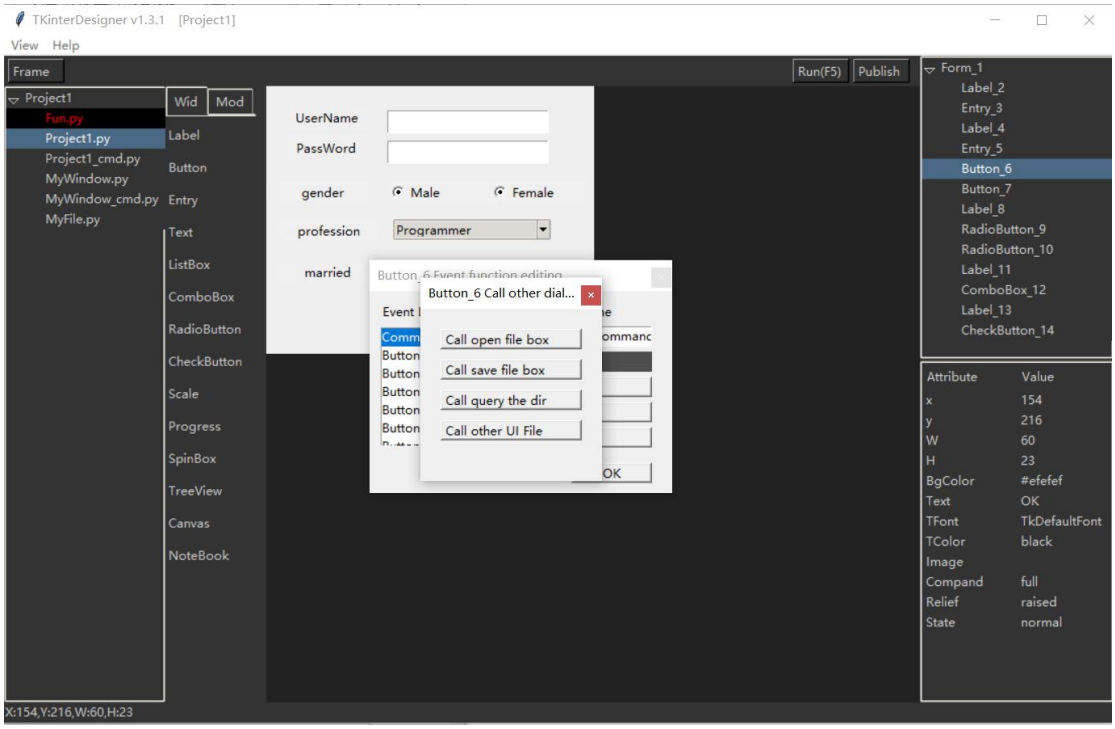


In the pop-up event response processing editing area, we can see that there is an event list on the left, which lists commonly used Python events, and an input box on the right, which shows the default function name. "Code" can directly enter the code editing area of the logic file, at this time, we can see the event response function has been added, we can manually edit the code in this function.



In addition, if we want to call other interfaces when the event is responding, we can also

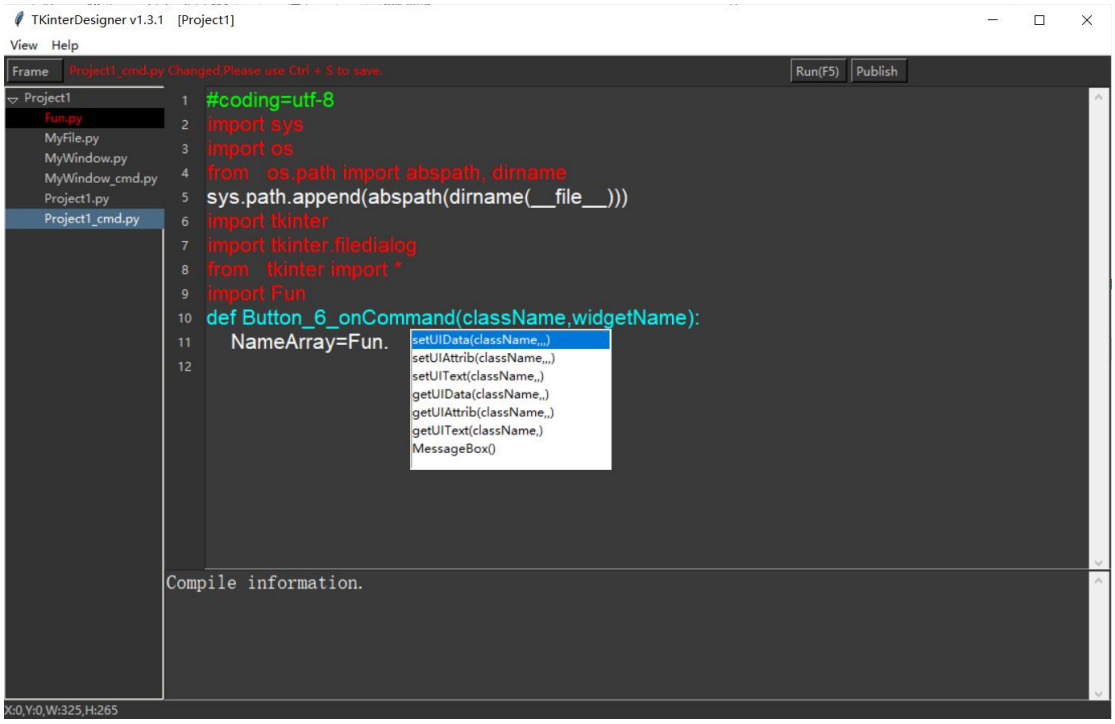
click the "Call Other Interfaces" button. At this time, an option dialog box will pop up, which we call as needed.

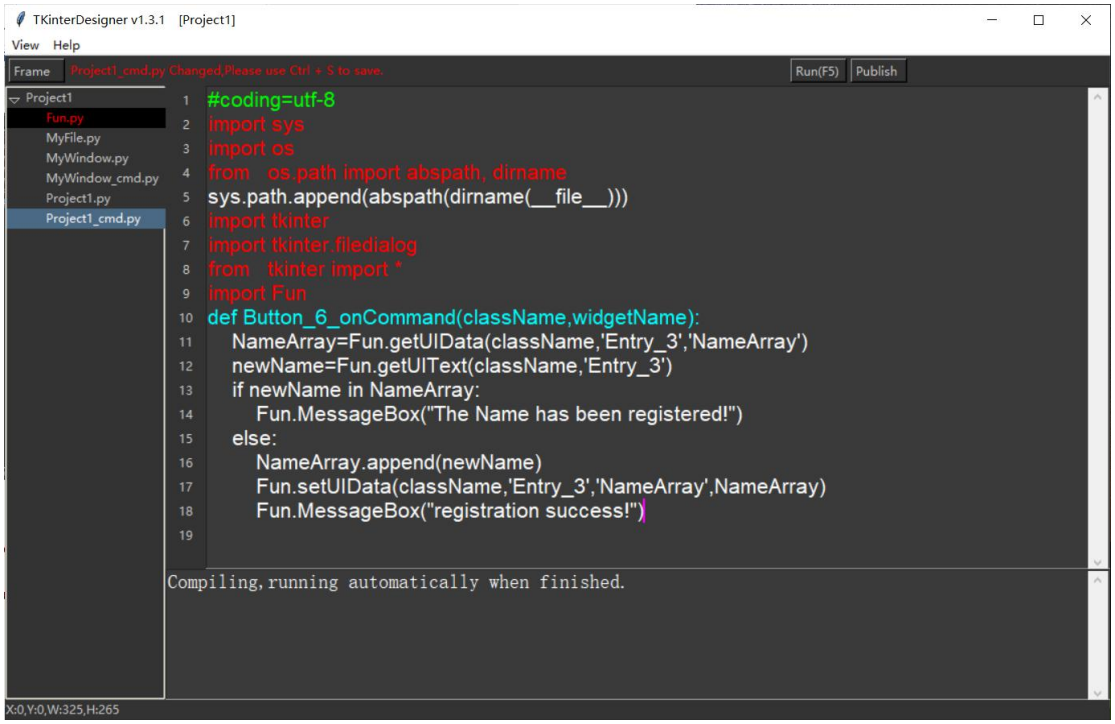


The most commonly used open and save file boxes can be directly selected here, but if you create a multi-window program, you need to call another window here, select "call custom interface" to find its Py file Just call. I have a CallTest project in the example project to demonstrate this.

6. Logic writing

In the Button_6_onCommand function in the code editing area of the logic file, we can write the following code:



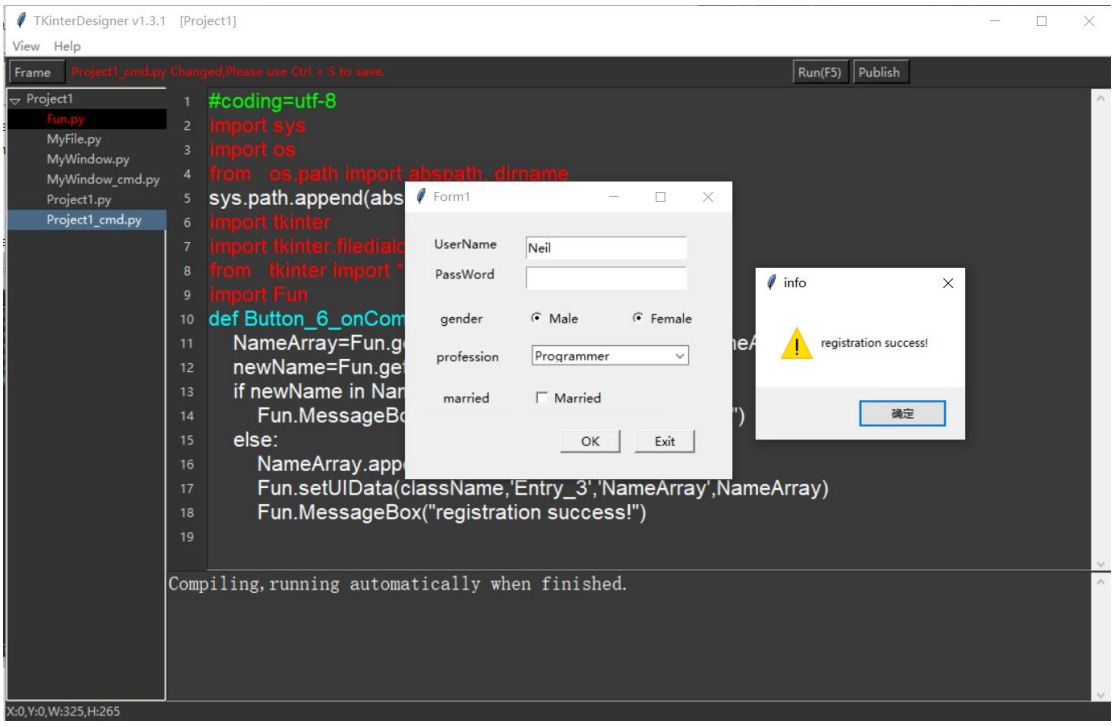


This code obtains the list variable bound to the input box through Fun.getUIData, and directly obtains the current input value through Fun.getUIText, and then compares it. If they are the same, a dialog box "the name has been registered" is displayed. If it is different, Add the input value to the list variable, and a dialog box "registration success" pops up.

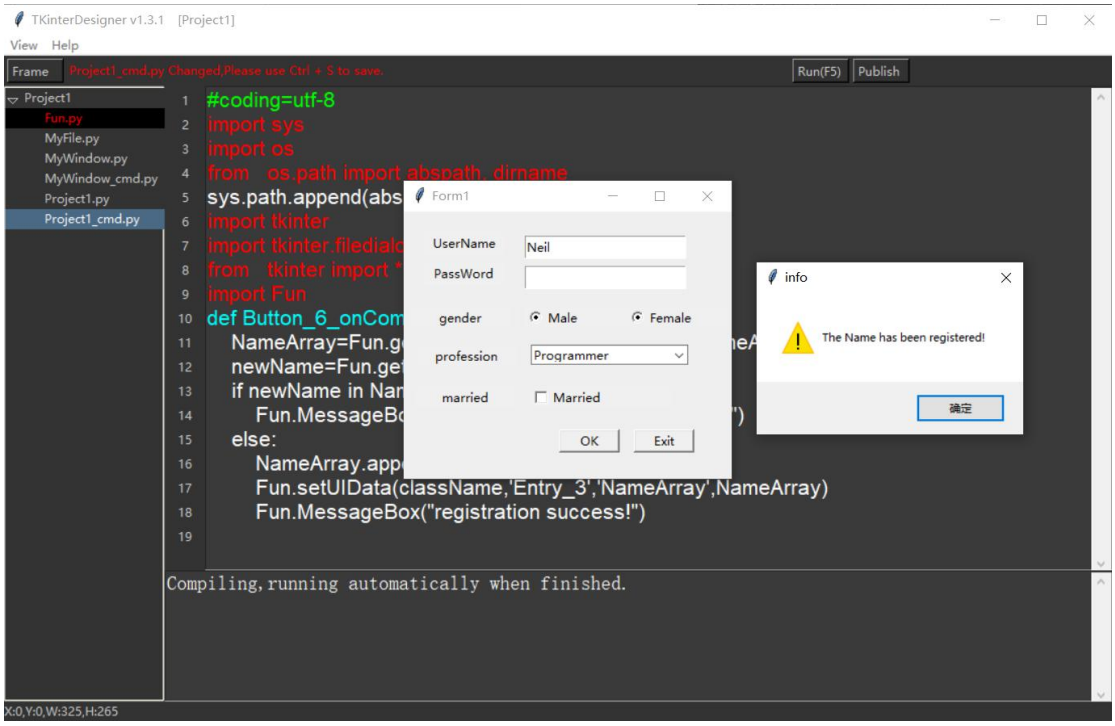
7. Compile and run

After developing here, you must want to compile and run immediately, then press F5 directly or click the "Run F5" button in the upper right, the program code will be automatically saved and start to compile and run. If there is an error in the code, it will be displayed in the compilation error information output window of the code area. If you add print to the function, it will also be displayed in real time.

We enter the guest name in the account of the running program and click "OK" for the first time, a "Registration Successful" dialog box will pop up.



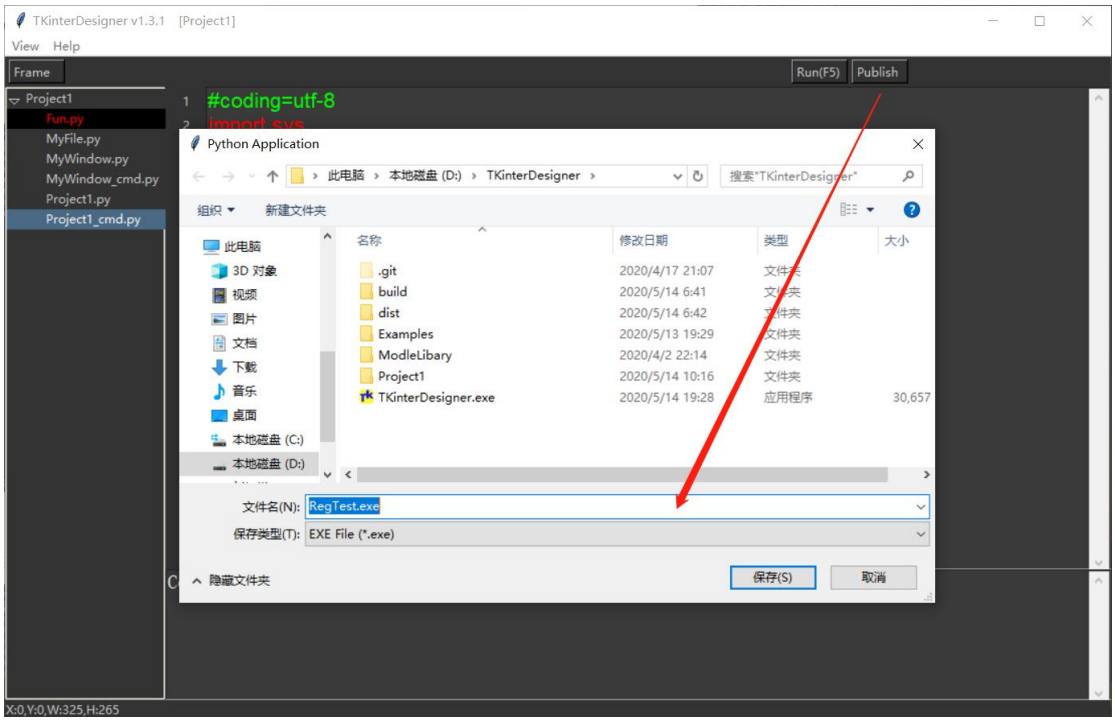
Then we click OK again, and the "Name has been registered" dialog box pops up.



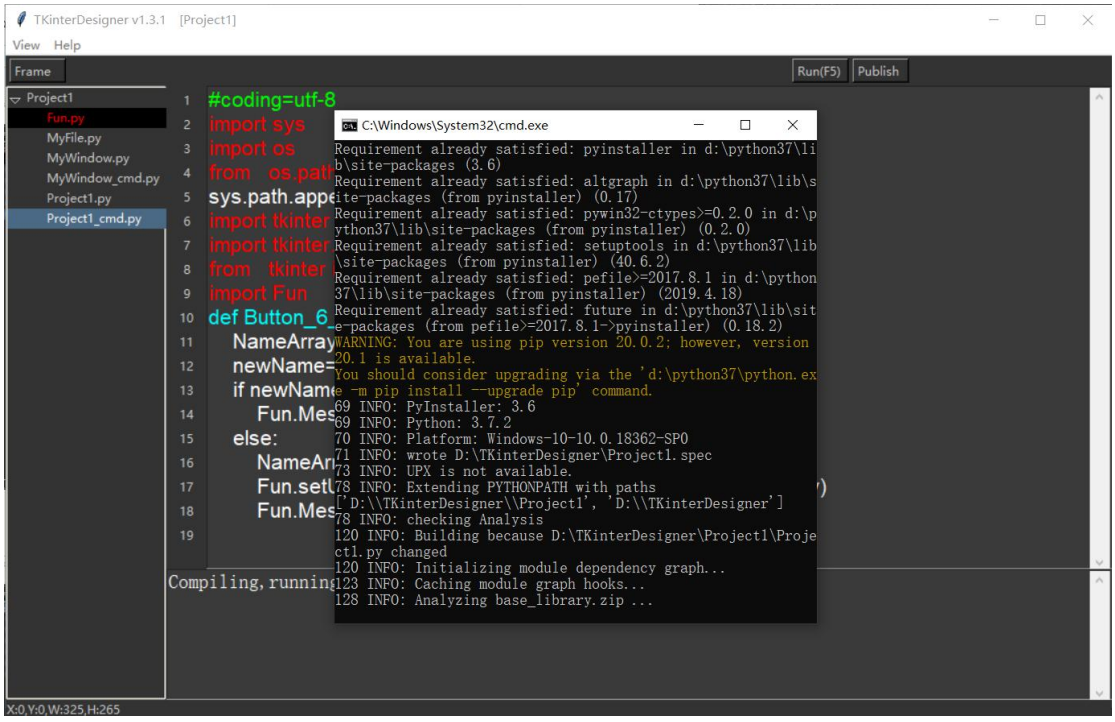
We can still register successfully by changing the name. It looks like everything is what we expected.

8. Pack EXE

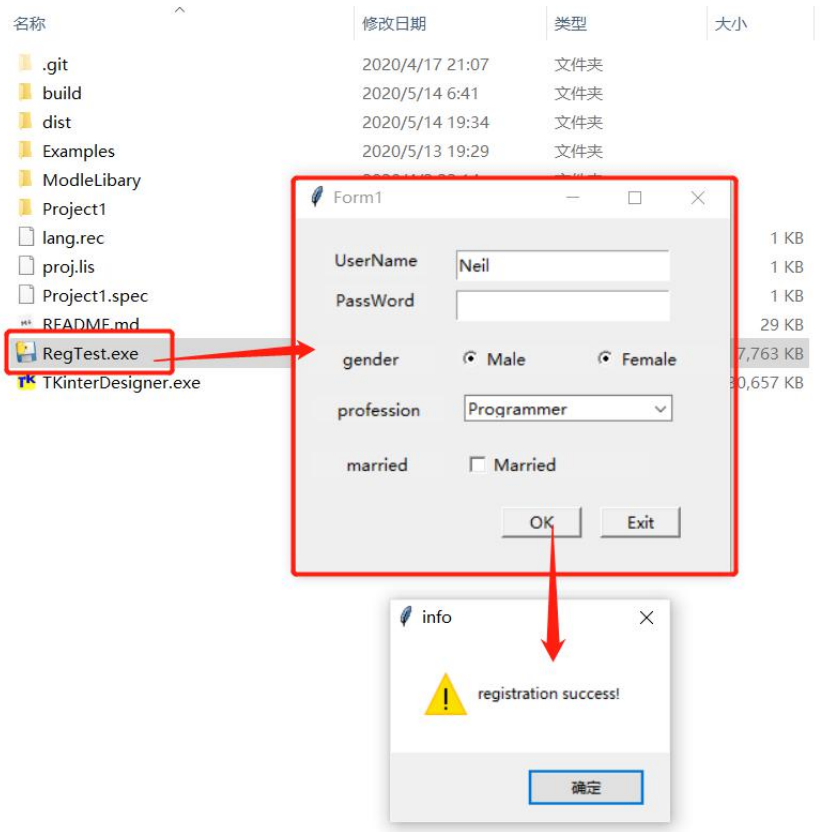
After we have perfected our own program, we hope to package the program as an EXE and publish it to users. We can directly click the "Publish" button on the upper right, and after selecting the output directory, enter the name of the EXE to be packed.



Click the "OK" button, TKinterDesigner will start calling the packaging program to package the project.



If it goes well, you can finally find the corresponding EXE program in the output directory：



9. Custom module import

I put the custom module at the end because you do n’ t have to use it, but it can easily extend your project. I use it in both the example projects Express_Query and Server.

Simply put, you can write a custom module class for multiple projects to use, you can easily set the properties of the module class in the designer, including throwing interface controls as a parameter to it.

Let's briefly explain the development process of the example project Express_Query.

First, we need to create a blank project, and right-click on the frame file tree item, create a Python file in the pop-up menu, and then name it Express.py. In this file, we need to create an Express class to pass the key Words to check the results of express delivery. The complete code for this class is as follows：

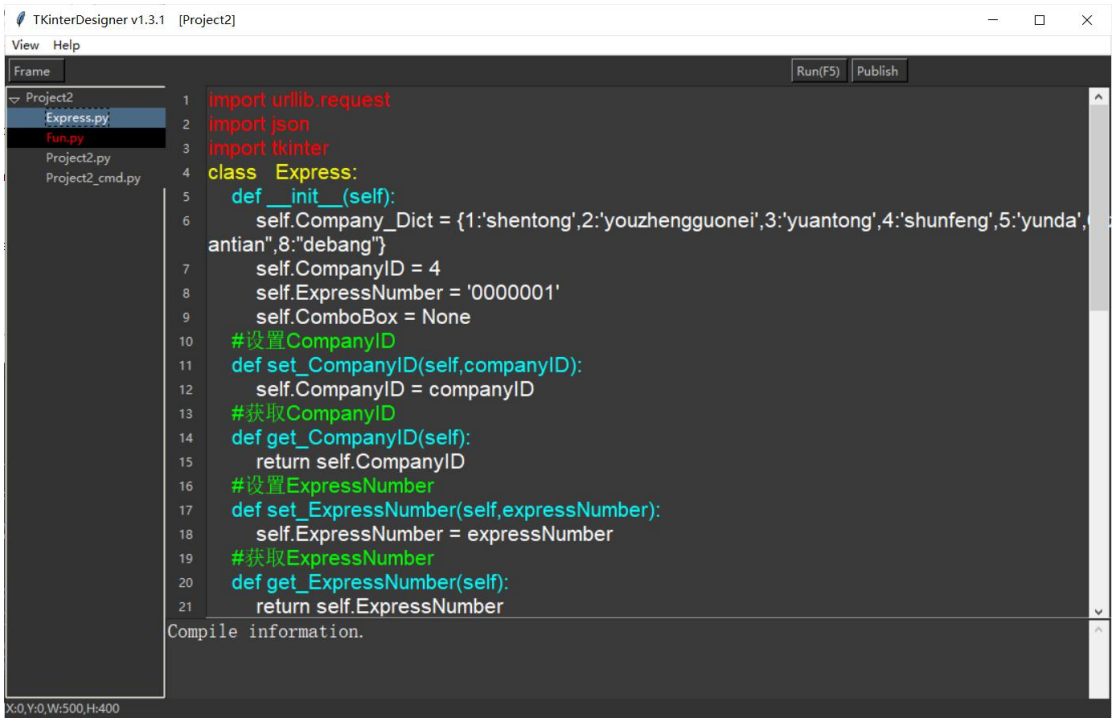
```
iimport urllib.request
import json
import msvcrt
import tkinter
```

```
class   Express:
    def __init__(self):
        self.Company_Dict = {1:'shentong',2:'youzhengguonei',3:'yuantong',4:'shunfeng',5:'yund
a',6:'zhongtong',7:"tiantian",8:"debang"}
        self.CompanyID = 4
        self.ExpressNumber = '0000001'
        self.ComboBox = None
    #Set CompanyID
    def set_CompanyID(self,companyID):
        self.CompanyID = companyID
    #Get CompanyID
    def get_CompanyID(self):
        return self.CompanyID
    #Set ExpressNumber
    def set_ExpressNumber(self,expressNumber):
        self.ExpressNumber = expressNumber
    #Get ExpressNumber
    def get_ExpressNumber(self):
        return self.ExpressNumber
    #Set ComboBox
    def set_ComboBox(self,comboBox):
        self.ComboBox = comboBox
        self.ComboBox['values'] = ['ShenTong Express','EMS','YuanTong Expree','SF
Express','YunDa delivery','China Express','Daily Express','Debon Express']
        self.ComboBox.current(4)
    #get ComboBox
    def get_ComboBox(self,comboBox):
        return self.ComboBox
    #Query
    def Query(self,ListBox):
        self.CompanyID = self.ComboBox.current() + 1
        ListBox.delete(0,tkinter.END)
        url = "http://www.kuaidi100.com/query?type=%s&postid=%s" % (self.Company_Dict[self.Com
panyID], self.ExpressNumber)
        response = urllib.request.urlopen(url)
        html = response.read().decode('utf-8')
        target = json.loads(html)
        #print(target)
        status = target['status']
        if status == '200':
            data = target['data']
            #print(data)
            data_len = len(data)
```

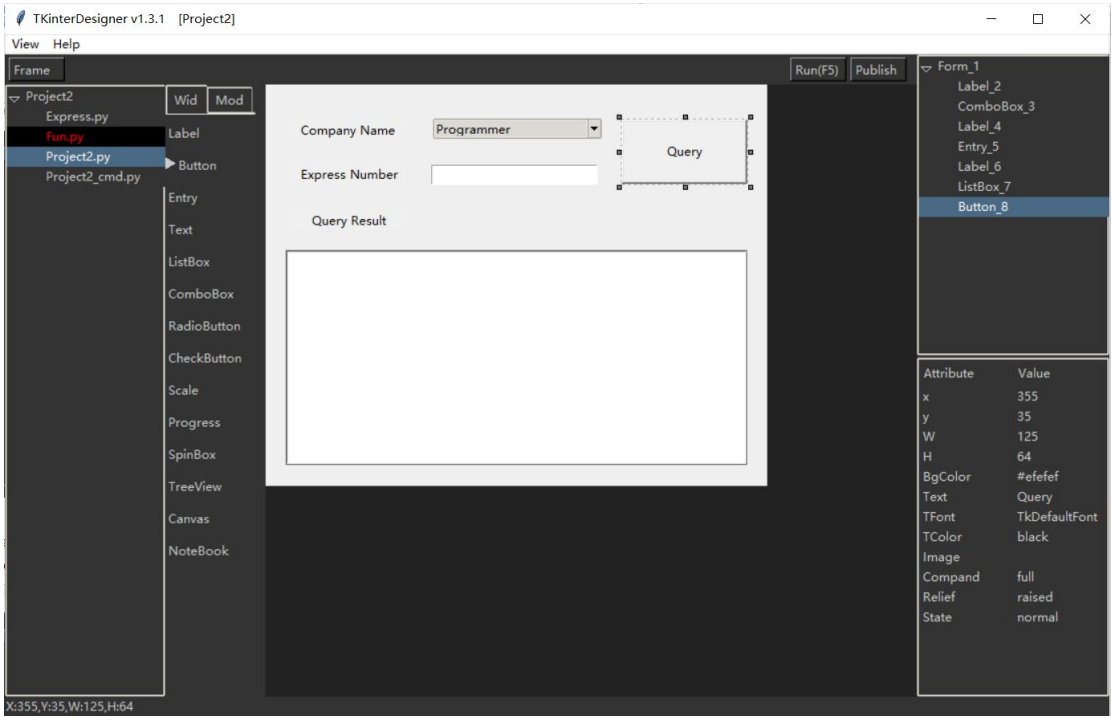
```
            for i in range(data_len):
                time_text = "Time: " + data[i]['time']
                ListBox.insert(tkinter.END,time_text)
                state_text = "State: " + data[i]['context']
                ListBox.insert(tkinter.END,state_text)
        else:
            ListBox.insert(tkinter.END,"Query error")
```

There is only one way to write a custom module class: if you want to pass parameters to it in the designer, you need to use variable access methods prefixed by set_ and get_. So here are a total of three variables related function design, including the company ID of the express company, express number ExpressNumber, and we hope to pass in a control ComboBox to accept the company name list.

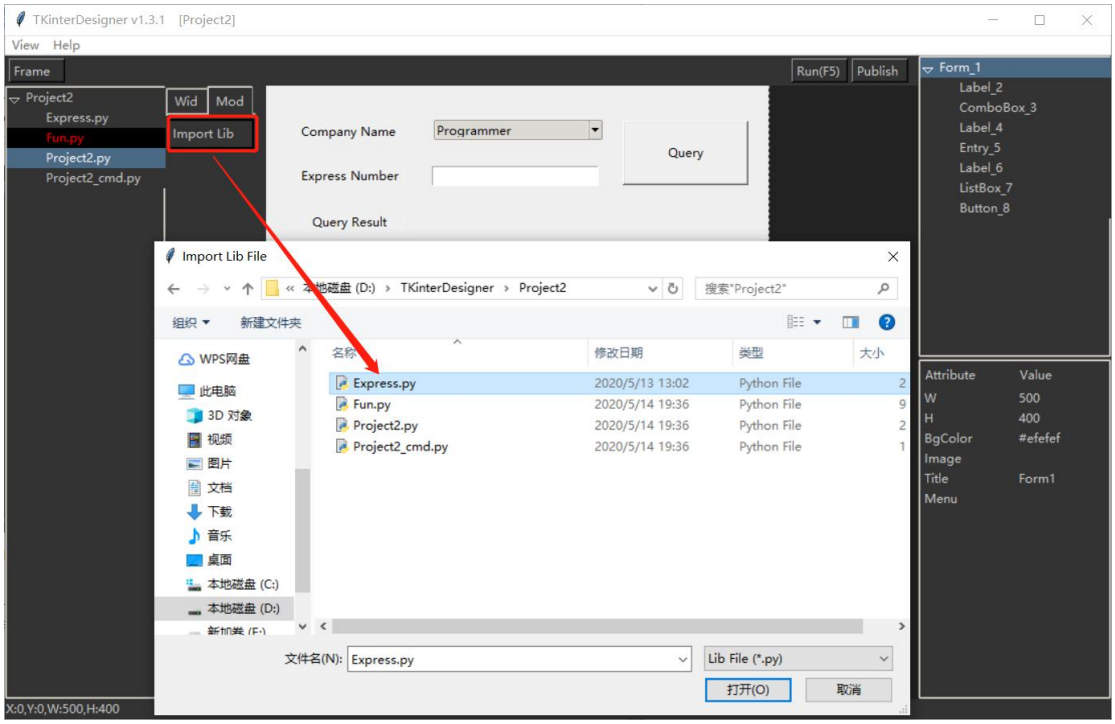
If you think the current engineering code is very inconvenient, you can also use VSCode or your favorite code editor to write the code, or directly import or copy Express.py from the example project



Back to the main interface design area, quickly build an interface：

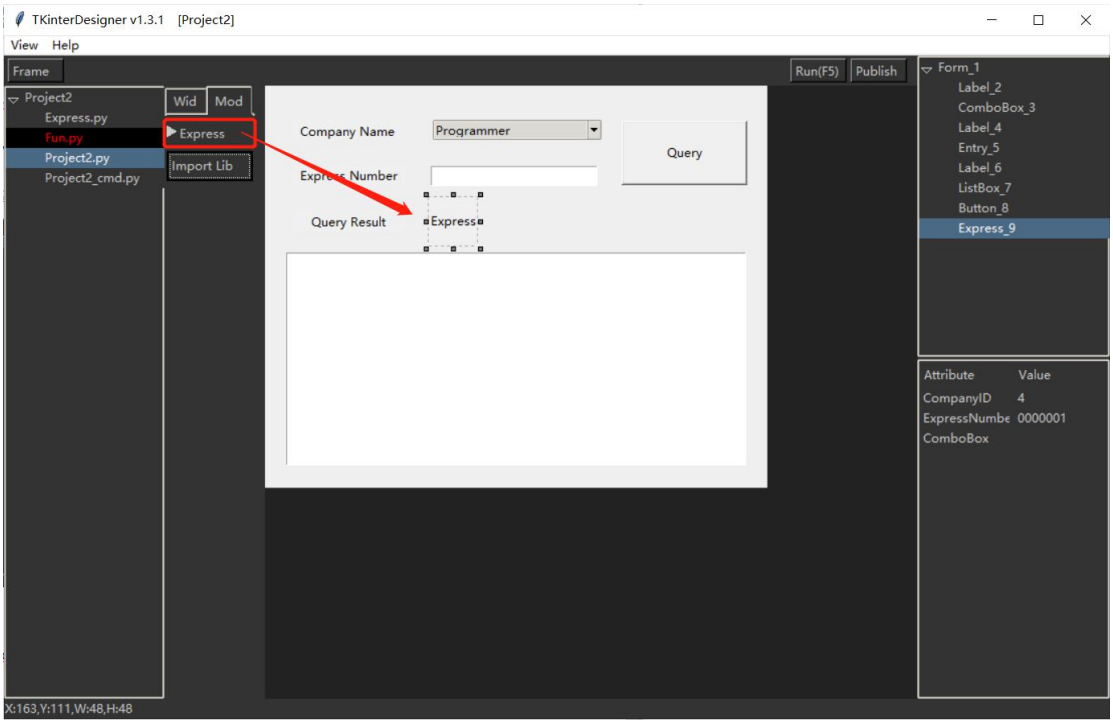


Now that we have completed the interface design work, we switch to the "Module" option in the "Control" and "Module" list selection areas on the left。

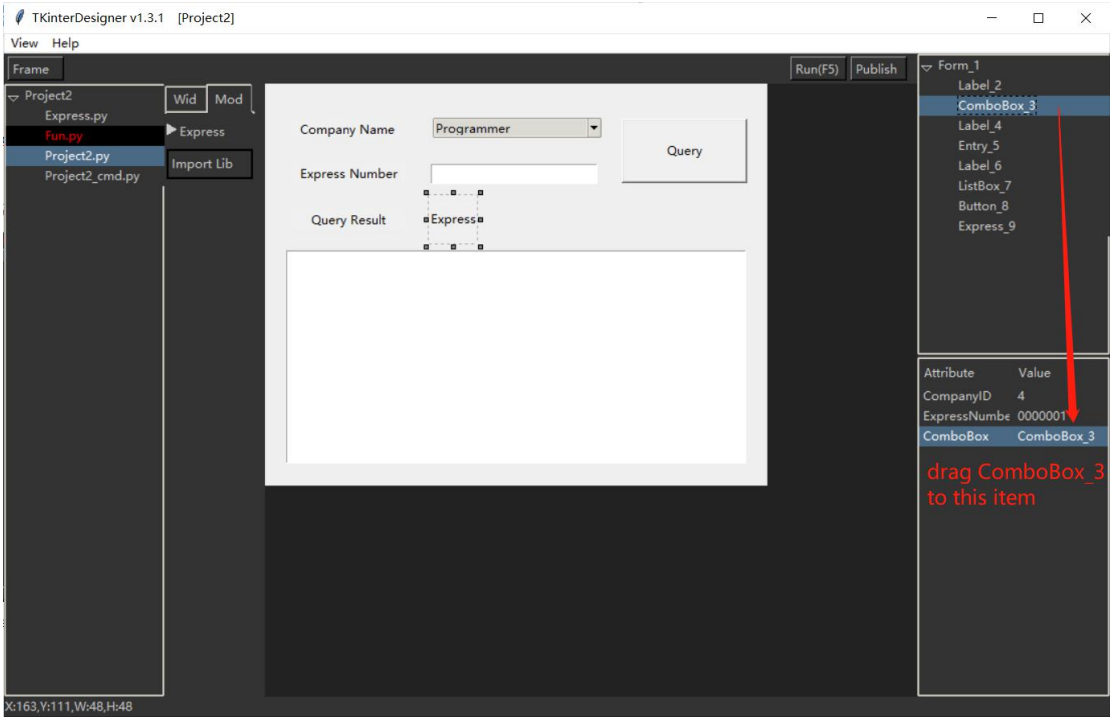


In the "Modules" option panel, we click the "Import Module" button, then find Express.py, and click "Open". Note here: In fact, Express.py does not have to be in the current project directory. You can use a module for multiple projects. You only need to import it here. You do n’t need to use every project that uses the same module. Create a module class file.

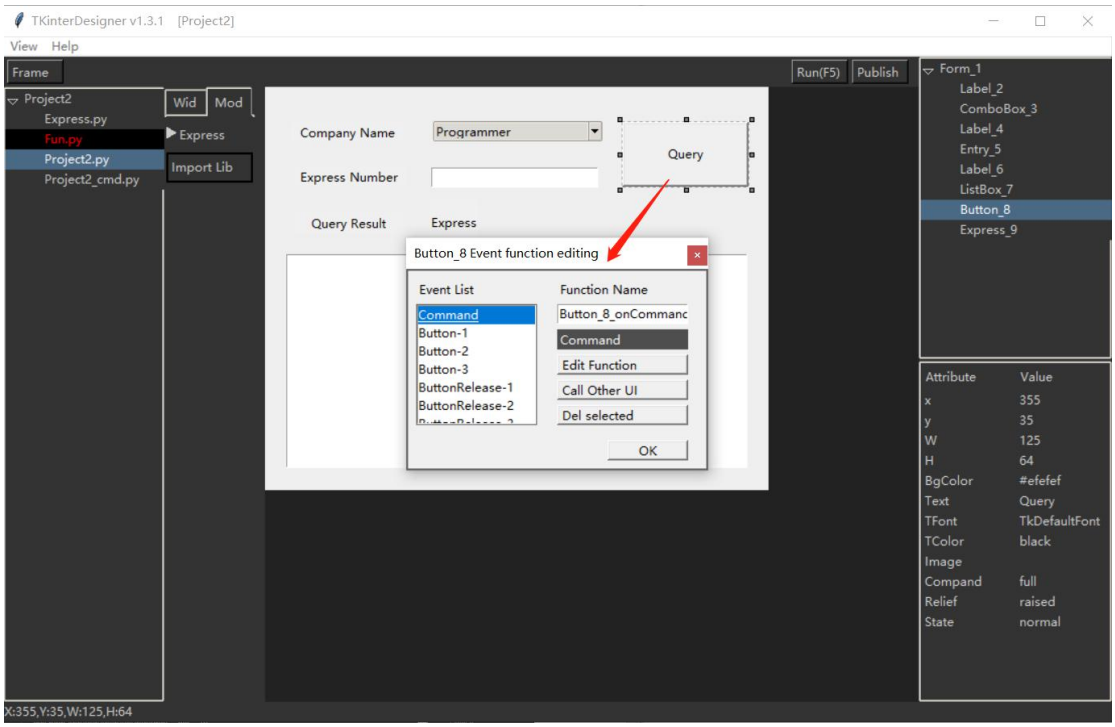
Ok, now the Express module item appears on the module panel, we drag it to the interface.



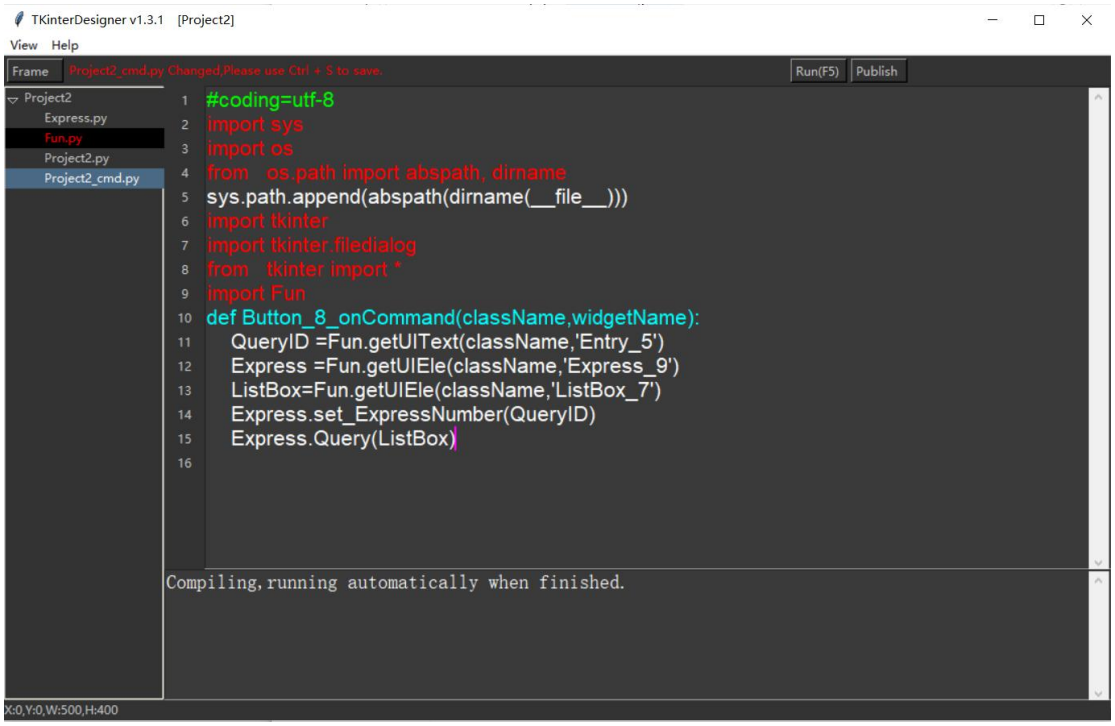
We can see that the three variables of the Express module are displayed in the property box in the lower right part. We can manually set the CompanyID and ExpressNumber, but how to set the ComboBox? Here you only need to find the corresponding ComboBox_3 under the control tree in the upper right and drag it to the value item position of the ComboBox of the property box.



Then we add Command response function to the "Query" button。



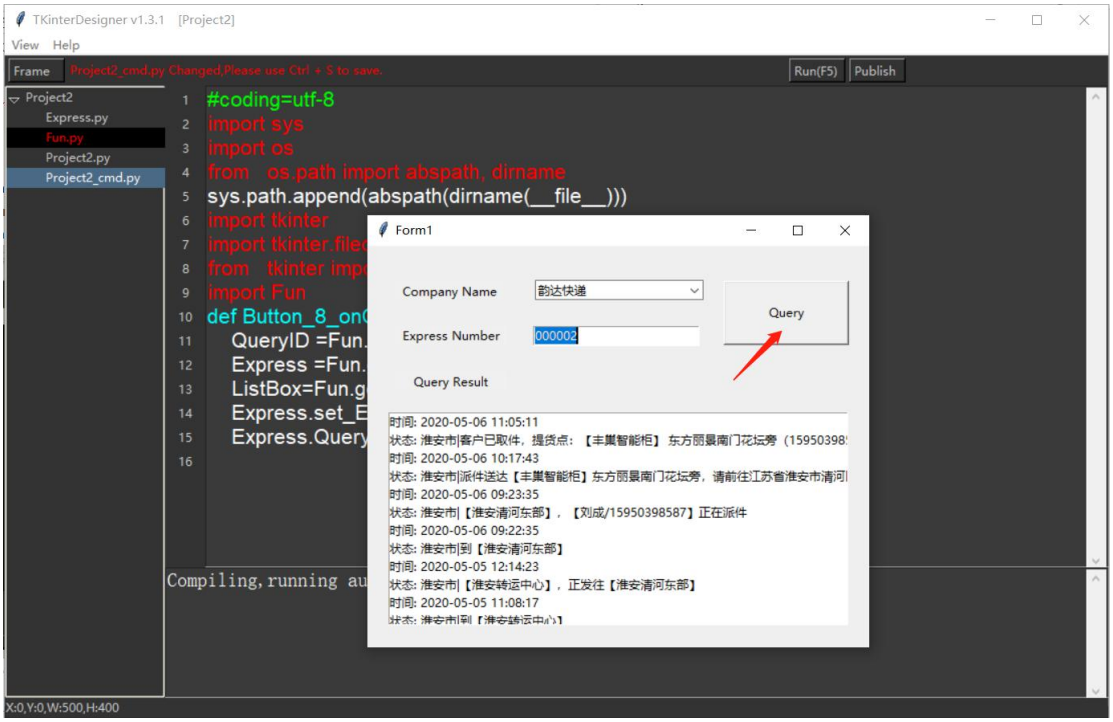
In the Button_6_onCommand function we can write the corresponding code：



This part of the code implements getting the express order number entered by Entry_5, and calling Fun.getUIEle through the module name "Express_9" to get the Express module, and using the same method to get the ListBox_7 object, then calling Express's set_ExpressNumber method,

setting the express order number Call the Query method to query, the parameter is the output ListBox object.

There are so many codes. After we finish, press F5 to run it. We can see the running program. After trying to enter the number of the express number, click "Query", and you will soon see the express information displayed in the list box.



This is how to use the custom module. Please forgive me for using an example of China Express inquiries. It outputs the diversion of packages from China Express.

About Me

NetName: Honghaier

Achievements: The original Engine director of the infinite world , the original COCOS game engine director , the CSDN blog expert, Cocos most valuable expert

QQ: 285421210

Email:honghaer2013@gmail.com

About TKinterDesigner

Because it's just a little motivation for learning Python intermittently for three months in my spare time, it has many questions

Topic, but I will continue to improve it, I hope that interested Python enthusiasts can communicate with me and give me more suggestions, because I have not read a Python book, there are too many needs and I do n't understand it, but I am very optimistic about Python's interface tool requirements for rapid prototyping, and I hope Python is getting better and better.

Finally, I wish you all a happy work and good health ~

Honghaier

2020/04/17