A Keypad Component

Create a Keypad using the Composite Pattern

Create a Keypad with a configurable set of keys and bind method that behaves like the bind of other tkinter components. The Keypad uses the Composite Design Pattern.

Benefits: 1. The Keypad looks and behaves like an ordinary tkinter component.
2. We can create many instances of Keypad or reuse Keypad in other applications.

Step 0. Create a repository from the starter code on Github.

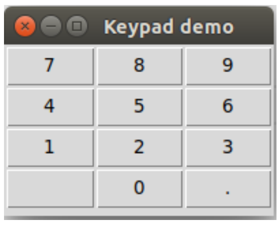
https://classroom.github.com/a/FHHMzPM6

Step 1. Complete the Keypad class.

Keypad should be a subclass of `ttk.Frame`. The constructor signature is:
`__init__(self, parent, keynames=[ ], columns=1, **kwargs)`

The constructor should invoke the superclass constructor first, and pass all parameters except `keynames` and `columns` to the superclass constructor,

1. Complete the init_components method. Create a Button component for each element in the keynames list.
2. Layout the buttons using a grid with `columns` (constructor parameter) columns and as many rows as necessary. In case there are not enough keynames to fill the last row of the grid, just leave the grid cells empty -- do **not** create more buttons than the keynames list size.
   Add 1 pixel of padding between keys.



```
Keypad(root, keynames=['7','8','9','4','5','6',...,'0','.'], columns=3)
```

Step 2. Make the Keys resizable.

If the user resizes the area containing a Keypad, the keys should all resize consistently.

Use self.columnconfigure and self.rowconfigure for this. tkinter provides a method to discover how many rows and columns are in a grid layout:

```
(cols, rows) = self.grid_size()
for row in range(rows):
     self.rowconfigure(row, weight=1)
#TODO set the column weights, too
```

Step 3: Override bind() to bind all the keys

This is the key step in applying the Composite pattern.
You want to make the composite behave exactly like a simple component.

We want the Keypad to behave like a single component. This means you must **override** important methods from the superclass, and pass the method call to each element of the composite.

How do you know what Buttons are in the Frame (the composite)? Use **`self.winfo_children()`** which returns a list of all widgets that are children of "self" widget (the Frame):

We want to override the bind() method, so **you must** know the formal parameters of the Frame class's bind method so you can correctly "pass" the method call to the superclass. Enter this:

```
>>> from tkinter import ttk
>>> help(ttk.Frame.bind)
bind(self, sequence=None, func=None, add=None)
    Bind to this widget at event SEQUENCE a call to function FUNC.
```

```
# Override bind of the Frame class (superclass)
def bind(self, sequence=None, func=None, add=None):
    """Bind the event sequence of all the keys to func.
    """
    super().bind(sequence, func, add)

    for child in self.winfo_children():
        child.bind(sequence, func, add)
```

In the __main__ block, add some code to verify that when you "bind" a button-click event to the keypad, your code receives notification when the user clicks on any of the keys.

Step 4: Create a mini-calculator with Display and Operators

To be completed.