

Table of Contents

Overview	1
Requirements.....	1
Capabilities and Workflow options	1
Command-Line Usage / Execution	2
Command-Line Input Parameters	2
Command-Line Examples.....	3
Python Code Sample	4
Release History.....	5

Overview

The 'datetimeUtils.py' Python script is a Date/Time support Utility, or Library, used by conversion scripts leveraged by the OverwriteFS.py script. The Functions and Classes found here can be used to perform specific Date and Time conversion or interpretation across many Conversion scripts.

Requirements

1. Python 3.x or Python Notebook

Capabilities and Workflow options

- Decode Date Time string and return a datetime object using the 'decodeDatetime' Function. Optional to include as UTC time zone or to include the time zone offset value from UTC. Currently the only function available!
- Starting at v2.1, the 'decodeDatetime' Function now supports returning the format string as part of a Tuple containing the Datetime object and the Format string. It can support Positive and Negative Timestamp "epoch" values as input. It allows for Ordinate Indicators on standalone numbers (1st, 2nd, 3rd, 12th)

Command-Line Usage / Execution

To execute script, open a Python Command Line Window and type:

```
'Python <path>\datetimeUtils.py "<a date and or time text string>" [<verbose> [<utcOut> [  
<returnFormat> [<asMicroseconds>]]]]'
```

Command-Line Input Parameters

- Available Input Parameters

< a date and or time text string >: (required) Enter text containing a Date and or Time string that should be decoded. Can include Month name (long or abbreviated or as number), Day name (long or abbreviated or as a number), Year (4 or 2 digit), Hour (12 or 24 hour), Minutes, Seconds, Microseconds, Time Zone name or offset. Or a Positive or Negative Timestamp value 13 digits long, 10 digits + microseconds or as a float. Standalone numbers can include Ordinate Indicators (1st, 2nd, 3rd, 12th)

<verbose>: (optional but required if <utcOut> is used) True or False to include or exclude details displayed to the console. Default is True

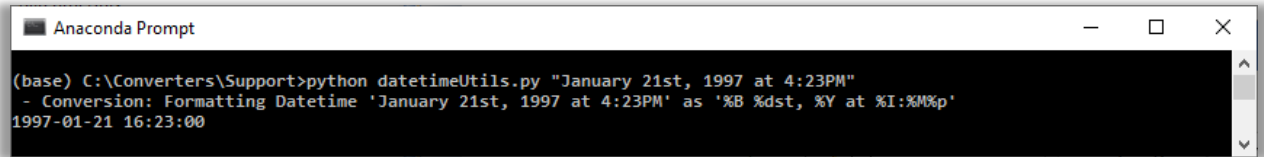
<utcOut>: (optional) True or False to convert to UTC date time or simply include offset from UTC. Default is False

<returnFormat>: (optional) True or False to return Tuple containing converted Datetime Object and Format string used to convert it. Default is False

<asMicroseconds>: (optional) True or False, is [epoch](#) time represented in Microseconds? Up to 13-digit Integer as Microseconds, True. Up to 10-digit Integer or Float as Seconds, False. Default: True

Command-Line Examples

- Decode a text date and time, no time zone.

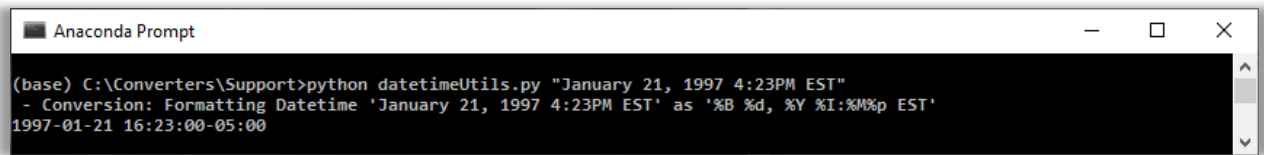


```

Anaconda Prompt
(base) C:\Converters\Support>python datetimeUtils.py "January 21st, 1997 at 4:23PM"
- Conversion: Formatting Datetime 'January 21st, 1997 at 4:23PM' as '%B %dst, %Y at %I:%M%p'
1997-01-21 16:23:00

```

- Decode a text date and time, with a time zone as UTC offset.

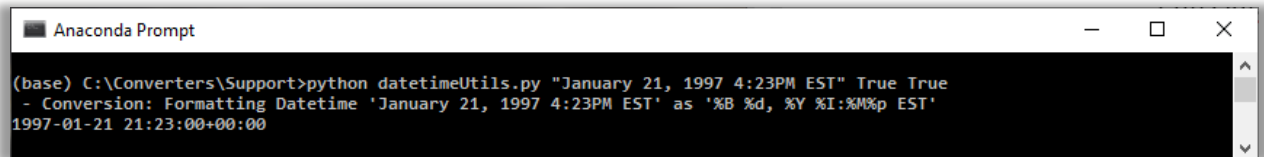


```

Anaconda Prompt
(base) C:\Converters\Support>python datetimeUtils.py "January 21, 1997 4:23PM EST"
- Conversion: Formatting Datetime 'January 21, 1997 4:23PM EST' as '%B %d, %Y %I:%M%p EST'
1997-01-21 16:23:00-05:00

```

- Decode a text date and time, with a time zone converted to UTC.

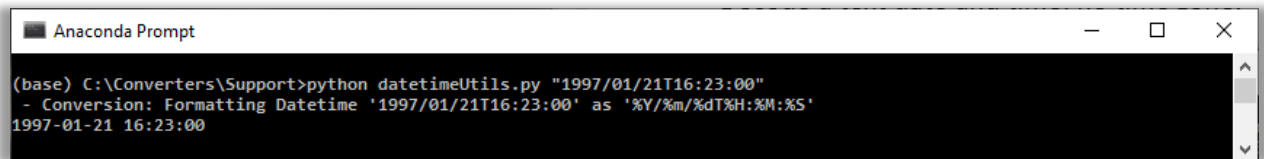


```

Anaconda Prompt
(base) C:\Converters\Support>python datetimeUtils.py "January 21, 1997 4:23PM EST" True True
- Conversion: Formatting Datetime 'January 21, 1997 4:23PM EST' as '%B %d, %Y %I:%M%p EST'
1997-01-21 21:23:00+00:00

```

- Decode a standard numeric date and time, no time zone.

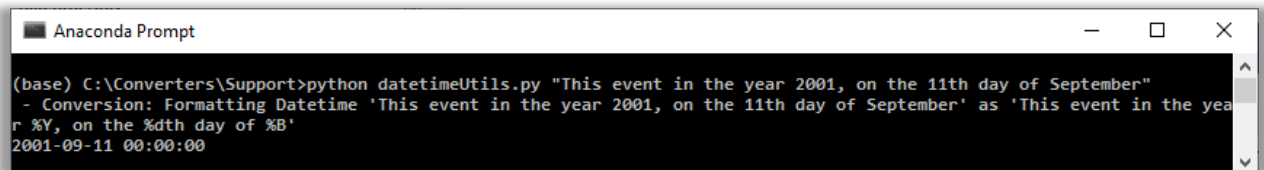


```

Anaconda Prompt
(base) C:\Converters\Support>python datetimeUtils.py "1997/01/21T16:23:00"
- Conversion: Formatting Datetime '1997/01/21T16:23:00' as '%Y/%m/%dT%H:%M:%S'
1997-01-21 16:23:00

```

- Decoding a text sentence containing a date and time.

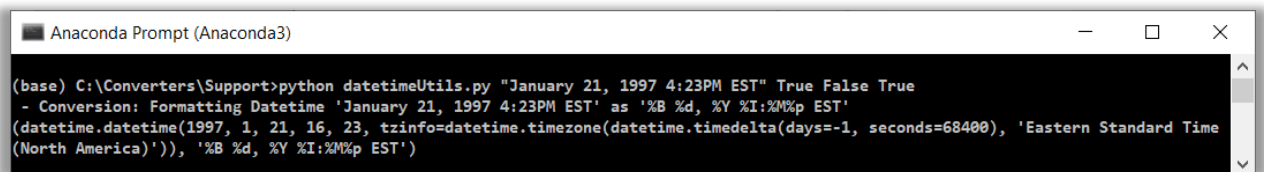


```

Anaconda Prompt
(base) C:\Converters\Support>python datetimeUtils.py "This event in the year 2001, on the 11th day of September"
- Conversion: Formatting Datetime 'This event in the year 2001, on the 11th day of September' as 'This event in the year %Y, on the %dth day of %B'
2001-09-11 00:00:00

```

- Return Tuple decoding a time zone enabled text sentence.

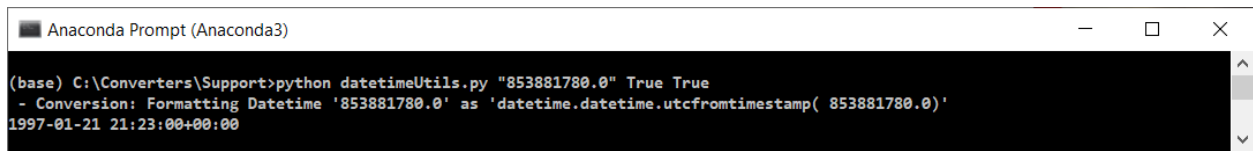


```

Anaconda Prompt (Anaconda3)
(base) C:\Converters\Support>python datetimeUtils.py "January 21, 1997 4:23PM EST" True False True
- Conversion: Formatting Datetime 'January 21, 1997 4:23PM EST' as '%B %d, %Y %I:%M%p EST'
(datetime.datetime(1997, 1, 21, 16, 23, tzinfo=datetime.timezone(datetime.timedelta(days=-1, seconds=68400), 'Eastern Standard Time (North America)')), '%B %d, %Y %I:%M%p EST')

```

- Decode [epoch](#) number as date and time.



```
Anaconda Prompt (Anaconda3)
(base) C:\Converters\Support>python datetimeUtils.py "853881780.0" True True
- Conversion: Formatting Datetime '853881780.0' as 'datetime.datetime.utcfromtimestamp( 853881780.0)'
1997-01-21 21:23:00+00:00
```

Python Code Sample

- Example of how to import and use the datetimeUtils library to decode a date time string, returning a Datetime object. This is the only Function available as of this time.

```
import datetimeUtils

dt = datetimeUtils.decodeDatetime( "In the year 2001 on the 11 th day of September at 9:00 AM")
print( dt)
```

Release History

- **September 2021, v2.0.0:** Initial public release.
- **October 2021, v2.1.0:** Added 'returnFormat' parameter option, providing ability to receive date format string with datetime object. Added support for Positive and Negative Timestamp "epoch" values. Added Ordinate Indicator detection for standalone numbers. Hardened Day and Month name detection. Fixed 'utcOut' request failure when date/time details do not contain a Timezone.
- **December 2021, v2.2.0:** Added 'asMicroseconds' to input options.