chronos

A CircuitPython helper class for converting time objects. It currently consists of three helpers for converting time values. The first helper confirms whether a specified year is a leap year. The second helper detects North American Daylight Saving Time (DST) from a Standard Time (xST) structured time object. The third helper detects and converts an xST structured time object to a DST value, if appropriate.

- Author(s): JG for Cedar Grove Studios

Implementation Notes

Hardware:

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://github.com/adafruit/circuitpython/releases

helper_class chronos.leap_year(*year*)

Helper representing the leap year detector. Returns a value of True if the input year value is a valid leap year. Returns False if not a valid leap year.

Parameters:

- **year** – The input year value. Can be any positive or negative integer. No default value.

Example:

```
>>> from cedargrove_unit_converter.chronos import leap_year
>>> leap_year(2000)  # Leap Year Determination
True
>>> leap_year(2020)
True
>>> leap_year(2021)
False
>>> leap_year(2024)
True
>>> leap_year(2100)
False
>>> leap_year(2104)
True
```

helper_class detect_dst(*datetime*)

Helper representing the daylight saving time detector. Returns a value of True if the input structured time value represents a date when a DST adjustment is to be applied. Returns False if the input value is not to be adjusted for DST.

Parameters:

- **datetime** – The Standard Time structured time input value. Can be any structured time value within the specified date calculation range of CircuitPython, currently January 1, 2000 00:00:00 to January 19, 2038 03:14:07. No default value.

helper_class adjust_dst(*datetime*)

Helper representing the daylight saving time adjuster. Returns an updated datetime value and True DST flag value if the input structured time value represents a date when a DST adjustment is to be applied. Returns the unchanged input datetime value and False DST flag value if the input value is not adjusted for DST. This helper uses the **detect_dst()** helper to determine whether to apply the DST conversion

Parameters:

- **datetime** – The Standard Time structured time input value. Can be any structured time value within the specified date calculation range of CircuitPython, currently January 1, 2000 00:00:00 to January 19, 2038 03:14:07. No default value.

Example:

```
# Example code
import time
from cedargrove_unit_converter.chronos import adjust_dst

# Today's date: 11/01/2020 00:00 Standard Time (xST)
datetime = time.struct_time((2020,11,1,0,0,0,6,0,-1))

# Check datetime and adjust if DST
adj_datetime, is_dst = adjust_dst(datetime)

if is_dst:
    flag_text = "DST"
else:
    flag_text = "xST"

# Print the submitted time
print("     {}/{}/{} {:02}:{:02}:{:02}  week_day={}".format(
      datetime.tm_mon, datetime.tm_mday, datetime.tm_year,
      datetime.tm_hour, datetime.tm_min, datetime.tm_sec,
      datetime.tm_wday))

# Print the adjusted time
print("{}: {}/{}/{} {:02}:{:02}:{:02}  week_day={}".format(flag_text,
      adj_datetime.tm_mon, adj_datetime.tm_mday, adj_datetime.tm_year,
      adj_datetime.tm_hour, adj_datetime.tm_min, adj_datetime.tm_sec,
      adj_datetime.tm_wday))
```

```
# for 11/1/2020 00:00 xST input; first Sunday of November 01:00 DST
code.py output:
     11/1/2020 00:00:00  week_day=6
DST: 11/1/2020 01:00:00  week_day=6
```

```
# for 11/1/2020 01:00 xST input; first Sunday of November 02:00 DST
# falls back to 01:00 xST
code.py output:
     11/1/2020 01:00:00  week_day=6
xST: 11/1/2020 01:00:00  week_day=6
```