# smartmirror.py

# requirements

# requests, feedparser, traceback, Pillow

**from** **Tkinter** **import** \*

**import** **locale**

**import** **threading**

**import** **time**

**import** **requests**

**import** **json**

**import** **traceback**

**import** **feedparser**

**from** **PIL** **import** Image, ImageTk

**from** **contextlib** **import** contextmanager

LOCALE\_LOCK = threading.Lock()

ip = '197.49.129.201'

ui\_locale = '' # e.g. 'fr\_FR' fro French, '' as default

time\_format = **12** # 12 or 24

date\_format = "%b %d, %Y" # check python doc for strftime() for options

news\_country\_code = 'EG'

weather\_api\_token = "3ca72348dd0be3026126487e3419b240" # create account at https://darksky.net/dev/

weather\_lang = 'en' # see https://darksky.net/dev/docs/forecast for full list of language parameters values

weather\_unit = 'si' # see https://darksky.net/dev/docs/forecast for full list of unit parameters values

latitude = '30.06263'# Set this if IP location lookup does not work for you (must be a string)

longitude = '31.24967' # Set this if IP location lookup does not work for you (must be a string)

xlarge\_text\_size = **94**

large\_text\_size = **48**

medium\_text\_size = **28**

small\_text\_size = **18**

**@contextmanager**

**def** **setlocale**(name): #thread proof function to work with locale

**with** LOCALE\_LOCK:

saved = locale.setlocale(locale.LC\_ALL)

**try**:

**yield** locale.setlocale(locale.LC\_ALL, name)

**finally**:

locale.setlocale(locale.LC\_ALL, saved)

# maps open weather icons to

# icon reading is not impacted by the 'lang' parameter

icon\_lookup = {

'clear-day': "assets/Sun.png", # clear sky day

'wind': "assets/Wind.png", #wind

'cloudy': "assets/Cloud.png", # cloudy day

'partly-cloudy-day': "assets/PartlySunny.png", # partly cloudy day

'rain': "assets/Rain.png", # rain day

'snow': "assets/Snow.png", # snow day

'snow-thin': "assets/Snow.png", # sleet day

'fog': "assets/Haze.png", # fog day

'clear-night': "assets/Moon.png", # clear sky night

'partly-cloudy-night': "assets/PartlyMoon.png", # scattered clouds night

'thunderstorm': "assets/Storm.png", # thunderstorm

'tornado': "assests/Tornado.png", # tornado

'hail': "assests/Hail.png" # hail

}

**class** **Clock**(Frame):

**def** **\_\_init\_\_**(self, parent, \*args, \*\*kwargs):

Frame.\_\_init\_\_(self, parent, bg='black')

# initialize time label

self.time1 = ''

self.timeLbl = Label(self, font=('Helvetica', large\_text\_size), fg="white", bg="black")

self.timeLbl.pack(side=TOP, anchor=E)

# initialize day of week

self.day\_of\_week1 = ''

self.dayOWLbl = Label(self, text=self.day\_of\_week1, font=('Helvetica', small\_text\_size), fg="white", bg="black")

self.dayOWLbl.pack(side=TOP, anchor=E)

# initialize date label

self.date1 = ''

self.dateLbl = Label(self, text=self.date1, font=('Helvetica', small\_text\_size), fg="white", bg="black")

self.dateLbl.pack(side=TOP, anchor=E)

self.tick()

**def** **tick**(self):

**with** setlocale(ui\_locale):

**if** time\_format == **12**:

time2 = time.strftime('%I:%M %p') #hour in 12h format

**else**:

time2 = time.strftime('%H:%M') #hour in 24h format

day\_of\_week2 = time.strftime('%A')

date2 = time.strftime(date\_format)

# if time string has changed, update it

**if** time2 != self.time1:

self.time1 = time2

self.timeLbl.config(text=time2)

**if** day\_of\_week2 != self.day\_of\_week1:

self.day\_of\_week1 = day\_of\_week2

self.dayOWLbl.config(text=day\_of\_week2)

**if** date2 != self.date1:

self.date1 = date2

self.dateLbl.config(text=date2)

# calls itself every 200 milliseconds

# to update the time display as needed

# could use >200 ms, but display gets jerky

self.timeLbl.after(**200**, self.tick)

**class** **Weather**(Frame):

**def** **\_\_init\_\_**(self, parent, \*args, \*\*kwargs):

Frame.\_\_init\_\_(self, parent, bg='black')

self.temperature = ''

self.forecast = ''

self.location = ''

self.currently = ''

self.icon = ''

self.degreeFrm = Frame(self, bg="black")

self.degreeFrm.pack(side=TOP, anchor=W)

self.temperatureLbl = Label(self.degreeFrm, font=('Helvetica', xlarge\_text\_size), fg="white", bg="black")

self.temperatureLbl.pack(side=LEFT, anchor=N)

self.iconLbl = Label(self.degreeFrm, bg="black")

self.iconLbl.pack(side=LEFT, anchor=N, padx=**20**)

self.currentlyLbl = Label(self, font=('Helvetica', medium\_text\_size), fg="white", bg="black")

self.currentlyLbl.pack(side=TOP, anchor=W)

self.forecastLbl = Label(self, font=('Helvetica', small\_text\_size), fg="white", bg="black")

self.forecastLbl.pack(side=TOP, anchor=W)

self.locationLbl = Label(self, font=('Helvetica', small\_text\_size), fg="white", bg="black")

self.locationLbl.pack(side=TOP, anchor=W)

self.get\_weather()

**def** **get\_ip**(self):

**try**:

ip\_url = "http://jsonip.com/"

req = requests.get(ip\_url)

ip\_json = json.loads(req.text)

**return** ip\_json['ip']

**except** **Exception** **as** e:

traceback.print\_exc()

**return** "Error: %s. Cannot get ip." % e

**def** **get\_weather**(self):

**try**:

**if** latitude **is** None **and** longitude **is** None:

# get location

location\_req\_url = "http://freegeoip.net/json/%s" % self.get\_ip()

r = requests.get(location\_req\_url)

location\_obj = json.loads(r.text)

lat = location\_obj['latitude']

lon = location\_obj['longitude']

location2 = "%s, %s" % (location\_obj['city'], location\_obj['region\_code'])

# get weather

weather\_req\_url = "https://api.darksky.net/forecast/%s/%s,%s?lang=%s&units=%s" % (weather\_api\_token, lat,lon,weather\_lang,weather\_unit)

**else**:

location2 = ""

# get weather

weather\_req\_url = "https://api.darksky.net/forecast/%s/%s,%s?lang=%s&units=%s" % (weather\_api\_token, latitude, longitude, weather\_lang, weather\_unit)

r = requests.get(weather\_req\_url)

weather\_obj = json.loads(r.text)

degree\_sign= u'**\N{DEGREE SIGN}**'

temperature2 = "%s%s" % (str(int(weather\_obj['currently']['temperature'])), degree\_sign)

currently2 = weather\_obj['currently']['summary']

forecast2 = weather\_obj["hourly"]["summary"]

icon\_id = weather\_obj['currently']['icon']

icon2 = None

**if** icon\_id **in** icon\_lookup:

icon2 = icon\_lookup[icon\_id]

**if** icon2 **is** **not** None:

**if** self.icon != icon2:

self.icon = icon2

image = Image.open(icon2)

image = image.resize((**100**, **100**), Image.ANTIALIAS)

image = image.convert('RGB')

photo = ImageTk.PhotoImage(image)

self.iconLbl.config(image=photo)

self.iconLbl.image = photo

**else**:

# remove image

self.iconLbl.config(image='')

**if** self.currently != currently2:

self.currently = currently2

self.currentlyLbl.config(text=currently2)

**if** self.forecast != forecast2:

self.forecast = forecast2

self.forecastLbl.config(text=forecast2)

**if** self.temperature != temperature2:

self.temperature = temperature2

self.temperatureLbl.config(text=temperature2)

**if** self.location != location2:

**if** location2 == ", ":

self.location = "Cannot Pinpoint Location"

self.locationLbl.config(text="Cannot Pinpoint Location")

**else**:

self.location = location2

self.locationLbl.config(text=location2)

**except** **Exception** **as** e:

traceback.print\_exc()

**print** "Error: %s. Cannot get weather." % e

self.after(**600000**, self.get\_weather)

**@staticmethod**

**def** **convert\_kelvin\_to\_fahrenheit**(kelvin\_temp):

**return** **1.8** \* (kelvin\_temp - **273**) + **32**

**class** **News**(Frame):

**def** **\_\_init\_\_**(self, parent, \*args, \*\*kwargs):

Frame.\_\_init\_\_(self, parent, \*args, \*\*kwargs)

self.config(bg='black')

self.title = 'News' # 'News' is more internationally generic

self.newsLbl = Label(self, text=self.title, font=('Helvetica', medium\_text\_size), fg="white", bg="black")

self.newsLbl.pack(side=TOP, anchor=W)

self.headlinesContainer = Frame(self, bg="black")

self.headlinesContainer.pack(side=TOP)

self.get\_headlines()

**def** **get\_headlines**(self):

**try**:

# remove all children

**for** widget **in** self.headlinesContainer.winfo\_children():

widget.destroy()

**if** news\_country\_code == None:

headlines\_url = "https://news.google.com/news?ned=us&output=rss"

**else**:

headlines\_url = "https://news.google.com/news?ned=%s&output=rss" % news\_country\_code

feed = feedparser.parse(headlines\_url)

**for** post **in** feed.entries[**0**:**5**]:

headline = NewsHeadline(self.headlinesContainer, post.title)

headline.pack(side=TOP, anchor=W)

**except** **Exception** **as** e:

traceback.print\_exc()

**print** "Error: %s. Cannot get news." % e

self.after(**600000**, self.get\_headlines)

**class** **NewsHeadline**(Frame):

**def** **\_\_init\_\_**(self, parent, event\_name=""):

Frame.\_\_init\_\_(self, parent, bg='black')

image = Image.open("assets/Newspaper.png")

image = image.resize((**25**, **25**), Image.ANTIALIAS)

image = image.convert('RGB')

photo = ImageTk.PhotoImage(image)

self.iconLbl = Label(self, bg='black', image=photo)

self.iconLbl.image = photo

self.iconLbl.pack(side=LEFT, anchor=N)

self.eventName = event\_name

self.eventNameLbl = Label(self, text=self.eventName, font=('Helvetica', small\_text\_size), fg="white", bg="black")

self.eventNameLbl.pack(side=LEFT, anchor=N)

**class** **Calendar**(Frame):

**def** **\_\_init\_\_**(self, parent, \*args, \*\*kwargs):

Frame.\_\_init\_\_(self, parent, bg='black')

self.title = 'Calendar Events'

self.calendarLbl = Label(self, text=self.title, font=('Helvetica', medium\_text\_size), fg="white", bg="black")

self.calendarLbl.pack(side=TOP, anchor=E)

self.calendarEventContainer = Frame(self, bg='black')

self.calendarEventContainer.pack(side=TOP, anchor=E)

self.get\_events()

**def** **get\_events**(self):

#TODO: implement this method

# reference https://developers.google.com/google-apps/calendar/quickstart/python

# remove all children

**for** widget **in** self.calendarEventContainer.winfo\_children():

widget.destroy()

calendar\_event = CalendarEvent(self.calendarEventContainer)

calendar\_event.pack(side=TOP, anchor=E)

**pass**

**class** **CalendarEvent**(Frame):

**def** **\_\_init\_\_**(self, parent, event\_name="Event 1"):

Frame.\_\_init\_\_(self, parent, bg='black')

self.eventName = event\_name

self.eventNameLbl = Label(self, text=self.eventName, font=('Helvetica', small\_text\_size), fg="white", bg="black")

self.eventNameLbl.pack(side=TOP, anchor=E)

**class** **Abdo**(Frame):

**def** **\_\_init\_\_**(self, parent, event\_name="Event 1"):

Frame.\_\_init\_\_(self, parent, bg='black')

# self.a=u'\u0627\u0644\u0633\u0644\u0627\u0645 \u0639\u0644\u064a\u0643\u0645'

#encoded=self.a.encode('utf-8')

self.a ='Ø§Ù„Ø³Ù„Ø§Ù… '

self.txt = Label(self, text=self.a, font=('Helvetica',**44**), fg="white", bg="black")

self.txt.pack(side=TOP, anchor=E)

**class** **FullscreenWindow**:

**def** **\_\_init\_\_**(self):

self.tk = Tk()

self.tk.configure(background='black')

self.topFrame = Frame(self.tk, background = 'black')

self.bottomFrame = Frame(self.tk, background = 'black')

self.topFrame.pack(side = TOP, fill=BOTH, expand = YES)

self.bottomFrame.pack(side = BOTTOM, fill=BOTH, expand = YES)

self.state = False

self.tk.bind("<Return>", self.toggle\_fullscreen)

self.tk.bind("<Escape>", self.end\_fullscreen)

#text

self.abdo =Abdo(self.topFrame)

self.abdo.pack(side=TOP, anchor=N, padx=**100**, pady=**60**)

# clock

self.clock = Clock(self.topFrame)

self.clock.pack(side=RIGHT, anchor=N, padx=**100**, pady=**60**)

# weather

self.weather = Weather(self.topFrame)

self.weather.pack(side=LEFT, anchor=N, padx=**100**, pady=**60**)

# news

self.news = News(self.bottomFrame)

self.news.pack(side=LEFT, anchor=S, padx=**100**, pady=**60**)

# calender - removing for now

# self.calender = Calendar(self.bottomFrame)

# self.calender.pack(side = RIGHT, anchor=S, padx=100, pady=60)

**def** **toggle\_fullscreen**(self, event=None):

self.state = **not** self.state # Just toggling the boolean

self.tk.attributes("-fullscreen", self.state)

**return** "break"

**def** **end\_fullscreen**(self, event=None):

self.state = False

self.tk.attributes("-fullscreen", False)

**return** "break"

**if** \_\_name\_\_ == '\_\_main\_\_':

w = FullscreenWindow()

w.tk.mainloop()