How to Create Dialogs in Python

Why Are These Dialogs Useful?

The dialogs in this post work on Windows, Linux and Mac and look exactly like the default dialogs the OS provides for other applications. Since they look the same as the 'default' dialogs, users are used to the look and feel, so they already know how to use them.

As these dialogs come with Python, they are easy to access and have had their time of debugging so are quite robust and predictable.

Dialog Types and Their Returns

Each dialog does something different, takes different input to be created and returns particular values. Understanding what each call returns in different situations is necessary to be able to use these dialogs correctly.

Some of these dialogs make sounds as you would typically find with dialogs displaying an error

In these examples, I will be referencing a `parent`. This parent is a `tkinter.Tk` object which acts as the "base" of the dialogs. Using a parent also allows you to **set an icon** on the dialog window and task-bar item. Here is an example:

```python
import tkinter

parent = tkinter.Tk() # Create the object
parent.overrideredirect(1) # Avoid it appearing and then disappearing quickly
parent.iconbitmap("PythonIcon.ico") # Set an icon (this is optional - must be in a .ico format)
parent.withdraw() # Hide the window as we do not want to see this one
```

When using an icon like this, the .ico must be an icon format - do not just rename the file.

You do not have to create a parent of your own and in that case, leave `parent=parent` out of the code snippets below. I do however recommend that you look at the side effects if you don't provide a parent as a `tkinter.Tk` object will be created which will then show a window that you cannot control. Using the parent snippet above makes things a lot easier.

It is important to note that all these dialog calls below are blocking. This means execution will not continue until the user has made a selection or this code is put into another thread.

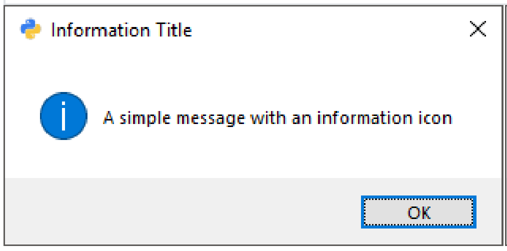
Message Dialogs

Message dialogs are a simple way to display a message with a title. There are three types of dialogs that simply provide an "OK" button; information, warning and error. Each of these dialogs display a title, message and icon that corresponds to the type of dialog.

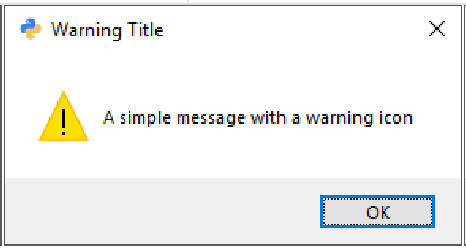
```python
from tkinter import messagebox

# After creating parent...

info = messagebox.showinfo('Information Title', 'A simple message with an information icon',
parent=parent)
```

```
warn = messagebox.showwarning('Warning Title', 'A simple message with a warning icon',
parent=parent)
error = messagebox.showerror('Error Title', 'A simple message with an error icon',
parent=parent)
```
Looking at the values of each `info`, `warn` and `error` after executing the lines, you will notice that they all have been assigned value "ok". If the user clicks the OK button or even closes the window using the cross, these dialogs will always return "ok".

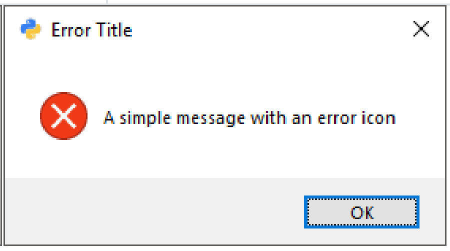
Method	OK Clicked	Dialog Closed (X)
messagebox.showinfo	"ok"	"ok"
messagebox.showwarning	"ok"	"ok"
messagebox.showerror	"ok"	"ok"


Information Title

A simple message with an information icon


Warning Title

A simple message with a warning icon


Error Title

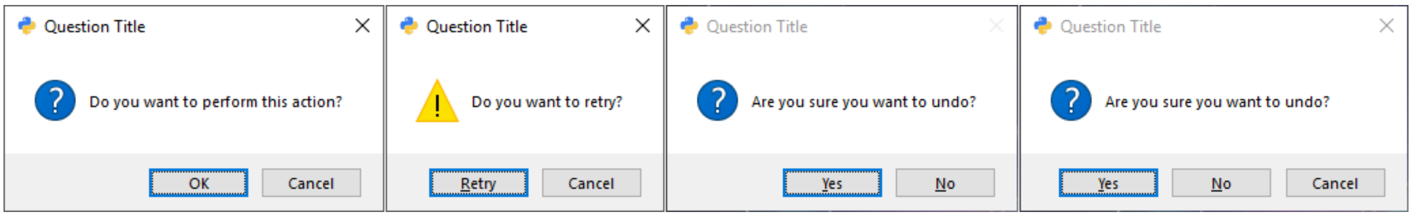
A simple message with an error icon

Question Dialogs

Question dialogs allow you to ask a user a particular question with a title and have them reply in one of two ways (or three in one case).

```
from tkinter import messagebox

# After creating parent...

okcancel = messagebox.askokcancel('Question Title', 'Do you want to perform this action?',
parent=parent) # OK / Cancel
retrycancel = messagebox.askretrycancel('Question Title', 'Do you want to retry?',
parent=parent) # Retry / Cancel
yesno = messagebox.askyesno('Question Title', 'Are you sure you want to undo?', parent=parent)
# Yes / No
yesnocancel = messagebox.askyesnocancel('Question Title', 'Are you sure you want to undo?',
parent=parent) # Yes / No / Cancel
```

Method	OK	Retry	Yes	No	Cancel	Dialog Closed (X)
messagebox.askokcancel	True				False	False
messagebox.askretrycancel		True			False	False
messagebox.askyesno			True	False		*Not Possible*
messagebox.askyesnocancel			True	False	None	None


Question Title

Do you want to perform this action?

OK Cancel

Question Title

Do you want to retry?

Retry Cancel

Question Title

Are you sure you want to undo?

Yes No

Question Title

Are you sure you want to undo?

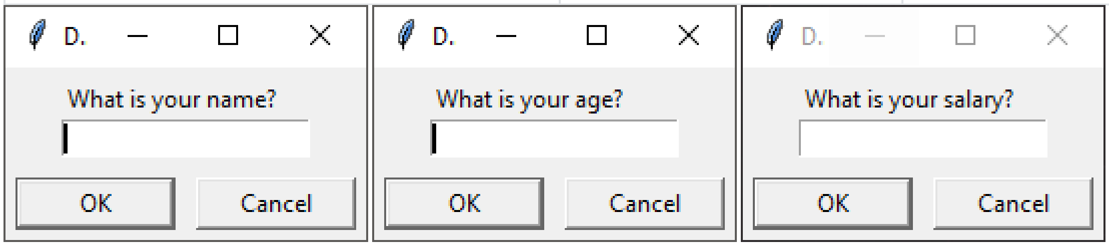
Yes No Cancel

Input Dialogs

Input dialogs ask for a particular type of value; in this case that is either of type string, integer or float. Unfortunately like all the other dialogs, the icon from the parent will not be used due to how these dialogs are constructed. If you plan on consistently taking input from a user I recommend creating your own input dialog.

```
from tkinter import simpledialog

# After creating parent...

string_value = simpledialog.askstring('Dialog Title', 'What is your name?', parent=parent)
integer_value = simpledialog.askinteger('Dialog Title', 'What is your age?', minvalue=0,
maxvalue=100, parent=parent)
float_value = simpledialog.askfloat('Dialog Title', 'What is your salary?', minvalue=0.0,
maxvalue=100000.0, parent=parent)
```

Method	OK	Cancel	Dialog Closed (X)
simpledialog.askstring	*Input String*	None	None
simpledialog.askinteger	*Input Integer*	None	None
simpledialog.askfloat	*Input Float*	None	None


D.

What is your name?

OK Cancel

D.

What is your age?

OK Cancel

D.

What is your salary?

OK Cancel

File / Folder Dialogs

File and folder dialogs allow you to ask the user select a folder, single file, multiple files or chose a location to save a file as.

```
from tkinter import filedialog

# After creating parent...

directory = filedialog.askdirectory(title='Select a folder', parent=parent)

# Ask the user to select a single file name.
file_name = filedialog.askopenfilename(title='Select a file', parent=parent)

# Ask the user to select a one or more file names.
file_names = filedialog.askopenfilenames(title='Select one or more files', parent=parent)
```

```
# Ask the user to select a single file name for saving.
save_as = filedialog.asksaveasfilename(title='Save as', parent=parent)
```

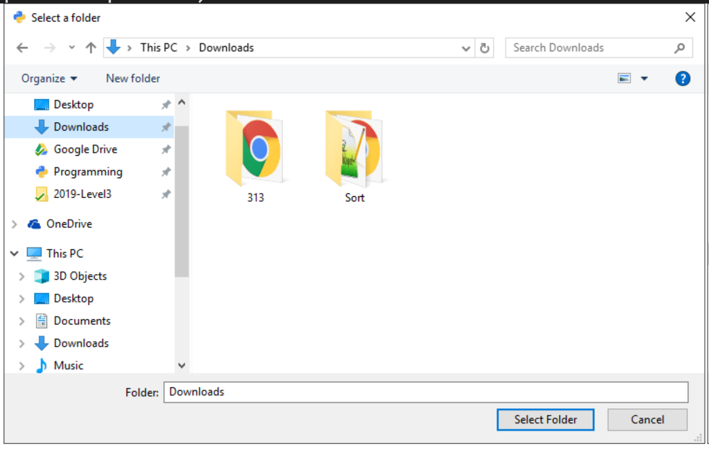
Method	OK	Cancel	Dialog Closed (X)
filedialog.askdirectory	*Directory Path*	"	"
filedialog.askopenfilename	*File Path*	"	"
filedialog.askopenfilenames	*List of File Paths*	"	"
filedialog.asksaveasfilename	*File Path*	"	"

When using any of these file dialog calls, you can also supply a string to the parameter `initialdir` which will put the user in a particular directory when the dialog first opens; for example:

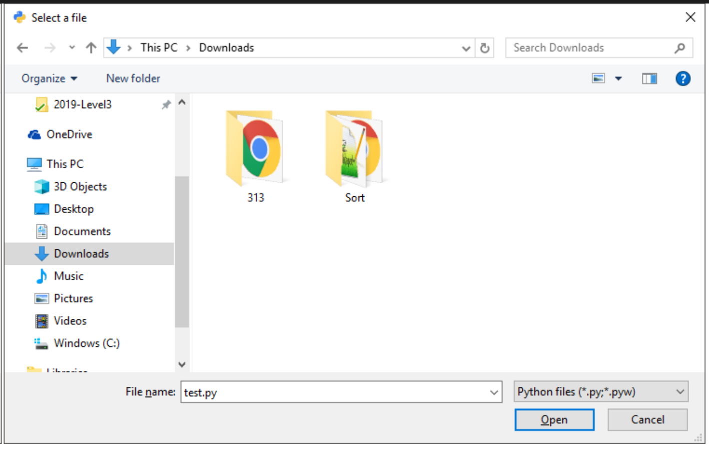
```
directory_to_start_from = 'C:/Users/User/Downloads/'
directory = filedialog.askdirectory(initialdir=directory_to_start_from, title='Please select a folder:', parent=parent)
```

When asking for a file to open or save as, you can specify what type of files the user can select (based off file extension) using the `filetypes` argument. This argument takes a list of tuples, where each of these tuples contain a string of the group name and a string of file extensions separated by `;` if required. The tuple that is the first in the list is the default selection. An example of the usage of `filetypes`:

```
file_types = [('Python files', '*.py;*.pyw'), ('All files', '*')] # Initially can select any .py or .pyw files but can change the selection to anything (*)
file_name = filedialog.askopenfilename(title='Select a file', filetypes=file_types, parent=parent)
```



Folder: Downloads



File name: test.py

Python files (*.py;*.pyw)

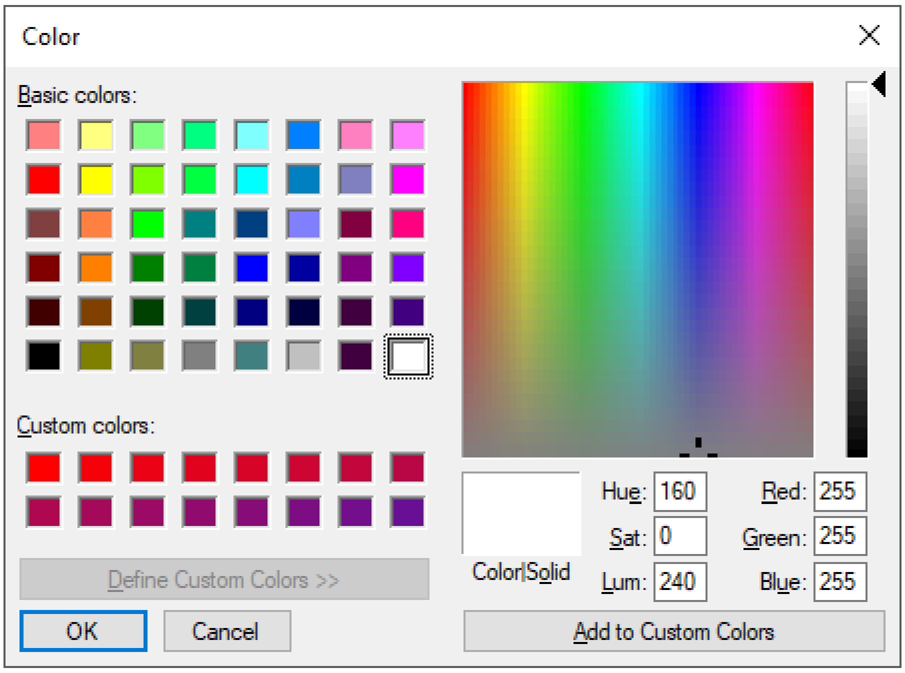
Color Picker Dialog

Color pickers are used to ask a user to select a color. These aren't seen in may places but when a color needs to be selected, this is a great option as it contains everything needed.

```
from tkinter import colorchooser

# After creating parent...

color = colorchooser.askcolor(initialcolor=(255, 255, 255), parent=parent) # Returns a tuple
```

Method	OK	Cancel	Dialog Closed (X)
filedialog.askdirectory	((*Red Int*, *Green Int*, *Blue Int*), '*Hex Value*')	(None, None)	(None, None)

As an example of the output, if I select cyan, I will get `((0.0, 255.99609375, 255.99609375), '#00ffff')`. The numbers are large as they are floats but this means 0 red, 256 green and 256 blue. We can also see that the second value in the tuple is the hex value for cyan.



Hue: 160
Sat: 0
Lum: 240
Red: 255
Green: 255
Blue: 255