```
"""
Simple calendar using ttk Treeview together with calendar and
datetime
classes.
"""
import calendar

try:
    import Tkinter
    import tkFont
except ImportError: # py3k
    import tkinter as Tkinter
    import tkinter.font as tkFont

from tkinter import ttk

def get_calendar(locale, fwday):
    # instantiate proper calendar class
    if locale is None:
        return calendar.TextCalendar(fwday)
    else:
        return calendar.LocaleTextCalendar(fwday, locale)

class Calendar(ttk.Frame):
    # XXX ToDo: cget and configure

    datetime = calendar.datetime.datetime
    timedelta = calendar.datetime.timedelta

    def __init__(self, master=None, **kw):
        """
        WIDGET-SPECIFIC OPTIONS

            locale, firstweekday, year, month, selectbackground,
            selectforeground
        """
        # remove custom options from kw before initializating
ttk.Frame
        fwday = kw.pop('firstweekday', calendar.MONDAY)
        year = kw.pop('year', self.datetime.now().year)
        month = kw.pop('month', self.datetime.now().month)
        locale = kw.pop('locale', None)
        sel_bg = kw.pop('selectbackground', '#ecffc4')
        sel_fg = kw.pop('selectforeground', '#05640e')

        self._date = self.datetime(year, month, 1)
        self._selection = None # no date selected

        ttk.Frame.__init__(self, master, **kw)

        self._cal = get_calendar(locale, fwday)

        self.__setup_styles()       # creates custom styles
        self.__place_widgets()      # pack/grid used widgets
```

```
        self.__config_calendar()    # adjust calendar columns and
setup tags
        # configure a canvas, and proper bindings, for selecting
dates
        self.__setup_selection(sel_bg, sel_fg)

        # store items ids, used for insertion later
        self._items = [self._calendar.insert('', 'end', values='')
                            for _ in range(6)]
        # insert dates in the currently empty calendar
        self._build_calendar()

        # set the minimal size for the widget
        self._calendar.bind('<Map>', self.__minsize)

    def __setitem__(self, item, value):
        if item in ('year', 'month'):
            raise AttributeError("attribute '%s' is not writeable" %
item)
        elif item == 'selectbackground':
            self._canvas['background'] = value
        elif item == 'selectforeground':
            self._canvas.itemconfigure(self._canvas.text,
item=value)
        else:
            ttk.Frame.__setitem__(self, item, value)

    def __getitem__(self, item):
        if item in ('year', 'month'):
            return getattr(self._date, item)
        elif item == 'selectbackground':
            return self._canvas['background']
        elif item == 'selectforeground':
            return self._canvas.itemcget(self._canvas.text, 'fill')
        else:
            r = ttk.tclobjs_to_py({item: ttk.Frame.__getitem__(self,
item)})
            return r[item]

    def __setup_styles(self):
        # custom ttk styles
        style = ttk.Style(self.master)
        arrow_layout = lambda dir: (
            [('Button.focus', {'children': [('Button.%sarrow' % dir,
None)]})]
        )
        style.layout('L.TButton', arrow_layout('left'))
        style.layout('R.TButton', arrow_layout('right'))

    def __place_widgets(self):
        # header frame and its widgets
        hframe = ttk.Frame(self)
        lbtn = ttk.Button(hframe, style='L.TButton',
command=self._prev_month)
```

```
        rbtn = ttk.Button(hframe, style='R.TButton',
command=self._next_month)
        self._header = ttk.Label(hframe, width=15, anchor='center')
        # the calendar
        self._calendar = ttk.Treeview(show='', selectmode='none',
height=7)

        # pack the widgets
        hframe.pack(in_=self, side='top', pady=4, anchor='center')
        lbtn.grid(in_=hframe)
        self._header.grid(in_=hframe, column=1, row=0, padx=12)
        rbtn.grid(in_=hframe, column=2, row=0)
        self._calendar.pack(in_=self, expand=1, fill='both',
side='bottom')

    def __config_calendar(self):
        cols = self._cal.formatweekheader(3).split()
        self._calendar['columns'] = cols
        self._calendar.tag_configure('header', background='grey90')
        self._calendar.insert('', 'end', values=cols, tag='header')
        # adjust its columns width
        font = tkFont.Font()
        maxwidth = max(font.measure(col) for col in cols)
        for col in cols:
            self._calendar.column(col, width=maxwidth,
minwidth=maxwidth,
                anchor='e')

    def __setup_selection(self, sel_bg, sel_fg):
        self._font = tkFont.Font()
        self._canvas = canvas = Tkinter.Canvas(self._calendar,
            background=sel_bg, borderwidth=0, highlightthickness=0)
        canvas.text = canvas.create_text(0, 0, fill=sel_fg,
anchor='w')

        canvas.bind('<ButtonPress-1>', lambda evt:
canvas.place_forget())
        self._calendar.bind('<Configure>', lambda evt:
canvas.place_forget())
        self._calendar.bind('<ButtonPress-1>', self._pressed)

    def __minsize(self, evt):
        width, height = self._calendar.master.geometry().split('x')
        height = height[:height.index('+')]
        self._calendar.master.minsize(width, height)

    def _build_calendar(self):
        year, month = self._date.year, self._date.month

        # update header text (Month, YEAR)
        header = self._cal.formatmonthname(year, month, 0)
        self._header['text'] = header.title()

        # update calendar shown dates
```

```
        cal = self._cal.monthdayscalendar(year, month)
        for indx, item in enumerate(self._items):
            week = cal[indx] if indx < len(cal) else []
            fmt_week = [('%02d' % day) if day else '' for day in
week]
            self._calendar.item(item, values=fmt_week)

    def _show_selection(self, text, bbox):
        """Configure canvas for a new selection."""
        x, y, width, height = bbox

        textw = self._font.measure(text)

        canvas = self._canvas
        canvas.configure(width=width, height=height)
        canvas.coords(canvas.text, width - textw, height / 2 - 1)
        canvas.itemconfigure(canvas.text, text=text)
        canvas.place(in_=self._calendar, x=x, y=y)

    # Callbacks

    def _pressed(self, evt):
        """Clicked somewhere in the calendar."""
        x, y, widget = evt.x, evt.y, evt.widget
        item = widget.identify_row(y)
        column = widget.identify_column(x)

        if not column or not item in self._items:
            # clicked in the weekdays row or just outside the
columns
            return

        item_values = widget.item(item)['values']
        if not len(item_values): # row is empty for this month
            return

        text = item_values[int(column[1]) - 1]
        if not text: # date is empty
            return

        bbox = widget.bbox(item, column)
        if not bbox: # calendar not visible yet
            return

        # update and then show selection
        text = '%02d' % text
        self._selection = (text, item, column)
        self._show_selection(text, bbox)

    def _prev_month(self):
        """Updated calendar to show the previous month."""
        self._canvas.place_forget()

        self._date = self._date - self.timedelta(days=1)
```

```
        self._date = self.datetime(self._date.year,
self._date.month, 1)
        self._build_calendar() # reconstuct calendar

    def _next_month(self):
        """Update calendar to show the next month."""
        self._canvas.place_forget()

        year, month = self._date.year, self._date.month
        self._date = self._date + self.timedelta(
            days=calendar.monthrange(year, month)[1] + 1)
        self._date = self.datetime(self._date.year,
self._date.month, 1)
        self._build_calendar() # reconstruct calendar

    # Properties

    @property
    def selection(self):
        """Return a datetime representing the current selected
date."""
        if not self._selection:
            return None

        year, month = self._date.year, self._date.month
        return self.datetime(year, month, int(self._selection[0]))

def test():
    import sys
    root = Tkinter.Tk()
    root.title('Ttk Calendar')
    ttkcal = Calendar(firstweekday=calendar.SUNDAY)
    ttkcal.pack(expand=1, fill='both')

    if 'win' not in sys.platform:
        style = ttk.Style()
        style.theme_use('clam')

    root.mainloop()

if __name__ == '__main__':
    test()
```