functool.partial

One of the functools classes is the **partial** class. You can use it create a new function with partial application of the arguments and keywords that you pass to it. You can use partial to “freeze” a portion of your function’s arguments and/or keywords which results in a new object. Another way to put it is that partial creates a new function with some defaults. Let’s look at an example!

```
from functools import partial
def add(x, y):
    return x + y

p_add = partial(add, 2)
print(p_add(4))
#6
```

Here we create a simple adding function that returns the result of adding its arguments, x and y. Next we create a new callable by creating an instance of partial and passing it our function and an argument for that function. In other words, we are basically defaulting the x parameter of our add function to the number 2. Finally we call our new callable, p_add, with the argument of the number 4 which results in 6 because 2 + 4 = 6.

One handy use case for partials is to pass arguments to callbacks. Let’s take a look using wxPython:

```
import wx

from functools import partial


class MainFrame(wx.Frame):
    """
    This app shows a group of buttons
    """

    def __init__(self, *args, **kwargs):
        """Constructor"""
        super(MainFrame, self).__init__(parent=None, title='Partial')
        panel = wx.Panel(self)

        sizer = wx.BoxSizer(wx.VERTICAL)
        btn_labels = ['one', 'two', 'three']
        for label in btn_labels:
            btn = wx.Button(panel, label=label)
            btn.Bind(wx.EVT_BUTTON, partial(self.onButton, label=label))
            sizer.Add(btn, 0, wx.ALL, 5)

        panel.SetSizer(sizer)
        self.Show()

    def onButton(self, event, label):
        """
        Event handler called when a button is pressed
        """
        print('You pressed: ' +  str(label))


if __name__ == '__main__':
    app = wx.App(False)
    frame = MainFrame()
    app.MainLoop()
```

Here we use partial to call the onButton event handler with an extra argument, which happens to be the button’s label. This might not seem all that useful to you, but if you do much GUI programming, you’ll see a lot of people asking how to do this sort of thing. Of course, you could also use a lambda instead for passing arguments to callbacks.

One use case that we’ve used at work was for our automated test framework. We test a UI with Python and we wanted to be able to pass a function along to dismiss certain dialogs. Basically you would pass a function along with the name of the dialog to dismiss, but it would need to be called at a certain point in the process to work correctly. Since I can’t show that code, here’s a really basic example of passing a partial function around:

```
from functools import partial


def add(x, y):
    """"""
    return x + y


def multiply(x, y):
    """"""
    return x * y



def run(func):
    """"""
    print(func())


def main():
    """"""
    a1 = partial(add, 1, 2)
    m1 = partial(multiply, 5, 8)
    run(a1)
    run(m1)

if __name__ == "__main__":
    main()
```

Here we create a couple of partial functions in our main function. Next we pass those partials to our run function, call it and then print out the result of the function that was called.